# Hesaplama Bilimi Matematik

Dr. Cahit Karakuş

“Balbiti”

# İçindekiler


**1. Yapay Zeka ve Matematik** ..... 4

**2. Lineer Cebir** ..... 9

2.1. Lineer Denklemler ..... 11

2.2. Korelasyon Analizi ..... 12

2.3. Regresyon ..... 17

**2.3.1. Interpolasyon** ..... 18

**2.3.2. Doğrusal (Lineer) Regresyon** ..... 20

**2.3.3. İkinci Mertebe En Küçük Kareler Yöntemi** ..... 24

**2.3.4. Genel Polinom Regresyon Modeli** ..... 25

2.4. Vektörler ..... 31

2.5. Matris ..... 38

2.6. Özdeğerler ve Özvektörler ..... 44

**3. Hesaplamalı Matematik** ..... 69

3.1. Derivatives ..... 70

3.2. Birinci ve İkinci türevin anlamı ..... 73

3.3. Integration ..... 75

3.4. Diferansiyel Denklemlerin Faz Düzlemleri ..... 77

3.5. Diferansiyel Denklemler - Doğrultu alanları ve çözüm eğrileri ..... 87

3.6. İkinci mertebeden diferansiyel denklemlerin faz düzlemleri ..... 95

3.7. Denge durum noktaları ve faz düzlemleri ..... 104

3.8. Doğrusal Olmayan Otonom Diferansiyel Denklem Sistemleri ..... 108

3.9. Directional Derivatives and Gradient Vectors ..... 111

3.10. The Potansiyel fonksiyon ..... 120

**4. Fonksiyon Optimizasyonu** ..... 121

4.1. Veri biliminde cebir ve diferansiyel hesaplama ..... 121

4.2. Fonksiyon optimizasyonu için tek boyutlu (1d) test fonksiyonları ..... 139

4.3. Two-Dimensional (2D) Test Functions for Function Optimization ..... 153

4.4. Optimization algorithms: the Newton Method ..... 167

**5. Fourier Transform** ..... 173

5.1. Matlab ile Sinyal Analizi ..... 183

5.2. Fourier analysis and Matlab ..... 185

**6. Laplace Transformation** ..... 199
**7. Tanımlayıcı İstatistik** ..... 206
**8. Olasılık** ..... 217
8.1. Hipotez Testi ..... 225
**8.1.1. T-Testi** ..... 229
**8.1.2. Chi-Square Bağımsızlık Testi** ..... 231
**8.1.3. Güç Analizi** ..... 235
8.2. Markov Zinciri ..... 238
**9. Standardization and Normalization** ..... 244
**10. Information Theory** ..... 250
**11. Robotiğin Arkasındaki Matematik** ..... 260
**12. Mühendislikte sistemlerin matematiksel modellenmesi** ..... 268

# 1. Yapay Zeka ve Matematik

Yapay zeka, sanal gerçeklikten işlevsel araçlara kadar, hayatlarımızı daha önce hiç kimsenin görmediği veya beklemediği şekillerde işgal etmeye başladı. Yapay Zeka araçları ve otonom sohbet algoritmaları, hızla gelişen teknoloji alanında bir atılımın eşiğinde. Yapay Zeka sihir değildir; sadece matematiktir. Düşünerek karar veren makinelerin arkasındaki fikirler ve insan davranışını taklit etme olasılığı matematiksel kavramlar yardımıyla yapılmaktadır.

Yapay Zeka, kökü Matematik olan bir ağacın gövdesi ve dallarıdır. Yapay Zeka alanında kariyer yapmak ve başarılı olmak istiyorsanız temel uygulamalı matematik çalışmanız gerekmektedir; sadece bilim kurgu hayranı olmak yeterli olmayacaktır. Bir Yapay Zeka kariyeri inşa edeceksin ve uzaydaki maden yağmalamasına hükmedeceksiniz, matematikle arkadaş olun, göreceksiniz dünyanız sallanmaya başlayacaktır. Bir matematikçi ve yapay zeka uzmanı olarak, yapay zeka ve matematik arasındaki büyülü bağlantıyı sizinle paylaşmak istiyorum.

Yapay zeka problemleri iki genel kategoride oluşur; Arama Problemleri ve Temsil Problemleri. Bunları, Kurallar, Çerçeveler, Mantıklar ve Ağlar gibi birbirine bağlı modeller ve araçlar takip eder. Hepsi çok matematiksel konular.

Yapay zekanın birincil amacı, insan anlayışı için kabul edilebilir bir model oluşturmaktır. Ve bu modeller, matematiğin çeşitli dallarından gelen fikir ve stratejilerle hazırlanabilir.
Kendi kendine giden arabaları düşünün; amaçları, video görüntülerindeki nesneleri ve insanları tanımaktır. Kameralardan ve algılayıcılardan akan veri nehrinden beslenen veri denizinin içnde yüzecek bir araba düşünün. Bu arabaların arkasında minimizasyon süreçleri ve geri yayılım şeklinde matematik vardır. Matematik, bilim adaamlarının yüzlerce yıldır bilinen geleneksel yöntem ve teknikleri kullanarak zorlu derin soyut problemleri çözmelerine yardımcı olur.

Makine öğrenimi mühendisi, veri bilimcisi veya robot bilimcisi olarak kariyer yapmak istiyorsanız, matematikte başarılı olmanız gerekir. Matematik, yapay zekada hayati önem taşıyan analitik düşünme becerileri geliştirir.

**Yapay Zeka'da ne tür matematik kullanılır?**
Tüm önemli ilerlemelerin arkasında matematik vardır. Lineer cebir, matematik, oyun teorisi, olasılık, istatistik, gelişmiş lojistik regresyon ve Gradient Descent kavramlarının tümü, veri biliminin temel dayanaklarıdır.

Matematik, mantıksal akıl yürütmeyi ve ayrıntılara gösterilen özeni anlamada yardımcı olur. Baskı altında düşünme yeteneklerinizi geliştirir ve zihinsel dayanıklılığınızı artırır. Matematiksel kavramlar, varsayımsal veya sanal problemlerin gerçek çözümünü verir. Bu, yapıyla, bileşenlerde herhangi bir değişiklik yapsanız bile doğru kalan ilkeleri geliştirmekle ilgilidir.

Yapay zekada matematiğin üç ana dalı lineer cebir, hesaplamalı matematik ve olasılıktır.

**1. Cebir**

Cebir bilgisi belki de genel olarak matematik için temeldir. Toplama, çıkarma, çarpma ve bölme gibi matematiksel işlemlerin yanı sıra aşağıdakileri de bilmeniz gerekir:

- Üsler
- Kökler
- Faktöriyeller
- Toplamlar
- Bilimsel gösterimler

**2. Lineer Cebir**

Lineer Cebir, yapay zeka uzmanlarının onsuz yaşayamayacağı bir şey olan uygulamalı matematik alanıdır. Bu alanda uzmanlaşmadan asla iyi bir yapay zeka uzmanı olamazsınız. Skyler Speakman'ın dediği gibi, “Lineer Cebir, 21. yüzyılın matematiğidir.”

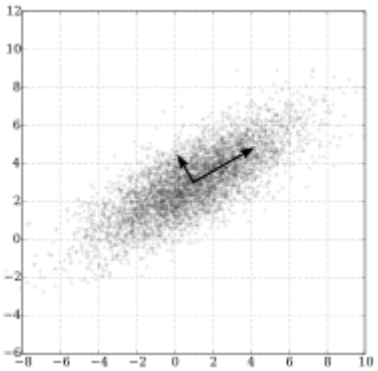
Lineer Cebirin skaler, vektörler, tensörler, matrisler, kümeler ve diziler yeni fikirler üretmeye yardımcı olur, bu yüzden yapay zeka uzmanları ve araştırmacıları için mutlaka öğrenilmesi gereken bir alandır. Topoloji, oyun teorisi, grafik teorisi, fonksiyonlar, doğrusal dönüşümler, özdeğerler ve özvektörler kavramları ile veri ve modelleri soyutlayabilirler.

Lineer Cebir, bilim ve mühendisliğin diğer birçok alanında olduğu gibi yapay zekada da birincil matematiksel hesaplama aracıdır. Bu alanla, temel matematiksel nesneleri ve özellikleri anlamanız gerekir:

- **Skaler büyüklükler:** Sayı ve birim kullanılarak belirtilebilen büyüklüklere skaler büyüklük denir. Örneğin 2 kg, 20 m gibi büyüklükler skaler büyüklüklerdir. Gerçek veya doğal tek bir sayı olabilir. Kütle, elektriksel yük, sıcaklık ve bir ortamdaki bir noktanın elektrik potansiyeli skaler niceliklerdir. 3 boyutlu evrende iki nokta arasındaki mesafe skalerdir, fakat birinden diğerine olan yön ile beraber belirtildiğinde vektörel bir nicelik olur. Fizikte ölçümü yapılan başlıca iki büyüklük vardır; skaler ve vektörel büyüklüklerdir.
- **Vektorel büyüklükler:** Büyüklüğü, başlangıç noktası, yönü ve ekseni ya da düzlemi olan büyüklüklere vektörel büyüklük denir.
- **Matematikte matris**, dikdörtgen bir sayılar tablosu veya daha genel bir açıklamayla, toplanabilir veya çarpılabilir soyut miktarlar tablosudur. Matrisler daha çok doğrusal denklemleri tanımlamak, doğrusal dönüşümlerde (lineer transformasyon) çarpanların

takibi ve iki parametreye bağlı verilerin kaydedilmesi amacıyla kullanılırlar. Matrislerin toplanabilir, çıkartılabilir, çarpılabilir, bölünebilir ve ayrıştırılabilir olmaları, doğrusal cebir kuramının temel kavramı olmalarını sağlamıştır.

- **Matematikte, tensör,** çok boyutlu verinin simgelenebildiği geometrik bir nesnedir. Skaler denilen yönsüz nicel büyüklükler, vektör denilen yönlü büyüklükler ve matris denilen iki boyutlu nesneler birer tensördür. Tensör, tüm bu nesnelerin genelleştirilmiş halidir ve çok boyutlu veri kümeleri için kullanılır. Nesnenin kaç boyutla ifade edildiğine de tensörün derecesi denilir. Bir skalerin derecesi sıfır, bir vektörün bir, bir matrisin ise ikidir. Tensörler üç ve üzeri dereceye sahip olabilir.Tensörler — N eksenli düzenli bir ızgara üzerinde düzenlenmiş sayıların bir N-D dizisi (N>2). Makine Öğrenimi, Derin Öğrenme ve Bilgisayarla Görmede Önemlidir.
- Özvektörler ve Özdeğerler — özel vektörler ve bunlara karşılık gelen skaler nicelik. Mühendislikte sık sık karşılaşılan problemlerin öznesidir. Bir vektörün özdeğerleri o matrisin karakteristik polinomunun kökleridir. Özdeğerler, özvektörler ve özuzaylar, bir matrisin özellikleridir ve matris hakkında önemli bilgiler verir. Matrislerin çarpanlarına ayrılmasında kullanılabilirler. Uygulamalı matematik alanlarında olduğu kadar finans ve kuantum mekaniğinde de kullanılır. Bir vektör üzerine uygulanan matris o vektörün hem büyüklüğünü, hem de yönünü değiştirir. Buna rağmen, bir matris bazı belirli vektörler üzerine etkidiğinde onların sadece büyüklüğünü değiştirir, doğrultularını değiştirmez (ancak vektörün yönü ters çevrilebilir). Doğrultusu değişmeyen bu vektörler söz konusu matrisin özvektörleri olarak adlandırılır. Bir matris, bir özvektörü üzerine etkidiğinde onun büyüklüğünü bir çarpan kadar katlar. Bu çarpan pozitif ise vektörün yönü değişmeden kalır, negatif ise vektörün yönü tersine döner (her iki durumda da vektörün doğrultusu değişmez). Bu çarpana, söz konusu özvektöre ilişkin özdeğer denir. Bir özuzay, aynı özdeğere sahip tüm özvektörlerin oluşturduğu kümedir. Bu kavramlar matrisler, vektörler ve doğrusal dönüşümler üzerinde tanımlıdır.
- İstatistikte**, temel bileşen analizi (TBA),** çok boyutlu uzaydaki bir verinin daha düşük boyutlu bir uzaya izdüşümünü, varyansı maksimize edecek şekilde bulma yöntemidir. Uzayda bir noktalar kümesi için, tüm noktalara ortalama uzaklığı en az olan "en uygun doğru" seçilir. Daha sonra bu doğruya dik olanlar arasından yine en uygun doğru seçilerek, bu adımlar, yeni bir boyutun varyansı belirli bir eşiğin altına inene kadar tekrarlanır. Bu sürecin sonunda elde edilen doğrular, bir doğrusal uzayın tabanlarını oluşturur. Bu taban vektörlerine temel bileşen denir. Verinin temel bileşenleri birbirinden bağımsız olur. Bu kavram bazen orijinal terimin kısaltması olan PCA (İngilizce: Principal component analysis) olarak da anılır. TBA'nın ana kullanım amaçları keşifsel veri analizi yapmak ve kestirimsel modeller oluşturmaktır.



“Temel Bileşenler Analizi” olan PCA tanıma, sınıflandırma, görüntü sıkıştırma alanlarında kullanılan yararlı bir istatistiksel tekniktir. Temel amacı yüksek boyutlu verilerde en yüksek varyans ile veri setini tutmak ancak bunu yaparken boyut indirgemeyi sağlamak olan bir tekniktir. Fazla boyutlu verilerdeki genel özellikleri bularak boyut sayısının azaltılmasını, verinin sıkıştırılmasını sağlar. Boyut azalmasıyla bazı özelliklerin kaybedileceği kesindir; fakat amaçlanan, bu kaybolan özelliklerin popülasyon hakkında çok az bilgi içeriyor olmasıdır. Bu yöntem, yüksek korelasyonlu değişkenleri bir araya getirerek, verilerdeki en çok varyasyonu oluşturan “temel bileşenler” olarak adlandırılan daha az sayıda yapay değişken kümesi oluşturur.

### 3. Matematiksel Hesaplama

Diferansiyel hesap, Çok değişkenli hesap, İntegral hesap, Gradyan iniş yoluyla hata minimizasyonu ve optimizasyonu, Limitler, Gelişmiş lojistik regresyonlar, matematiksel modellemede kullanılan tüm kavramlardır.

Matematik, parametrelerdeki, fonksiyonlardaki, hatalardaki ve yaklaşıklıklardaki değişikliklerle ilgilenir. Yapay Zekada çok boyutlu hesaplamalarda, sapma bulmada, yörünge öngörmede kullanılan matematiksel hesaplamada en önemli kavramlar (ayrıntılı olmamakla birlikte):

- Türevler — kurallar (toplama, çarpım, zincir kuralı vb.), hiperbolik türevler (tanh, cosh vb.) ve kısmi türevler.
- Vektör/Matris Hesabı — farklı türev operatörleri (Gradient, Jacobian, Hessian ve Laplacian)
- Gradyan (eğim) Algoritmaları — yerel/global maksimumlar ve minimumlar, eyer noktaları, dışbükey fonksiyonlar, yığınlar ve mini yığınlar, stokastik gradyan inişi ve performans karşılaştırması.

## 4. İstatistik & Olasılık Kavramları

Yapay zeka dünyasında çok fazla soyut problem var. Belirsizlik ve stokastikliği birçok biçimde yaşayabilirsiniz. Olasılık teorisi, belirsizlikle başa çıkmak için araçlar sunar. Bir olayın meydana gelme sıklığını analiz etmek için, bir olayın meydana gelme şansı olarak tanımlandığı için olasılık kavramları kullanılır.

Bir Robot düşünelim. Bir robot yalnızca belirli bir süre ileri gidebilir, ancak belirli bir mesafeye gidemez. Bilim adamları robotu ilerletmek için programlamasında matematiği kullanıyor. Ayrık rastgele değişkenler, sürekli rastgele değişkenler, Bayes Formülü ve normalleştirme, diğer lineer cebir kavramlarıyla birlikte Robotik navigasyon ve harekette kullanılan bazı olasılık kavramlarıdır.

Bu konu muhtemelen zamanınızın önemli bir bölümünü alacaktır. İyi haber: Bu kavramlar zor değil, bu yüzden bu kavramlarda ustalaşmamanız için hiçbir neden yok.

- Temel İstatistikler — Ortalama, medyan, mod, varyans, kovaryans vb.
- Olasılık — olaylarda (bağımlı ve bağımsız), örnek uzaylarda, koşullu olasılıkta temel kurallar.
- Rastgele değişkenler — sürekli ve ayrık, beklenti, varyans, dağılımlar (ortak ve koşullu).
- Bayes Teoremi — inançların geçerliliğini hesaplar. Bayesian yazılımı, makinelerin kalıpları tanımasına ve karar vermesine yardımcı olur.
- Maksimum Olabilirlik Tahmini (MLE) — parametre tahmini. Temel olasılık kavramları hakkında bilgi gerektirir (ortak olasılık ve olayların bağımsızlığı).
- Ortak Dağılımlar — binom, poisson, bernoulli, gauss, üstel.

## 5. Bilgi Teorisi Kavramları

Bu konu, AI ve Derin Öğrenmeye önemli katkılarda bulunan ve henüz birçok kişi tarafından bilinmeyen önemli bir alandır. Bunu matematik, istatistik ve olasılığın bir karışımı olarak düşünün.

- Entropi — Shannon Entropisi olarak da adlandırılır. Bir deneydeki belirsizliği ölçmek için kullanılır.
- Çapraz Entropi — iki olasılık dağılımını karşılaştırır ve bize bunların ne kadar benzer olduğunu söyler.
- Kullback Leibler Divergence — iki olasılık dağılımının ne kadar benzer olduğunun başka bir ölçüsü.
- Viterbi Algoritması — Doğal Dil İşleme (NLP) ve Konuşmada yaygın olarak kullanılır.
- Enkoder-Dekoder — Makine Çevirisi RNN'lerinde ve diğer modellerde kullanılır.

# 2. Lineer Cebir

Makine öğrenimi ve derin öğrenme sistemlerinin temeli tamamen matematik ilke ve kavramlarına dayanmaktadır. Matematiksel ilkelerin temel temellerini anlamak zorunludur. Modelin temel çizgisi ve inşası sırasında, boyutsallık laneti, düzenlileştirme, ikili, çok sınıflı, sıralı regresyon ve diğerleri gibi birçok matematiksel kavram akılda sanatsal olmalıdır. Yaygın olarak nöron olarak adlandırılan temel derin öğrenme birimi, tamamen matematiksel kavramına dayanır ve bu, girdi ve ağırlığı içeren çarpım değerlerinin toplamını içerir. Sigmoid, ReLU ve diğerleri gibi aktivasyon fonksiyonları matematiksel teoremler kullanılarak oluşturulmuştur.

Bunlar, makine öğrenimi ve derin öğrenmenin temel kavramlarını uygun şekilde anlamak için temel matematiksel alanlardır:

- Lineer Cebir.
- Vektör Analizi.
- Matris Ayrıştırma.
- Olasılık ve Dağılımlar.
- Analitik Geometri.

**Makine Öğrenimi ve Derin Öğrenmede Lineer Cebir**

Lineer cebir, vektörlerin mevcudiyeti ve vektörleri işlemek için çeşitli kurallar nedeniyle makine öğreniminde gerekli bir rol oynar. Makine öğreniminde çoğunlukla sınıflandırıcılar veya regresyon problemlerini ele alıyoruz ve ardından gerçek değerden tahmin edilen değere kadar hesaplama yaparak hata minimizasyon teknikleri uygulanıyor. Sonuç olarak, daha önce bahsedilen hesaplama setlerini işlemek için lineer cebir kullanıyoruz. Doğrusal cebir, büyük miktarda veriyi işler veya başka bir deyişle, “doğrusal cebir, verilerin temel matematiğidir”.

Makine öğrenimi (ML) ve derin öğrenmede kullandığımız lineer cebir alanlarından bazıları şunlardır:

- Vektör ve Matris.
- Lineer Denklemler Sistemi.
- Vektör alanı.
- Temel (Basis).

Ayrıca, lineer cebir yöntemlerini uyguladığımız makine öğrenimi (ML) ve derin öğrenme alanları şunlardır:

- Regresyon Doğrusunun Türetilmesi.
- Hedef değeri tahmin etmek için Doğrusal Denklem.
- Destek Vektör Makinesi Sınıflandırması (SVM).
- Boyutsal küçülme.
- Ortalama Kare Hata veya Kayıp fonksiyonu.

- Düzenlileştirme.
- Kovaryans matrisi.
- Evrişim.

$$xy^T \in \mathbb{R}^{m \times n} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \begin{bmatrix} y_1 & y_2 & \cdots & y_n \end{bmatrix} = \begin{bmatrix} x_1 y_1 & x_1 y_2 & \cdots & x_1 y_n \\ x_2 y_1 & x_2 y_2 & \cdots & x_2 y_n \\ \vdots & \vdots & \ddots & \vdots \\ x_m y_1 & x_m y_2 & \cdots & x_m y_n \end{bmatrix}$$

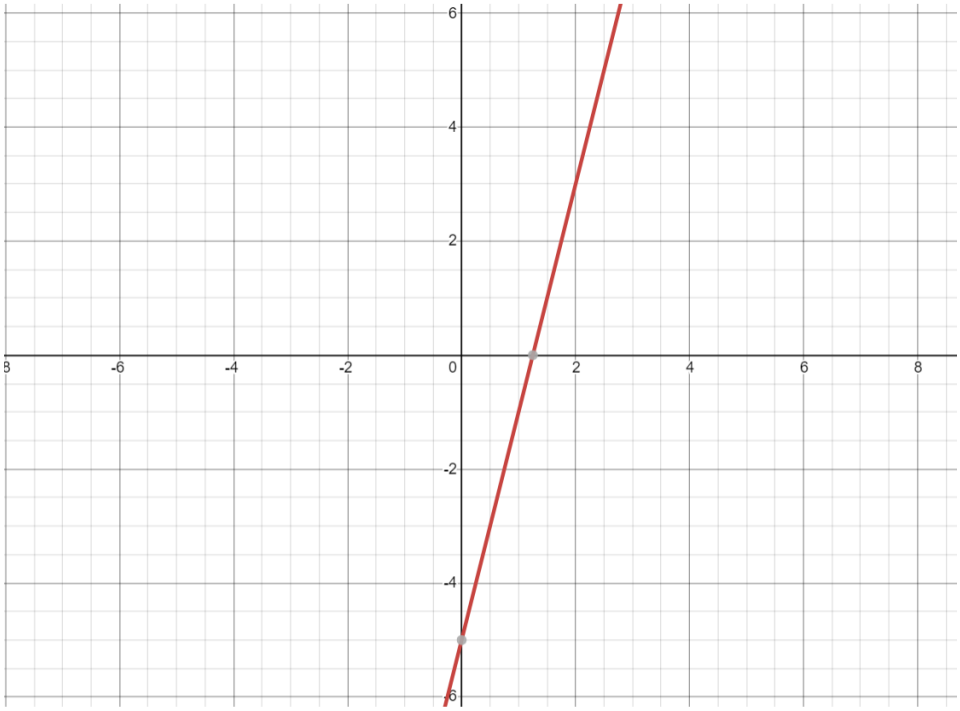
Şekil 2: Bir vektör örneği.

# 2.1. Lineer Denklemler

The linear equation is the central part of linear algebra by which many problems are formulated and solved. It is an equation for a straight line.

We represent the linear equation in figure:

$$y = 4x - 5$$




Example: y=4x-5 linner denkleminde *x = 0 için y=4*(0)-5, y=-5 bulunur. Denklemde y=0 için x=5/4 bulunr.*

**Doğrusal Regresyonda Doğrusal Denklemler**

Regresyon, düz çizgi için denklemi veren bir süreçtir. Belirli bir veri kümesiyle en uygun satırı bulmaya çalışır. Düz çizginin denklemi, doğrusal denklemi temel alır:

y = bx + a

a = Bu bir y kesişimidir ve doğrunun y eksenini kestiği noktayı belirler.

b = Bir eğimdir ve doğrunun eğim yönünü ve derecesini belirler.

Doğrusal regresyonda kaç adet değişken vardır. Anlamları nelerdir?

## 2.2. Korelasyon Analizi

Korelasyon analizi, bir fonksiyonun değişkenleri arasındaki ilişkinin yönünü, derecesini ve önemini ortaya koyan istatistiksel yöntemdir. Değişkenler arasındaki ilişkinin yönünü ve derecesini belirten katsayıya korelasyon katsayısı denir.

Korelasyon katsayısı küçük r harfi ile gösterilir ve r değeri -1 ile +1 arasında değerler alır. Eğer r değeri -1'e yakın değerler alıyor ise değişkenler arasında negatif yönde, +1'e yakın değerler alıyor ise pozitif yönde bir ilişki olduğu belirlenir. Eğer r değeri sıfıra yakın değerler alıyor ise iki değişken arasında bir ilişki olmadığı sonucuna varılır.

Korelasyon katsayısı negatif ise iki değişken arasında ters ilişki vardır, yani "değişkenlerden biri artarken diğeri azalmaktadır" denir. Korelasyon katsayısı pozitif ise "değişkenlerden biri artarken diğeride artmaktadır" yorumu yapılır.

Çok sayıda korelasyon analizi mevcuttur. Ancak en yaygın kullanılan korelasyon analizleri;

- Pearson çarpım moment korelasyon katsayısı
- Spearman'ın sıralama korelasyon katsayısı

Verilerin normal dağılıma sahip olması durumunda Pearson korelasyon katsayısı, verilerin normal dağılmadığı durumda ise Spearman Rank korelasyon katsayısı tercih edilir. Bir korelasyon katsayısının yorumlanabilmesi için p değerinin 0.05 den daha küçük olması gerekir.

**Pearson Çarpım Moment Korelasyon Katsayısı:**

$$r = \frac{n(\sum x_i y_i) - (\sum x_i)(\sum y_i)}{\sqrt{n(\sum x_i^2) - (\sum x_i)^2}\sqrt{n(\sum y_i^2) - (\sum y_i)^2}}$$

**Korelasyon katsayısı (r) nın yorumu;**

- r<0.2 ise çok zayıf ilişki yada korelasyon yok
- 0.2-0.4 arasında ise zayıf korelasyon
- 0.4-0.6 arasında ise orta şiddette korelasyon
- 0.6-0.8 arasında ise yüksek korelasyon
- 0.8> ise çok yüksek korelasyon olduğu yorumu yapılır.

**Örnek:**

| SUBJECT | AGE X | GLUCOSE LEVEL Y | XY | $X^2$ | $Y^2$ |
|---|---|---|---|---|---|
| 1 | 43 | 99 | 4257 | 1849 | 9801 |
| 2 | 21 | 65 | 1365 | 441 | 4225 |
| 3 | 25 | 79 | 1975 | 625 | 6241 |
| 4 | 42 | 75 | 3150 | 1764 | 5625 |
| 5 | 57 | 87 | 4959 | 3249 | 7569 |
| 6 | 59 | 81 | 4779 | 3481 | 6561 |
| Σ | 247 | 486 | 20485 | 11409 | 40022 |

The answer is: r=**2868 / 5413.27 = 0.529809**

From our table:

- Σx = 247
- Σy = 486
- Σxy = 20,485
- $\Sigma x^2$ = 11,409
- $\Sigma y^2$ = 40,022
- n is the sample size, n = 6

The correlation coefficient = 6(20,485) – (247 × 486) / [√[[6(11,409) – ($247^2$)] × [6(40,022) – $486^2$]]]

= 0.5298

The range of the correlation coefficient is from -1 to 1. Our result is 0.5298 or 52.98%, which means the variables have a moderate positive correlation.

**Örnek:**

Σx = 60, Σy = 60, Σxy = 400, Σx^2 = 400, Σy^2= 400, n = 10

$$r = \frac{n \sum xy - (\sum x)(\sum y)}{\sqrt{n \sum x^2 - (\sum x)^2}\sqrt{n \sum y^2 - (\sum y)^2}}$$

r=1, çok yüksek korelasyon olduğu yorumu yapılır

**Örnek:**
Aşağıdaki veriler için korelasyon katsayısı r'yi hesaplayın.

| $x$ | $y$ | $xy$ | $x^2$ | $y^2$ |
|---|---|---|---|---|
| 1 | $-3$ | $-3$ | 1 | 9 |
| 2 | $-1$ | $-2$ | 4 | 1 |
| 3 | 0 | 0 | 9 | 0 |
| 4 | 1 | 4 | 16 | 1 |
| 5 | 2 | 10 | 25 | 4 |
| $\Sigma x = 15$ | $\Sigma y = -1$ | $\Sigma xy = 9$ | $\Sigma x^2 = 55$ | $\Sigma y^2 = 15$ |

$$r = \frac{n\Sigma xy - (\Sigma x)(\Sigma y)}{\sqrt{n\Sigma x^2 - (\Sigma x)^2}\sqrt{n\Sigma y^2 - (\Sigma y)^2}} = \frac{5(9) - (15)(-1)}{\sqrt{5(55) - 15^2}\sqrt{5(15) - (-1)^2}}$$

$$= \frac{60}{\sqrt{50}\sqrt{74}} \approx 0.986$$

X ve y arasında güçlü bir pozitif doğrusal korelasyon vardır.

**Spearman'ın sıralama korelasyon katsayısı:**
İstatistik bilim dalında, Spearman'ın sıralama korelasyon katsayısı ismi verilen istatistiksel ölçüyü ilk ortaya atan Amerikan istatistikçi Charles Spearman'a atfen adlandırılmıştır. Matematik notasyon olarak çok defa eski Yunan harfi ρ (rho okunur) ile belirtilir.

Parametrik olmayan istatistik ölçüsüdür ve iki değişken arasındaki bağımlılık, yani korelasyon, ölçüsü olarak bulunup kullanılır. Bu demektir ki Spearman'in ρ katsayısı iki değişken için çokluluklar dağılımı hakkında hiçbir varsayım yapmayarak, bu iki değişken arasında bulunan bağlantının herhangi bir monotonik fonksiyon ile ne kadar iyi betimlenebilineceğini değerlendirmek amaçlı incelemedir.

Prensip olarak Spearman'ın sıralama korelasyon katsayısı ρ, Pearson çarpım-moment korelasyon katsayısının özel bir halidir. ρ değerinin hesaplanması için iki değişken (Y ve X) içinde örneklem verilerinin sıralama düzeninde olmaları gereklidir. Genel olarak, örneklem verileri için bu koşul uygun değildir ve veriler sıralama düzeni halinde olmadan oransal ölçekli veya aralıksal ölçekli veya sırasal ölçekli olarak bulunur ve bu halde bir dönüşümle sıralama düzeni haline sokulurlar. Böylece ρ formulü için sıralama düzenli xi ve yi örneklem verileri kullanılır.

Sonra iki değişken için karşılıklı veri elemanları (xi ve yi)'nin sıra numaraları arasındaki fark di, i=1,...n olarak bulunur. Bu tüm karşılıklı veriler (i=1...n) için uygulanır. Eğer sıra numaraları arasında hiç beraberlik yoksa, ρ değerini bulmak için şu formül kullanılır:

$$\rho = 1 - \frac{6\sum_{i=1}^{n} d_i}{n(n^2-1)}$$

Burada
di=xi-yi: i elamanı Xi ile Yi sıra numaraları arasındaki fark;
n : iki değişkenli örneklemde toplam gözlem sayısıdır.

**Örnek:**
Aşağıdaki tabloda iki değişken X ve Y için n=8 gözlem sayılı örneklem verileri için Spearman'ın sıralama korelasyon katsayısı ρ hesaplanması için örneğin verilmektedir. [A] ve [B] sütunlarında bu iki değişken X ve Y için örneklem verileri verilmiştir. [C] ve [D] sütunlarında bu iki değişkenlerin verileri için ayrı ayrı sıralama düzeni uygulanıp sıra numaraları x ve y olarak verilmiştir. X için verilerde 2 değişik beraberlik görülmektedir: 3 ve 10. Bu nedenle iki tekrarlı 3 için verilen sıra numaraları ortalaması (2+3)/2= 2,5 dur. Aynı şeklide 2 tekrarlı 10 için sıra numaraları 7,5 7,5 olarak verilmiştir. Y için verilerde ise 1,5 için 2 beraberlik ve 5 için 2 beraberlik bulunmaktadır ve bunlara da ortalama sıra numaraları verilmiştir. Sütun [E]de sıra numaraları farkları d verilmekte ve son [F] sütununda fark kareleri $d^2$ hesaplanmaktadır.

| [A] | [B] | [C] | [D] | [E] | [F] |
|---|---|---|---|---|---|
| X | Y | x : X için sıralama | y : Y için sıralama | d : Sıralama farkları | $d^2$ : Farkların karesi |
| 2 | 1,5 | 1 | 2,5 | -1,5 | 2,25 |
| 3 | 1,5 | 2,5 | 2,5 | 0 | 0 |
| 3 | 4 | 2,5 | 5 | -2,5 | 6,25 |
| 5 | 3 | 4 | 4 | 0 | 0 |
| 5,5 | 1 | 5 | 1 | 4 | 16 |
| 8 | 5 | 6 | 6,5 | -0,5 | 0,25 |
| 10 | 5 | 7,5 | 6,5 | 1 | 1 |
| 10 | 9,5 | 7,5 | 8 | -0,5 | 0,25 |
| | | | | Kareler Toplamı | 26 |

$$\rho = 1 - \frac{6x26}{8(8^2 - 1)}$$

ρ=0.6 bulunur.

Bu ρ=0.6 değeri sıfıra yakın pozitiftir. Sıfıra yakınlığı X ve Y sıralamaları arasındaki bağlantının (korelasyonun) az olduğunu gösterir ve negatif olma ise var zayıf bağlantının aksi yönde olduğunu ifade eder (yani X sıralaması artarsa Y sıralaması düşer ve aksi olur).

Bu veriler içinde beraberlikler bulunmaktadır. Bu nedenle kullanılan genel ρ formülü uygun sonuç vermeyebilir. Daha uygun sonuç bulmak için x ve y sıra numaraları için Pearson'un çarpım-moment korelasyon katsayısı bulunması tavsiye edilmektedir.

Örnek: Pearsonr fonksiyonunu içe aktarabilir ve iki dizi arasındaki Pearson korelasyon katsayısını hesaplayabiliriz.

```
#create two arrays
x = [3, 4, 4, 5, 7, 8, 10, 12, 13, 15]
y = [2, 4, 4, 5, 4, 7, 8, 19, 14, 10]

from scipy.stats.stats import pearsonr

#calculation correlation coefficient and p-value between x and y
r=pearsonr(x, y)
print(r)
```

sonuç: (0.8076177030748631, 0.004717255828132089)

Çıktıyı nasıl yorumlayacağınız aşağıda açıklanmıştır:
Pearson korelasyon katsayısı (r): 0.8076
İki kuyruklu p değeri: 0.0047

Korelasyon katsayısı 1'e yakın olduğu için bu bize iki değişken arasında güçlü bir pozitif ilişki olduğunu söyler.

Karşılık gelen p değeri .05'ten küçük olduğundan, iki değişken arasında istatistiksel olarak anlamlı bir ilişki olduğu sonucuna varırız.

## 2.3. Regresyon

En küçük kareler yöntemi, birbirine bağlı olarak değişen iki fiziksel büyüklük arasındaki matematiksel bağlantıyı, mümkün olduğunca gerçeğe uygun bir denklem olarak yazmak için kullanılan, standart bir regresyon yöntemidir. Bir başka deyişle bu yöntem, ölçüm sonucu elde edilmiş veri noktalarına "mümkün olduğu kadar yakın" geçecek bir fonksiyon eğrisi bulmaya yarar. Gauss-Markov Teoremi'ne göre en küçük kareler yöntemi, regresyon için optimal yöntemdir.

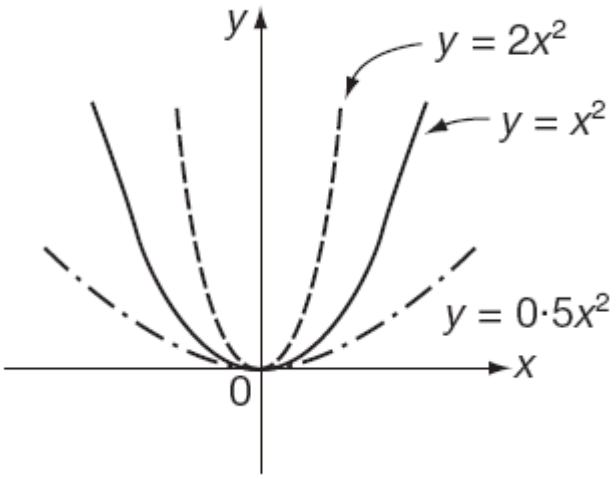
The variance of $\{x_1, \ldots, x_N\}$, denoted by $\sigma_x^2$, is

$$\sigma_x^2 = \frac{1}{N}\sum_{n=1}^{N}(x_i - \overline{x})^2;$$

the standard deviation $\sigma_x$ is the square root of the variance:

$$\sigma_x = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(x_i - \overline{x})^2}.$$

**Standard curves -** *Second-degree curves*




The simplest second-degree curve is expressed by:

$$y = x^2$$

Its graph is a parabola, symmetrical about
The y-axis and existing only for $y \geq 0$. $y = ax^2$ gives a thinner parabola if $a > 1$ and a flatter parabola if $0 < a < 1$. The general second-degree curve is:

$$y = ax^2 + bx + c$$

where *a*, *b* and *c* determine the position,'width' and orientation of the parabola.

Linear Regression: $y(x) = ax + b$

Polynomial Regression: $y(x) = a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n$

1.order (linear): $y(x) = ax + b$

2.order: $y(x) = ax^2 + bx + c$

## 2.3.1. Interpolasyon

İnterpolasyon, uygulamalı matematiğin bir dalı olan sayısal analiz yöntemlerini kullanılarak farklı bir yerde ve değeri bilinmeyen bir noktadaki olası değeri bulmaya ya da tahmin etmeye yarayan yöntemlerin tümüne verilen genel isimdir. En basit tanımı ile "varolan sayısal değerleri kullanarak, boş noktalardaki değerlerin tahmin edilmesi" olarak açıklanmaktadır. Türkçede bazen kolaylık olsun diye "interpolasyon" sözcüğü yerine yalnızca "tahmin" de kullanılmaktadır.

İnterpolasyon genelde mühendislik ve deneylere ya da ölçümlere dayalı benzeri bilim dallarında, toplanan verilerin bir fonksiyon eğrisine uydurulması amacıyla kullanılmaktadır. İnterpolasyon kavramı, verilen bir fonksiyon sınıfından, grafiği verilen sınırlı sayıdaki veri noktasından geçecek şekilde bir y=p(x) fonksiyonu seçme işlemidir. Kestirilen çıkış değeri çok boyutlu enterpolasyon ile sağlanmaktadır. Sistemin öğrenmesi ise, kestirilmiş değer ile yapılan kullanım sonrası ortaya çıkan hataya dayanarak gerçekleştirilir. Makine öğrenmesinde, hataların minimize edilmesi, temel kuraldır. **Önceden kestirilmiş olan yaklaşık değerler ile karşılaştırma yapılarak düzeltmeler gerçekleştirilir ve öğrenme sağlanır**. Çeşitli elemanların karakteristiklerinin otomatik olarak çıkartılması ve zamanla değişimlerinin izlenmesi yapılmalıdır. İzlenen sistem zamanla değişen bir sistem ise, algoritma öğrenmeye devam ederek, oluşturduğu fonksiyondaki katsayılarda gerekli değişiklikleri yapacaktır.

Bir fonksiyonun *xi* (*i=0,1,...N*) noktalarında bilinen *fi* (*i=0,1,...N*) değerlerinden hareketle herhangi bir *xo* ara noktasında bilinmeyen *f* (*xo*) ara değerinin bulunması anlamına gelen interpolasyon teknikleri aynı zamanda sayısal türev, integrasyon, adi ve kısmi türevli diferansiyel denklemlerin sayısal çözümü gibi başka sayısal yöntemlerin de esasını teşkil eder.

İnterpolasyon yöntemleri genellikle mevcut (xi , fi) veri noktalarına eğri veya eğriler uydurulması yoluyla uygulanır. Bu amaçla kullanılan fonksiyonlara interpolasyon fonksiyonları denir.

İnterpolasyon fonksiyonu olarak çoğu zaman çeşitli mertebeden (genellikle:,1,2,3) polinomlar kullanılır. Ancak bazı hallerde logaritmik, eksponansiyel, hiperbolik gibi daha özel fonksiyonlar, periyodik veri değerleri için trigonometrik fonksiyonlar kullanılabilir.

Veri noktaları eşit aralıklı olarak dağılmışsa sonlu fark esaslı interpolasyon yöntemleri, eşit aralıklı değilse doğrusal interpolasyon, Lagrange interpolasyonu vb yöntemler daha uygun olur.

**İnterpolasyon polinomları:**

| | | | | |
|---|---|---|---|---|
| Xi | 3.2 | 2.7 | 1 | 4.8 |
| yi | 22 | 17.8 | 14.2 | 38.3 |

Bu noktalardan $y(x) = a_0 + a_1 x + a_2 x^2 + a3\, x^3$ şeklinde üçüncü dereceden bir polinom (kübik) geçirmek mümkündür. Herbir noktanın koordinatları bu denklemi sağlayacağı için

$$a_0 + (3.2)a_1 + (3.2)^2 a_2 + (3.2)^3 a_3 = 22.0$$
$$a_0 + (2.7)a_1 + (2.7)^2 a_2 + (2.7)^3 a_3 = 17.8$$
$$a_0 + (1.0)a_1 + (1.0)^2 a_2 + (1.0)^3 a_3 = 14.2$$
$$a_0 + (4.8)a_1 + (4.8)^2 a_2 + (4.8)^3 a_3 = 38.3$$

$$\begin{bmatrix} 1 & 3.2 & 10.24 & 32.768 \\ 1 & 2.7 & 7.29 & 19.683 \\ 1 & 1.0 & 1.00 & 1.000 \\ 1 & 4.8 & 23.04 & 110.592 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{Bmatrix} = \begin{Bmatrix} 22.0 \\ 17.8 \\ 14.2 \\ 38.3 \end{Bmatrix}$$

*Ax=b, linner denklem sisteminde,* $x=A^{-1}b$

$a_0=24.3499;\ a_1=16.1177\ a_2=6.4952\ a_3=0.5275$

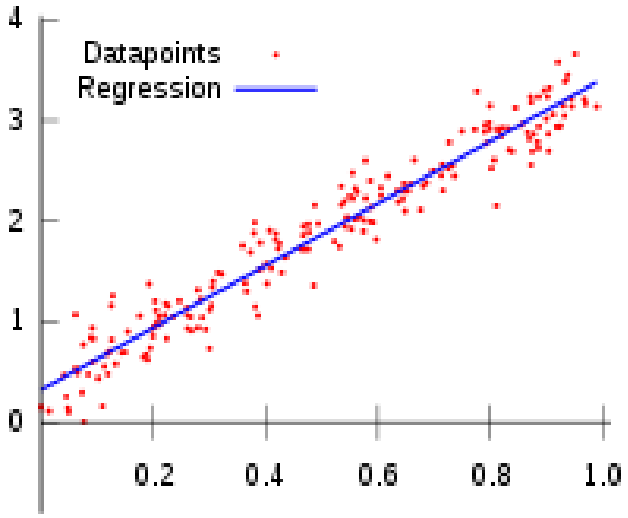
Yani interpolasyon fonksiyonu,

$y(x) = 24.3496 - 16.1176x + 6.4952x^2 - 0.5275x^3$

Buna göre örneğin *x=3.0* noktasındaki ara değer için *y=20.212* elde edilir.

Makine öğrenmesinde, verilen veri setinden öğrenen bir yapı elde edildi. Ardından test edilerek performansı artırılır. Performans artımı devamlılığı süreklidir.

## 2.3.2. Doğrusal (Lineer) Regresyon




Kırmızı noktalar ölçümle elde edilmiş veri noktalarını, mavi çizgi ise en küçük kareler yöntemi ile bulunmuş teorik bağlantıyı ifade eder.

Çoğu zaman veri tablosuna tam olarak uyan bir fonksiyon bulmak mümkün olmaz; veri tablosuna en iyi uyan fonksiyon belirlenmeye çalışılır. Bir veri tablosuna en iyi uyan fonksiyonu bulma sürecine **regresyon analizi** denir. Regresyon analizi yaparken en çok kullanılan yöntemlerden biri en küçük kareler yöntemidir.

Belli ölçümler sonucunda i = 1, 2, . . . , n için ($x_i$ , $y_i$) verileri elde edilmiş olsun. Burada, her bir $y_i$'nin $x_i$'ye bağlı olarak değiştiği varsayılmaktadır. Yapılan ölçümlerin doğası gereği, her i = 1, 2, . . . , n için $y_i = f(x_i)$ olacak biçimde bir fonksiyonun var olduğu, ölçümlerde yapılan hata nedeniyle bu eşitliklerin bazıları veya hepsinin sağlanmadığı kabul edilebilir. Bu düşünceyle, ölçülen $y_i$ değeri $y(x_i)$ için yaklaşık değer kabul edilerek bu yaklaşımdaki hatanın minimum olduğu y fonksiyonu belirlenmeye çalışılır. Bu amacı gerçekleştirmek için f fonksiyonunun bir takım parametrelere bağlı bir ifadesi bulunduğu varsayılıp eldeki veriler yardımıyla bu parametreler belirlenmeye çalışılır. Örneğin, y fonksiyonu $y = y(x) = mx + b$ ifadesinde olduğu gibi bir doğrusal fonksiyon veya $y = f(x) = ax^2 + bx + c$ ifadesinde olduğu gibi bir karesel fonksiyon olabilir ki bu durumda belirlenmesi gereken parametereler a , b , c , m dir.

En küçük kareler yönteminde aranan fonksiyon, ya da onun parametreleri, tüm farkların kareleri toplamı olan

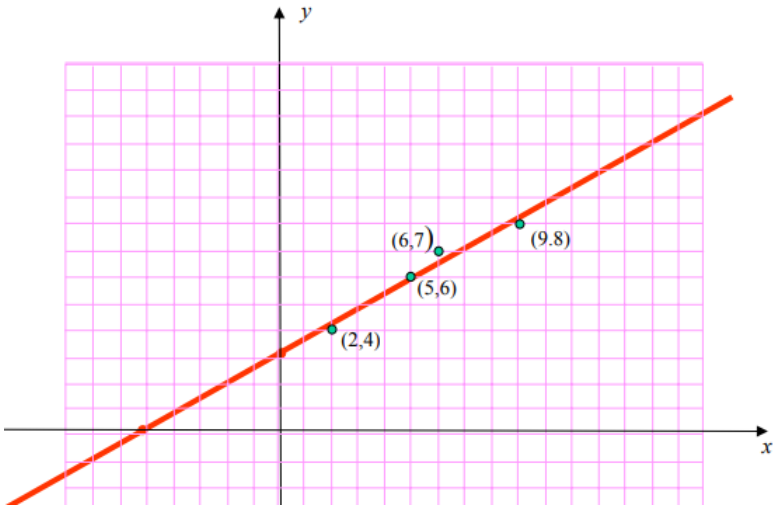
$$\sum_{i=1}^{n}\left(y_i-\hat{y}_i\right)^2=\sum_{i=1}^{n}\hat{\varepsilon}_i^2$$

ifadesini minimum yapacak şekilde belirlenir. Sözü edilen kareler toplamının minimum olması için her bir hatanın küçük olması gerekir. En Küçük Kareler, Kare Farklarının Toplamını (SSE) en aza indirir (Sum of the Squared Differences: SSE)

**Örnek:**

Bir makineden üretilen ürünlerin tablosu şu şekilde tutulmuştur,

2. ayda 4 adet
5. ayda 6 adet
6. ayda 7 adet
9.ayda 8 adet




f(x) = mx + b

$y_i = f(x_i)$

$$\sum_{i=1}^{n}(y_i - f(x_i))^2 = (y_1 - f(x_1))^2 + \cdots + (y_n - f(x_n))^2$$

2. ayda 4 adet , y1-f(x1)=4-2m-b
5. ayda 6 adet, y2-f(x2)=6-5m-b
6. ayda 7 adet, y3-f(x3)=7-6m+b
9.ayda 8 adet, y4-f(x4)=8-9m+b

$$F(m,b)=(4-2m-b)^2 + (6-5m-b)^2 + (7-6m-b)^2 + (8-9m-b)^2.$$

Eğimlerini bulabilmek için kısmi türevleri alınır ve sıfıra eşitlenir.

$$F_m(m,b)=2(4-2m-b)(-2)+2(6-5m-b)(-5)+2(7-6m-b)(-6)+2(8-9m-b)(-9)=0,$$

$$F_b(m,b)=2(4-2m-b)(-1)+2(6-5m-b)(-1)+2(7-6m-b)(-1)+2(8-9m-b)(-1)=0.$$

Sadeleştirmelerden sonra aşağıdaki denklemler elde edilir.
146m +22b=152
22m + 4b=25

Bu iki bilinmeyenli iki denklemin çözümünden m=0.58, b=3.06 bulunur.

f(x) = 0.58x + 3.06

m ve b nin doğrudan hesaplanmasını sağlayacak formüller farkların karelerinin toplamında elde edilir.

$$F(m,b)=\sum_{i=1}^{n}(y_i-mx_i-b)^2=(y_1-mx_1-b)^2+\cdots+(y_n-mx_n-b)^2$$

Yukarıdaki denklemde m ve b ye göre kısmi türevler alınıp sıfıra eşitlenirse aşağıdaki denklemler elde edilir.

$$F_m(m,b)=\sum_{i=1}^{n}2(y_i-mx_i-b)(-x_i)=-2\left(\sum_{i=1}^{n}x_i^2\right)m-2\left(\sum_{i=1}^{n}x_i\right)b+2\left(\sum_{i=1}^{n}x_iy_i\right)=0$$

$$F_b(m,b)=\sum_{i=1}^{n}2(y_i-mx_i-b)(-1)=-2\left(\sum_{i=1}^{n}x_i\right)m-2\left(\sum_{i=1}^{n}1\right)b+2\left(\sum_{i=1}^{n}y_i\right)=0$$

ya da

$$\begin{cases}\left(\sum_{i=1}^{n}x_i^2\right)m+\left(\sum_{i=1}^{n}x_i\right)b=\sum_{i=1}^{n}x_iy_i\\ \left(\sum_{i=1}^{n}x_i\right)m+\quad nb\quad=\sum_{i=1}^{n}y_i\end{cases}$$

denklem sistemi çözülerek bulunur.

Bu denklem sisteminin daima tek bir çözümü bulunduğuna dikkat ediniz.

$$m = \frac{n(\sum_{k=1}^{n} x_k y_k) - (\sum_{k=1}^{n} x_k)(\sum_{k=1}^{n} y_k)}{n(\sum_{k=1}^{n} x_k^2) - (\sum_{k=1}^{n} x_k)^2} \quad , \quad b = \frac{\sum_{k=1}^{n} y_k - m(\sum_{k=1}^{n} x_k)}{n}.$$

**Örnek:**

(0 , 6.4), (1 , 2.6), (2 , 0.5), (3 , 0.6) ve (4 , 0.3) veri noktalarına en iyi uyan y=mx + b, doğru denklemini bulunuz.

$$\sum_{k=1}^{5} x_k = 0+1+2+3+4 = 10 \quad , \quad \sum_{k=1}^{5} y_k = 6.4+2.6+0.5+0.6+0.3 = 10.4$$

$$\sum_{k=1}^{5} x_k y_k = 0\cdot(6.4)+1\cdot(2.6)+2\cdot(0.5)+3\cdot(0.6)+4\cdot(0.3) = 2.6+1+1.8+1.2 = 6.6$$

$$\sum_{k=1}^{5} x_k^{\,2} = 0+1+4+9+16 = 30 \quad , \quad (\sum_{k=1}^{5} x_k)(\sum_{k=1}^{5} y_k) = 10\cdot(10.4) = 104$$

$$m = \frac{n(\sum_{k=1}^{n} x_k y_k) - (\sum_{k=1}^{n} x_k)(\sum_{k=1}^{n} y_k)}{n(\sum_{k=1}^{n} x_k^2) - (\sum_{k=1}^{n} x_k)^2} = \frac{5\cdot(6.6)-104}{5\cdot 30-100} = \frac{33-104}{50} = \frac{-71}{50} = -1.42,$$

$$b = \frac{\sum_{k=1}^{n} y_k - m(\sum_{k=1}^{n} x_k)}{n} = \frac{10.4-(-1.42)\cdot 10}{5} = \frac{10.4+14.2}{5} = \frac{24.6}{5} = 4.92.$$

$y$=-1.42$x$+4.92

## 2.3.3. İkinci Mertebe En Küçük Kareler Yöntemi

$$E = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

f(x)=$a_2x^2 + a_1 x + a_0$

$$E = \sum_{i=1}^{N} (y_i - a_2 x_i^2 - a_1 b x_i - a_0)^2$$

$E_a = \sum_{i=1}^{N} 2(y_i - a_2 x_i^2 - a_1 x_i - a_0)(-x_i^2)$=0

$E_b = \sum_{i=1}^{N} 2(y_i - a_2 x_i^2 - a_1 x_i - a_0)(-x_i)$=0

$E_c = \sum_{i=1}^{N} 2(y_i - a_2 x_i^2 - a_1 x_i - a_0)(-1)$=0

$$\sum_{i=1}^{N} y_i x_i^2 = a_2 \sum_{i=1}^{N} x_i^4 + a_1 \sum_{i=1}^{N} x_i^3 + a_0 \sum_{i=1}^{N} x_i^2$$

$$\sum_{i=1}^{N} y_i x_i = a_2 \sum_{i=1}^{N} x_i^3 + a_1 \sum_{i=1}^{N} x_i^2 + a_0 \sum_{i=1}^{N} x_i$$

$$\sum_{i=1}^{N} y_i = a_2 \sum_{i=1}^{N} x_i^2 + a_1 \sum_{i=1}^{N} x_i + a_0 N$$

## 2.3.4. Genel Polinom Regresyon Modeli

Genel polinom regresyon modeli, en küçük kareler yöntemi kullanılarak geliştirilebilir. En küçük kareler yöntemi, polinomdan tahmin edilen değerler ile veri kümesinden beklenen değerler arasındaki farkı en aza indirmeyi amaçlamaktadır.

The coefficients of the polynomial regression model ($a_k$,$a_{k-1}$,⋯,$a_1$, $a_0$) may be determined by solving the following system of linear equations.

$$\begin{bmatrix} N & \sum_{i=1}^{N} x_i & \cdots & \sum_{i=1}^{N} x_i^k \\ \sum_{i=1}^{N} x_i & \sum_{i=1}^{N} x_i^2 & \cdots & \sum_{i=1}^{N} x_i^{k+1} \\ \vdots & \vdots & \vdots & \vdots \\ \sum_{i=1}^{N} x_i^k & \sum_{i=1}^{N} x_i^{k+1} & \cdots & \sum_{i=1}^{N} x_i^{2k} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_k \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} y_i \\ \sum_{i=1}^{N} x_i y_i \\ \vdots \\ \sum_{i=1}^{N} x_i^k y_i \end{bmatrix}$$

**Örnek:**
Aşağıdaki veri kümesine uygun 2. dereceden bir polinom eğrisinin nasıl geliştirileceğini gösterin.

| x | -3 | -2 | -1 | -0.2 | 1 | 3 |
|---|---|---|---|---|---|---|
| y | 0.9 | 0.8 | 0.4 | 0.2 | 0.1 | 0 |

Bu veri kümesinde N = 6 ve 2. dereceden bir polinom için k = 2 dir. En küçük kareler yönteminin uygulanması aşağıdaki doğrusal sistemi sağlar.

$$\begin{bmatrix} 6 & -2.2 & 24.04 \\ -2.2 & 24.04 & -8.008 \\ 24.04 & -8.008 & 180.0016 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 2.4 \\ -4.64 \\ 11.808 \end{bmatrix}$$

Sistemi çözmek için Cramer kuralını kullanarak, M matrisini alarak ve sütun vektörü b'yi i sütununa yerleştirerek Mi matrislerinin her birini üretiyoruz, örneğin M0, M1, M2,:

$$M_0 = \begin{bmatrix} 2.4 & -2.2 & 24.04 \\ -4.64 & 24.04 & -8.008 \\ 11.808 & -8.008 & 180.0016 \end{bmatrix}$$

$$M_1 = \begin{bmatrix} 6 & 2.44 & 24.04 \\ -2.2 & -4.64 & -8.008 \\ 24.04 & 11.808 & 180.0016 \end{bmatrix}$$

$$M_2 = \begin{bmatrix} 6 & 2.44 & 2.4 \\ -2.2 & -4.64 & -4.64 \\ 24.04 & 11.808 & 11.808 \end{bmatrix}$$

$$a_0 = \frac{det(M_0)}{det(M)} = \frac{2671.20}{11661.27} = 0.2291$$
$$a_1 = \frac{det(M_1)}{det(M)} = \frac{-1898.46}{11661.27} = -0.1628$$
$$a_2 = \frac{det(M_2)}{det(M)} = \frac{323.76}{11661.27} = 0.0278$$

$$y = 0.0278x^2 - 0.1628x + 0.2291$$

**Örnek:**

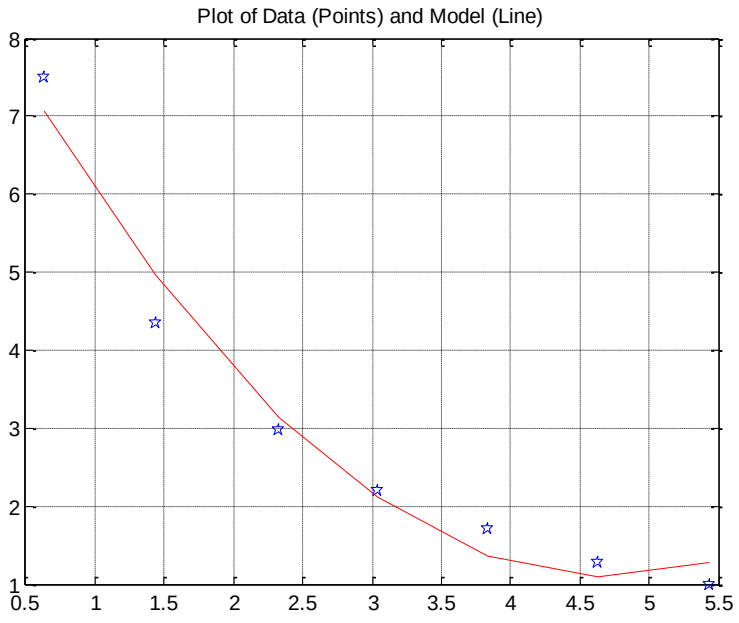
ikinci dereceden en küçük kareler ve Matlab ; p = polyfit(x,y,n), n=2

```
clear all
close all
x = [5.435   4.635   3.835   3.035   2.325   1.435   0.635];
y = [1.00     1.28     1.70     2.20     2.97     4.35     7.50  ];
p = polyfit(x, y, 2)                        % Quadratic Function Fit
v = polyval(p, x)                           % Evaluate
TSE = sum((v - y).^2)                 % Total Squared Error
figure(1)
plot(x, y, 'bp')
hold on
plot(x, v, '-r')
hold off
grid
title('Plot of Data (Points) and Model (Line)')
```

p = 0.3582 -3.3833 9.0748

v = 1.2664 1.0877 1.3674 2.1056 3.1447 4.9573 7.0708

TSE = 0.8110



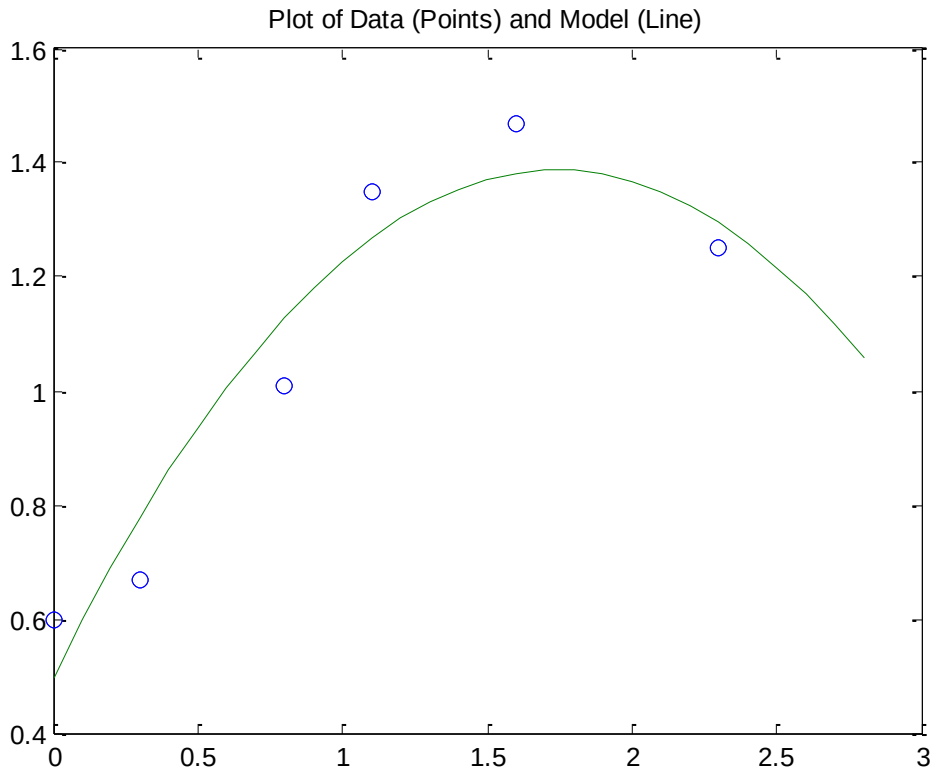
**Örnek:**

ikinci dereceden en küçük kareler ve Matlab ; p = polyfit(x,y,n), n=2

```
clear all
close all
t = [0 0.3 0.8 1.1 1.6 2.3];
y = [0.6 0.67 1.01 1.35 1.47 1.25];
figure, plot(t,y,'o')
title('Plot of y Versus t')
p = polyfit(t,y,2)
t2 = 0:0.1:2.8;
y2 = polyval(p,t2);
hold on
plot(t,y,'o',t2,y2)
title('Plot of Data (Points) and Model (Line)')
```

p = -0.2942 1.0231 0.4981

f(x) = $-0.2942x^2 + 1.0231x + 0.4981$



**Örnek:**

Uyum İşlevini Kullanarak Üstel Modellere Sığdırma

```
clear all
close all
x = (0:0.2:5)';
y = 2*exp(-0.2*x) + 0.1*randn(size(x));
f = fit(x,y,'exp1')
plot(f,x,y)
```

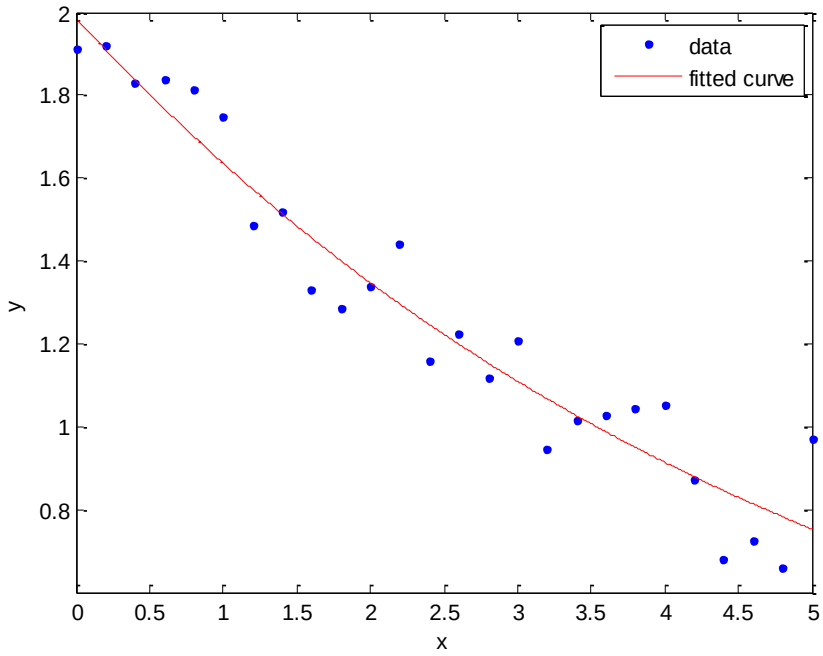
f = General model Exp1:

f(x) = a*exp(b*x)

Coefficients (with 95% confidence bounds):

a = 2.103 (1.999, 2.208)

b = -0.2222 (-0.2464, -0.1981)



**Doğrusal Regresyon**

Bir evin fiyatını tahmin etmek için birden fazla özellik vardır. Aşağıda özellikleri ve hedef değeri (fiyat) ile farklı evlerin tablosu bulunmaktadır.

| No. of Rooms (x0) | Size (Square feet) (x1) | Year Built (x2) | No. of Floors (x3) | Price (y) |
|---|---|---|---|---|
| 2 | 1434 | 2010 | 1 | 8500 |
| 3 | 1534 | 2019 | 2 | 9600 |
| 2 | 962 | 1996 | 3 | 25880 |
| ... | ... | ... | ... | ... |
| ... | ... | ... | ... | ... |

Şekil: Fiyat tahmini için evlerin özelliklerini gösteren Tablo.

Where,

$x_i$ = features of a house

y = target variable

Şekil: Özellikler ve hedef değişkenler.

Therefore, to calculate the hypothesis:

$$x = \begin{pmatrix} x0 \\ x1 \\ x2 \\ x3 \end{pmatrix} \quad \& \quad \theta = \begin{pmatrix} \theta 0 \\ \theta 1 \\ \theta 2 \\ \theta 3 \end{pmatrix}$$

$$h_\theta(x) = \theta_0 x0 + \theta_1 x1 + \quad \cdots \quad + \theta_n xn$$

# 2.4. Vektörler

**Vektör Normları**

Vektör normları bir vektörün büyüklüğünü ölçer. Temel olarak, belirli bir x değişkeninin boyutu, onun normu ||x|| ile temsil edilebilir ve norm, x ve y iki değişkeni arasındaki mesafeyi temsil eder ve ||x-y|| ile temsil edilir.

Vektör normunun genel denklemi:

$$||x|| = \left( \sum_{i=1}^{n} |x|^{p} \right)^{1/p}$$

Where,
p = class of p-norm

Bunlar, p-normlarının genel sınıflarıdır:

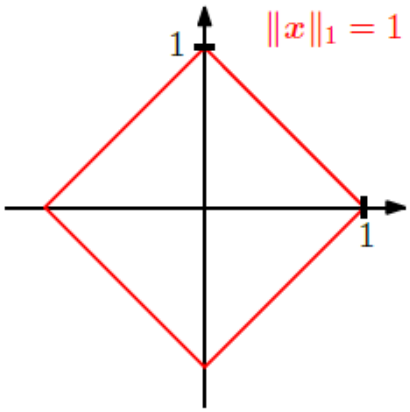
- L1 normu veya Manhattan Normu.
- L2 normu veya Öklid Normu.

Düzenlemede L1 ve L2 normları kullanılmaktadır.

**L1 norm or Manhattan Norm**

The L1 norm on **R^***n* is defined for x ∈ **R^**n as shown in figure:

$$||x||_1 = \left( \sum_{i=1}^{n} |x_i| \right)$$

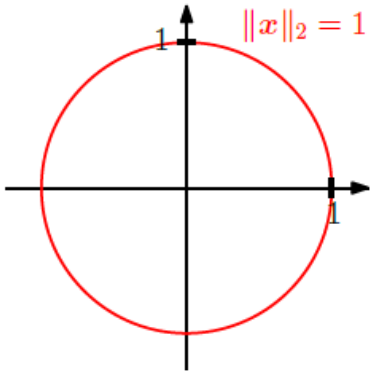



As shown in figure, the red lines symbolize the set of vectors for the L1 norm equation.

**L2 norm or Euclidean Norm**

The L2 norm of x∈**R**^*n* is defined as:

$$\|x\|_2 = \left(\sum_{i=1}^{n} |x_i|^2\right)^{1/2}$$




As shown in figure, the red lines symbolize the set of vectors for the L2 norm equation.

**Regularization in Machine Learning**

Regularization is a process of modifying the loss function to penalize specific values of the weight on learning. Regularization helps us avoid overfitting.

It is an excellent addition in machine learning for the operations below:

- To handle collinearity.
- To filter out noise from data.
- To prevent overfitting.
- To get good performance.

These are the standard regularization techniques:

- L1 Regularization (Lasso)
- L2 Regularization (Ridge)

Regularization is the application of the norm.

**L1 Regularization (Lasso)**

Lasso is a widespread regularization technique. Its formula is shown in figure:

$$||x||_1 = \left( \sum_{i=1}^{n} |x_i| \right)$$

**L2 Regularization (Ridge)**

The equation of L2 regularization (Ridge):

$$\lambda/2 \, ||x||^2 = \lambda/2 \sum_{i=1}^{n} x_i^2$$

Where, λ = Controls the tradeoff of complexity by adjusting the weight of the penalty term.

## Özellik Çıkarma ve Özellik Seçimi

Öznitelik çıkarma ve öznitelik seçiminin temel amacı, sınıflandırma verimliliğini artırmak için en uygun düşük boyutlu öznitelikler kümesini seçmektir. Bu terimler esasen boyutsallık probleminin lanetini ele alır. Özellik seçimi ve özellik çıkarımı matriste gerçekleştirilir.

## Feature Extraction

Öznitelik çıkarmada, bazı işlev eşlemeleri yoluyla mevcut özniteliklerden bir dizi öznitelik buluruz.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ . \\ . \\ x_N \end{bmatrix} \xrightarrow{f(\mathbf{x})} \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_K \end{bmatrix}$$

K<<N

**Feature Selection**

Özellik seçiminde, orijinal özelliklerin bir alt kümesini seçin.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ . \\ . \\ x_N \end{bmatrix} \rightarrow \mathbf{y} = \begin{bmatrix} x_{i_1} \\ x_{i_2} \\ . \\ . \\ x_{i_K} \end{bmatrix}$$

K<<N

Ana özellik çıkarma yöntemleri şunlardır:

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)

PCA kritik bir özellik çıkarma yöntemidir ve PCA kavramını anlamak için kovaryans matrisi, özdeğerler veya özvektörler kavramlarını bilmek hayati önem taşır.

## Covariance Matrix

The covariance matrix is an integral part of PCA derivation. The below concepts are important to compute a covariance matrix:

- Variance.
- Covariance.

**Variance**

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

or

$$= \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})$$

The limitation of variance is that it does not explore the relationship between variables.

## Covariance

Kovaryans, iki rastgele değişkenin ortak değişkenliğini ölçmek için kullanılır.

$$\text{Covariance}(x, y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$$

## Covariance Matrix

A covariance matrix is a squared matrix that gives the covariance between each pair of given random vector elements.

$$Cov\,(\sum) = \begin{bmatrix} cov(x_1, x_1) & cov(x_1, x_2) & \cdots & cov(x_1, x_m) \\ cov(x_2, x_1) & cov(x_2, x_2) & \cdots & cov(x_2, x_m) \\ \vdots & \vdots & \vdots & \vdots \\ cov(x_m, x_1) & cov(x_m, x_2) & \cdots & cov(x_m, x_m) \end{bmatrix}$$

The equation of the covariance matrix:

$$Cov\,(\sum) = \frac{1}{n}(X - \bar{X})(X - \bar{X})^T; where\ X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

**Orthogonality**

Two vectors v and w are called orthogonal if their dot product is zero.

**v.w = 0**

Example:

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \& \begin{bmatrix} 6 \\ -3 \end{bmatrix} \text{ are orthogonal in } R^2$$

Or to write like $x \perp y$

**Orthonormal Set**

If all vectors in the set are mutually orthogonal and all of the unit lengths, then it is called an Orthonormal set. An orthonormal set that forms a basis is called an orthonormal basis.

**Span**

Let **V** is a vector space and its elements are v*1*, v*2*, ….., v*n* ∈ V.

Any sum of these elements multiplied by the scalars representing the equation shown in figure 53, a set of all linear combinations is called the span.

$$a_1v_1 + a_2v_2 + \cdots\cdots\cdots + a_nv_n$$

Example:

$$v1 = \begin{bmatrix} 2 \\ 1 \\ -1 \end{bmatrix} \quad v2 = \begin{bmatrix} 0 \\ 2 \\ 2 \end{bmatrix} \quad v3 = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

Span (v1, v2, v3) = av1 + bv2 + cv3

$$= \begin{bmatrix} 2a \\ a \\ -a \end{bmatrix} + \begin{bmatrix} 0 \\ 2b \\ 2b \end{bmatrix} + \begin{bmatrix} -c \\ -c \\ -c \end{bmatrix} = \begin{bmatrix} 2a - c \\ a + 2b - c \\ -a + 2b - c \end{bmatrix}$$

**Basis**

A basis for a vector space is a sequence of vectors that form a set that is linearly independent and that spans the space [9].

Example:

The vector's sequence below is a basis:

$$\left[ \begin{bmatrix} 2 \\ 4 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 1 \end{bmatrix} \right]$$

It is linearly independent as given below:

$$c1\begin{pmatrix} 2 \\ 4 \end{pmatrix} + c2\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$2c1 + 1c2 = 0$$
$$4c1 + 1c2 = 0$$
$$\Longrightarrow c1 = c2 = 0$$

**Scalar Product of Two Vectors**

If **A** & **B** are vectors, their **Scalar Product** is defined as: **A•B ≡ AB cosθ**

- In terms of vector components & unit vectors **i,j,k** are along the **x,y,z** axes:
  **$A = A_x i + A_y j + A_z k$** **$B = B_x i + B_y j + B_z k$**
- **$A \bullet B = A_x B_x + A_y B_y + A_z B_z$**
- Using **i•i = j•j = k•k = 1,**
- **And i•j= i•k = j•k = 0** gives
- **Dot Product** clearly a **SCALAR**.

**Vector Addition.** Adding vectors is easy, just add each corresponding component! If $\vec{v}$ and $\vec{w}$ are complex vectors written explicitly as:

$$\vec{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \quad \text{and} \quad \vec{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

adding them gives:

$$\vec{v} + \vec{w} = \begin{bmatrix} v_1 + w_1 \\ v_2 + w_2 \end{bmatrix}$$

## 2.5. Matris

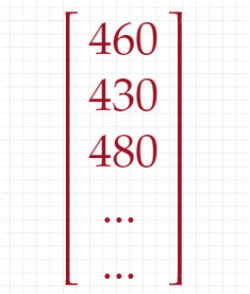
Bir matris, lineer cebirin önemli bir parçasıdır. m*n veri öğesini depolar ve biz onu doğrusal denklem sisteminin veya doğrusal eşlemelerin hesaplanması için kullanırız. Gerçek değerli öğelerin bir m*n demetidir.


Şekil 3: Matris gösterimi.

Satır ve sütun sayısına matrisin boyutu denir.

**Vektör**

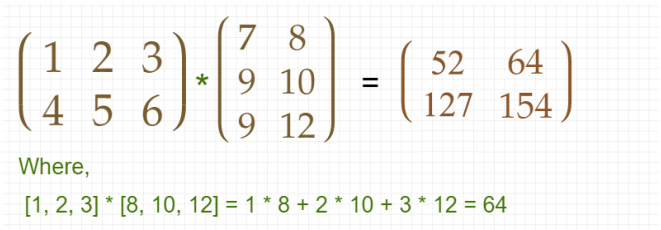
Lineer cebirde, bir vektör bir n*1 matrisidir. Sadece bir sütunu vardır.




Şekil: Vektör gösterimi.

**Matris Çarpımı**

Matris çarpımı, matrisin satırının başka bir matris sütunuyla çarpıldığı ve toplandığı satır ve sütunların bir nokta çarpımıdır.



**Matris çarpmaya örnekler**

**Satır vektör ve sütun vektör**

Aşağıdaki gibi iki matris verilsin; $\mathbf{A} = \begin{pmatrix} a & b & c \end{pmatrix}$, $\mathbf{B} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$,

Burada matris çarpma işlemi şöyle:

$$\mathbf{AB} = \begin{pmatrix} a & b & c \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = ax + by + cz, \quad \text{Benzer şekilde;} \quad \mathbf{BA} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \begin{pmatrix} a & b & c \end{pmatrix} = \begin{pmatrix} xa & xb & xc \\ ya & yb & yc \\ za & zb & zc \end{pmatrix}$$

**AB** ile **BA**nın çok farklı matrisler olduğuna dikkat edin. İlk matris $1 \times 1$ boyutlu matris iken, ikincisi $3 \times 3$ boyutlu matristir.

**Kare matris ve sütun vektörü**

Aşağıdaki gibi iki matris verilsin;

$$\mathbf{A} = \begin{pmatrix} a & b & c \\ p & q & r \\ u & v & w \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \quad \mathbf{AB} = \begin{pmatrix} a & b & c \\ p & q & r \\ u & v & w \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} ax + by + cz \\ px + qy + rz \\ ux + vy + wz \end{pmatrix}$$

Bu örnekte **BA** tanımlı değildir.

**Transpose Matrix**

For **A** ∈ R^*m*n* the matrix **B** ∈ R^*n*m* with **b***ij* = ***a***ij is called transpose of **A**. It is represented as **B** = **A**^*T*.

Example:

$$A = \begin{pmatrix} 1 & 4 \\ -2 & 3 \end{pmatrix} \qquad A^T = \begin{pmatrix} 1 & -2 \\ 4 & 3 \end{pmatrix}$$

**Inverse Matrix**

Consider a square matrix **A** ∈ R^*n*n*. Let matrix **B**∈R^*n*n* has the property that **AB** = In = **BA,** **B** is called the inverse of **A** and denoted by **A**^-*1*.

$$A = \begin{pmatrix} 1 & 2 & 1 \\ 4 & 4 & 5 \\ 6 & 7 & 7 \end{pmatrix} \qquad B = \begin{pmatrix} -7 & -7 & 6 \\ 2 & 1 & -1 \\ 4 & 5 & -4 \end{pmatrix}$$

$$A*B = \begin{pmatrix} -7+4+4 & -28+8+20 & -42+14+28 \\ -7+2+5 & -28+4+25 & -42+4+35 \\ 6-2-4 & 24-4-20 & 36-7-28 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

**Birim Matris**

- Birim matris bir matrise etki ettiği zaman o matrisin kendisi elde edilir. Birim matrisin tüm köşegen elemanlar 1 dir. Köşegen dışındaki elemanları 0 dır.
- Bir kare matrisin eşleniği ile çarpımı birim matristir. (Matrisin transposesi alınır, ardından kompleks ifadelerde a+ib yerine a-ib; a-ib yerine a+ib yazılarak eşdeniği elde edilir.)
- Bir kare matrisin quantum lojik kapısı olabilmesi için matrisin tranposesnin eşleniği ile çarpımı birim matris olmak zorundadır (Hermisyen).
- $IA = A$
- $I = \begin{pmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix}$
- Bir vektör birim matris ile çarpıldığında vektörü değiştirmez.
- $\begin{pmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$

**Orthogonal Matrix**

Bir A∈R^n*n kare matrisi, ancak ve ancak sütunları ortonormal (birim uzunluk) ise, ortogonal bir matristir:

$$AA^T = I = A^T A$$

Where, $A^{-1} = A^T$

Example:

$$A = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}, \quad A^T = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$

Consequently,

$$A * A^T = \begin{bmatrix} (-1)(-1) & (0)(0) \\ (0)(0) & (1)(1) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

**Diagonal Matrix**

A square matrix A∈R^*n*n* is a diagonal matrix where all the elements are zero except those on the main diagonal like:

Aij =0 for all i != j

Aij = 0 for some or all i = j

Example:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

**Transpose Matrix and Inverse Matrix in Normal Equation**

The normal equation method minimizes *J* by explicitly taking its derivatives concerning theta j and setting them to zero. We can directly find out the value of **θ** without using Gradient Descent.

$$\theta = (x^T x)^{-1} x^T y$$

Where

$x^T$ = Transpose of X

y = Prediction

Implementation by taking the data from the table above, “Table1” is shown in figure.

$$x = \begin{pmatrix} 2 & 1434 & 1 \\ 3 & 1534 & 2 \\ 2 & 962 & 3 \end{pmatrix} \qquad y = \begin{pmatrix} 8500 \\ 9600 \\ 258800 \end{pmatrix}$$

# Adjoints

Associated with any linear operator $A$ is its *adjoint* $A^\dagger$ which satisfies

$$\langle v|Aw\rangle = \langle A^\dagger v|w\rangle$$

In terms of matrices, $A^\dagger = (A^*)^T$

where $*$ denotes complex conjugation and $T$ denotes transposition.

$$\begin{bmatrix} 1+i & 1-i \\ -1 & 1 \end{bmatrix}^\dagger = \begin{bmatrix} 1-i & -1 \\ 1+i & 1 \end{bmatrix}$$

$A^*$ - complex conjugate of matrix $A$.

$$\text{if } A = \begin{bmatrix} 1 & 6i \\ 3i & 2+4i \end{bmatrix} \text{ then } A^* = \begin{bmatrix} 1 & -6i \\ -3i & 2-4i \end{bmatrix}$$

$A^T$ - transpose of matrix $A$.

$$\text{if } A = \begin{bmatrix} 1 & 6i \\ 3i & 2+4i \end{bmatrix} \text{ then } A^T = \begin{bmatrix} 1 & 3i \\ 6i & 2+4i \end{bmatrix}$$

$A^\dagger$ - Hermitian conjugate (adjoint) of matrix $A$.

Note $A^\dagger = (A^T)^*$

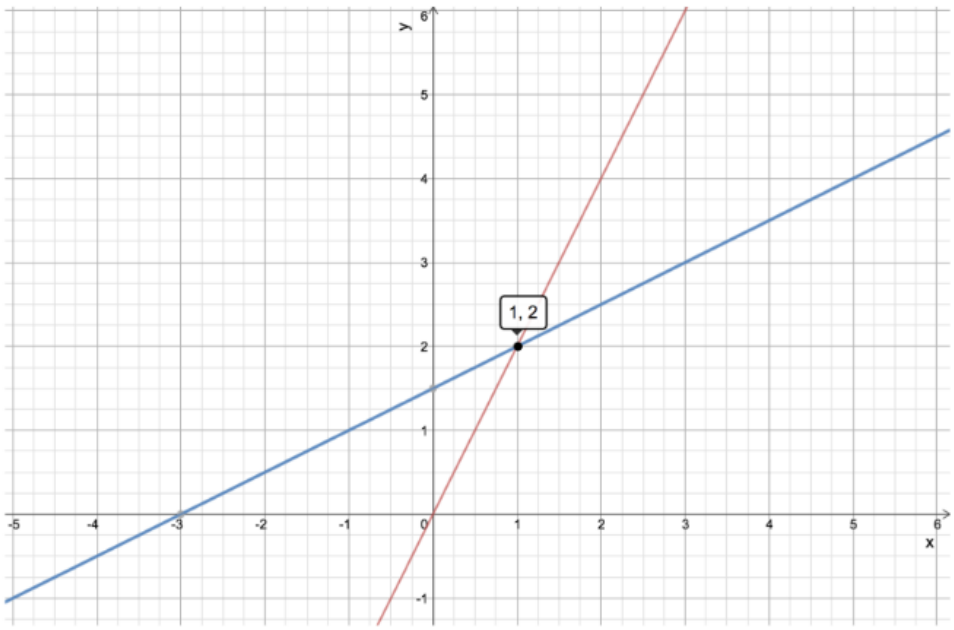
$$\text{if } A = \begin{bmatrix} 1 & 6i \\ 3i & 2+4i \end{bmatrix} \text{ then } A^\dagger = \begin{bmatrix} 1 & -3i \\ -6i & 2-4i \end{bmatrix}$$

Bir matrisin konjugesi, matrisin kompleks elemanlarının işaretinin tersinin alınmasıdır.
Bir matrisin transposesi, matrisin sütunları ile satırlarının yer değiştirmesidir.
Hermisyen konjugesi, matrisin transposesi ile konjugesinin alınmasıdır.

**Matris yorumu:**

Makine öğrenimi algoritması incelenip uygulandığında, algoritmalar hızlandırmak için matris-matris çarpımı (tüm matris-vektörü ve vektör-vektörü genelleştiren) kullanmak çok önemli ve ustalık gerektirir. Ancak güçlü lineer cebir altyapısı gerektiren vektörleştirilmiş ifadeleri yorumlanabilmelidir. Bu nedenle doğrusal cebirdeki bazı temel kavramlar özetlenebilmeli ve daha çok yorumlamaya odaklanılmalıdır; bu, Sinir Ağları (yalnızca bir matris-matris çarpımı zinciri) gibi bazı makine öğrenimi algoritmalarını anlamamıza çok yardımcı olabilir.

Example with N = 2 (2-dimension): $\begin{cases} 2x - y = 0 \\ -x + 2y = 3 \end{cases} = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \end{bmatrix}$




Bu, yukarıdaki iki denklemi sıklıkla yorumlama şeklimizdir: 2-B uzayda sadece iki çizgi ve çözüm, noktanın her iki çizgi üzerinde olmasıdır.

# 2.6. Özdeğerler ve Özvektörler

Özdeğerler, özvektörler ve özuzaylar, bir matrisin özellikleridir ve matris hakkında önemli bilgiler verir. Matrislerin çarpanlarına ayrılmasında kullanılabilirler. Uygulamalı matematik alanlarında olduğu kadar finans ve kuantum mekaniğinde de uygulama alanları vardır.

Genel olarak, bir vektör üzerine uygulanan matris o vektörün hem büyüklüğünü, hem de yönünü değiştirir. Buna rağmen, bir matris bazı belirli vektörler üzerine etkidiğinde onların sadece büyüklüğünü değiştirir, doğrultularını değiştirmez (yani belli bir yönelime sahip vektör, tam tersi yöne yönelebilir). Doğrultusu değişmeyen bu vektörler söz konusu matrisin özvektörleri olarak adlandırılır. Bir matris, bir özvektörü üzerine etkidiğinde onun büyüklüğünü bir çarpan kadar katlar. Bu çarpan pozitif ise vektörün yönü değişmeden kalır, negatif ise vektörün yönü tersine döner (dikkat edilirse her iki durumda da vektörün doğrultusu değişmez.). Bu çarpana, söz konusu özvektöre ilişkin özdeğer denir. Bir özuzay, aynı özdeğere sahip tüm özvektörlerin oluşturduğu kümedir. Bu kavramlar matrisler, vektörler ve doğrusal dönüşümler hakkında bir anlayışa sahip olunmaksızın biçimsel olarak tanımlanamaz.

Lineer zamanla değişmeyen ve $\dot{x} = f(x)$ bağıntısıyla ifade edilen bir sistemin denge noktaları $f(x)$=0 eşitliğinin reel kökleridir.

$p(\lambda) = det(A - \lambda I) = 0$ denklemini sağlayan $\lambda$ değerlerine $A$ matrisinin **özdeğer**leri denir. Bu $\lambda$ değerini $Ax = \lambda x$ veya $(A - \lambda I)x = 0$ da yerine koyarak elde edilen sıfırdan farklı çözümler, $A$ matrisinin bu özdeğerine karşı gelen **özvektörleridir**.

**A** bir *n x n* boyutlu kare matris olsun. Eğer $\lambda$ bir skaler ve **x** vektörü de sıfır olmayan, $\mathbf{x} \neq \mathbf{0}$ , bir sütun vektörü olmak üzere, $\mathbf{Ax} = \lambda\mathbf{x}$ eşitliği sağlanıyorsa **x** vektörü, **A** matrisinin özvektörü, $\lambda$ skaleri de **A** matrisinin özdeğeridir. Aynı zamanda **x**, $\lambda$ özdeğerine karşılık gelen özvektördür. Bir skaler olan $\lambda$, *nxn* boyutlu **A** matrisi için $\mathbf{Ax} = \lambda\mathbf{x}$ denkleminde **x**'in sonsuz çözümü olduğu durumda bir özdeğer tanımlar.

Özdeğerler, bir matrisin orijinal yapısını görmek için kullanılan alternatif bir yoldur. Bazı vektörler bir **A** matrisi ile çarpıldıkları zaman yön değiştirir, bazıları ise değiştirmezler. Bazı özel **x** vektörleri, A**x** vektörü ile aynı yönde kalmaktadır. İşte bu vektörlere "özvektörler" denir. Bir özvektörün **A** matrisi ile çarpımı olan A**x** vektörü, orijinal **x** vektörünün $\lambda \epsilon R$ olmak üzere $\lambda$ katıdır.

Özdeğerler ve özvektörler, diferensiyel denklemler içeren denklem sistemlerinin çözümlerinde, sınır-değer problemlerinde ortaya çıkabilir. Bu tür denklemlere, kuantum mekaniğinde elektriksel bir potansiyel içinde bulunan bir parçacığın enerjisini hesaplarken, elastik çarpışma problemlerinde, akışkanlar mekaniğinde, titreşim yapan cisimlerin

hareketlerinde sıkça karşılaşılır. Titreşim yapan bir sistemin doğal frekansı ile dışarıdan uygulanan sürücü kuvvetin frekansı birbirine eşit veya yakın olması sistemin kararlılığı açısından veya malzemelerin elastik özelliklerinin incelendiği durumlarda, şekil bozukluklarının başladığı noktaların belirlenmesinde özfonksiyonların alacağı özdeğerler önemli olmaktadır.

**A matrisinin özdeğerleri olan $\lambda_1$ ve $\lambda_2$ değerlerinin bulunması:**
Av=λv
A, n x n kare matrisdir. v, bir sütun matrisdir, özvektörlerdir. λ, özdeğerdir. v ve λ değerleri, A matrisinin özdeğerleri ve özvektörleridir.

$(A - \lambda I)v = 0$

$|A - \lambda I| = 0$ , bu ifadeyi 0 yapan λ'lar özdeğerlerdir.

**Özvektörlerin bulunması:**
Bulunan herbir λ değeri için

$(A - \lambda_i I)v = 0$ , i=1,2, ...

denklemi yardımıyla bulanan v'değerleri ise özvektörlerdir. Bu matrisde herbir satır diğer satırların mutlaka katı olmak zorundadır. Bu denklemler birbirinin katı olacağı için çözüm için bir denklemi almamız yeterli olacaktır. Her zaman ikinci vektör birimi olan $v_2$'ye bir değer atarak $v_1$ bulunur.

İki farklı özdeğer elde edersek, onlara tekabül eden özvektörler birbirinden bağımsızdır. Bu illa ki birbirlerine dikgen anlamına gelmez, sadece aynı çizgi üzerinde değiller demektir.

Kritik noktalar: $\dot{x} = 0 \; ve \; \dot{y} = 0$

λ değerlerine bakılarak faz düzlem çizgileri çizilebilmekte ve denge noktaları bulunabilmektedir.

Şimdi genel çözümü yazalım, bu çözüm iki çözümün üst üste konmuş hali olacaktır.

$\mathrm{x(t)} = \; c_1 e^{\lambda_1 t} v_1 + \; c_2 e^{\lambda_2 t} v_2$

$c_{1,}\, c_2$ herhangi birer sabittir. Bu ifade genel çözüm "özçözümlerin" lineer bir kombinasyonudur.

Özdeğerlerin ve Özvektörlerin özellikleri:

Özdeğerlerin çarpımı A matrisinin determinatını verir. Det(A)= $\lambda_1 \lambda_2$
Özdeğerler, matrisin köşegenlerindeki elemanların toplamına eşittir.
Trace(A)= $\lambda_1 + \lambda_2$
Bir matrisi köşegen hale getirmek için özdeğerler ve özvektörler kullanılır.
Bir matrisin çok büyük üssünü hesaplamak istendiğin de özdeğerler ve özvektörler kullanılır.
Özdeğerler ve özvektörler diferansiyel denklemleri çözmede kullanılır.

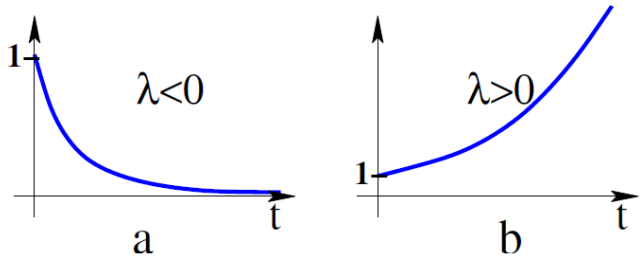
Özdeğerler:
$\lambda_1$, $\lambda_2$ gerçek ve aynı işarete sahip (+ veya -)
$\lambda_1$, $\lambda_2$ gerçek ve zıt işaretli
$\lambda_1$, $\lambda_2$ sıfır olmayan gerçek parçalara sahip kompleks eşleniklerdir
$\lambda_1$, $\lambda_2$, gerçek parçaları 0'a eşit olan kompleks eşleniklerdir
$\lambda_1 = \lambda_2$, kökler gerçek ve birbirine eşit

**Özdeğerlerden kararlı ya da kararsız durumların belirlenmesi:**
$\lambda_1$ ve $\lambda_2$ özdeğerlerine bağlı olarak birkaç farklı denge türü ortaya çıkmaktadır. Bilindiği gibi $\lambda_1$ ve $\lambda_2$, genel ikinci dereceden bir denklem olan $|A - \lambda I| = 0$ , karakteristik denklemin kökleridir. Bu nedenle kökler gerçek veya karmaşık sayılar olabilir ve aşağıdaki denge kararlılığı durumları ortaya çıkarır.

Özdeğerler, çözüm denkleminde $e^{\lambda_1 t}$ ve $e^{\lambda_2 t}$ katkı verirler. Bu durumda üssel negatif özdeğeri olan fonksiyon $t \to \infty$ olurken çözüm eğrisine yakınsar. üssel pozitif özdeğeri olan fonksiyon $t \to \infty$ olurken çözüm eğrisinden ıraksar.

$e^{\lambda t}$, fonksiyonunun iki ana davranış türü vardır.

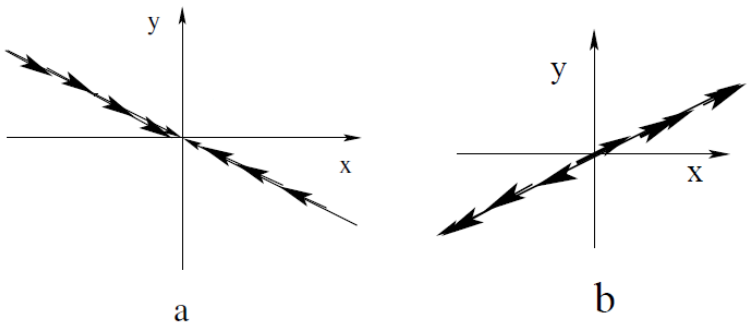
- $\lambda<0$ iken t artığında $e^{\lambda t}$ kararlı bir noktaya yaklaşır.
- $\lambda>0$ iken t artığında $e^{\lambda t}$ kararsız sonsuza gider.




Şekil 4.5.

Bu durumda,
$\lambda<0$ iken, aşağıdaki denklem kararlı bir durum belirleyecektir.
$\lambda>0$ iken, aşağıdaki denklem kararsız bir durum belirleyecektir.

$$\begin{pmatrix} x \\ y \end{pmatrix} = C \begin{pmatrix} V_1 \\ V_2 \end{pmatrix} e^{\lambda t}$$




Şekil 4.6.

Kararlı durumlar:

1) Kararlı düğüm noktası: $\lambda_1<0$ ve $\lambda_2<0$; kökler reel.
2) Kararlı spiral: $\lambda_{1,2} = \alpha \pm i\beta$; $\alpha<0$;

Kararsız durumlar:

1) Kararsız düğüm noktası: $\lambda_1>0$ ve $\lambda_2>0$; kökler reel
2) Kararsız spiral: $\lambda_{1,2} = \alpha \pm i\beta$; $\alpha>0$
3) Dairesel spiral: $\lambda_{1,2} = \pm i\beta$; $\alpha=0$;
4) Eyer ya da denge noktası: $\lambda_1>0$, $\lambda_2<0$ ya da $\lambda_1<0$, $\lambda_2>0$; kökler reel

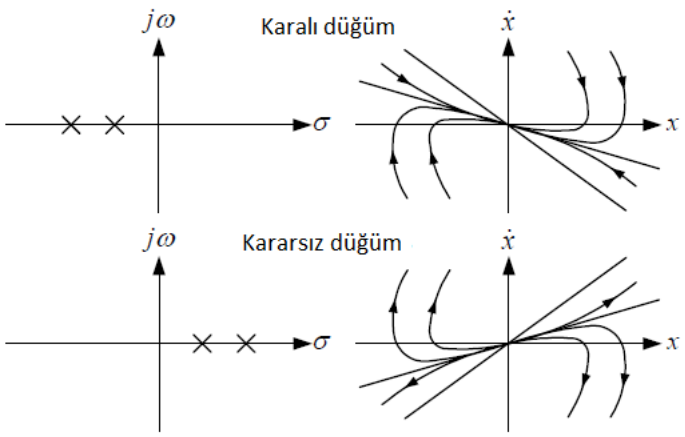
**Özvektörlerden Eyer, Düğüm çizgileri belirlenir.**

**λ1, λ2 gerçek ve aynı işarete sahip (+ veya -) özdeğerler ise:**
Düğümleri kontrol eden karekökün mevcudiyetir. $\tau^2 - 4\Delta > 0$ ise iki reel kök mevcuttur. Eğer bir karekök var ise, bir düğüm var demektir. Çeken düğüm noktası ile iten düğüm noktasının durumları topolojik olarak aynıdır, tek fark okların terse çevirilmiş olmasıdır. Bu durumda her iki λ reel ve işaretleri de aynı olur.

$\tau = \lambda_1 + \lambda_2$
$\Delta = \lambda_1 \lambda_2$

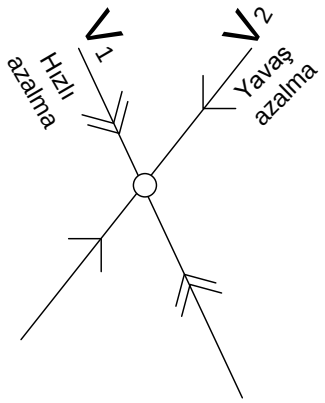
Lamdaların her ikisi de negatif işaretli ise çeken bir noktayı karakterize eden bir özellik olur. Bu durumda $\Delta>0$, $\tau<0$ dır. Kararlı düğüm noktası vardır.

Lamdaların her ikisi de pozitif işaretli ise iten bir noktayı karakterize eden bir özellik olur. Bu durumda $\Delta>0$, $\tau>0$ dır. Kararsız düğüm noktası vardır.




Şekil 4.7.

λ1<λ2<0 ise, ve $v_1, v_2$ hala bağımsız.




Şekil 4.8.

Yine iki özvektör var ama bu sefer gidiş stabil noktaya doğru. Dikkat gösterilen çizgiler $v_1, v_2$ vektörlerinin kendisi değil, onların her türlü tek sayısal çarpımdan gelen katlarından oluşan bir uzay, yani "özuzay". λ1 daha negatif demiştik, o zaman $v_1$'in azalması daha hızlı (fast decay) olacaktır, $v_2$ daha yavaş (slow decay) olacaktır.

**Eigenvectors** are the vectors which when multiplied by a matrix (linear combination or transformation) results in another **vector having same direction** but scaled (hence scaler multiple) in forward or reverse direction by a **magnitude of the scaler multiple** which can be termed as **Eigenvalue**. In simpler words, eigenvalue can be seen as the **scaling factor** for eigenvectors. Here is the formula for what is called **eigenequation**.

$Ax=\lambda xAx=\lambda x$

In the above equation, the **matrix A** acts on the **vector x** and the outcome is another **vector Ax** having **same direction as original vector x** but scaled / shrunk in forward or reverse direction by a magnitude of scaler multiple, λλ. The **vector x** is called as **eigenvector of A** and λλ is called its **eigenvalue**. Let's understand what pictorially what happens when a matrix A acts on a vector x. Note that the new vector Ax has different direction than vector x.

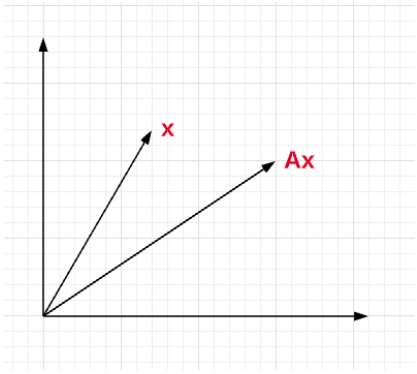



**Fig. Matrix A acts on x resulting in another vector Ax**

When the **matrix multiplication with vector** results in **another vector in the same / opposite direction** but scaled in forward / reverse direction by a magnitude of **scaler multiple** or **eigenvalue** (λλ), then the vector is called as **eigenvector** of that matrix. Here is the diagram representing the eigenvector x of matrix A because the vector Ax is in the same / opposite direction of x.

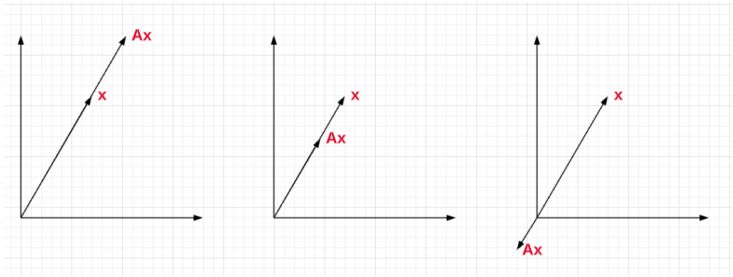



**Fig. x is eigenvector of A**

Here is further information on the value of eigenvalues:

- $\lambda \in \mathbf{R}$, $\lambda > 0$: $v$ and $Av$ point in same direction
- $\lambda \in \mathbf{R}$, $\lambda < 0$: $v$ and $Av$ point in opposite directions
- $\lambda \in \mathbf{R}$, $|\lambda| < 1$: $Av$ smaller than $v$
- $\lambda \in \mathbf{R}$, $|\lambda| > 1$: $Av$ larger than $v$

Many disciplines traditionally represent vectors as matrices with a **single column** rather than as matrices with a single row. For that reason, the word "**eigenvector**" in the context of matrices almost always refers to a **right eigenvector**, namely a ***column* vector**.
How to Calculate Eigenvector & Eigenvalue?
Here are the steps to calculate the eigenvalue and eigenvector of any matrix A.

- Calculate one or more eigenvalues depending upon number of dimensions of square matrix
- Determine the corresponding eigenvectors

For calculating the eigenvalues, one needs to solve the following equation:
Ax=λxAx–λx=0(A–λI)x=0Ax=λxAx–λx=0(A–λI)x=0

For non-zero eigenvector, the eigenvalues can be determined by solving the following equation:
A–λI=0A–λI=0

In above equation, I is identity matrix and λλ is eigenvalue. Once eigenvalues are determined, eigenvectors are determined by solving the equation (A–λI)x=0(A–λI)x=0
When to use Eigenvalues & Eigenvectors?

Whenever there is a complex system having large number of dimensions with a large number of data, eigenvectors and eigenvalues concepts help in transforming the data in a set of most important dimensions (principal components). This will result in processing the data in a faster manner.

Here are some learnings from this post:

- Eigenvector is a vector which when multiplied with a transformation matrix results in another vector multiplied with a scaler multiple having same direction as Eigenvector. This scaler multiple is known as Eigenvalue
- Eigenvectors and Eigenvalues are key concepts used in feature extraction techniques such as Principal Component analysis which is an algorithm used to reducing dimensionality while training a machine learning model.

- Eigenvalues and Eigenvector concepts are used in several fields including machine learning, quantum computing, communication system design, construction designs, electrical and mechanical engineering etc.

The definition of Eigenvalues:
Let m be an n*n matrix. A scalar λ is called the eigenvalue of m if there exists a non-zero vector x in **R**^n such that mx = λx.
and for Eigenvector:
The vector x is called an eigenvector corresponding to λ.

**Calculation of Eigenvalues & Eigenvectors**
Let m be **n*n** matrix with eigenvalues ***λ*** *and* corresponding eigenvector x. So, **mx = *λx***. This equation can be written as below:
**mx — λx = 0**
So, the equation:

$$(m - \lambda I_n)x = 0$$

**Example**:
Calculate eigenvalues and eigenvectors of given matrix ***m***:

$$m = \begin{bmatrix} -4 & -6 \\ 3 & 5 \end{bmatrix}$$

Solution:
Here, the size of the matrix is 2. So:

$$m - \lambda I_2 = \begin{bmatrix} -4 & -6 \\ 3 & 5 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$m - \lambda I_2 = \begin{bmatrix} -4-\lambda & -6 \\ 3 & 5-\lambda \end{bmatrix}$$

$$|m - \lambda I_2| = (-4 - \lambda)(5 - \lambda) + 18 = \lambda^2 - \lambda - 2$$

$$\lambda^2 - \lambda - 2 = 0$$

$$(\lambda - 2)(\lambda + 1) = 0$$

$$\lambda = 2 \text{ or } -1$$

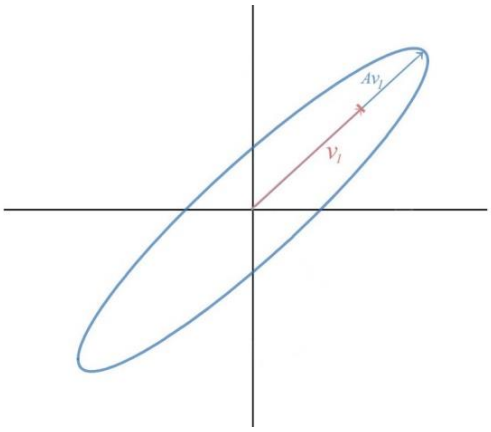
Here, the Eigenvalues of **m** are **2** and **-1**.
There are multiple eigenvectors available to each eigenvalue.

Özdeğer ve özvektör, lineer cebirdeki muhtemelen en önemli kavramlardan biridir. Av = λv gibi basit bir denklemin bu kadar anlamlı olmasını kim bekleyebilir? Bir matrisin özdeğerini ve özvektörlerini bularak makine öğreniminden kuantum hesaplamaya kadar birçok problem çözülebilir. Bu yazıda neden bu kadar önemli olduğunu ve nasıl uygulayabileceğimizi keşfedeceğiz.

Tanım olarak, skaler λ ve vektör v, eğer A'nın özdeğeri ve özvektörüdür:

$$Av = \lambda v$$

Görsel olarak, Av özvektör v ile aynı doğru üzerinde yer alır.




*Ax genellikle λx'e eşit değildir. Yalnızca bazı istisnai vektörler koşulu sağlar. İşte bazı özvektör örnekleri.*

$$\begin{bmatrix} 1 & -3 & 3 \\ 3 & -5 & 3 \\ 6 & -6 & 4 \end{bmatrix} \begin{bmatrix} 1/2 \\ 1/2 \\ 1 \end{bmatrix} = 4 \begin{bmatrix} 1/2 \\ 1/2 \\ 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & -3 & 3 \\ 3 & -5 & 3 \\ 6 & -6 & 4 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = -2 \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

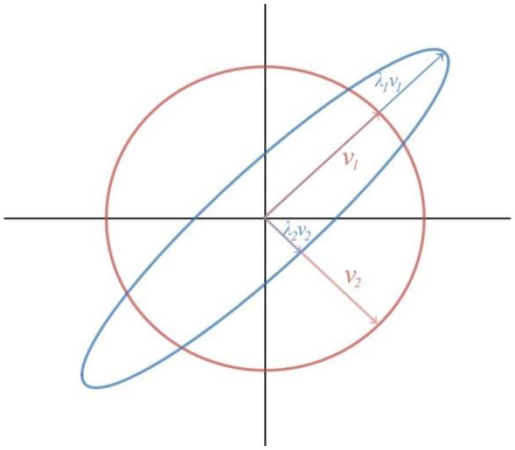
$A$ — *eigenvalue* — *eigenvector*

$$\begin{bmatrix} 1 & -3 & 3 \\ 3 & -5 & 3 \\ 6 & -6 & 4 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} = -2 \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

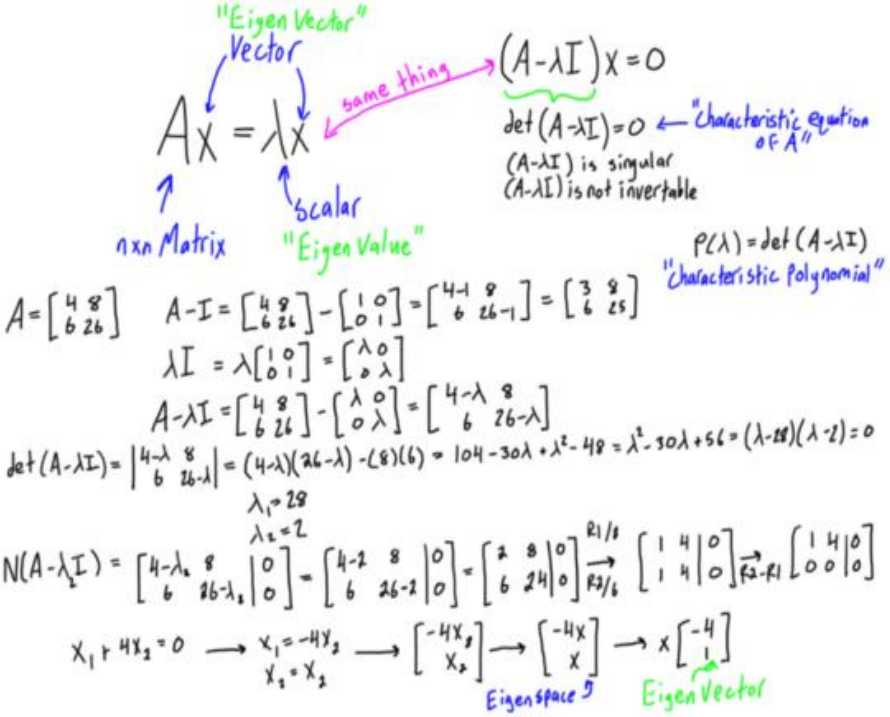
Özdeğer birden büyükse, karşılık gelen $Av_i$ genişleyecektir. Birden küçükse küçülür.

$Ax = \lambda x$ denkleminden A matrisi x, özvektörünü λ, özdeğeri kadar büyültür ya da küçültür. Özdeğer, λ değeri pozitif ise x ve λx aynı yönde, aksi halde ters yöndedir.

- λ<0 iken t artığında $e^{\lambda t}$ kararlı bir noktaya yaklaşır.
- λ>0 iken t artığında $e^{\lambda t}$ kararsız sonsuza gider.
- Bir sıfırdan küçük diğeri sıfırdan büyükse denge durumu söz konusudur.




- Özdeğerler (veya karakteristik değerler) ve özvektörleri (karakteristik özvektörler), fiziksel bir sistemin sahip olabileceği özel değerlerde nasıl davrandıklarını belirlemek için önemlidir.
- Bu değerler sisteme ait özel bir enerji, özel bir frekans değeri, dalgaların girişimi veya kuvvet dengesinin sağlandığı bir duruma ait olabilir.
- Özvektörler ise, fiziksel sistemin sahip olduğu özdeğerlerdeki (örneğin bir dalga) fonksiyonları olabilir. Özdeğerler ve özvektörler, diferensiyel denklemler içeren denklem sistemlerinin çözümlerinde, sınır-değer problemlerinde ortaya çıkabilir.
- Bu tür denklemlere, kuantum mekaniğinde elektriksel bir potansiyel içinde bulunan bir parçacığın enerjisini hesaplarken, elastik çarpışma problemlerinde, akışkanlar mekaniğinde, titreşim yapan cisimlerin hareketlerinde sıkça karşılaşılır.
- Titreşim yapan bir sistemin doğal frekansı ile dışarıdan uygulanan sürücü kuvvetin frekansı birbirineeşit veya yakın olması sistemin kararlılığı açısından veya malzemelerin elastik özelliklerinin incelendiği durumlarda, şekil bozukluklarının başladığı noktaların belirlenmesinde özfonksiyonların alacağı özdeğerler önemli olmaktadır.




**Uygulama**

Ancak ayrıntılara girmeden önce duralım ve böylesine soyut bir kavramın güzelliğini değerlendirelim. Spesifik olarak, birçok problem, özdeğerlerden ve özvektörlerden türetilen çözümlerle lineer cebir ile modellenebilir.

Birçok sistemde, durumlarını bir vektörde, değişim oranları mevcut durumlara doğrusal olarak bağlı olarak ifade edebiliriz. Genel denklem

$$\frac{du}{dt} = A\,u \qquad \begin{bmatrix} \frac{du_1}{dt} \\ \vdots \\ \frac{du_n}{dt} \end{bmatrix} = A \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix}$$

burada n özellikten oluşur. Öyleyse yukarıdaki denklemi sağlayan u(t) için bir çözüm tahmin edelim. Üstel bir fonksiyonun türevi kendisine eşit olduğundan, tahmine üstel bir t fonksiyonu ile başlayabilir ve onu bir x vektörü ile çarpabiliriz. Çıktı da bir vektör olacak ve sonra x ve λ değerini hesaplayacağız.

$$u(t) = e^{\lambda t}x \quad \text{— our guess}$$

Solve $\frac{du}{dt} = A\,u$

$$\frac{du}{dt} = \lambda e^{\lambda t}x \qquad (\text{put } u(t) = e^{\lambda t}x)$$
$$= \lambda u$$

i.e. $A\,u = \lambda u$

$$A e^{\lambda t}x = \lambda e^{\lambda t}x$$
$$A\,x = \lambda\,x$$ — $\lambda$ & $x$ is the eigenvalue & eigenvector of $x$

Yukarıdaki hesaplamadan, x ve λ A'nın özvektörleri ve özdeğerleri ise, tahminimiz du/dt = Au'yu sağlayacaktır.

$$u(t) = e^{\lambda t}x$$

eigenvalue of $A$ — eigenvector of $A$

$$\frac{du}{dt} = A\,u$$

*Sistemlerde, problem özdeğerler ve özvektörler tarafından yeniden ifade edilebilir ve çözülebilir. Şimdi tam çözümünü bulalım. Birinci mertebeden türev denklemimiz lineer bir fonksiyondur.*

$$\frac{du}{dt} = A\,u \quad \text{is a linear function, i.e. } f(u+v) = f(u) + f(v)$$

$$\frac{d(u+v)}{dt} = \frac{du}{dt} + \frac{dv}{dt}$$

Doğrusal fonksiyonlar için tam çözüm, belirli çözümlerin doğrusal birleşimidir. Çözümler u ve v ise, $C_1u + C_2v$ de çözümdür. Özdeğerleri λ = 4, -2 ve -2 olan önceki örneğimizden, tam çözüm şöyle olacaktır:

$$u(t) = C_1 e^{4t}\begin{bmatrix}1/2\\1/2\\1\end{bmatrix} + C_2 e^{-2t}\begin{bmatrix}1\\1\\0\end{bmatrix} + C_3 e^{-2t}\begin{bmatrix}-1\\0\\1\end{bmatrix}$$

eigenvalue — eigenvector — constant

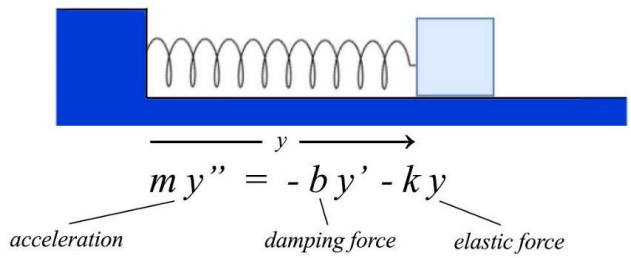
Bir sistem kararlı bir duruma ulaşacaksa, o zaman tüm özdeğerler negatif olmalıdır. Aksi takdirde, negatif olmayan terim sürekli olarak değişecektir. t=0 anında u(0) başlangıç durumunu ölçebilir, $[u_{01}, u_{02}, u_{03}]^T$ diyebilir ve $C_1$, $C_2$ ve $C_3$ sabitini çözebiliriz.

$$u(0) = \begin{bmatrix}u_{01}\\u_{02}\\u_{03}\end{bmatrix} = C_1\begin{bmatrix}1/2\\1/2\\1\end{bmatrix} + C_2\begin{bmatrix}1\\1\\0\end{bmatrix} + C_3\begin{bmatrix}-1\\0\\1\end{bmatrix}$$

Harmonik bir osilatör örneğini seçelim. Harmonik osilatör ve onun yakın kuzeni olan kuantum harmonik osilatör parçacık fiziği, kuantum mekaniği, sicim teorisi veya genel olarak fizik çalışmalarında temeldir. Ünlü F=ma denklemiyle başlayalım ve ikinci dereceden türevi çözmek için özdeğerleri ve özvektörleri kullanalım. Kütle birimini seçme özgürlüğüne sahip olduğumuz için, fizikçiler genellikle tartışmayı basitleştirmek için m=1'i ayarlarlar;

$$F = ma$$
$$= my''$$
$$F = y''$$

Harmonik osilatör problemini aşağıda sağ alttaki matris formuna yeniden formüle ediyoruz.




$$y' = \frac{d}{dt} y$$
$$y'' = \frac{d}{dt} y' = -ky - by'$$

$$\frac{d}{dt}\begin{bmatrix} y \\ y' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -k & -b \end{bmatrix}\begin{bmatrix} y \\ y' \end{bmatrix}$$
$$\frac{d}{dt} u = A u$$

Sönümleme (Damping) harmonik osilatör

Bu, son örneğimizle aynı forma sahiptir ve bu nedenle, tam çözümü oluşturmak için A'nın özdeğerlerini ve özvektörlerini kullanabiliriz.

Ünlü zamandan bağımsız Schrödinger denklemi, özdeğerler ve özvektörlerle ifade edilir. Gözlenen tüm özellikler, kuantum mekaniğindeki özdeğerlerle modellenir. Bunlar, makine öğrenimi de dahil olmak üzere birçok başka örnektir. Hesaplanan en büyük özvektörlerden biri, Google aramanın temel parçası olan Google PageRank'tir.

$$\hat{H}|\psi\rangle = E|\psi\rangle$$
eigenvalues eigenvectors

Schrödinger equation

$$\begin{pmatrix} (\sigma_z)_{11} & (\sigma_z)_{12} \\ (\sigma_z)_{21} & (\sigma_z)_{22} \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = 1\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \text{ and}$$
$$\begin{pmatrix} (\sigma_z)_{11} & (\sigma_z)_{12} \\ (\sigma_z)_{21} & (\sigma_z)_{22} \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = -1\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$
observed value

Observable in quantum mechanics

Fundamentally, many systems can be modeled as

$$\frac{du}{dt} = A u \qquad \text{or} \qquad u_{k+1} = A u_k$$

Schrödinger equation: $i\hbar\frac{\partial}{\partial t}|\Psi, t\rangle = H(t)|\Psi, t\rangle$.

Makine öğrenimi amacıyla aşağıdaki zaman dizisi modelini inceleyelim.

$$u_{k+1} = A\,u_k$$

İlk olarak, $u_o$ başlangıç durumunun A'nın bir özvektörü olduğunu varsayıyoruz. Bu nedenle, gelecekteki durumları şu şekilde hesaplanabilir:

$$u_{k+1} = A\,u_k$$

$$u_k = A^k\,u_0$$

$$u_1 = A\,u_0 = \lambda\,u_0$$
$$u_2 = A\,u_1 = \lambda^2\,u_0$$
$$u_k = \lambda^k\,u_0 \quad \text{——— much easier}$$

Bu denklem, bir matrisin gücünü ($A^k$) hesaplaması çok kolay olan bir skalerin gücüyle değiştirerek basitleştirilebilir. Ardından, sorunu herhangi bir başlangıç durumunu içerecek şekilde genelleştirelim. A'nın, $R^n$'nin temelini oluşturan n lineer bağımsız özvektöre sahip olduğunu düşünün. Herhangi bir $R^n$ vektörünü bu temele ayrıştırabiliriz. Yine özdeğerin gücünü yeniden hesaplayarak zaman dizisi hesaplamasını basitleştirebiliriz.

$$v_0 = c_1u_1 + c_2\,u_2 + \ldots + c_nu_n \quad \text{(eigenvectors)}$$

$$v_{k+1} = A\,v_k$$
$$v_k = A^k\,v_0$$
$$= c_1A^k\,u_1 + c_2A^k\,u_2 + \ldots + c_nA^k\,u_n$$
$$= c_1\lambda_1{}^k\,u_1 + c_2\,\lambda_2{}^k\,u_2 + \ldots + c_n\lambda_n{}^ku_n$$

Bir sistem kararlı bir duruma ulaşacaksa, yine $\lambda_i$'nin 1'den küçük veya 1'e eşit olmasını beklemeliyiz. Ek olarak, bu kararlı durumu hesaplamak için $\lambda_i{}^k$ değeri 1'den küçük olan terimleri göz ardı edebiliriz. Zaman geçtikçe bu terimler azalır 0. Dolayısıyla, kararlı bir durum bulmak için $\lambda_i = 1$ ile ilişkili özvektörlere odaklanabiliriz.

Tam potansiyelini gerçekleştirmek için gerçek bir multi-milyarlık fikri tartışalım. Tartışmamızı basitleştirelim ve tüm internetin yalnızca üç web sayfası içerdiğini varsayalım. A matrisinin $A_{ij}$ öğesi, kullanıcı j sayfasındayken i sayfasına gitme olasılığıdır.

*Markov matrix*

$$A = \begin{bmatrix} 0.5 & 0.1 & 0.7 \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.4 & 0.1 \end{bmatrix}$$

*all column values add up to 1.0*

$$A_{ij} = Pr(next\ page = i \mid page = j)$$

Given a specific page, if we sum over all the possibilities of the next page, it equals 1. i.e., all columns of *A* sum up to 1.0 and this kind of matrix is called the **stochastic matrix**, **transition matrix** or **Markov matrix**.

$$Pr(1 \mid page=1) + Pr(2 \mid page=1) + \ldots + Pr(n \mid page=1) = 1$$

Markov matrix has some important properties. The result of $Ax$ or $A^kx$ always sums up to one with its columns. The right-hand term is the chance of being on pages 1, 2 and 3 respectively after each click. So it is obvious that it should sum up to one.

$$\begin{bmatrix} 0.5 & 0.1 & 0.7 \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.4 & 0.1 \end{bmatrix} \begin{bmatrix} 0.2 \\ 0.4 \\ 0.4 \end{bmatrix} = \begin{bmatrix} 0.42 \\ 0.34 \\ 0.24 \end{bmatrix}$$

*sum up to 1*

$$\begin{bmatrix} 0.5 & 0.1 & 0.7 \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.4 & 0.1 \end{bmatrix} \begin{bmatrix} 0.42 \\ 0.34 \\ 0.24 \end{bmatrix} = \begin{bmatrix} 0.412 \\ 0.344 \\ 0.244 \end{bmatrix}$$

*sum up to 1*

Any Markov matrix *A* has an eigenvalue of 1 and the absolute value of other non-one eigenvalues to be smaller than one. This behavior is very important. In our example,

$$\lambda_1 = 1 \qquad \lambda_1 = 0.1 \qquad \lambda_1 = 0$$

$$u_1 = \frac{1}{90}\begin{bmatrix} 37 \\ 31 \\ 22 \end{bmatrix} \quad u_2 = \begin{bmatrix} -2 \\ 1 \\ 1 \end{bmatrix} \quad u_3 = \begin{bmatrix} -3 \\ 1 \\ 2 \end{bmatrix}$$

*sum up to 1*

For a Markov matrix, we can choose the eigenvector for $\lambda$=1 to have elements sum up to 1.0. So vectors *v0* can be decomposed using the eigenvectors of *A* with $c_1$ equals to 1 below.

$$
\begin{aligned}
v_0 &= c_1 u_1 + c_2 u_2 + \ldots + c_n u_n \\
&= u_1 + c_2 u_2 + \ldots + c_n u_n \\
v_k &= \underbrace{\lambda_1^k}_{=1} u_1 + c_2 \lambda_2^k u_2 + \ldots + c_n \lambda_n^k u_n \\
&\approx u_1 \qquad (\lambda_1^k = 1,\ otherwise\ \lambda_i^k \to 0)
\end{aligned}
$$

approach 0

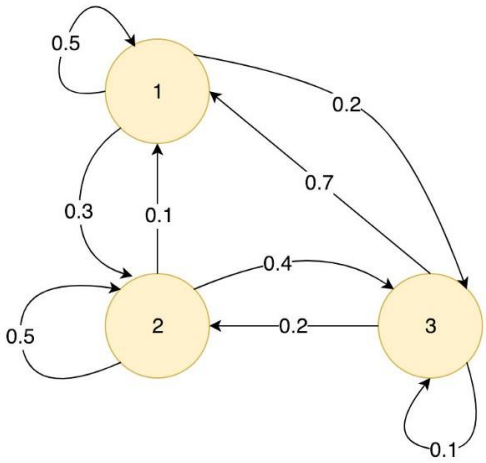
Since $u_1$, $u_2$, …, and *un* are eigenvectors, $A^k$ can be replaced by $\lambda^k$. Except for eigenvalue $\lambda$=1, *t*he power of the eigenvalue ($\lambda^k$) for a Markov matrix will diminish. So the system reaches a steady-state that approaches the eigenvector $u_1$ regardless of the initial state *v0*. The steady-state will equal the eigenvector $u_1$ as shown below.

$$\lambda_1 = 1$$

$$u_1 = \frac{1}{90}\begin{bmatrix} 37 \\ 31 \\ 22 \end{bmatrix} \qquad A^k \to \frac{1}{90}\begin{bmatrix} 37 & 37 & 37 \\ 31 & 31 & 31 \\ 22 & 22 & 22 \end{bmatrix} \qquad v_k \to \frac{1}{90}\begin{bmatrix} 37 \\ 31 \\ 22 \end{bmatrix} = \begin{bmatrix} 0.41\dot{} \\ 0.34\dot{} \\ 0.44\dot{} \end{bmatrix}$$

sum up to 1

From our example, the chance we land on pages 1, 2 and 3 are about 0.41, 0.34 and 0.44 respectively. This concept has many potential applications. For instance, many problems can be easily modeled with **Markov processes** and a Markov/transition matrix.




$$\begin{bmatrix} 0.5 & 0.1 & 0.7 \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.4 & 0.1 \end{bmatrix}$$

Markov process and the transition matrix

**Google PageRank**

Robot crawlers first retrieve the webpages, and then index and catalog them by assigning Page Rank Values. The credibility of each page depends on the number of links to that page. The rank r (P) of a given page P is assumed as,

$$r(P) = \sum_{Q \in Bp} \frac{r(Q)}{|Q|}$$

Where,

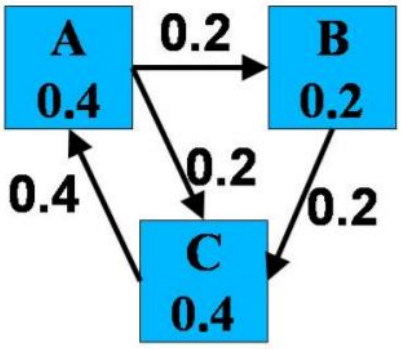
Bp = all pages pointing to P

|Q| = number of outlinks from Q.

P is the matrix with the terms,

$$Pi,j = \begin{cases} \frac{1}{Pi}, & if\ Pi\ links\ to\ \text{Pj}; \\ 0 & ,otherwise. \end{cases}$$

To find the convergence and convergence rates, the matrix P is adjusted. When the row Google matrix P reaches the sum to 1, then it is called a row stochastic matrix. The Page Rank iteration represents the evolution of a Markov Chain in which the web directed graph is represented as a transition probability matrix P.




It shows the probability of a random surfer at each of the three pages at any point in time. First, a binary Adjacency matrix is created to signify the link structure, then it is transformed into a probability matrix by normalizing it.

To compute the Page Rank, it is essential to solving an eigenvector problem for the linear system,

$$V^T(1 - P) = 0$$

The eigenvalues of Stochastic matrix P can be assumed as 1> λ1 ≥ λ2 ≥ … ≥ λn,

and V1, V2, V3, ……, Vn be the corresponding eigenvectors.

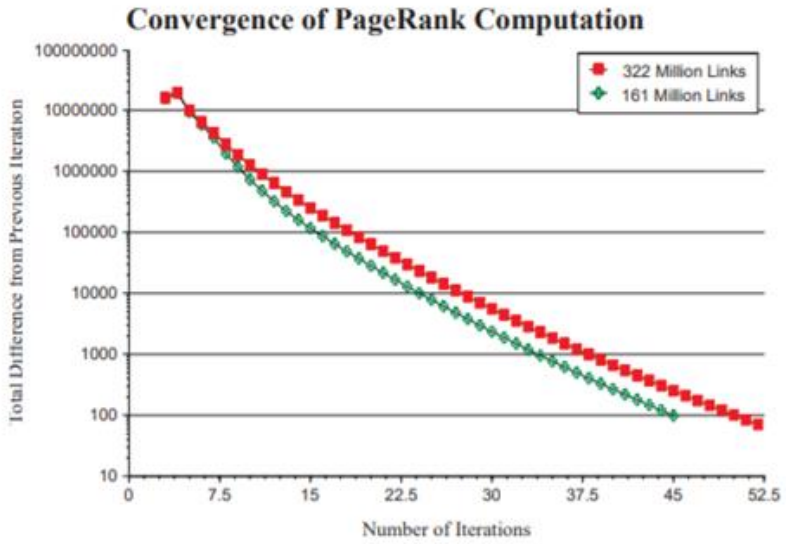
After the process of convergence, the dominant eigenvalue of matrix P should be λ=1 to satisfy,

$$V^T = V^T P$$

With,

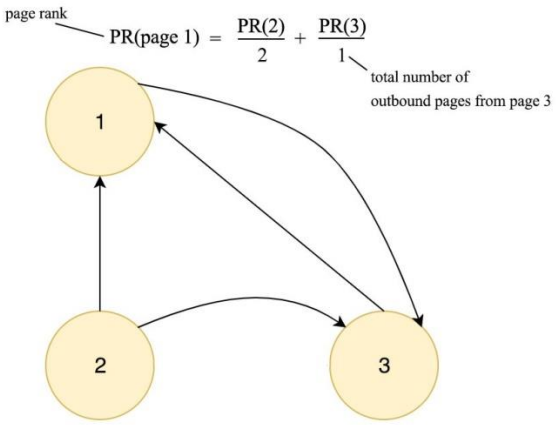
$$V^T e = 1$$

which is the steady state distribution of the Markov Model.
The process of PageRank convergence is shown in this graph,




This is how Google automatically characterizes the Page Rank value of each site.

The core idea can be conceptualized as the following. PageRank outputs a probability distribution of pages that you may land on by following links on a web page after many many clicks. This probability will act as the ranking of a Web page. When many pages link to your Web page, Google will rank it higher considering it as a good indicator of popularity.
In our previous example, we have not discussed ways to derive the values in the Markov matrix. For page rank, we compute the values as the sum of other page ranks linked to this page divided by its total number of outbound pages.




Mathematically, PageRank tries to solve the PageRank vector *R* (an eigenvector) in the following equation. It initializes R with equal probability at the beginning and performs the calculation iteratively until it reaches a steady-state.

$l$(pi, pj): the ratio between number of outbound from page j to page i to the total number of outbound page of j.

$$\mathbf{R} = \begin{bmatrix} (1-d)/N \\ (1-d)/N \\ \vdots \\ (1-d)/N \end{bmatrix} + d \begin{bmatrix} \ell(p_1,p_1) & \ell(p_1,p_2) & \cdots & \ell(p_1,p_N) \\ \ell(p_2,p_1) & \ddots & & \vdots \\ \vdots & & \ell(p_i,p_j) & \\ \ell(p_N,p_1) & \cdots & & \ell(p_N,p_N) \end{bmatrix} \mathbf{R}$$

0.85 in the Brin & Page 1998 page to reflect we don't random walk forever.

All columns sum up to one: A markov matrix

The matrix element $l_{ij}$ above is the ratio between the number of the outbound page from page j to page i to the total number of the outbound page of j. Each column of *l* adds up to 1 and it is a Markov matrix. This equation has an enormous similarity with the example we discussed before if we ignore the damping factor *d*. This factor is introduced because we will not have a random walk to take forever steps.

For Google, they do not compute the eigenvectors directly. In our previous example, the power of *A* converges very fast which the column of $A^3$ converge to eigenvector $u_1$ already.

$$A = \begin{bmatrix} 0.5 & 0.1 & 0.7 \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.4 & 0.1 \end{bmatrix} \quad A^2 = \begin{bmatrix} 0.42 & 0.38 & 0.44 \\ 0.34 & 0.36 & 0.33 \\ 0.24 & 0.26 & 0.23 \end{bmatrix} \quad A^3 = \begin{bmatrix} 0.41 & 0.41 & 0.41 \\ 0.34 & 0.35 & 0.34 \\ 0.24 & 0.25 & 0.24 \end{bmatrix} = A^4$$

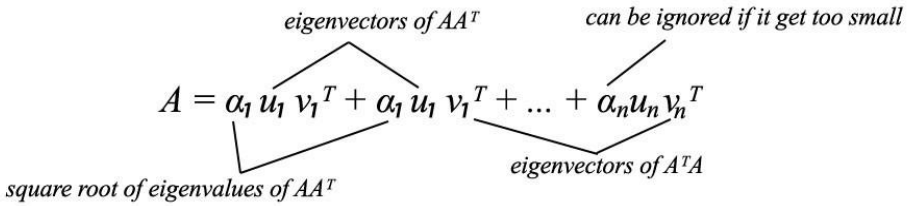
The PageRank paper has demonstrated that with 322 million page links, the solution converges to a tolerable limit in 52 iterations. So the solution scales unexpectedly well.

The Markov matrix leads us to the equation below which the steady-state depends on one principal component.

$$v_k = \lambda_1^k u_1 + c_2 \lambda_2^k u_2 + \cdots + c_n \lambda_n^k u_n \quad (c_2 \lambda_2^k u_2 + \cdots \to 0)$$

principal component

In machine learning, information is tangled in raw data. Intelligence is based on the ability to extract the principal components of information inside a stack of hay. Mathematically, eigenvalues and eigenvectors provide a way to identify them. Eigenvectors identify the components and eigenvalues quantify its significance. As a preview, the equation below decomposes information in *A* into components. We can prioritize them based on the square root of eigenvalues and ignore terms with small $\alpha$ values. This reduces the noise and helps us to extract the core information in *A*. (Details on a later article.)




**Solution**

I hope you can see the beauty of $Ax = \lambda x$ for now. Let's see how can we solve it. Eigenvalues and eigenvectors can be calculated by solving $(A - \lambda I)\, v = 0$. To have a solution other than $v$=0 for $Ax = \lambda x$, the matrix $(A - \lambda I)$ cannot be invertible. i.e. it is singular. Its determinant is zero. $\det(A - \lambda I) = 0$ is called the **characteristic polynomial** and the eigenvalues are the root of this polynomial.

$$\begin{aligned} Av &= \lambda v \\ (A - \lambda I)v &= 0 \\ \Longrightarrow \det(A - \lambda I) &= 0 \end{aligned}$$

if $(A - \lambda I)$ is invertible,
$v = (A - \lambda I)^{-1} 0 = 0$.
So for $v$ to have a solution other than 0, it has to be not invertible. Its determinant equals 0.

**Example**

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

The eigenvalues are:

$$det\,|A - \lambda I| = det \begin{bmatrix} 2-\lambda & 1 \\ 1 & 2-\lambda \end{bmatrix} = 3 - 4\lambda + \lambda^2 = 0$$

*characteristic polynomial*

$$\lambda = 1 \text{ or } 3$$

Apply $Av = \lambda v$, we solve:

$$v_{\lambda=1} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad v_{\lambda=3} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

Let's detail the step with a more complicated example,

$$A = \begin{bmatrix} 1 & -3 & 3 \\ 3 & -5 & 3 \\ 6 & -6 & 4 \end{bmatrix}$$

To find the eigenvalue $\lambda$,

$$det(A - \lambda I) = 0$$

$$\begin{vmatrix} 1-\lambda & -3 & 3 \\ 3 & -5-\lambda & 3 \\ 6 & -6 & 4-\lambda \end{vmatrix} = 0$$

$$(1-\lambda)\begin{vmatrix} -5-\lambda & 3 \\ -6 & 4-\lambda \end{vmatrix} - (-3)\begin{vmatrix} 3 & 3 \\ 6 & 4-\lambda \end{vmatrix} + 3\begin{vmatrix} 3 & -5-\lambda \\ 6 & -6 \end{vmatrix} = 0$$

$$16 + 12\lambda - \lambda^3 = 0$$

$$\lambda^3 - 12\lambda - 16 = 0$$

The possible factors for 16 are 1, 2, 4, 8, 16.

when $\lambda = 4$, $\lambda^3 - 12\lambda - 16 = 0$

So $\lambda^3 - 12\lambda - 16 = (\lambda - 4)(\lambda^2 + 4\lambda + 4) = 0$

By solving the root, the eigenvalues are 4, -2.

Let's calculate the eigenvector for eigenvalue $\lambda = 4$ through **row reduction**.

$$A - 4I = \begin{vmatrix} -3 & -3 & 3 \\ 3 & -9 & 3 \\ 6 & -6 & 0 \end{vmatrix}$$

Doing row reduction to solve the linear equation $(A - \lambda I)v = 0$

$$\begin{vmatrix} -3 & -3 & 3 & 0 \\ 3 & -9 & 3 & 0 \\ 6 & -6 & 0 & 0 \end{vmatrix}$$ Appending 0

Perform $R_1 = -\frac{1}{3}R_1$

$$\begin{vmatrix} 1 & 1 & -1 & 0 \\ 3 & -9 & 3 & 0 \\ 6 & -6 & 0 & 0 \end{vmatrix}$$

Perform row subtraction/multiplication:

$$\begin{vmatrix} 1 & 1 & -1 & 0 \\ 0 & -12 & 6 & 0 \\ 0 & -12 & 6 & 0 \end{vmatrix}$$

After many more reductions:

$$\begin{vmatrix} 1 & 0 & -1/2 & 0 \\ 0 & 1 & -1/2 & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

$$x_1 - \frac{1}{2}x_3 = 0$$
$$x_2 - \frac{1}{2}x_3 = 0$$

We have three variables with 2 equations. We set $x_3$ arbitrary to 1 and compute the other two variables. So for $\lambda=4$, the eigenvector is:

$$\begin{bmatrix} 1/2 \\ 1/2 \\ 1 \end{bmatrix}$$

We repeat the calculation for $\lambda=-2$ and get

$$x_1 - x_2 + x_3 = 0$$

With 3 variables and 1 equation, we have 2 degrees of freedom in our solution. Let's set the value to one in one of the degrees of freedom while other(s) to 0. i.e. setting $x_2=1$, $x_3=0$, and $x_2=0$, $x_3=1$ separately, the calculated eigenvectors will be:

$$\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \quad \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

Note that the solution set for eigenvalues and eigenvectors are not unique. we can rescale the eigenvectors. We can also set different values for $x_2$, $x_3$ above. Hene, it is possible and desirable to **choose** our eigenvectors to meet certain conditions. For example, for a symmetric matrix, it is always possible to choose the eigenvectors to have unit length and orthogonal to each other.

In our example, we have a repeated eigenvalue “-2”. It generates two different eigenvectors. However, this is not always the case — there are cases where repeated eigenvalues do not have more than one eigenvector.

**Diagonalizable**

Let’s assume a matrix *A* has two eigenvalues and eigenvectors.

$$Av_1 = \lambda_1 v_1$$

$$Av_2 = \lambda_2 v_2$$

We can concatenate them together and rewrite the equations in the matrix form.

$$A\begin{bmatrix} v_1 & v_2 \end{bmatrix} = \begin{bmatrix} \lambda_1 v_1 & \lambda_2 v_2 \end{bmatrix} = \begin{bmatrix} v_1 & v_2 \end{bmatrix}\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

We can generalize it into any number of eigenvectors as

$$\mathrm{A\,V} = \mathrm{V}\ \Lambda$$

since

$$\begin{bmatrix} | & & | \\ v_1 & \cdots & v_n \\ | & & | \end{bmatrix}\begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix} = \begin{bmatrix} | & & | \\ \lambda_1 v_1 & \cdots & \lambda_n v_n \\ | & & | \end{bmatrix} = \begin{bmatrix} | & & | \\ Av_1 & \cdots & Av_n \\ | & & | \end{bmatrix} = AV$$

where *V* concatenates all the eigenvectors and Λ (the capital letter for λ) is the diagonal matrix containing the eigenvalues.

$$\underbrace{\begin{pmatrix} \uparrow & \uparrow & & \uparrow \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \\ \downarrow & \downarrow & & \downarrow \end{pmatrix}}_{\mathrm{V}} \underbrace{\begin{pmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_n \end{pmatrix}}_{\text{Diagonal matrix } \Lambda}$$

A square matrix ***A*** is **diagonalizable** if we can convert it into a diagonal matrix, like

$$\mathrm{V}^{-1}\mathrm{A\,V} = \Lambda$$

i.e.,

$$P^{-1}AP = \begin{pmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{pmatrix}$$

Bir n × n kare matris, n doğrusal olarak bağımsız özvektöre sahipse köşegenleştirilebilir. Bir matris simetrik ise, köşegenleştirilebilir. Bir matris tekrarlanan özdeğere sahip değilse, her zaman bir vektörü köşegenleştirmeye yetecek kadar doğrusal olarak bağımsız özvektörler üretir. Tekrarlanan özdeğerleri varsa, yeterli özvektöre sahip olduğumuzun garantisi yoktur. Bazıları köşegenleştirilemez.

**Eigendecomposition**

If ***A*** is a square matrix with *N* linearly independent eigenvectors ($v_1$, $v_2$, … & *vn* and corresponding eigenvalues $\lambda_1$, $\lambda_2$, … & $\lambda n$), we can rearrange

$$V^{-1} A V = \Lambda$$

into

$$\forall = \Lambda V \Lambda_{-1}$$

For example,

$$A = \begin{bmatrix} | & | & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \\ | & | & | \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix} | & | & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \\ | & | & | \end{bmatrix}^{-1}$$

square matrix

the inverse exists only if eigenvectors are linearly independent

However, there are some serious limitations on eigendecomposition. First, *A* needs to be a square matrix. Second, *V* may not be invertible. As a preview, SVD solves both problems and SVD decomposes a matrix into orthogonal matrices which are easier to manipulate. But for matrices like symmetric matrices, this is not a problem. The power of a diagonalizable matrix *A* is easy to compute.

$$A^k = V \Lambda^k V^{-1}$$

This demonstrates the power of decomposing a matrix that can be manipulated easily.
*To recap, for states that can be expressed as*

$$u_{k+1} = A u_k$$

$A^k$ *can be computed with eigenvalues.*

**Properties of eigenvalue & eigenvectors**

- *Ax, x özvektörü ile aynı doğru üzerinde bulunur (aynı veya ters yön).*
- *Özdeğerlerin toplamı, bir matrisin izine eşittir (köşegen elemanların toplamı).*
- *Özdeğerlerin çarpımı determinant'a eşittir.*
- *Yukarıdaki her iki koşul da, özdeğerlerin hesaplanmasında iyi bir kontrol işlevi görür.*
- *Hiçbir özdeğer tekrarlanmıyorsa, tüm özvektörler doğrusal olarak bağımsızdır. Böyle bir n × n matris, n özdeğere ve n doğrusal olarak bağımsız özvektöre sahip olacaktır.*
- *Özdeğerler tekrarlanırsa, bir kare matrisi köşegenleştirmek için n lineer bağımsız özvektöre sahip olabilir veya olmayabiliriz.*
- *Pozitif özdeğerlerin sayısı, pozitif pivotların sayısına eşittir.*
- *Ax = λx için,*

$$A^k x = \lambda^k x,$$

$$A^{-1} x = \lambda^{-1} x,$$

$$(A + cI)x = (\lambda + c)x$$

- A tekil ise, özdeğeri 0'dır. Tersine çevrilebilir bir matrisin tüm özdeğerleri sıfır değildir.
- Özdeğerler ve özvektörler karmaşık sayılar olabilir.
- İzdüşüm matrislerinin her zaman yalnızca 1 ve 0 özdeğerleri vardır. Yansıma matrislerinin özdeğerleri 1 ve -1'dir.

Özdeğerler, özvektörler doğrultusunda bilginin önemini ölçer. Bu bilgilerle donatılmış olarak, bilgilerin hangi kısmının göz ardı edilebileceğini ve bilgilerin nasıl sıkıştırılacağını (SVD, Boyut küçültme ve PCA) biliyoruz. Ayrıca, makine öğrenimi modellerini geliştirirken özellikleri çıkarmamıza da yardımcı olur. Bazen, karışık bilgilerin azalması nedeniyle modelin eğitilmesini kolaylaştırır. Ayrıca karışık ham verileri görselleştirme amacına da hizmet eder. Diğer uygulamalar, öneri sistemlerini veya finansal risk analizini içerir. Örneğin, kişisel izleme davranışınıza ve diğerlerine göre filmler öneriyoruz. Veriler arasındaki korelasyonları anlamak için özvektörleri de kullanabiliriz. Belirli bir tür hastalığı tetikleyen genlerin kombinasyonu gibi ortak faktörleri bulmak için bilgi ve küme bilgi trendlerini geliştirin. Ve hepsi basit denklemden başlıyor:

$$Av = \lambda v$$

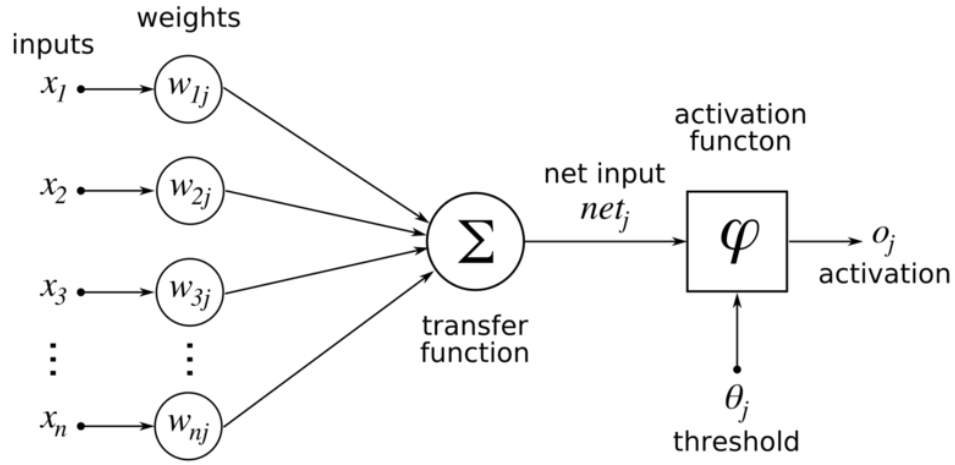
Matrisler eşit olarak oluşturulmaz. Makine öğreniminde bazı yaygın matrislerin özelliklerini bilmek önemlidir. Sonra, ona bir göz atacağız.

# 3. Hesaplamalı Matematik

This article is organized as follows:

1. Derivatives
2. Integration
3. Gradients
4. Gradient Visualization
5. Limit
6. Mathematical Optimization

Calculus is important because in order to optimize a neural network, we use variations of *gradient descent*, the most common of which is stochastic gradient descent.



## 3.1. Derivatives

The definition of a derivative can be understood in two different ways. The first way is in a geometric perspective, where the derivative is the slope of a curve. Essentially it is how much a curve is changing at a particular point.

The second is a physical perspective, where the derivative is the rate of change.
As we can see both of these perspectives are related to change.

# Differentiation

$$(uv)' = u'v + uv', \quad \left(\frac{u}{v}\right)' = \frac{u'v - uv'}{v^2}$$

$$(uv)^{(n)} = u^{(n)}v + nu^{(n-1)}v^{(1)} + \cdots + {}^nC_r u^{(n-r)}v^{(r)} + \cdots + uv^{(n)}$$

$$\text{where } {}^nC_r \equiv \binom{n}{r} = \frac{n!}{r!(n-r)!}$$

$$\frac{\mathrm{d}}{\mathrm{d}x}(\sin x) = \cos x \qquad \frac{\mathrm{d}}{\mathrm{d}x}(\sinh x) = \cosh x$$

$$\frac{\mathrm{d}}{\mathrm{d}x}(\cos x) = -\sin x \qquad \frac{\mathrm{d}}{\mathrm{d}x}(\cosh x) = \sinh x$$

$$\frac{\mathrm{d}}{\mathrm{d}x}(\tan x) = \sec^2 x \qquad \frac{\mathrm{d}}{\mathrm{d}x}(\tanh x) = \operatorname{sech}^2 x$$

$$\frac{\mathrm{d}}{\mathrm{d}x}(\sec x) = \sec x \tan x \qquad \frac{\mathrm{d}}{\mathrm{d}x}(\operatorname{sech} x) = -\operatorname{sech} x \tanh x$$

$$\frac{\mathrm{d}}{\mathrm{d}x}(\cot x) = -\operatorname{cosec}^2 x \qquad \frac{\mathrm{d}}{\mathrm{d}x}(\coth x) = -\operatorname{cosech}^2 x$$

$$\frac{\mathrm{d}}{\mathrm{d}x}(\operatorname{cosec} x) = -\operatorname{cosec} x \cot x \qquad \frac{\mathrm{d}}{\mathrm{d}x}(\operatorname{cosech} x) = -\operatorname{cosech} x \coth x$$

**Temel türev özellikleri ve kuralları:**

$$\frac{d}{dx}\left(cf(x)\right) = cf'(x),\ c \text{ is any constant.} \qquad \left(f(x) \pm g(x)\right)' = f'(x) \pm g'(x)$$

$$\frac{d}{dx}\left(x^n\right) = nx^{n-1},\ n \text{ is any number.} \qquad \frac{d}{dx}(c) = 0,\ c \text{ is any constant.}$$

$$(fg)' = f'g + fg' \quad \textbf{– (Product Rule)} \qquad \left(\frac{f}{g}\right)' = \frac{f'g - fg'}{g^2} \quad \textbf{– (Quotient Rule)}$$

$$\frac{d}{dx}\left(f\left(g(x)\right)\right) = f'\left(g(x)\right)g'(x) \quad \textbf{(Chain Rule)}$$

$$\frac{d}{dx}\left(\mathbf{e}^{g(x)}\right) = g'(x)\mathbf{e}^{g(x)} \qquad \frac{d}{dx}\left(\ln g(x)\right) = \frac{g'(x)}{g(x)}$$

**Derivative Notation**

The derivative operator is written as: ddxddx

This could read as the derivative with respect to xx.

**Scalar Derivative Rules**

This is not an exhaustive list of the rules, but let's review a few of the basics.

Constants - the derivative of any constant is 0.

dxc=0

For example: dx5=0

This makes sense when you think about change, because the constant 5 is not changing at all.

Power Rule - this is how you deal with variables in derivatives. If we have a derivative of $x^n$ with respect to xx, we calculate this as follows: $dx^n=nx^{n-1}$

For example: $dx^4=4x^3$

Multiplication - this is how we deal with multiplication of derivatives:

$dxcx^n=cnx^{n-1}$

The Sum Rule - if we have two functions and they are added together we can take the derivative of them both indivually:

dx(f(x)+g(x))=dxf(x)+dxg(x)

The Product Rule - multiplying two functions together is a bit more complicated, we take the derivate of the first function times the second plus the derivative of the second times the first: dx(f(x)g(x))=g(x)dxf(x)+f(x)dxd(x)

The Chain Rule - the chain rule is important for machine learning because when we have multiple layers in a neural network we need this calculate the derivative of earlier layers. If we have a function g(x)g(x) inside another function f(x), we take the derivative of the outside function f'' times the derivative of the inside function g'':

dx(f(g(x))=f'(g(x))g'(x))

These are just a few important rules, but let's move on to partial derivatives.

**Örnek: Matlab**

```
syms f(x)
f(x) = sin(x^2);
Df = diff(f,x)
```

Df(x) = 2 *x* cos(*x*2**)**

**Partial Derivatives**
Partial derivatives are used for equations with more than one variable.
If we have two variables we have the partial with respect to xx and the partial with respect to yy: ddxddyddxddy
Now let's move on to integrals, which are essential to probability theory.

**İvme:**
Fizikte ivme, hızın zamana göre türevi olarak tanımlanır. Büyüklüğü uzaklık/zaman$^2$ olan bir vektörel niceliktir ve cismin hem hızının hem de yönünün şiddetlerindeki değişimini gösterir. İvmeölçer yardımıyla ölçülen ivmenin SI birimi metre/saniye²'dir.

**Ortalama İvme:**
Bir cismin ortalama ivmesi belli bir zaman aralığı başına düşen belli bir hız miktarıdır. Matematiksel olarak aşağıdaki gibi ifade edilir:

$$\boldsymbol{a} = \frac{\Delta \boldsymbol{v}}{\Delta \boldsymbol{t}}$$

**Anlık İvme**
Bir cismin anlık ivmesi cismin anlık zamanda sahip olduğu hız miktarına eşittir. Matematiksel olarak aşağıdaki gibi ifade edilir:

$$a = \log_{\Delta t \to 0} \frac{\Delta v}{\Delta t} = \frac{dv}{dt}$$

İvme yolun ikinci türevi olarak da tarif edilebilir:

$$a = \frac{d^2x}{dt^2}$$

x (t) burada yolun fonksiyonunu temsil eder.

Genel olarak *ivme* terimi hızdaki (hız vektörünün şiddetindeki) artış olarak kullanılır; hızdaki azalışa ise *yavaşlama* denir. Fizikte, hız vektöründeki bir değişim ivme olarak kabul edilir: dairesel harekette, hız vektörünün yönündeki değişim *merkezcil (merkeze doğru) ivme*'ye yol açar. Bir cismin kazandığı ivmelenme, ona uygulanan kuvvetin kütlesine bölümünün bir fonksiyonudur.

İvme kelimesi köken olarak iv kökünden gelir ve ivedi:acele, iven:acele eden, ivmek:acele etmek gibi kelimelerle aynı ailede bulunur.

Klasik mekanikte sabit kütleli bir cismin ivmesi, cisme etki eden net kuvvetle orantılıdır (Newton'un ikinci yasası):

$$F = ma$$
$$a = F/m$$

Formülde F cisme etki eden net kuvvet, *m* cismin kütlesi ve a da cismin ivmesini temsil eder. Ortalama ivme kavramı hız vektöründeki değişimin *(Δv)* geçen süreye *(Δt)* bölümüdür. Anlık ivme de, Δt sıfıra yaklaşırken, çok kısa zaman aralıklarında, belirli bir noktanın ivmesidir.

Kütleçekim ivmesi
İvme kütleçekim sebebiyle de ortaya çıkar. Dünya simetrik bir küre olmadığı için, deniz seviyesinde kütleçekimi her yerde aynı değildir. Mesela Paris kentinde ivme;
$g = 9.80665 m/s^2$

## 3.2. Birinci ve İkinci türevin anlamı

Birinci türevin verdiği bilgilerden f'(t) veya df/dt olarak yazılan f(t) fonksiyonunun ilk türevi, t noktasındaki teğet çizgisi fonksiyonun eğimidir. Bunu grafik olmayan terimlerle ifade etmek gerekirse, ilk türev bize bir fonksiyonun nasıl arttığını veya azaldığını ya da ne kadar artacağını veya azalacağını söyler. Pozitif eğim t arttıkça f(t)'nin de arttığını söyler. Negatif eğim ise t arttıkça f(t)'in azaldığını söyler. Sıfır eğim özel bir şey söylemez: fonksiyon o noktada artar, ne azalır veya yerel maksimumda veya yerel minimumda olabilir. Türevler açısından bu bilgiler yazılırsa, bir f fonksiyonun herhangi bir nokta kararlı olup olmadığını belirlemek için

$\frac{df(p)}{dt} > 0$, ise f(t), t= p'de artan bir fonksiyondur.

$\frac{df(p)}{dt} < 0$, ise f(t), t= p'de azalan bir fonksiyondur.

$\frac{df(p)}{dt} = 0$, ise o zaman t = p, f (t)'in kritik noktası olarak adlandırılır ve bu noktada f (t) 'nin davranışı hakkında yorum yapılamaz.

Bir fonksiyonun ikinci türevi, f'' (t) veya $\frac{d^2f}{dt^2}$ olarak yazılır.

t = p'de $\frac{d^2f(p)}{dt^2}$ >0 ise, f(t), t = p'de yukarı doğru kavislidir, bölgesel mimum vardır.

t = p'de $\frac{d^2f(p)}{dt^2}$ <0 ise, f(t), t = p'de aşağı doğru kavislidir, bölgesel maksimum vardır.

t = p'de $\frac{d^2f(p)}{dt^2}$ =0 ise, o zaman f(t) 'in t = p'deki davranışı hakkında bir yorum yapamıyoruz.

Birinci türevin anlamından t, f(t) fonksiyonunun kritik bir noktası olduğunda, o noktada fonksiyonun davranışı hakkında bir yorum yapabilmek için, t'in bölgesel maksimum veya bölgesel minimum olduğunu öğrenmek için genellikle işlevin ikinci türevi kullanılır. Böylece t, f(t)'in kritik bir noktasıysa ve f(t)'in ikinci türevi pozitifse, t, f(t)'in bölgesel minumumdur. Benzer şekilde, t, f(t)'in kritik bir noktasıysa ve f(t)'in ikinci türevi negatifse, t'in f(t) fonksiyonun bölgesel maksimum olduğu anlaşılır. f(t) fonksiyonun t kritik bir noktasında f(t) fonksiyonunun ikinci türevi sıfır olduğunda, nokta hakkında yeni bir yorum yapamıyoruz. İkinci türev testi, kritik noktayı sınıflandırmak ve özellikle bölgesel maksimum ve bölgesel minimum bulmak için bir yol sağlar. İkinci türev testini özetlemek gerekirse:

$\frac{df(p)}{dt} = 0$ ve $\frac{d^2f(p)}{dt^2} > 0$ ise, f(t) 'in bölgesel bir minimum değeri t = p'dir.

$\frac{df(p)}{dt} = 0$ ve $\frac{d^2f(p)}{dt^2} < 0$ ise, f (t) 'in bölgesel bir maksimum değeri t = p'dir.

$\frac{df(p)}{dt} = 0$ ve $\frac{d^2f(p)}{dt^2} = 0$ ise, f (t) 'in t = p'deki davranışı hakkında bir yorum yapamıyoruz.

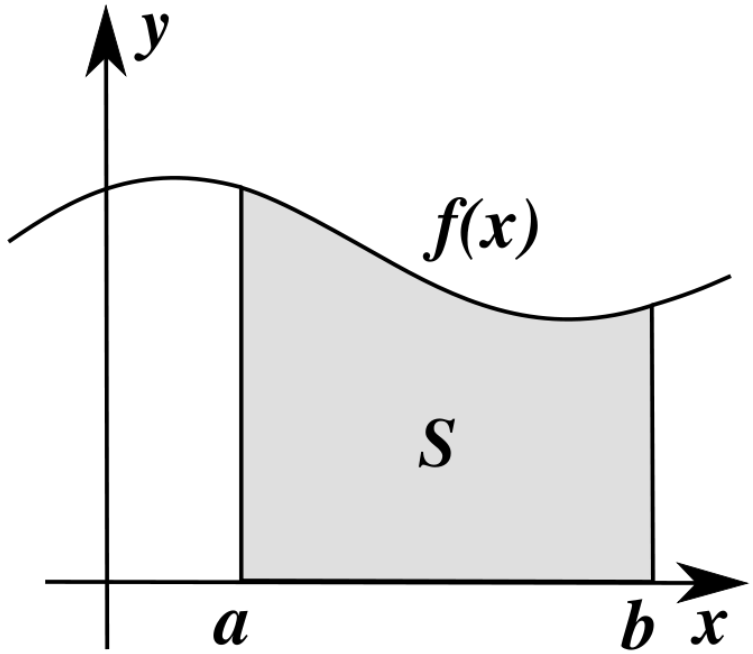
**Bükülme Noktaları:**

Bir f(t) fonksiyonunun grafiğinin t'de bir bükülme noktası olduğunu varsayalım. Fonksiyonun grafiği yukarı doğru kavisden aşağı doğru kavise veya fonksiyonun grafiği aşağı doğru kavisden yukarı doğru kavise gitsin. Açık bir şekilde, bir bükülme noktası sadece ikinci türevin 0 olduğu noktalarda meydana gelebilir, aksi takdirde grafik ya tamamen yukarı doğru kavisli ya da o noktada tamamen aşağı doğru kavisli olacaktır.

Bununla birlikte, bölgesel maksimum ve bölgesel minimumu ve ilk türevde olduğu gibi, bir fonksiyonun ikinci türevinin 0 olduğu bir noktanın varlığı, noktanın bir bükülme noktası olduğunu otomatik olarak söylemez. Dolayısıyla, ikinci türev bize bir fonksiyonun grafiğinin şekli hakkında çok şey söylese de, bize her şeyi söyleyemez: bir fonksiyonun grafiğinin bir bükülme noktası olup olmadığını bize söyleyemez; bize sadece bir bükülme noktası olabileceğini söyleyebilir. Kıvrım noktalarını kesin olarak bulmak için kullanabileceğimiz ikinci türev testi gibi bir testi düşünebilir misiniz?

## 3.3. Integration

Let's now discuss integrals in the context of machine learning. **Integration is used to find the area under a curve. The integral is the opposite of differentiation, and is often referred to as the anti-derivative.**




There are two types of integrals: definite and indefinite.

Definite integrals have limits, and can be written as follows: $\int_a^b f(x)dx \int_a^b f(x)dx$

Indefinite integrals have no limits, so we're taking the integral from negative infinity to positive infinity: $\int f(x)dx$

As mentioned, the integral can be understood as the anti-derivative as it reverses differentiation. In the equation below we have the integral of f'(x) is equal to f(x), so if we take the integral of a function we get back the original function.
$\int f'(x)dx=f(x)+C$

The constant of integration and is there because the derivative of a constant is 0, so we don't know if it was there or not.

**The Rules of Integration**

Let's review a few of the rules of integration.
The Power Rule - this is the reverse of the power rule in derivatives:
$\int x^n dx=x^{n+1}/(n+1)+C$

Constants - these follow the power rule:

∫kdx=kx+C

**Evaluating Definite Integrals**

If we have limits on our integral a and b we evaluate them as follows:

∫baf(x)dx

This is the same as saying the (value of f(x) + C at x=b) - (value of f(x) + C at x=a). Since the C's are going to cancel out we're left with:

∫baf(x)dx=f(b)−f(a)

∫abf(x)dx=f(b)−f(a)

This can also be written as:

∫baf(x)dx=f(x)|ba=f(b)−f(a)∫abf(x)dx=f(x)|ab=f(b)−f(a)

We'll discuss integrals more in our article on probability theory, but let's now move on to the concept of gradients.

## Integration

$$\int x^n \,\mathrm{d}x = \frac{x^{n+1}}{n+1} + c \qquad \text{for } n \neq -1$$

$$\int \frac{1}{x} \,\mathrm{d}x = \ln x + c \qquad \int \ln x \,\mathrm{d}x = x(\ln x - 1) + c$$

$$\int \mathrm{e}^{ax} \,\mathrm{d}x = \frac{1}{a}\mathrm{e}^{ax} + c \qquad \int x\,\mathrm{e}^{ax} \,\mathrm{d}x = \mathrm{e}^{ax}\left(\frac{x}{a} - \frac{1}{a^2}\right) + c$$

$$\int x \ln x \,\mathrm{d}x = \frac{x^2}{2}\left(\ln x - \frac{1}{2}\right) + c$$

$$\int \frac{1}{a^2 + x^2} \,\mathrm{d}x = \frac{1}{a}\tan^{-1}\left(\frac{x}{a}\right) + c$$

$$\int \frac{1}{a^2 - x^2} \,\mathrm{d}x = \frac{1}{a}\tanh^{-1}\left(\frac{x}{a}\right) + c = \frac{1}{2a}\ln\left(\frac{a+x}{a-x}\right) + c \qquad \text{for } x^2 < a^2$$

$$\int \frac{1}{x^2 - a^2} \,\mathrm{d}x = -\frac{1}{a}\coth^{-1}\left(\frac{x}{a}\right) + c = \frac{1}{2a}\ln\left(\frac{x-a}{x+a}\right) + c \qquad \text{for } x^2 > a^2$$

$$\int \frac{x}{(x^2 \pm a^2)^n} \,\mathrm{d}x = \frac{-1}{2(n-1)}\frac{1}{(x^2 \pm a^2)^{n-1}} + c \qquad \text{for } n \neq 1$$

$$\int \frac{x}{x^2 \pm a^2} \,\mathrm{d}x = \frac{1}{2}\ln(x^2 \pm a^2) + c$$

$$\int \frac{1}{\sqrt{a^2 - x^2}} \,\mathrm{d}x = \sin^{-1}\left(\frac{x}{a}\right) + c$$

$$\int \frac{1}{\sqrt{x^2 \pm a^2}} \,\mathrm{d}x = \ln\left(x + \sqrt{x^2 \pm a^2}\right) + c$$

## 3.4. Diferansiyel Denklemlerin Faz Düzlemleri

bir diferansiyel denkleme örnek olarak

$y''' + 3x^2 - 4y = xe^x + 2cotx$

verilebilir.

Bir diferansiyel denklemde en yüksek mertebeli türevin mertebesi diferansiyel denklemin ***mertebe***sini verir. Örneğin üstteki denklem 3. mertebedendir denir. Bunun yanında, mertebeyle sıkça karıştırılan bir kavram olan ***derece***'ye de değinmek gerekir. Bir diferansiyel denklemde bulunan en yüksek mertebeli türevin üssüne, bu diferansiyel denklemin ***derece***si denecektir.

Bir diferansiyel denklemdeki bağımlı değişken ve tüm türevleri birinci dereceden ise, diferansiyel denkleme ***lineer*** diferansiyel denklem denir.

Dolayısıyla içerisinde $y^3$ ,$(y'')^2$ , $yy'$, $y'y'''$, $\sin y$, $e^y$ gibi terimler bulunan denklemler lineer değildir. Bunun yanında denklem $x^2$ , $xy''$ , $\sin x$, $\ln x$ türünden ifadeler içerebilir.

Daha genel bir ifadeyle eğer bir diferansiyel denklem

$$y^{(n)} + f_1(x)y^{(n-1)} + f_2(x)y^{(n-2)} + \cdots + f_n(x)y = R(x)$$

formunda ifade edilebiliyorsa denkleme lineerdir diyeceğiz, aksi halde lineer olmayan bir diferansiyel denklem söz konusudur. Bu denklemde eğer $R(x) = 0$ ise lineer diferansiyel denklem ***homojen***dir. Aksi durumda denklem ***homojen olmayan*** diferansiyel denklem adını alır.

Aşağıdaki diferansiyel denklemleri sınıflandırınız.

**Çözüm**

(1) $y''' + 3y = 0$ (2. mertebe-lineer-homojen)

(2) $y'' + 3y = 2x + \sin x$ (2. mertebe-lineer-homojen değil)

(3) $y'' + 3yy' = 0$ (2. mertebe-lineer değil)

(4) $y''' + \sin x\, y' + \cos y = e^{-2x}$ (3. mertebe-lineer değil-homojen değil)

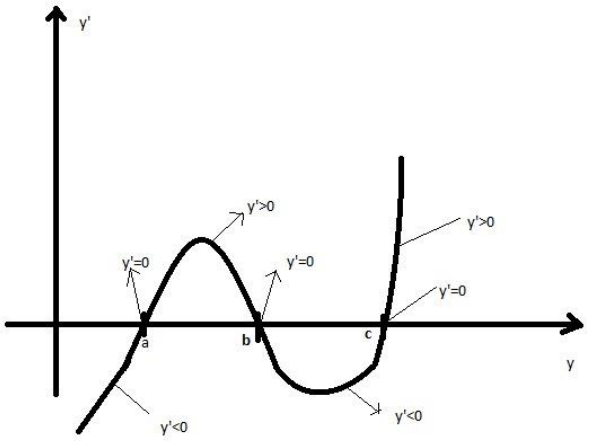
I ⊆ R olmak üzere t bağımsız değişkeninin türevlenebilir bir x : I→R fonksiyonunu ele alalım. Bu bölümde f : R→R olmak üzere

$\dot{x}$= f (x) (

diferansiyel denkleminde bilinmeyen x fonksiyonunda türev t değişkenine göre, f fonksiyonunda ise x değişkenine göre olacaktır. Sıklıkla karışıklık olmaması için dx/dt ve df /dx türevlerinin sırasıyla $\dot{x}$ ve $f'$notasyonları kullanılmaktadır. Bu denkleme bir skaler otonom denklem denir, skaler denmesinin sebebi bir boyutlu (reel değişkenli) olması ve otonom denmesinin sebebi de f fonksiyonunun t değişkeninden bağımsız olmasıdır.

Otonom denklemlerin genel formu, $\dot{x}$= f(x) dır. Bir diferansiyel denklemin otonom olabilmesi için türev eşitliğinin karşısında fonksiyonda sadece x ifadeleri olmasıdır. Başka bir değişken olmamasıdır. Bu formdaki denklemlere otonom difarensiyel denklemler adı verilir. Otonom diferansiyel denklemlerin davranışı genellikle $\dot{x}$-x grafiği çizilerek anlaşılmaya çalışılır. Davranıştan kastedilen çözüm eğrilerini tahmin etmektir. $\dot{x}$-x grafiği yorumlanarak çözüm eğrilerinin görünümü tahmin edilebilir. Grafik çizilerek çözüm eğrilerinin tahmin edilmesine faz düzlem analizi denir.

Öncelikle $\dot{x}$-x grafiğini çizilir.



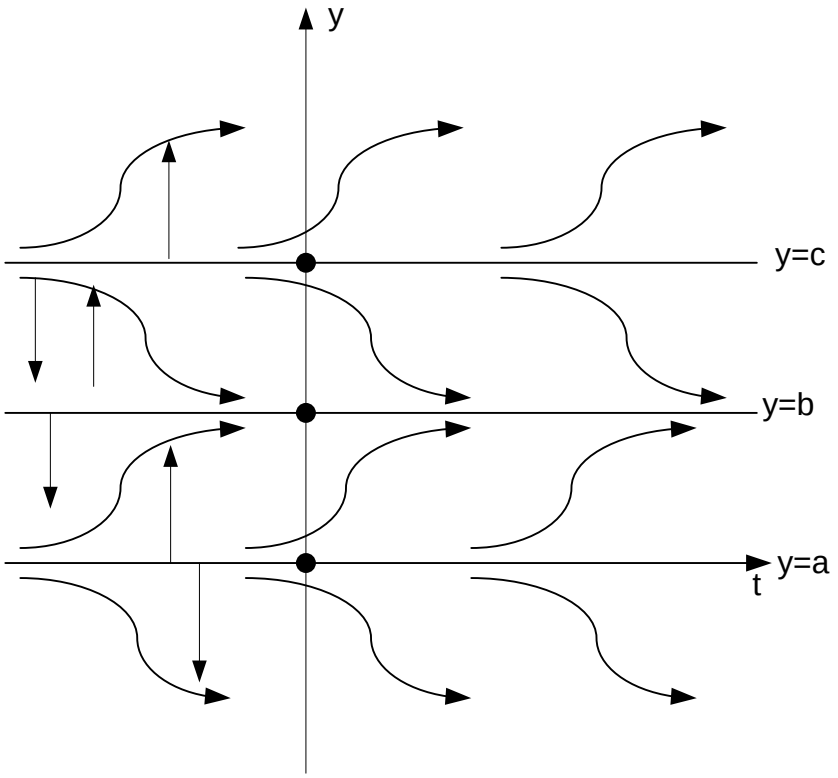

Şekil 2.1.1

Bu grafiği çizdikten sonra şu yorumları yaparız. y=a, y=b, y=c değerleri y'=y fonksiyonunun kökleridir.

y<a ise y'<0
y=a ise y'=0
a<y<b ise y'>0
y=b ise y'=0
b<y<c ise y'<0
y=c ise y'=0
y>c ise y'>0

Grafiğe bakarak bu yorumları kolaylıkla yapabiliriz. Bu işlemden sonra elde ettiğimiz bilgilerden yararlanarak çözüm eğrilerinin şeklini tahmin ederiz. Şöyleki; y'>0 ise çözüm eğrileri yukarı doğru meyilli olacaktır. Çünkü türevin pozitif olabilmesi için y nin grafiği bu şekilde bir eğri olmalıdır. Dolayısıyla buna çizilen teğet yani türev pozitif olmalıdır.Teğetin eğimi şekildeki gibi pozitiftir. y'<0 ise çözüm eğrileri aşağı meyilli olacaktır. y'=0 ise çözüm eğrileri eğimi sıfır olan düz birer çizgi olacaktır. Bu bilgileri kullanarak basitçe çözüm eğrilerinin şeklini çizebiliriz.

Çözüm eğrileri çizilir. Y-ekseni ve y nin bağlı olduğu değişken cinsinden eksenler çizilir. Biz t değişkeni kullanacağız. Yukarıda faz düzlem analizinde y'-y grafiği çizilip yorumlandı. Bu ise çözüm eğrilerini gösteren y-t grafiği olacaktır.




Şekil 2.1.2

y=a , y=b ve y=c noktalarında türevimiz "0" a eşit olduğu için eğrilerimizin eğimi sıfır olan düz çizgiler olmak zorundadır. Düz olarak çizdiğimiz çözüm eğrilerine denge çözümleri adı verilir. Dikkat edin hiçbir çözüm eğrisi birbirine değmemelidir. Denge çözümleri ile de kesişmez. Grafikte görülen çözüm eğrilerinin şekilleri böylelikle tahmin edilir. Faz düzlem analizi y'-y grafiğini inceleyerek bu sonuçlara vardık. Diğer çözümler denge çözümlerine (y'=0 olan çizgilere) yaklaşmaya çalışır. Sanki diğer çözüm eğrileri bir şekilde dengesini arıyor ve oraya doğru yaklaşmaya çalışıyor. Bu yüzden denge çözümleri olarak adlandırılmıştır. Bu denge çözümleri örneğimiz için; y=a, y=b ve y=c olmak üzere üçtanedir.

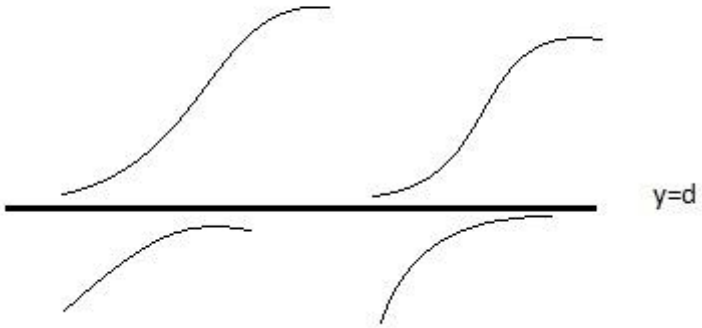
Denge çözümlerinin 3 çeşidi vardır:

- Kararlı denge çözümüleri
- Kararsız denge çözümleri
- Yarı kararlı denge çözümleri

Bunları anlamak için tekrar grafiğe bakılırsa, y=c denge çözümüne bakılırsa altında ve üstünde kalan çözüm eğrilerine göre yorumlanır. t sonsuza giderken altındaki ve üstündeki iki çözüm eğrisi de y=c denge çözümünden uzaklaşmaktadır. Dolayısıyla "y=c" ye kararsız denge çözümü denir. Kararsız dengede, çözümden biraz dışarı çıkıldığında çözümler dengeden uzaklaşır.

y=b denge çözümüne bakarsak, t sonsuza giderken çözümden biraz dışarı çıkıldığında alttaki ve üsttteki iki çözüm eğrisi de y=b denge çözümüne yaklaşmaktadır. Dolayısıyla "y=b" ye kararlı denge çözümü denir. Kararlı denge çözümünde, çözümden biraz dışarı çıkıldığında çözümler dengeye yaklaşma eğilimi gösterir.

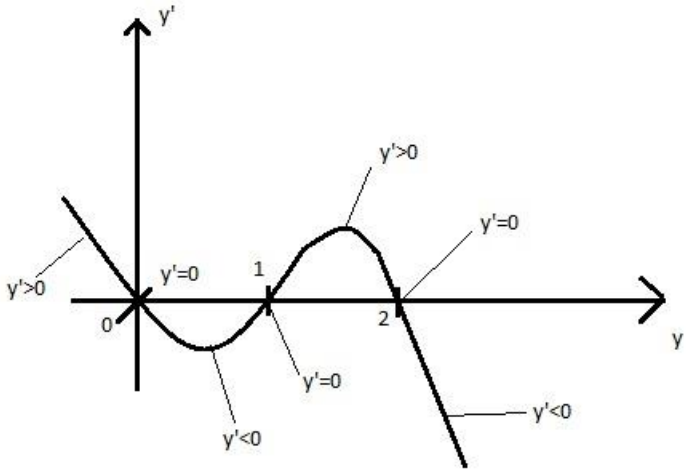
y=a denge çözümüne bakarsak, t sonsuza giderken altında ve üstünde kalan iki çözüm eğrisi de "y=a" çözümünden uzaklaşmaktadır. Dolayısıyla "y=a" ya kararsız denge çözümü denir. Yarı kararlı denge çözümününü göstermek gerekirse "y=d" de denge çözümümüz var diyelim.




Şekil 2.1.3

t sonsuza doğru giderken "y=d" doğrusundan yukarı doğru yaklaştığında "y=d" doğrusundan uzaklaşırken. Aşağıdaki çözüm eğrisine yaklaşıldığında "y=d" doğrusuna yakınlaşmaktadır. Yarı kararlı denge çözümü= çözüm eğrilerinden biri çözüm doğrusuna yaklaşırken diğeri uzaklaşıyorsa buna yarı kararlı denge çözümü denir.
Faz düzlem analizini daha iyi kavramak için başka bir örnek üzerinden gidelim

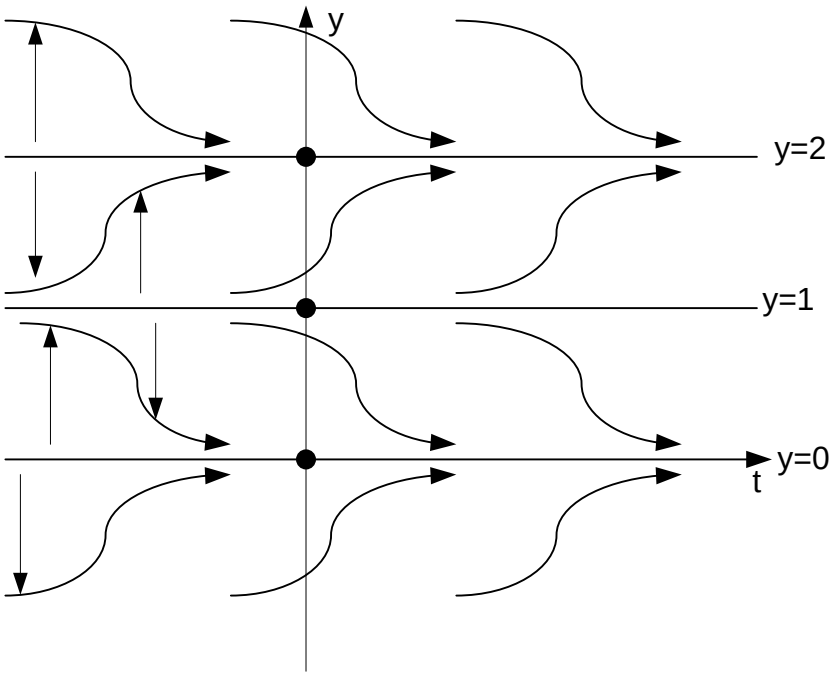
Örnek:
y’=y(y-1).(2-y) diferansiyel denklemi için öncelikle y’-y grafiğini çizelim ve çözüm eğrilerini tahmin edelim. Çözüm y(t) şeklinde ifade edilmektedir. Daha sonra denge çözümlerini bulalım ve kararlılık durumlarını inceleyelim. Daha sonra y(2)=5 , y(10)=2 , y(1)=3/2 , y(3)=1, y(1)=-2 başlangıç noktaları olmak üzere t sonsuza giderken y(t) ne olur inceleyelim.
Denklem gördüğümüz üzere tek değişken “y” olduğu için otonom bir denklemdir. Çünkü, denklemde sadece y'li’ifadeler bulunmaktadır. İlk işimiz y’-y grafiğini çizmek olacaktır.




Şekil 2.1.4

**Çözüm eğrileri:**
Çözüm eğrilerini tahmin etmek için faz düzlemi analiz yapılır. Grafik üzerinde türevin sıfırdan büyük, sıfırdan küçük ve sıfıra eşit olması yorumlanır. Çözüm eğrilerini tahmin edelim.

y<0 değerleri için y’>0 olur. O halde çözüm eğrisi yukarı doğru meyildir ve y=0 çizgisine değmez.
y=0 ile y=1 aralığında y’<0 olur. O halde çözüm eğrisi aşağı doğru meyildir ve y=0 ve y=1 çizgilerine değmez.
y=1 ile y=2 aralığında y’>0 olur. O halde çözüm eğrisi yukarı doğru meyildir ve y=1 ile y=2 çizgilerine değmez.
y>2 değerleri için y’<0 olur. O halde çözüm eğrisi aşağı doğru meyildir ve y=2 çizgisine değmez.




Şekil 2.1.5 Çözüm eğrileri

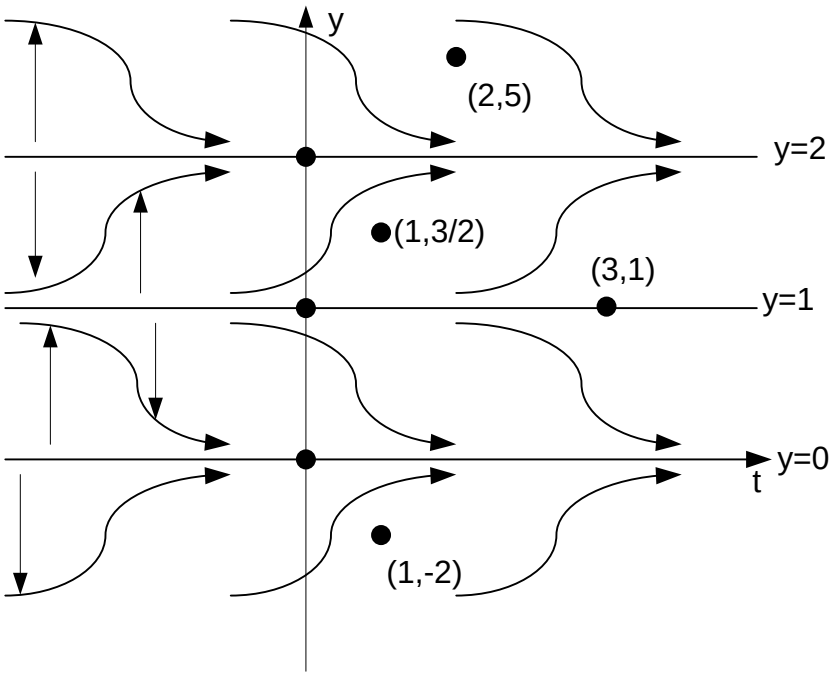
Denge Çözümleri ve Kararlılık Durumları:
Denge çözümleri türevin sıfıra eşit olduğu durumlardır. Karalılık durumları analiz edlirken denge çözümlerine bakılır. Çözüm eğrileri kesişmez; y=0, y=1 ve y=2 denge çözümleridir. Şimdi denge çözümlerine ve kararlılık durumlarına bakalım.

Şekil 3.5'de y=2'den biraz yukarı doğru çıkılırsa y>2 bölgesindeki eğriyi yakalar ve t büyüdükce y=2 çizgisine yaklaşır. y=2'den biraz aşağı doğru inilirse y<2 bölgesindeki eğriyi yakalar ve t büyüdükce y=2 çizgisine yaklaşır. O halde y=2 çizgisi kararlı bir denge çözümüdür.

Şekil 3.5'de y=1'den biraz yukarı doğru çıkılırsa y>1 bölgesindeki eğriyi yakalar ve t büyüdükce y=2 çizgisine yaklaşır. y=1'den biraz aşağı doğru inilirse y<1 bölgesindeki eğriyi yakalar ve t büyüdükce y=0 çizgisine yaklaşır. O halde y=1 çizgisi kararsız bir denge çözümüdür.

Şekil 3.5'de y=0'den biraz yukarı doğru çıkılırsa y>0 bölgesindeki eğriyi yakalar ve t büyüdükce y=0 çizgisine yaklaşır. y=0'den biraz aşağı doğru inilirse y<0 bölgesindeki eğriyi yakalar ve t büyüdükce y=0 çizgisine yaklaşır. O halde y=0 çizgisi karalı bir denge çözümüdür.

y(2)=5 , y(10)=2 , y(1)=3/2 , y(3)=1, y(1)=-2,  başlangıç noktaları için t sonsuza gittiğinde çözüm eğrilerinin davranışlarının anliz edilmesi:




Şekil 2.1.6

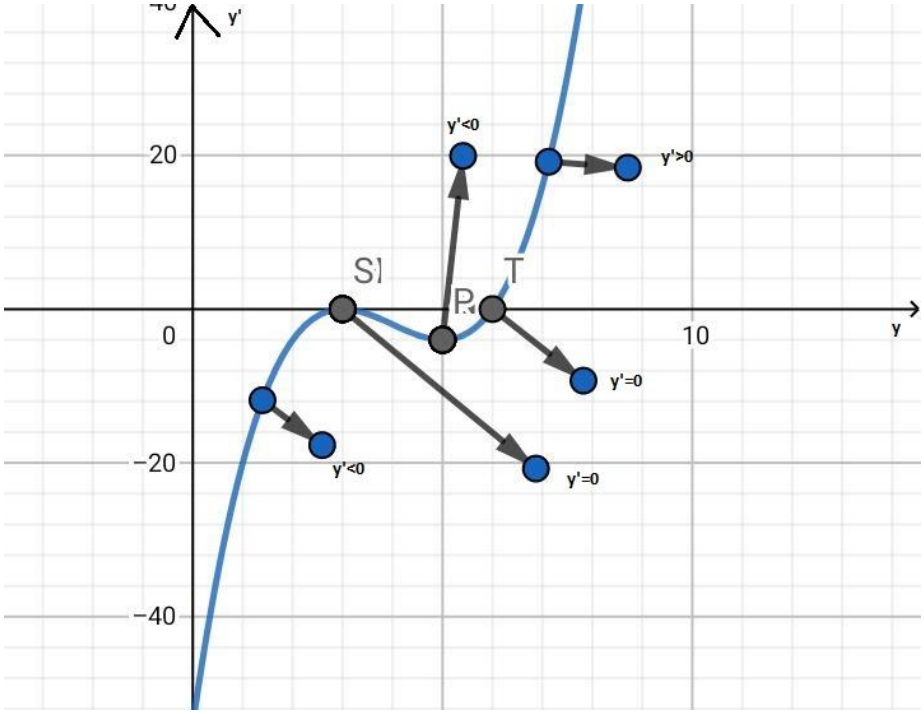
t=2, iken y=5 noktası y>2 çözüm eğrisi üzerine düşer. t → ∞ olursa, y=2 çizgisine yaklaşır. $\log_{t\to\infty} y(t) = 2$. Kararlıdır.

t=1, iken y=3/2 noktası y<2 veya y>1 çözüm eğrisi üzerine düşer. t → ∞ olursa, y=2 çizgisine yaklaşır. $\log_{t\to\infty} y(t) = 2$. Karalıdır.

t=3, iken y=1 noktası ya y>1 ya da y<1 çözüm eğrisi üzerine düşer. t → ∞ olursa, ya y=2 ya da y=0 çizgisine yaklaşır. Kararsızdır.

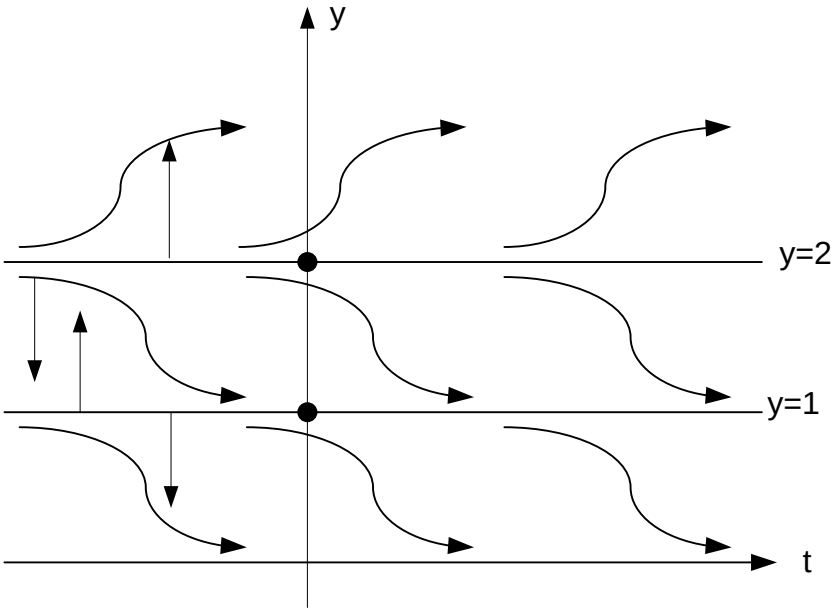
t=1, iken y=-2 noktası y<0 çözüm eğrisi üzerine düşer. t → ∞ olursa, y=0 çizgisine yaklaşır. $\log_{t\to\infty} y(t) = 0$. Kararlıdır.

Örnek:
y'=$y^3$-4$y^2$+5y-2 diferansiyel denklem için, y'-y grafiğini çizelim ve çözüm eğrileri grafiğini yapalım. Denge çözümleri ve kararlılık durumlarını analiz edelim. Başlangıç noktası y(0)=$y_0$ kabul edilirse $y_0$ ın hangi değerleri için çözüm eğrilerinin limit davranışı ne olur?
Öncelikle denklemimizin y eksenini nerelerde kestiğini bulmak için çarpanlarına ayırmamız gerekiyor
y'=(y-2).( $y^2 - 2y + 1$)
y'=(y-2).$(y-1)^2$
y=2 ve y=1 kesim noktaları
y=1 çift kat kök olduğu için teğet kesecektir.




Şekil 2.1.7

Grafiğimizi çizdikten sonra çözüm eğrilerini tahmin etmeye başlayabiliriz.

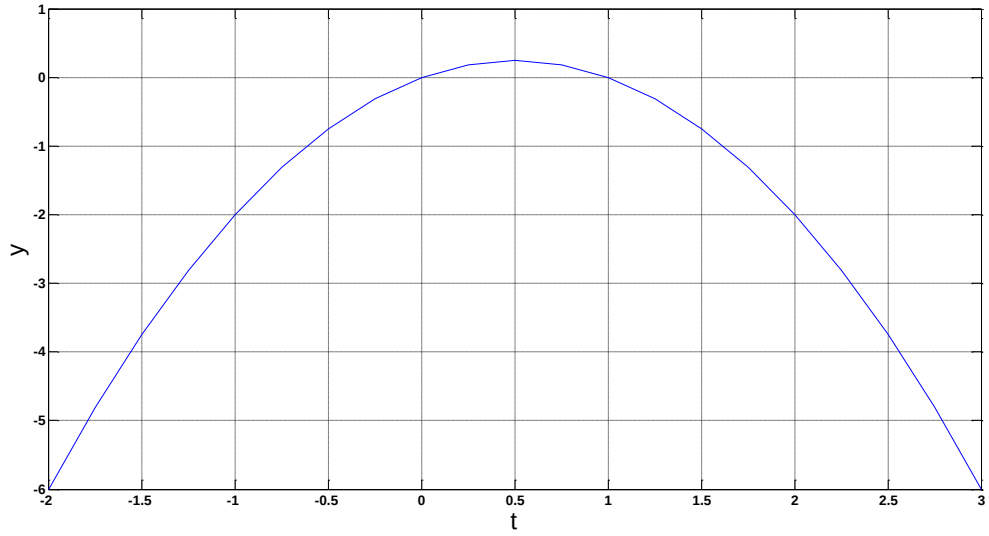



Şekil 2.1.8

y=2 ve y=1 denge çözümleridir, aralarında kalan eğriler ise çözüm eğrilerini vermektedir. Denge çözümlerinin kararlılık durumları incelenirse,
y=2 için t sonsuza giderken iki çözüm eğrisi de y=2 doğrusundan uzaklaştığı için kararsız denge çözümüdür.
y=1 için t sonsuza giderken üstte kalan çözüm eğrisini yakaladığımızda y=1 doğrusuna yaklaşıyor, altta kalan çözüm eğrisi ise y=1 doğrusundan uzaklaşmaktadır. Buna dayanarak y=1 için yarı kararlı denge çözümü olduğunu söyleriz.

(0, $y_0$) noktasına baktığımızda y=1 doğrusunun altında kalan bir çözüm eğrisi yakaladığını görüyoruz. T sonsuza giderken y ise eksi sonsuza gider.
(0, $y_0$)
$y_0$<1 için t→∞ y→-∞
1≤ $y_0$<2 için t→∞ y→1
$y_0$=2 için t→∞ y→2
$y_0$>2 için t→∞ y→∞
Otonom denklemler ve faz düzlem analizi yeterince iyi anlaşıldığında sürü davranışı sergileyen insansız hava araçlarının takip ettikleri yörüngelerinin denge çözümlerine bakılarak kararlı yada kararsız olduğu hakkında yorum yapabilir.

**Örnek:**
y'=-y(y-1)=y(1-y)
y=0 ve y=1 denge çözümleridir.




y<0 olduğunda y'<0 olur. O halde y<0 olduğu yerlerde çözüm eğrileri aşağı doğru meyildir, çözüm eğrileri y=0 çizgisine değmez.

0<y<1 aralığında y'>0 olur. O halde bu aralıkta çözüm eğrileri yukarı doğru meyildir, çözüm eğrileri y=0 ve y=1 çizgilerine değmez.

1<y, y’<0 olur. O halde çözüm y>1 olduğu yerlerde çözüm eğrileri aşağı doğru meyildir, y=1 çizgisine değmez.

y=0 denge çözüm çizgisinden biraz yukarı gidildiğinde, çözüm eğrisi yukarı doğru y=1 çizgisine yönelir. Biraz aşağı gidildiğinde çözüm eğrileri aşağı doğru -∞ a yönelir. Bu nedenle y=0 denge çözümü kararsızdır.

y=1 denge durumundan biraz yukarı gidildiğinde, çözüm eğrisi aşağı doğru y=1 çizgisine yönelir. Biraz aşağı gidildiğinde de çözüm eğrisi yukarı doğru y=1 çizgisine yönelir. Bu nedenle y=1 denge çözümü kararlıdır.

## 3.5. Diferansiyel Denklemler - Doğrultu alanları ve çözüm eğrileri

Bir doğrunun eğimi ya da gradyanı o doğrunun dikliğini, eğimliliğini belirtir.
$y = mx + k$
$y'$=m

Eğim, bir doğrunun herhangi iki noktası arasındaki dikey değişimin yatay değişe oranı olarak tanımlanabilir. Bir doğru üzerinde ($x_1$,$y_1$) ve ($x_2$,$y_2$) koordinatlarında iki nokta verildiğinde doğrunun eğimi, m

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

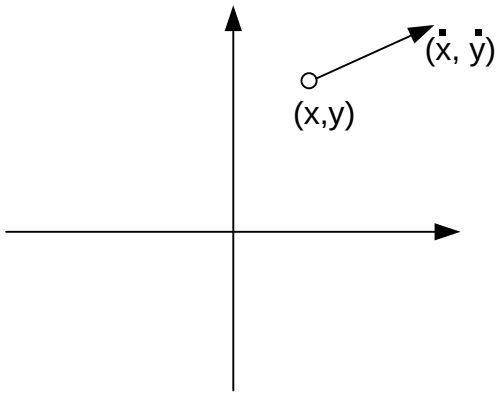
ile bulunur. Bir doğrunun pozitif x aksisiyle yaptığı θ açısı, tanjant fonksiyonu aracılığıyla m eğimiyle yakından ilgilidir,

$m = tan\theta.$

y2-y1 işareti + ve x2-x1 işareti + ise 1. Bölge
y2-y1 işareti + ve x2-x1 işareti - ise 2. Bölge
y2-y1 işareti - ve x2-x1 işareti - ise 3. Bölge
y2-y1 işareti - ve x2-x1 işareti + ise 4. Bölge

Birinci dereceden lineer diferansiyel denklemleri faz düzlemi analizinde kullanılmaktadır. Diferansiyel denklemleri, cebirsel denklemlerden ayıran en önemli özellik “fonksiyon türevleri” içermeleridir. Bir ya da daha fazla fonksiyonun türevlerini içeren denklemlere diferansiyel denklem diyoruz. Diğer bir ifadeyle diferansiyel denklem bir takım fonksiyonlar ile bunların türevleri arasındaki ilişkiyi temsil eder. Çoğu bilimsel problemlerin tarif edilmesi bazı anahtar değişkenlerin diğer değişkenlere göre olan değişimlerini içerir. Genellikle bu değişkenlerdeki çok küçük değişimlerin dikkate alınması daha genel ve hassas bir tanımlama sağlar.

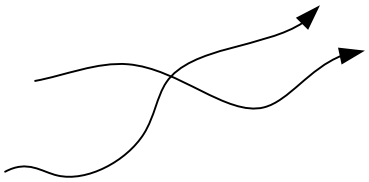
Grafiksel olarak faz düzlemi:




Şekil 4. 1.

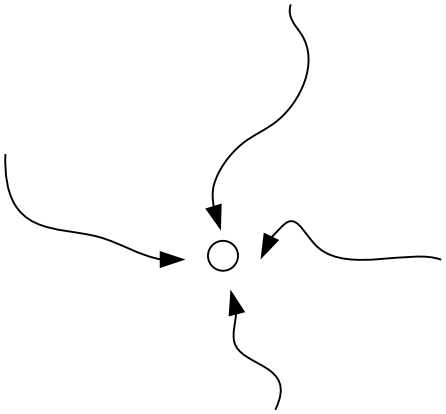
(x,y) noktasından çıkan bir vektörün yönü ( x,y)'den sonra gittiği noktanın diğer ucu olacaktır. Tabii o noktada da bir başka vektör olur, sonra o vektör takip edilir, bu süreç böyle devam eder. Sürekli vektörleri takip ederek bir gidiş yolu üzerindeki yörüngede hareket edilir.

Eğer çözümler özgün ise, gidiş yolları birbiriyle kesişemezler. Yani faz portrelerini çizerken şöyle bir şekil görmemiz mümkün değildir.



Şekil 4. 2.

Bu mümkünsüzlüğün sebebini anlamak için herhangi bir hayali kesişme noktasında isek bir sonraki gidişatın ne olacağını düşünmek yeterli. Öyle bir durumda kesişme noktasından sonra iki tane farklı gidiş yönü olurdu, ve bu iki seçenek varlığı çözümlerin özgünlüğü prensibine aykırıdır.
Diğer yandan gidiş yolları sabit noktalarda birbirine yaklaşabilir, alttaki gibi bir şekil mümkündür,

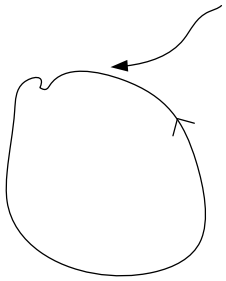


Şekil 4. 3.

ve sabit bir noktada kalır, o noktaya doğru işaret eden geliş yolları sabit noktaya dokunmaz aslında, sadece o noktaya yaklaşır.

İki boyutlu durumda sabit noktaların oluşmasının şartı bir x,y noktası için hem $\dot{x}$=0, hem de $\dot{y}$=0 olmasıdır. Ya da vektör notasyonu ile f($x^*$)=0. Eşitliğin sağındakine dikkat, bu bir "sıfır vektörü", içinde sadece sıfır değerleri var.

Gidiş yollarının kesişmeme şartının topolojik sonuçlarına bir bakalım. Diyelim ki elimizde kapalı bir yörünge (closed orbit) var, $R^2$ için olsun,



Şekil

Bir noktadan başlanır, gidiş yolu gide gide tekrar kendisine döner. Bu tür bir yol diferansiyel denklemlerin dönemsel bir çözümü olduğunu gösterir. Şimdi dışarıdan döngüye yaklaşan bir gidiş yolu düşünelim (üstteki resimde yine), yol döngünün içine "giremez"; o zaman özgünlük prensibi ihlal edilmiş olur. Hadi birleşim olduğunu varsayalım, ve birleşim sonrası bir noktadan zamanı geri saralım; birleşme anında zaman hangi gidiş yoluna sapacak? Bu bir çelişki ortaya çıkartır. Yani bir döngünün varlığı iki boyutta diğer gidiş yolları için ciddi bazı kısıtlamalara sebep oluyor.

$R^3$'te gidiş yolları birleşebilir, çünkü üç boyutta bir döngü uzayda bir ayrım yaratamaz, ve bu durum bazı şartlarda kaos ortaya çıkartır. İki boyutta kaos olması mümkün değildir. Hatırlarsak, tek boyutta yapılabilecekler çok basitti, sabit noktaya, ya da sonsuzluğa doğru

akıyor her şey. İki boyutta sabit noktaya, ya da dönemsel bir yörüngeye gidilebilir.. farklı biraz daha egzotik şeyler de mümkün, fakat kaos mümkün değil. Üç boyutta kaos olabilir, zaten üç boyut kaosun ilk ortaya çıkabileceği yer.

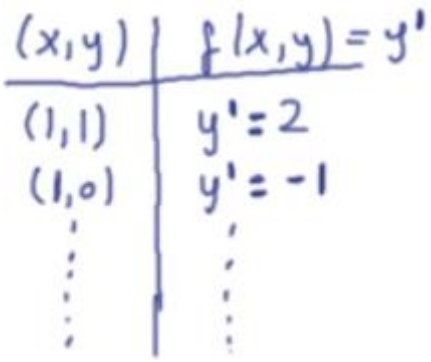
Bir x'=f(x) sistemi için faz portresini (grafiğini) çizmek, bunu niteliksel olarak farklı olan tüm gidiş yollarının göstererek yapmak, sabit noktaları, stabilitesi, ve muhtemel kapalı yörüngeler hakkında yine niteliksel bilgi toplamak. Bu işlem iki boyutta sabit nokta etrafında lineerizasyon kullanarak yapılır. Tek boyutta sistemi olduğu gibi analiz etmek kolaydı fakat iki boyutta bu biraz daha zor.

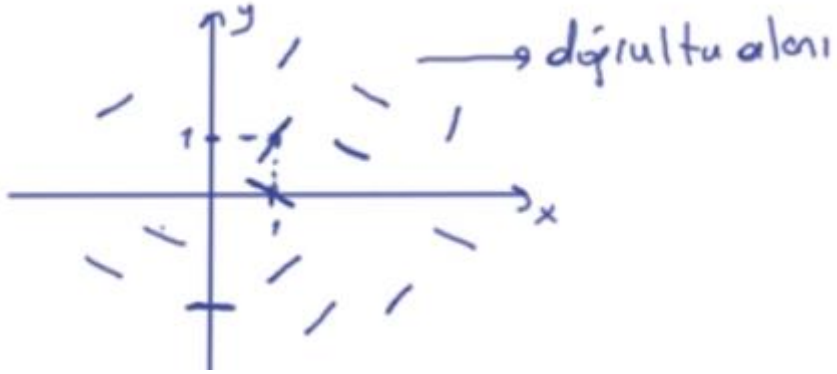
Lineer veya lineer olmayan tipten, $\ddot{x} = f(t, x, \dot{x})$ denklemler teorisinde x(t) çoğu kez x-ekseninde hareket eden bir noktanın t anındaki yerini ve y(t)=$\dot{x}(t)$ de t anındaki hızını tanımlar. (x(t),y(t)) ikilisi, birlikte, sistemin t anındaki durumunu belirler.

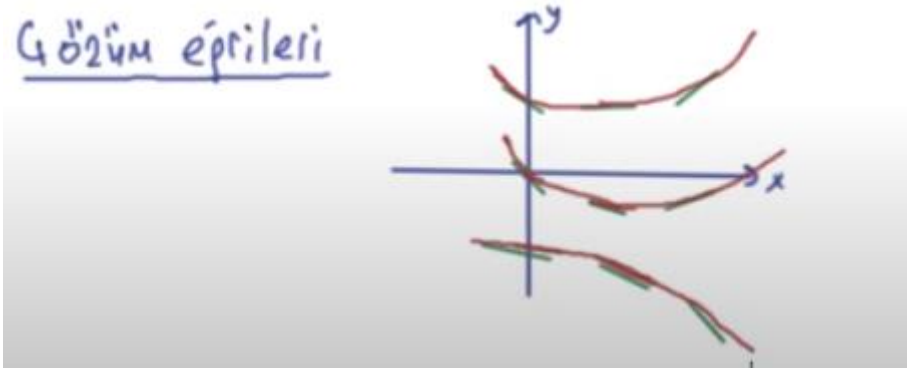
Sistemin davranışı, (x,y)-düzleminde (x(t),y(t)) noktasının geometrik yeri ile tarif edilebilir. Bu biçimde diferansiyel denklem ile ilişkilendirilen (x,y)-düzlemi faz düzlemi olarak adlandırılır. (x(t),y(t)) parametrik çözüm eğrisine yörünge ve onun görüntüsüne de orbit veya iz denir. Bir yörünge ile orbit arasındaki fark, yörüngenin çözüm eğrisinin oryantasyonunu veren t parametresi ile donatılmış olmasıdır.

İkinci mertebeden lineer diferansiyel denklemlerin veya daha genel olarak iki boyutlu lineer diferansiyel denklemler sisteminin kalitatif davranışını faz düzlemindeki yörüngeleri çizerek inceleyeceğiz.

Bir diferansiyel denklem y'=f(x,y) biçiminde yazılabiliyorsa, bu eşitlik doğrultu alanı oluşturmak için yazılabilir.






Diferansiyel denklemler farklı görünümler içerebilir. Genel çözüm, $y = ce^{x^2}$ dir. Burada c, integral sabitidir. Her c değerine karşın farklı bir çözüm eğrisi bulunur. Bu eğerilerin herbirisi diferansiyel denklemini sağlar. Diferansiyel denklemlerin çözümü olarak sonsuz fonksiyon bulunur. Sonsuz farklı fonksiyon sonsuz çözüm eğrisini temsil eder. Belirli bir c değeri için ise belirli bir eğriyi temsil eder. Belirli bir başlangıç koşulu için belli bir c değeri bulunduğunda, bu çözüme bağlangıç koşulunu sağlayan özel çözüm elde edilmiş olur. O da çözüm eğrilerinden bir tanesine karşılık gelir. Çözüm eğrileri hiçbir zaman birbirini kesmez. Kesmesi saçma bir durum olur; çünkü o zaman bir nokta için iki ayrı çözüm eğrisi olmuş olurdu. Anlamsızlık olur. Her bir çözüm eğrisi, genel çözümdeki sabitin bir değerine karşılık gelir. Özel çözüm, ifade eden çözüm eğrisi ise balangıç değerlerini sağlayan integral sabitinin bulunmasıyla ayırt edilir.

Örnek:

$y' = x + 2y$

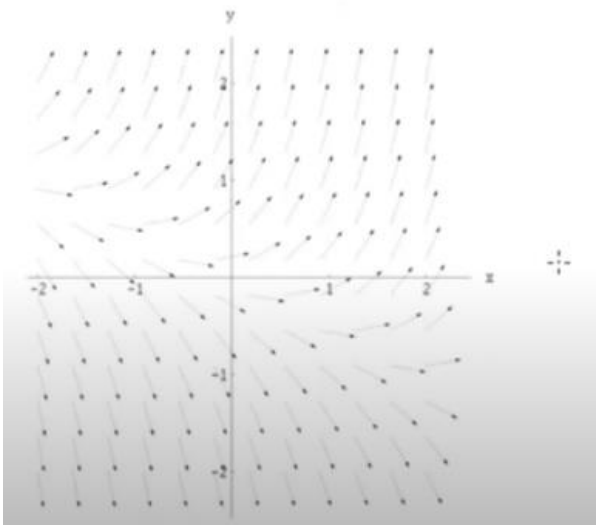
a) Diferansiyel denklemin doğrultu alanını çiziniz

b) Çözüm eğrilerini çiziniz

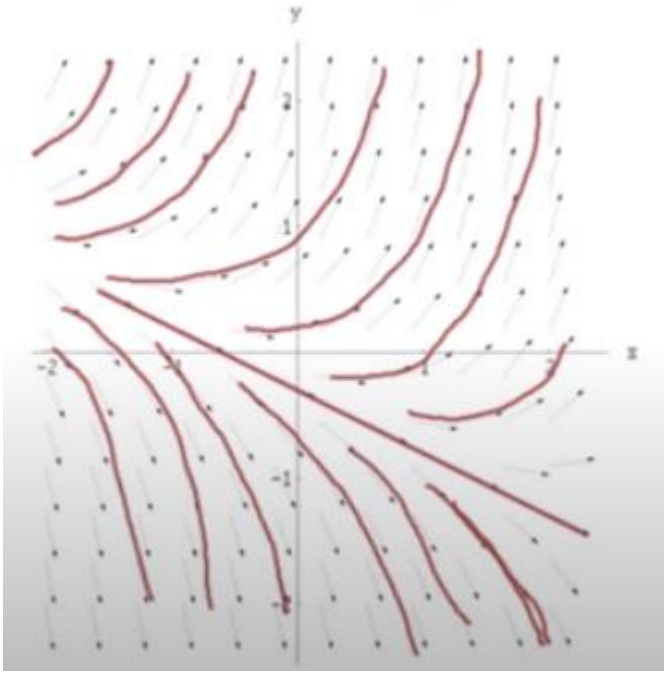
c) $y = \frac{1}{2}x - \frac{1}{4} + ce^{2x}$ ifadesinin genel çözüm olduğunu doğrulayınız.

d) Çözüm eğrilerinin hangi c değerlerine karşılık geldiğini yorumlayınız.

| (x,y) | y' |
|---|---|
| (0,0) | 0 |
| (1,0) | 1 |
| (0,1) | 2 |
| (1,1) | 3 |
| (-1,0) | -1 |
| (-1,1) | 1 |
| (-2,0) | 2 |
| (-2,1) | 0 |
| (-2,-1) | -4 |
| (-1,-1) | -3 |
| (0,-1) | -2 |
| (1,-1) | -1 |
| (2,-1) | 0 |
| (2,0) | 2 |
| (2,1) | 4 |




b)

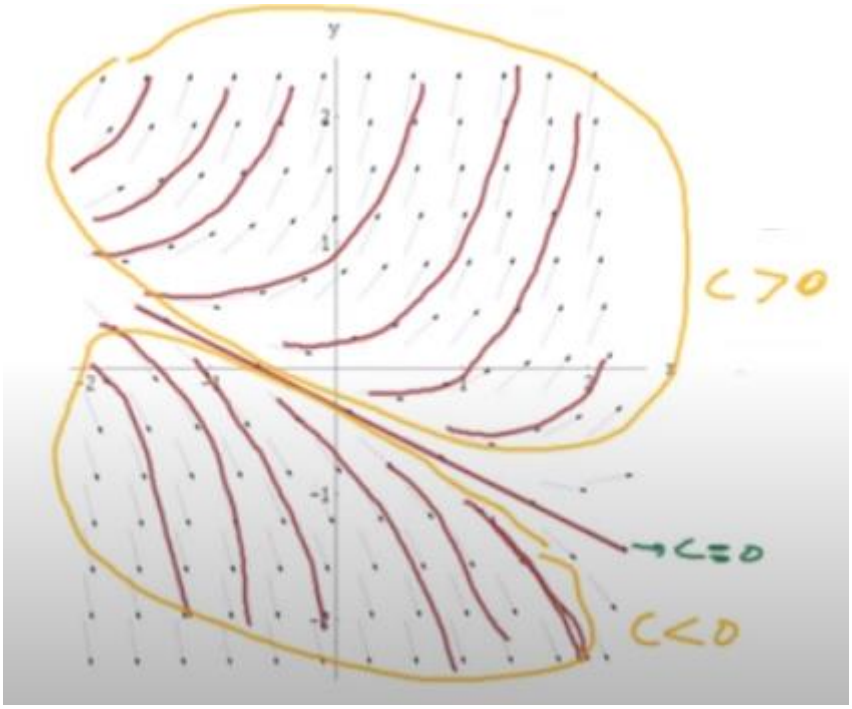



c)

$y = \frac{1}{2}x - \frac{1}{4} + ce^{2x}$ ifadesinin türevi alınır, $y' = x + 2y$ ifadesinde yerine konursa

$-\frac{1}{2} + 2ce^{2x} = x + (-x - \frac{1}{2} + 2ce^{2x})$ ifadesinde görüleceği üzere eşitlik sağlandı, dolayısıyla bu bir genel çözümdür.

d) $y = \frac{1}{2}x - \frac{1}{4} + ce^{2x}$ ifadesinde c=0 olduğunda $y = \frac{1}{2}x - \frac{1}{4}$ bir doğrusal denklem olduğu görülmektedir. c>0 olduğunda yukarı doğru olan eğrileri, c<0 olduğunda aşağı doğru eğrileri temsil etmektedir.




Çözüm eğrileri hiçbir zaman kesişemez.

## 3.6. İkinci mertebeden diferansiyel denklemlerin faz düzlemleri

Genel bir biçimde yazılabilen düzlem otonom linner denklem sistemini göz önüne alalım.

$$\frac{dx}{dt} = ax + by$$

$$\frac{dy}{dt} = cx + dy$$

burada x (t) ve y (t), t zamanının bilinmeyen fonksiyonlarıdır ve a, b, c, d sabitlerdir.

Örnek:
x‘ = 5x-y
y’=3x+y

Özdeğerlerin buluması,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 5 & -1 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Det($\begin{bmatrix} 5-\lambda & -1 \\ 3 & 1-\lambda \end{bmatrix}$)=0, $\lambda^2$ - 6λ + 8=0, λ1=2, λ2=4
λ1=2 için Özvektörlerin bulunması,

$$\begin{bmatrix} 5 & -1 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = 2 \begin{bmatrix} u \\ v \end{bmatrix}$$

3u=v, u=1 olursa v=3 olur. Özvektör: $\begin{bmatrix} 1 \\ 3 \end{bmatrix}$
λ1=4 için Özvektörlerin bulunması,

$$\begin{bmatrix} 5 & -1 \\ 3 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = 4 \begin{bmatrix} u \\ v \end{bmatrix}$$

u=v, u=1 olursa v=1 olur. Özvektör: $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$
Özdeğerler gerçek ve pozitiftir, dolayısıyla kritik nokta kararsız bir düğümdür.

$$m = \frac{y'}{x'}$$

$$\theta = \arctan(m)$$

Burada y’ ve x’ nin işaretleri yönün bölgesini , $\theta$ açısı ise dikliğini vermektedir.

İkinci mertebeden lineer diferansiyel denklemlerin veya daha genel olarak iki boyutlu lineer diferansiyel denklemler sisteminin kalitatif davranışı faz düzlemindeki yörüngeleri çizilerek incelenmektedir.

En basit durumda, p,q sabitler olmak üzere $u'' + pu' + qu = 0$ denklemi için yazılır. Buradan (u,v)

$$\begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -p & -q \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

denkleminin çözümü olur.

$\dot{x}$ = ax + by
$\dot{y}$ = cx + dy

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

A=$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$

Det(λI-A)=0,

$$\begin{vmatrix} a-\lambda & b \\ c & d-\lambda \end{vmatrix} = 0$$

$$\lambda^2 - (\mathrm{a}+\mathrm{d})\lambda + \mathrm{ad} - \mathrm{bc} = 0$$

$$\lambda^2 - \tau\lambda + \Delta$$

τ matrisin izidir (trace), yani A matrisinin köşegenindeki öğelerin toplamıdır. Δ ise A matrisin determinantıdır.

τ=traceA=a+d
Δ=detA=ad−bc
İkinci dereceden kök bulma formülü kullanarak λ'ları bulabiliriz,

$$\lambda_{1,2} = \frac{\tau \pm \sqrt{\tau^2 - 4\Delta}}{2}$$

$\tau = \lambda_1 + \lambda_2$
$\Delta = \lambda_1 \lambda_2$

Sabit katsayılı, homojen lineer sistemlerde, A bir sabit katsayılar matrisi, ve sayılar reel. Bu tür sistemlerde orijin her zaman bir sabit noktadır. Faz portresi A'nin özdeğerleri ve

özvektörleri üzerinden tanımlıdır, ki özdeğer / vektörler sayesinde lineer diferansiyel denklemleri çözülebilir.

Orjin (x(t),y(t))$\equiv 0$ her zaman yukarıdaki denklemin bir çözümüdür ve kritik nokta, sabit çözüm veya denge noktası olarak adlandırılır. Eğer A matrisi tekil değilse, yani $|A| \neq 0$ ise (0,0) yukarıdaki denklemin kritik noktasıdır. $|A| = 0$ durumu dejenere durum olarak adlandırılır.

**Örnek:**
$\ddot{x} + 6\dot{x} + 5x = 0$, denklemin faz düzlemi analizini yapınız.

$\dot{x}$ = y
$\dot{y}$ = -5x -6y

$$\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix} = \begin{pmatrix}0 & 1\\ -5 & -6\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix}$$

A=$\begin{pmatrix}0 & 1\\ -5 & -6\end{pmatrix}$

Det(λI-A)=0,

$$\begin{vmatrix}-\lambda & 1\\ -5 & -6-\lambda\end{vmatrix} = 0$$

$\lambda^2 + 6\lambda + 5 = 0$
(λ + 5)(λ + 1)=0

Özdeğerler: $\lambda_1 = -5$ , $\lambda_2 = -1$ olarak bulunur.

$\lambda^2 - \tau\lambda + \Delta = 0$
$\tau = \lambda_1 + \lambda_2 = -6$
$\Delta = \lambda_1 \lambda_2 = 5$

Özvektörlerin bulunması:
$(A - \lambda_i I)v = 0$ , i=1,2, ...

$\lambda_1$=-5 değeri için özvektörün bulunması:
$\begin{pmatrix}0+5 & 1\\ -5 & -6+5\end{pmatrix}\begin{pmatrix}v_1\\ v_2\end{pmatrix} = \begin{pmatrix}0\\ 0\end{pmatrix}$

$5v_1 + v_2 = 0$
$-5v_1 - v_2 = 0$

Bu denklemler birbirinin katı olacağı için çözüm için bir denklemi almamız yeterli olacaktır. Her zaman ikinci vektör birimi olan $v_2$'ye bir değer atanarak $v_1$ bulunur.

$v_2 = 1$ ise $v_1$ =-1/5

$\lambda_1$=-5 değeri için özvektör, $\begin{pmatrix} 1 \\ -1/5 \end{pmatrix}$

$\lambda_1$ negatif olduğunda kararlı noktaya yaklaşan bir özellik gösterir. Lineer denklem değerini bulmak için, $v_2 = y$ ve $v_1$ =x değeri konduğunda y=-3x elde edilir.

$\lambda_2$=-1 değeri için özvektörün bulunması:

$$\begin{pmatrix} 0+1 & 1 \\ -5 & -6+1 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$v_1 + v_2 = 0$

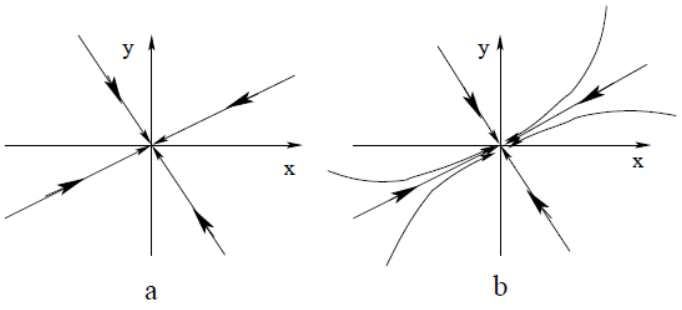
$-5v_1 - 5v_2 = 0$

Bu denklemler birbirinin katı olacağı için çözüm için bir denklemi almamız yeterli olacaktır. Her zaman ikinci vektör birimi olan $v_2$'ye bir değer atarak $v_1$ bulunur.

$v_2 = 1$ ise $v_1$ =-1

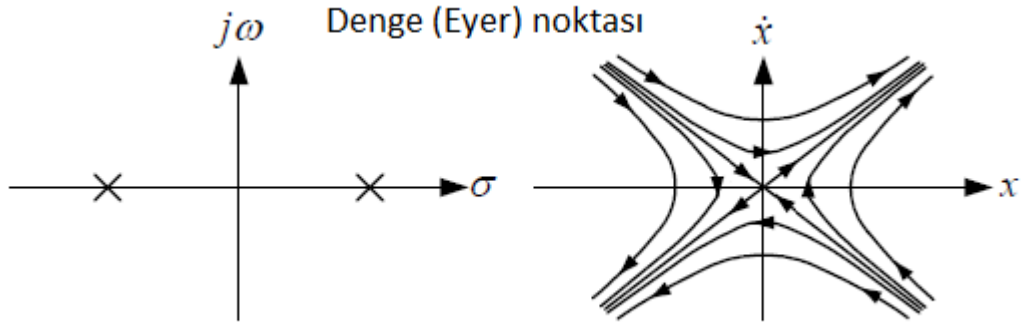
$\lambda_2$=-1 değeri için özvektör, $\begin{pmatrix} -1 \\ 1 \end{pmatrix}$

Lineer denklem değerini bulmak için, $v_2 = y$ ve $v_1$ =x değeri konduğunda y=-x elde edilir.




Şekil 4.9.

**λ1, λ2 gerçek ve zıt işaretli:**




Şekil 4.10.

Bir çizgi üzerinde ve λ1>0 ise, orijinden büyüyerek uzaklaşacak şekilde hareket eder. Fakat diğer "özdeğer" için $\lambda_2$<0 ise üstel bir azalma var demektir. Olanlar denge(eğer) noktası tanımına tam uygun, bir sabit noktaya hem giden hem de ondan kaçan gidiş yolları vardır. Kıvrımlı gidiş yörüngeleri doldurulurken negatif ve pozitif sonsuzluktan geliş ve gidiş özdeğerler göz önüne alınır.

**Örnek:**

$\dot{x}$ =4x -3y

$\dot{y}$ = 6x -7y

Çözüm öngörüsü: $e^{\lambda t}\vec{v}$ ya da $te^{\lambda t}\vec{v}$ dir. Burada $\vec{v}$, özvektördür.

Kritik noktalar: $\dot{x}$ =0 , $\dot{y}$ =0 dır.

$$\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix} = \begin{pmatrix}4 & -3\\ 6 & -7\end{pmatrix}\begin{pmatrix}x\\ y\end{pmatrix}$$

$$A=\begin{pmatrix}4 & -3\\ 6 & -7\end{pmatrix}$$

Det(λI-A)=0,

$$\begin{vmatrix}4-\lambda & -3\\ 6 & -7-\lambda\end{vmatrix} = 0$$

$$\lambda^2 + 3\lambda - 10 = 0$$

(λ + 5)(λ -2)=0

Özdeğerler: $\lambda_1 = -5$ , $\lambda_2 = 2$ olarak bulunur.

Özdeğer, $\lambda_1$<0 için üstel bir azalma söz konusudur. $\lambda_2$>0 olduğundan orijinden büyüyerek uzaklaşacak şekilde hareket eder. Olanlar denge(eyer) noktası tanımına tam uygundur.

Özvektörlerin bulunması:
$(A - \lambda_i I)v = 0$ , i=1,2, ...

$\lambda_1$=-5 değeri için özvektörün bulunması:

A=$\begin{pmatrix} 4 & -3 \\ 6 & -7 \end{pmatrix}$

$\begin{pmatrix} 4+5 & -3 \\ 6 & -7+5 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

$9v_1 - 3v_2 = 0$
$6v_1 - 2v_2 = 0$
Bu denklemler birbirinin katı olacağı için çözüm için bir denklemi almamız yeterli olacaktır. Her zaman ikinci vektör birimi olan $v_2$'ye bir değer atarak $v_1$ bulunur.

$v_2 = 1$ ise $v_1 =$-1/3

$\lambda_1$=-5 değeri için özvektör, $\begin{pmatrix} 1 \\ -1/3 \end{pmatrix}$

$\lambda_1$ negatif olduğundan kararlı noktaya yaklaşan bir özellik gösterir. Lineer denklem değerini bulmak için, $v_2 = y$ ise $v_1 =$x değeri konduğunda y=-3x elde edilir.

$\lambda_2$=2 değeri için özvektörün bulunması:
$\begin{pmatrix} 4-2 & -3 \\ 6 & -7-2 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

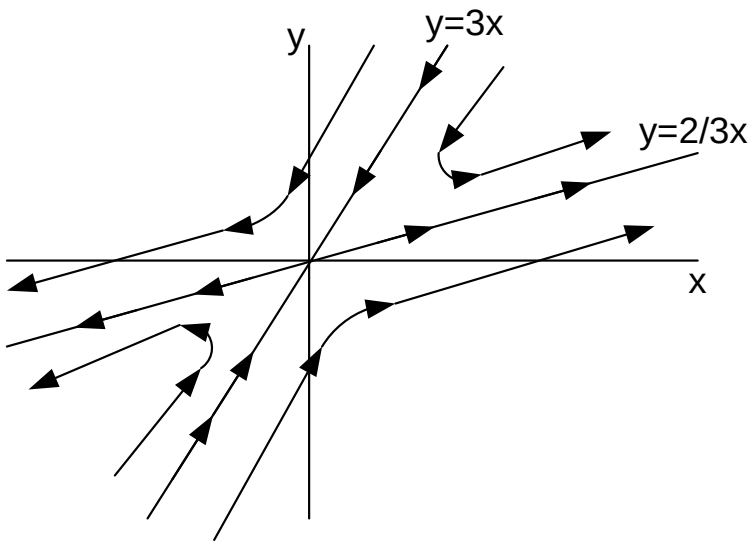
$2v_1 - 3v_2 = 0$
$6v_1 - 9v_2 = 0$

Bu denklemler birbirinin katı olacağı için çözüm için bir denklemi almamız yeterli olacaktır. Her zaman ikinci vektör birimi olan $v_2$'ye bir değer atarak $v_1$ bulunur.

$v_2 = 1$ ise $v_1 =$3/2

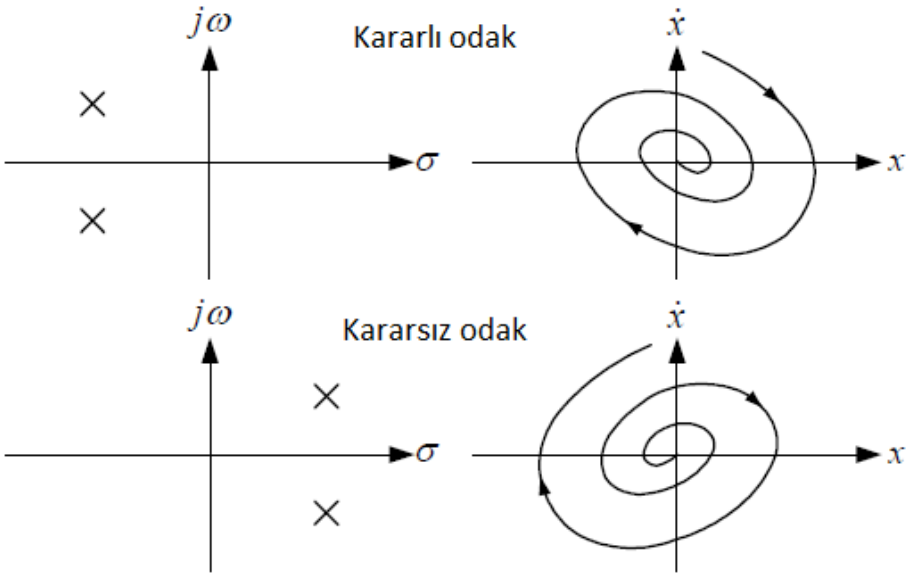
$\lambda_2$=2 değeri için özvektör, $\begin{pmatrix} 1 \\ 3/2 \end{pmatrix}$

$\lambda_2$ pozitif olduğunda kararlı noktadan uzaklaşan bir özellik gösterir. Lineer denklem değerini bulmak için, $v_2 = y$ ise $v_1 =$x değeri konduğunda $y = \frac{2}{3}x$ elde edilir.




Şekil 4.11.

**λ1, λ2 sıfır olmayan gerçek parçalara sahip kompleks eşleniklerdir.**




Şekil 4.12.

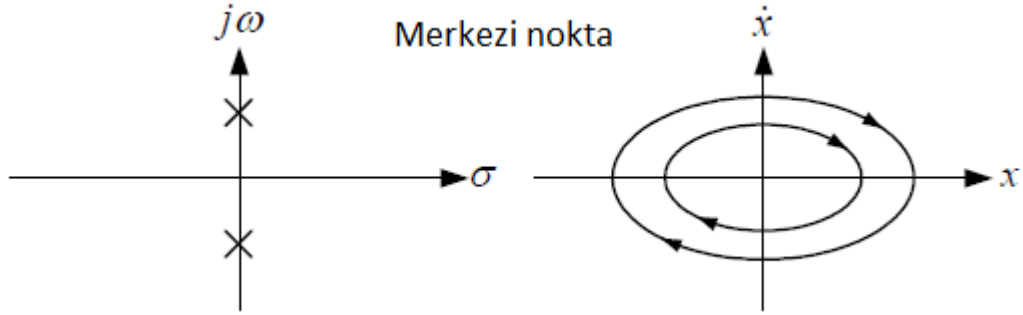
Bu durumda Δ>0, ve nagatif iz τ<0, onları çeken (attracting) hale getiren $\tau^2-4\Delta<0$ olur, o zaman kökler, λ'lar, kompleks, ve özdeğerler reel olmayacak. Faz düzleminde kompleks özvektörlerle ilgilenilmez, düzlemde sadece reel özvektörler görünür. Özdeğerler birbirinin kompleks eşleniği olarak tabii ki belli ve birbirinden ayrıdır (distinct), $\lambda=\sigma \pm i\omega$ formunda gösterelir.

$\sigma$,ω'nin fiziksel yorumlaması yapılabilir. $\sigma$<0 durumu azalma ya da artma oranını kontrol eder, ω ise sarmalın dönme oranını temsil eder. Eğer x(t) çözümleri yazılırsa, lineer cebirde x'in her bileşeni $e^{\sigma t}\cos\omega t$ ve $e^{\sigma t}\sin\omega t$ 'nin bir lineer kombinasyonudur. Sarmallar sönümlü titreşimlerin geometrik karşılığıdır.

Karalı ve karasız düğümlerin olduğu gibi kararlı ya da kararsız sarmallar da olabilir. Okların yönü değiştirilerek elde edilir. Aslında sarmalı daha eliptik bir şekilde çizilmesi gerekir, mesela üstten daha basık bir yuvarlaktır.

**λ1, λ2, gerçek parçaları 0'a eşit olan kompleks eşleniklerdir**

Sarmalların özel bir hali $\sigma$ =0 olduğu zamandır.




Şekil 4.13.

Bu durumda Δ>0, τ=0 ve λ=±iω, yani pür kompleks. Merkez halinde her gidiş yolu kapalı, tipik yukarıdaki resime benzer. Gidiş yollarının hepsi aynı yöne işaret eder. Merkez'in en iyi bilinen örneklerinden biri, sönümsüz basit harmonik titreşirlerin faz düzlemi çizilince bu şekil çıkar, x¨+x=0 sisteminin faz düzlemi bu yapıya benzer.

x˙=y

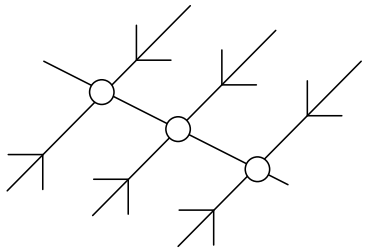
y˙=-x

İlk önce eksenlerden çıkan oklar çizilir, ardından birkaç tane diğer yerlerden ekleri çizilir. Gidiş yolları bir sabit noktaya yaklaşmaz. Her gidiş yolu kapalı, yani gidip gidip kendilerine dönerler. Orijinde bir sabit nokta var, fakat diğer tüm gidiş yolları onun etrafında bir kapalı yörüngedir.

**λ1= λ2, kökler gerçek ve birbirine eşit**

Eğer Δ=0 ise o zaman matrisin determinantı sıfır demektir, ve sabit nokta bulmak için Ax=0'i çözmeye uğraşırken bir özgün çözüm bulunamaz. Mümkün çözümü oluşturan tüm sabit noktalar bir çizgi, hatta bir düzlem oluşturacaklardır. Düzlem, yani o düzlemdeki her nokta sabit - bu çok sıkıcı bir dinamik sistemdir. x˙=0. Nerede olursan ol, hiç hareket edemiyorsun.

Bu sistemin analizi çok kolay (!). Çizgide sabit noktalar belki biraz daha ilginç, alttaki şekilde görüldüğü gibi bir durum olabilir.



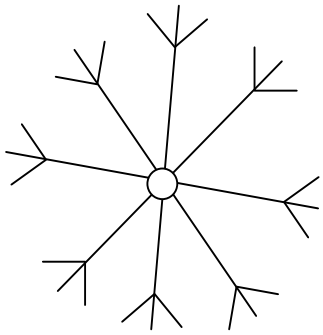
Şekil 4.14.

Çok yaygın olmasa da ortaya çıkabiliyor. Üstteki duruma 0'da izole olmayan sabit nokta durumu" ismi verilir, çünkü orijinin yakınında diğer sabit noktalar var.

$\boldsymbol{\tau^2}-4\Delta=0$ durumu:
Burada tekrarlanan kökler var, ve dejenere düğümler ortaya çıkabiliyor. Her yön bir özvektör oluyor, ve mesela tüm gidiş yolları düz bir şekilde tek bir sabit noktaya akıyorlar, bir yıldız şekli oluşturuyorlar. Ya da tek bir özyön var, mesela

$$\begin{bmatrix}\lambda & 1\\ 0 & \lambda\end{bmatrix}$$

matrisinin özvektör ya da özdeğerleri bunu verir. Kontrol edebilirsiniz, üstteki matrisin iki değil sadece bir tane özvektör vardır. Mekanik bağlamında bu durum kritik sönümlüdür (critically damped).



Şekil 4.15

## 3.7. Denge durum noktaları ve faz düzlemleri

Lineer veya lineer olmayan tipten $\ddot{x} = f(t, x, \dot{x})$, denklemler teorisinde x(t) çoğu kez x-ekseninde hareket eden bir noktanın t anındaki yerini ve y(t)= $\dot{x}(t)$ de t anındaki hızını tanımlar. (x(t), y(t)) ikilisi, birlikte, sistemin t anındaki durumunu belirler.

Sistemin davranışı, (x,y) -düzleminde (x(t), y(t)) noktasının geometrik yeri ile tarif edilebilir. Bu biçimde diferansiyel denklem ile ilişkilendirilen (x,y)-düzlemi faz düzlemi olarak adlandırılır. (x(t), y(t)) parametrik çözüm eğrisine yörünge ve onun görüntüsüne de orbit veya iz denir. Bir yörünge ile orbit arasındaki fark, yörüngenin çözüm eğrisinin oryantasyonunu veren t parametresi ile donatılmış olmasıdır.

$\ddot{x} = f(t, x, \dot{x})$, genel denklemi için, denge noktaları x ekseninde bulunur ve tüm f(x, 0) = 0 çözümleri tarafından tanımlanır. ( x , y ) ( y= $\dot{x}$ ) düzlemdeki faz yolları birinci mertebeden denklemin çözümleri yardımıyla belirlenir.

$$y(t) = \dot{x} = \frac{dx}{dt}$$

$$\ddot{x} = \frac{dy}{dt} = \frac{dy}{dx}\frac{dx}{dt} = y\frac{dy}{dx} = f(x, y)$$

$$\frac{dy}{dx} = \frac{f(x, y)}{y}$$

$$\frac{dy}{dx} = \frac{\ddot{x}}{\dot{x}}$$

x ve y eksenlerindeki ölçeklerin her zaman aynı olmadığı unutulmamalıdır. Genellikle eş zamanlı diferansiyel denklemler olarak muamele edilen f(x, y) çözümü olan (x (t), y (t)), t cinsinden parametrik olarak elde edilir.

**Örnek:**

$\ddot{x} - 8\dot{x}x = 0$ diferansiyel f(x, y) = 8xy olur. f(x, 0) = 0 olduğundan, x eksenindeki her nokta bir denge noktasıdır. Faz yolları için diferansiyel denklem,

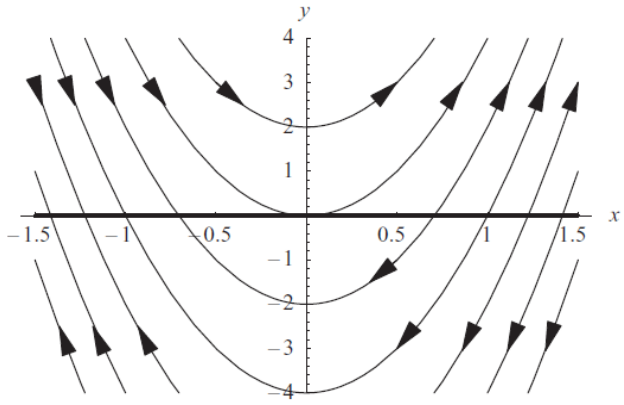
$\ddot{x} = 8\dot{x}x$

$\dot{x} = y$

$\dot{y} = 8xy$

$\frac{dy}{dx} = 8x$, genel çözüm, $y=4x^2 + C$ dir.

f(x, y) = 8xy denkleminin faz çözüm çizgileri aşağıdaki şekilde verilmiştir.

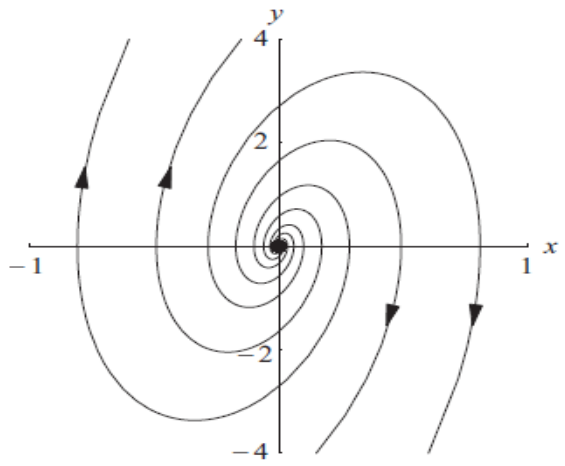



Şekil 3.

Matlab Yazılım kodu:

```
clear all
close all
x = linspace(-1.5,1.5,15);
y = linspace(-4,4,15);
[M1,M2]=size(y);
[X,Y] = meshgrid(x,y);
U = Y;
for i=1:M2
   for j=1:M2
  V(i,j)=-8*X(i,j)*Y(i,j);
   end
end
figure(1), quiver(X,Y,U,V,'r')
```

Örnek: $\ddot{x} - 4\dot{x} + 40x = 0$ diferansiyel f(x, y) = 4y-40x olur. f(x, 0) = 0 olduğundan, x eksenindeki her nokta bir denge noktasıdır. Faz yolları için diferansiyel denklem, f(x, y) =4y-40x denkleminin faz çözüm çizgileri aşağıdaki şekilde verilmiştir.



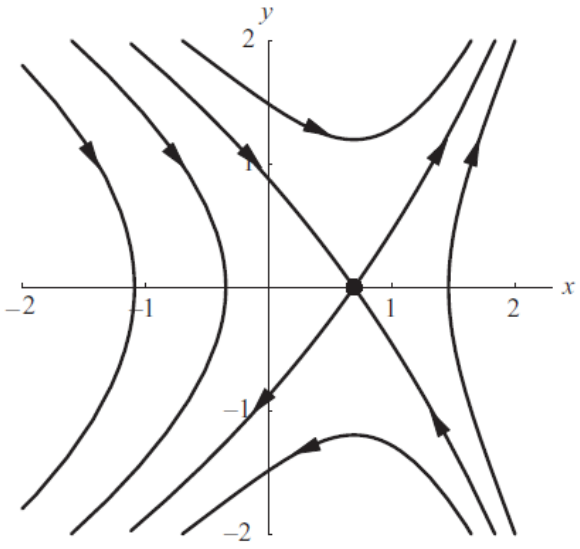

Şekil 3.2

Matlab Yazılım Kodu:

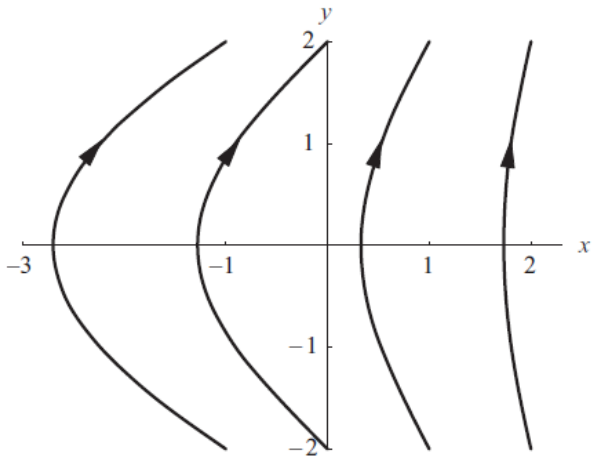
```
clear all
close all
x = linspace(-1,1,20);
y = linspace(-4,4,20);
[M1,M2]=size(y);
[X,Y] = meshgrid(x,y);
U = Y;
for i=1:M2
   for j=1:M2
   F(i,j)=-40*X(i,j)+4*Y(i,j);
   end
end
figure(1), quiver(X,Y,U,F,'r')
```

Örnek: $\ddot{x} - e^x - a = 0$ diferansiyel f(x, y) = $e^x + a$ olur.
f(x, y) = $e^x + a$ denkleminin a=-2, a=2 ve a=0 değerleri için faz çözüm çizgilerinin bulunması.

a<0 için tek denge noktası vardır.
a>0 için denge noktası yoktur.
a=0 için denge noktası yoktur.




Şekil: a<0 için faz diyagramı ve denge noktası




Şekil: a>0 için faz diyagramı ve denge noktası yoktur.

## 3.8. Doğrusal Olmayan Otonom Diferansiyel Denklem Sistemleri

Örnek:

$x'=10x-5xy$
$y'=3y+xy-3y^2$

To find the critical points, we need to find every ordered pair of real numbers (x, y) at which both x' and y' are zero. This means algebraically solving the system

$0= 10x - 5xy$
$0 = 3y + xy - 3y^2$

$5x(2-y)=0$ denkleminden $x=0$ ve $y=2$ elde edilir.

$x=0$ değeri ikinci denklemde yerine konulursa, $3y-3y^2=0$ elde edilir. Burdan $3y(1-y)=0$ denkleminin kökleri $y=0$ ve $y=1$ bulunur.

$y=2$ değeri ikinci denklemde yerine konulursa, $6+2x-12=0$ elde edilir. Burdan da $x=3$ bulunur.

Diferansiyel denklemin üç kritik noktası vardır, (0, 0), (0,1), (3,2).

## Lineer Olmayan Sistemlerin Kararlılığı:

Düzgün ODE'lerin tipik dengelerinin kararlılığı, Jacobian matrisinin özdeğerlerinin gerçek kısmının işaretiyle belirlenir. Bu özdeğerler genellikle 'dengenin özdeğerleri' olarak adlandırılır. Düzgün ODE'lerden oluşan bir sistemin Jacobian matrisi, durum değişkenlerine göre sağ tarafın kısmi türevlerinin matrisidir.

Özdeğerleri, dengenin doğrusal kararlılık özelliklerini belirler. Tüm özdeğerlerin negatif reel kısımları varsa, bir denge asimptotik olarak kararlıdır; en az bir özdeğerin pozitif reel kısmı varsa kararsızdır.

Jacobian matrisinin tüm özdeğerleri sıfır olmayan reel kısımlara sahipse dengenin hiperbolik olduğu söylenir.

### The Jacobian Matrix of a System

Associated with the regular system

$$x' = f(x, y)$$
$$y' = g(x, y)$$

is the *Jacobian matrix* of the system, also called the Jacobian matrix of $f$ and $g$ with respect to $x$ and $y$, or the Jacobian matrix of the vector-valued function $\mathbf{F} = [f, g]^{\mathsf{T}}$. This is the matrix-valued function of $x$ and $y$, normally denoted by either $\mathbf{J}$ or $\partial(f,g)/\partial(x,y)$, given by

$$\mathbf{J}(x, y) = \frac{\partial(f, g)}{\partial(x, y)} = \begin{bmatrix} \partial f/\partial x & \partial f/\partial y \\ \partial g/\partial x & \partial g/\partial y \end{bmatrix} .$$

x'=10x-5xy
y'=3y+xy-3y$^2$

f(x,y)= 10x-5xy
g(x,y)= 3y+xy-3y$^2$

$$J(x,y) = \begin{bmatrix} 10 - 5y & -5x \\ y & 3 + x - 6y \end{bmatrix}$$

$$J(0,0) = \begin{bmatrix} 10 & 0 \\ 0 & 3 \end{bmatrix}$$

$$J(0,1) = \begin{bmatrix} 5 & 0 \\ 1 & -3 \end{bmatrix}$$

$$J(3,2) = \begin{bmatrix} 0 & -15 \\ 2 & -6 \end{bmatrix}$$

Bir sistem düşünün

x’(t) = f(x; y);
y’(t) = g(x; y); olsun.

Burada f; g sürekli kısmi türevlerle ifade edilebilir ve her ikisi de (x0; y0) noktasında sıfır olmaktadır. J, bu noktada Jacobian matrisini göstersin, yani

$$J = \begin{bmatrix} f_x(x_0, y_0) & f_y(x_0, y_0) \\ g_x(x_0, y_0) & g_y(x_0, y_0) \end{bmatrix}$$

J'nin tüm özdeğerleri negatif reel kısma sahipse, (x0; y0) asimptotik olarak kararlıdır. Ve eğer J'nin özdeğerlerinden birinin reel kısmı pozitifse, o zaman (x0; y0) kararsızdır.

**Örnek:**
f(x,y)=x’(t) = 1-y
g(x,y)=y’(t) = $x^2 – y^2$

Bu durumda, kritik noktalar tatmin edilebilir.
1-y=0, y=1
$x^2 – y^2$=0, $x^2 = y^2$

Kritik noktalar A(-1,1), B(1,1) olarak elde edilir.

Kararlılık özelliklerini belirlemek için Jacobian matrisine bakılır.

$$J = \begin{bmatrix} f_x & f_y \\ g_x & g_y \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ 2x & -2y \end{bmatrix}$$

A(-1; 1) kritik noktasında, J(-1,1)=$\begin{bmatrix} 0 & -1 \\ -2 & -2 \end{bmatrix}$

bu matrisin özdeğerleri Det(J-λI)=0 dan bulunur. $\lambda^2 + 2\lambda - 2 = 0$, $\lambda = -1 \pm \sqrt{3}$, özdeğerlerden biri pozitif olduğu için A (-1,1) kritik noktasında kararsızdır.

B(1; 1) kritik noktasında, J(-1,1)=$\begin{bmatrix} 0 & -1 \\ 2 & -2 \end{bmatrix}$

bu matrisin özdeğerleri Det(J-λI)=0 dan bulunur. $\lambda^2 + 2\lambda + 2 = 0$, $\lambda = -1 \pm i$, özdeğerlerin reel kısımları negatif olduğuna B (1,1) kritik noktasında asimtotik kararlıdır.

Doğrusal olmayan düzenli otonom sistemin x’ = F(x) taslağı (kaba) faz portreleri var. İlk olarak, aşağıdaki prosedürle kritik noktalara yakın yörüngelerin davranışı belirlenir:

1) Sistem için J(x, y) Jacobian matrisini hesaplanır.
2) Tüm kritik noktalar bulunur.
3) Her kritik nokta için ($x_0$, $y_0$) :
   - A = J($x_0$, $y_0$) noktasındaki Jacobian matrisini değerlendirilir. Bu, ($x_0$, $y_0$)'da karşılık gelen lineer sistem için matristir.
   - A için özdeğerleri ve uygunsa özvektörleri bulunur.
   - Matris A'nın özdeğerleri ve uygunsa, özvektörleri kullanılarak, her bir kritik noktanın kararlılığı ve tipi belirlenir ve kritik noktanın yakınındaki bölgedeki yörüngeler çizilir. Yörüngelerin “hareket yönü” göstergelerinin eklendiğinden emin olunur.

## 3.9. Directional Derivatives and Gradient Vectors

**Eğim (Gradient)**

Gradients are extremely important in machine learning because this is how we optimize neural networks.

To understand gradients we need to return to partial derivatives, which we can reorganize into a vector as shown below: ddxandddyddxandddy

∇=→ddx,

dy∇=dx,

dy→

where ∇ is the gradient.

Let's look at an example - if we have the gradient for f(x,y) this is the same as writing the partial of ff with respect to xx and the partial with respect to y:

∇f(x,y)=→df(x,y)dx, df(x,y)dy

**Understanding the Gradient**

As you may have guessed, like the derivative, the gradient represents the slope of a function.

**The gradient points in the direction of the largest rate of increase of the function, and its magnitude is the slope in that direction.**

This is very valuable because we can determine the direction to change our variables that will have the greatest impact. In the context of machine learning we take the negative gradient direction to minimize our loss function.

**Gradient Vectors**

It's important to note that gradients are not restricted to two dimensions like we've seen, and are often expanded into higher dimensions. In the context of machine learning we rarely stop at 3 dimensions, so gradient vectors can be organized as partial derivatives for a scalar function. If there are multiple functions, we use the Jacobian, which is a vector of gradients.

Let's now look at convex optimization and see how we can use gradients to minimize the loss function in our machine learning algorithm.

**Örnek:**

```
[X,Y] = meshgrid(-2:0.25:2,-1:0.2:1);
Z = X.*exp(-X.^2 - Y.^2);
[U,V,W] = surfnorm(X,Y,Z);
```

Display the vectors as a 3-D quiver plot. Then, display the surface in the same axes. Adjust the display so that the vectors appear normal to the surface by calling axis equal.

```
quiver3(X,Y,Z,U,V,W)
hold on
surf(X,Y,Z)
axis equal
```

**Örnek:**

```
spacing = 0.2;
[X,Y] = meshgrid(-2:spacing:2);
Z = X.*exp(-X.^2 - Y.^2);
[DX,DY] = gradient(Z,spacing);
```

Display the gradient vectors as a quiver plot. Then, display contour lines in the same axes. Adjust the display so that the gradient vectors appear perpendicular to the contour lines by calling axis equal.

```
quiver(X,Y,DX,DY)
hold on
contour(X,Y,Z)
axis equal
hold off
```

Örnek:

```
[X,Y] = meshgrid(-pi:pi/8:pi,-pi:pi/8:pi);
U = sin(Y);
V = cos(X);
quiver(X,Y,U,V,'r')
```

**Gradients, Gradient Plots and Tangent Planes**

**Visualizing Gradients of Functions of Two Variables**

Birkaç değişkenli bir fonksiyonun gradyanı, bileşenleri fonksiyonun kısmi türevleri olan vektör değerli fonksiyondur.

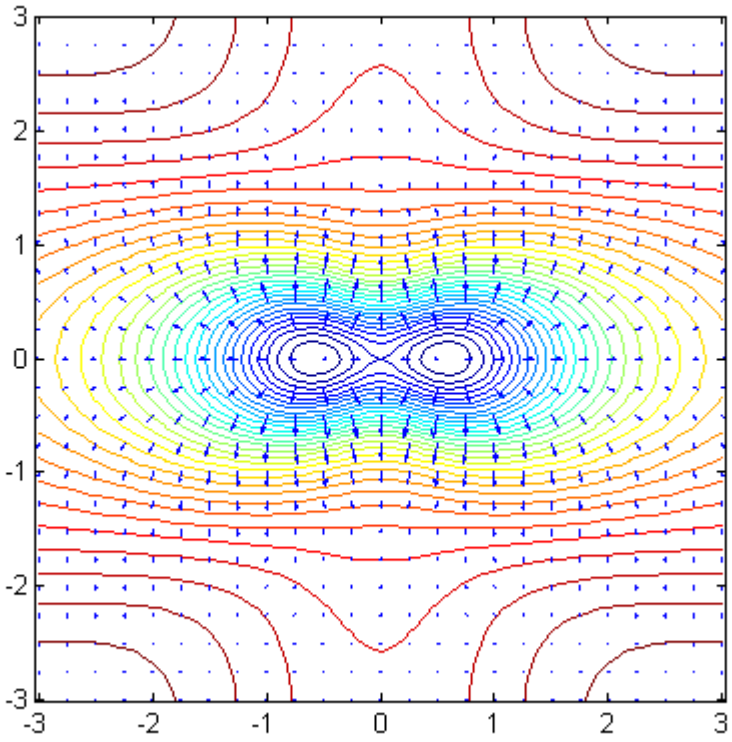
```
syms x y z
f=((x^2-1)+(y^2-4)+(x^2-1)*(y^2-4))/(x^2+y^2+1)^2
 f =((x^2 - 1)*(y^2 - 4) + x^2 + y^2 - 5)/(x^2 + y^2 + 1)^2
```

f'nin gradyanı, simgesel araç kutusundaki jacobian işlevi kullanılarak hesaplanabilir. Jacobian'ın ikinci argümanı olarak bileşenleri değişken olan bir vektöre ihtiyaç duyulduğuna dikkat edin. Bu arada, gradyan adında bir MATLAB komutu da vardır, ancak gradyan için sembolik bir form değil, gradyan için sayısal bir yaklaşım üretir.

```
gradf=jacobian(f,[x,y])
 gradf =
 [ (2*x + 2*x*(y^2 - 4))/(x^2 + y^2 + 1)^2 - (4*x*((x^2 - 1)*(y^2 - 4) + x^2 + y^2 - 5))/(x^2 + y^2 + 1)^3,
(2*y + 2*y*(x^2 - 1))/(x^2 + y^2 + 1)^2 - (4*y*((x^2 - 1)*(y^2 - 4) + x^2 + y^2 - 5))/(x^2 + y^2 + 1)^3]
```

Quiver kullanarak f'nin gradyanını çizebiliriz. Bunu f'nin kontur grafiği üzerine bindirmek özellikle ilginçtir. Hem titreme hem de kontur, önce meshgrid kullanılmasını gerektirir. İyi bir resim elde etmek için, genellikle kontur için oldukça yakın aralıklı bir ağa, ancak titreme için daha seyrek bir ağa ihtiyacınız olduğunu unutmayın.

```
[xx, yy] = meshgrid(-3:.1:3,-3:.1:3);
ffun = @(x,y) eval(vectorize(f));
fxfun = @(x,y) eval(vectorize(gradf(1)));
fyfun = @(x,y) eval(vectorize(gradf(2)));
contour(xx, yy, ffun(xx,yy), 30)
hold on
[xx, yy] = meshgrid(-3:.25:3,-3:.25:3);
quiver(xx, yy, fxfun(xx,yy), fyfun(xx,yy), 0.6)
axis equal tight, hold off
```

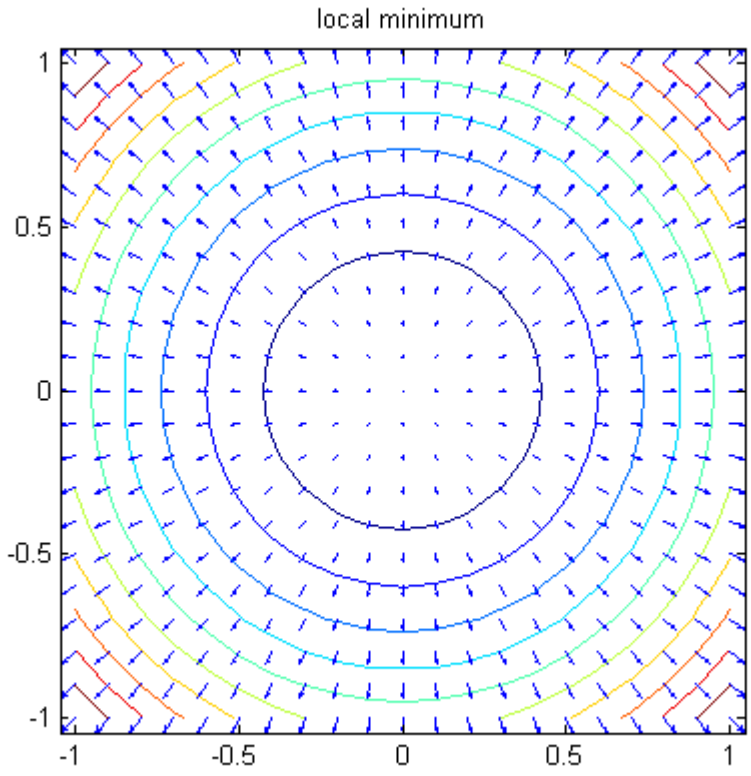



Gradyan grafiğindeki okların uzunlukları, hem adım boyutu hem de uzunluk parametresi olarak adlandıracağımız (isteğe bağlı) son sayısal parametre tarafından belirlenir. Burada olan şey, sayısal 'gradyan' işlevinin, işlevin ızgara üzerindeki bitişik noktalardaki değerlerini karşılaştırması ve okları son parametrenin fark katlarıyla orantılı hale getirmesidir. Yararlı bir gradyan grafiği elde etmek için birkaç farklı adım boyutu ve uzunluk parametresi seti ile deneme yapmak gerekli olabilir. Degrade okların her zaman seviye eğrilerine dik olduğuna dikkat edin. Bu her zaman böyledir.

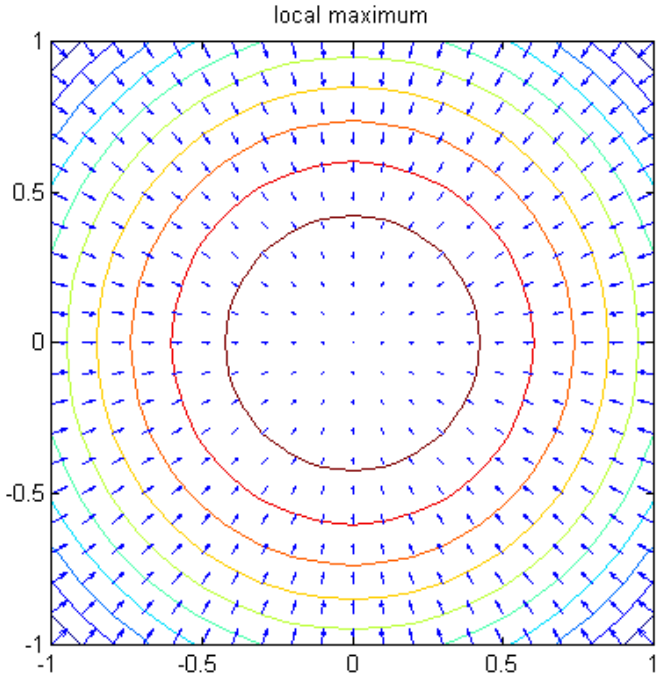
**Behavior Near Critical Points**

Bunun gibi bir çizim, fonksiyonun kritik noktaları hakkında bilgi vermek için yorumlanabilir. Kritik noktalar, gradyan vektörünün kaybolduğu noktalardır. Fonksiyonun kritik noktaya yakın davranışı ikinci türevler tarafından kontrol ediliyorsa (böylece 'ikinci türev testi' uygulanır) kritik bir nokta dejenere olarak adlandırılır. İki değişkenli fonksiyonlar için, üç tür dejenere olmayan kritik nokta vardır. Bunları aşağıdaki üç tür resimden tanıyabilirsiniz:

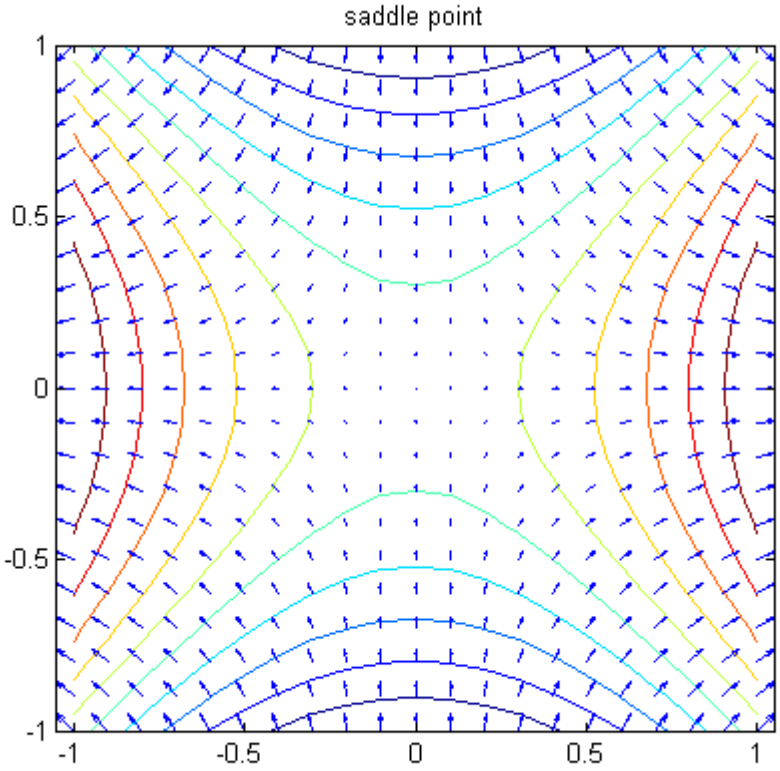
```
f1 = x^2 + y^2; gradf1 = jacobian(f1,[x,y]);
f1fun = @(x,y) eval(vectorize(f1));
f1xfun = @(x,y) eval(vectorize(gradf1(1)));
f1yfun = @(x,y) eval(vectorize(gradf1(2)));
[xx, yy] = meshgrid(-1:.1:1,-1:.1:1);
contour(xx, yy, f1fun(xx, yy), 10)
hold on
quiver(xx, yy, f1xfun(xx, yy), f1yfun(xx, yy), 0.5)
title('local minimum'), axis equal tight, hold off
set(gca, 'YTick', -1:.5:1)
```




```
contour(xx, yy, -f1fun(xx, yy), 10)
hold on
quiver(xx, yy, -f1xfun(xx, yy), -f1yfun(xx, yy), 0.5)
title('local maximum'), axis equal tight, hold off
set(gca, 'YTick', -1:.5:1)
```



```
f2 = x^2 - y^2; gradf2 = jacobian(f2,[x,y]);
f2fun = @(x,y) eval(vectorize(f2));
f2xfun = @(x,y) eval(vectorize(gradf2(1)));
f2yfun = @(x,y) eval(vectorize(gradf2(2)));
contour(xx, yy, f2fun(xx, yy), 10)
hold on
quiver(xx, yy, f2xfun(xx, yy), f2yfun(xx, yy), 0.5)
title('saddle point'), axis equal tight, hold off
set(gca, 'YTick', -1:.5:1)
```




f fonksiyonu durumunda, x ekseninde iki yerel minimum ve bir eyer noktası olduğunu görebiliriz. İki minimum, gradyan oklarının dışarıyı gösterdiği kapalı konturlarla çevrili oldukları gerçeğinden belirlenebilir. Eyer noktası, iki minimum etrafındaki kapalı kontur çiftlerinden oluşan konturlardan, her iki minimumu çevreleyen tek eğrilere geçişi işaretler. Bir eyer noktasından geçen seviye eğrisinin kendi kendine kesiştiğine dikkat edin. Simetri ile, eyer noktasının orijinde olduğu ve minimumların simetrik olarak etrafına yerleştirildiği sonucuna varmak güvenli görünüyor. Bir sonraki defterde bu örneği takip etmeye devam edeceğiz.

**Problem 1:**

- (a) Obtain a simultaneous contour and gradient plot of the function

$$g(x,y) = \frac{x^4 + 2x^3y - 6x^2y^2 + y^4}{x^4 + y^4 + 1},$$

defined in Problem 1 of the previous lesson.

- (b) Explain what information the plot conveys regarding the location and nature of the critical points of the function.

**Gradients of Functions of Three Variables, and Tangent Planes to Surfaces**

İki değişkenli bir fonksiyonun gradyanının her zaman fonksiyonun seviye eğrilerine normal olduğu gözleminin üç boyutlu benzeri, üç boyutlu bir fonksiyonun gradyanının her zaman fonksiyonun seviye yüzeylerine normal olmasıdır. Bundan, herhangi bir noktadaki fonksiyonun gradyanının, o noktadaki teğet düzleme o noktadan geçen düz yüzeye dik olduğu sonucu çıkar. Bu, seçilen bir noktada bir yüzeye teğet düzlemi çizmek için kullanılabilir. Yüzeyi çizelim

$$3z^3 - 2x^2 - y^2 = 0$$

together with its tangent plane at the point (2,4,2). We begin by checking that the indicated point satisfies the equation.

```
g = 3*z^3-2*x^2-y^2; subs(g,[x,y,z],[2,4,2])
ans =      0
```

Next we determine the gradient of g at (2,4,2).

```
gradg=jacobian(g, [x,y,z])
planenormal=subs(gradg, [x,y,z], [2,4,2])
```

```
gradg =  [ (-4)*x, (-2)*y, 9*z^2]
 planenormal =     -8    -8    36
```

Next we write the function that determines the plane.

```
realdot = @(u, v) u*transpose(v);
planeq = realdot(planenormal, [x-2,y-4,z-2])
```

```
planeq = 36*z - 8*y - 8*x - 24
```

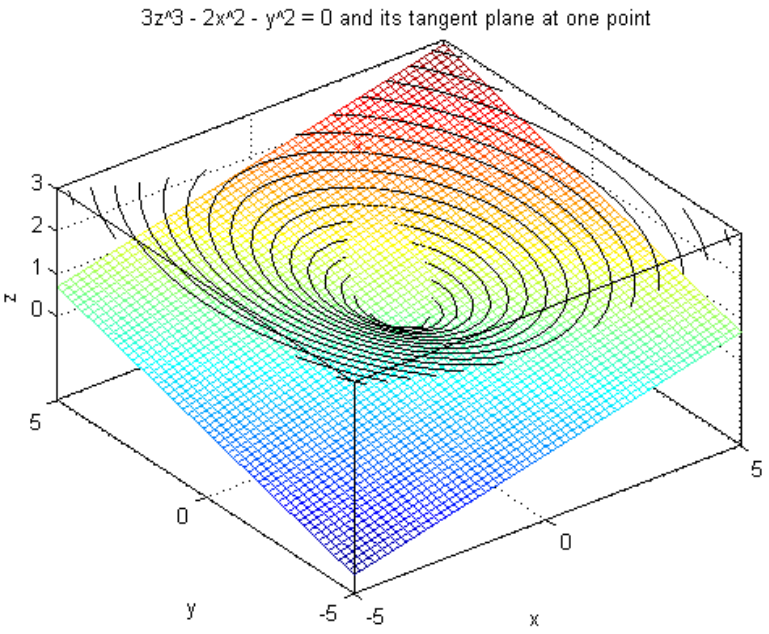
Then we solve for z in terms of x and y to make this easier to plot.

```
planefun=solve(planeq,z)
```

```
planefun = (2*x)/9 + (2*y)/9 + 2/3
```

Finally we plot the surface g = 0 and add a plot of the plane so we can see the tangency.

```
implicitplot3d(g, 0, -5, 5, -5, 5, 0, 3, 20), hold on
plot3(2, 4, 2, 'xr')
ezmesh(planefun, [-5, 5, -5, 5]), hold off
set(gca, 'YTick', -5:5:5, 'ZTick', 0:3)
title '3z^3 - 2x^2 - y^2 = 0 and its tangent plane at one point'
```




Note that there is another way we could have done this. We could have instead solved for z as a function of x and y in the equation g(x, y, z) = 0 and represented our surface as the graph of a function of two variables: z = h(x, y). Then we could have gotten the equation of the tangent plane from the gradient of h. Here are the details:

```
h=solve(g,z)
```

```
h =
                    (3^(2/3)*(2*x^2 + y^2)^(1/3))/3
 (3^(2/3)*(2*x^2 + y^2)^(1/3)*((3^(1/2)*i)/2 - 1/2))/3
-(3^(2/3)*(2*x^2 + y^2)^(1/3)*((3^(1/2)*i)/2 + 1/2))/3
```

Since h should be real, we want the first solution.

```
h=h(1)
```

```
 h = (3^(2/3)*(2*x^2 + y^2)^(1/3))/3
```

The equation of the tangent plane is then given by:

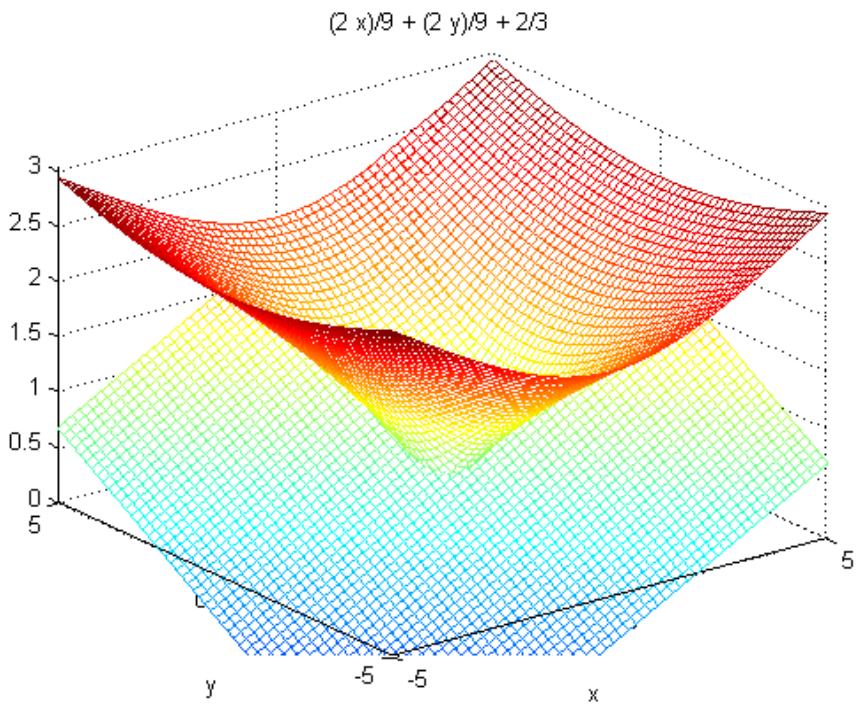
$z = \vec{\nabla} h(2,4) + h(2,4) \cdot (x-2, y-4).$

```
gradh=simplify(subs(jacobian(h,[x,y]), [x,y], [sym(2),sym(4)]))
planefun=simplify(subs(h,[x,y], [sym(2),sym(4)]))+realdot(gradh,[x-2,y-4])
```

```
gradh =  [ 2/9, 2/9]
planefun =  (2*x)/9 + (2*y)/9 + 2/3
```

This is the same answer as before. Again we can produce a plot:

```
ezmesh(h,[-5,5,-5,5]), hold on, ezmesh(planefun,[-5,5,-5,5]), hold off
```




**Problem 2:**

Obtain a plot showing the hyperboloid together with its tangent plane at the point (3,4,5).

**Additional Problems:**

- 1. Obtain simultaneous contour and gradient plots for the function

$f(x,y) = (x^2 + y^2)^2 - x^2 + y^2.$

What information does your plot give you regarding the location and classification of the critical points of in the region -1.5 < x, y < 1.5?

- 2. Obtain a plot showing the ellipsoid

$$\frac{x^2}{4} + \frac{y^2}{9} + z^2 = 3$$

together with its tangent plane at the point (2,3,1).

- 3. Find the equation of the tangent plane to the surface x y z = 8 at the point (-2, -2, 2) in two different ways: by thinking of the surface as a graph of a function of two variables, and also by thinking of the surface as a level surface of a function of three variables. Check that your two answers agree. Then plot the surface (in a vicinity of this point) along with the tangent plane, so that the tangency is visible (you need only do this once).

## 3.10. The Potansiyel fonksiyon

Konservatif bir sistemin potansiyel enerjisi *V(x)* süreklidir ve x <−1 için kesinlikle artmakta, |x| ≤1 için sıfır ve x> 1 için kesinlikle azalmaktadır. Denge noktalarını bulun ve sistemin faz diyagramını çizin.

*V(x)* potansiyeli olan bir sistem geçerli denkleme sahiptir.

$$\ddot{x} = -\frac{dV(x)}{dx}$$

$$V(x) = -\int f(x,y)dx$$

Denge noktaları $\ddot{x}$= 0 olduğunda veya dV(x)/dx = 0 olduğunda oluşur; bu, x eksenindeki tüm noktaların |x| ≤1 denge noktaları olduğu anlamına gelir. Ayrıca, faz diyagramları $\frac{1}{2}y^2 = V(x) + C$ yardımıyla verilir.

Bu potansiyel ifade, Schwarzian türevi ile kaotik davranış analizinde kullanılmaktadır.

quiver(X,Y,U,V) plots arrows with directional components U and V at the Cartesian coordinates specified by X and Y.

**The Potansiyel fonksiyon:**

$u = \dot{x}$
$v = \dot{y}$

x ve y değişkenlerinin ivmeleri, $\dot{u}\ ve\ \dot{v}$ dir.
$\dot{u} = \ddot{x}$
$\dot{v} = \ddot{y}$

Potansiyel bir fonksiyon bu ivmeleri verir.

$$F = -\nabla V(x,y) = \big(F_u(x,y), F_v(x,y)\big) = (\dot{u}, \dot{v})$$

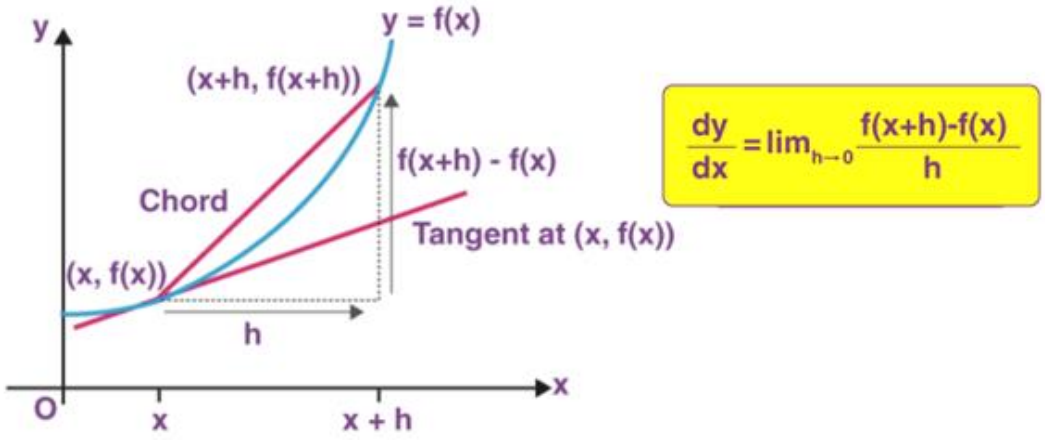
# 4. Fonksiyon Optimizasyonu

## 4.1. Veri biliminde cebir ve diferansiyel hesaplama

Türev, bir bağımlı değişkenin diğer bağımsız değişkene göre değişim oranını tanımlar. Makine öğreniminde türevler, fonksiyon optimizasyonu için önemlidir. Türevler, "Ne kadar hızlı?" "Ne kadar dik?" ve "Ne kadar hassas?" gibi değişim oranları ve yönü ile ilgili sorulara yanıt verir. Bu değişim oranları $\delta y / \delta x$ ile gösterilir, dolayısıyla bağımsız değişken $\delta x$'teki bir değişikliğe göre bağımlı değişken $\delta y$'deki bir değişikliği tanımlar. Bir araba saatte 100 kilometre hızla gidiyor dediğimizde sadece değişim hızını belirtmiş oluyoruz. Sıklıkla kullandığımız ortak terim hızın, farklı iki anlamı vardır. Önce ikisini birbirinden ayırmamız en iyisi olacaktır. Zamansal Hız (speed), bir nesnenin bir yol boyunca hareket ettiği zaman hızıdır, Hareketsel hız (Velocity) ise bir nesnenin hareketinin hızı ve yönüdür. Zamansal hız her zaman pozitiftir, oysa hareketsel hız bir yön kavramı getirir ve bu nedenle hem pozitif hem de negatif değerler gösterebilir.

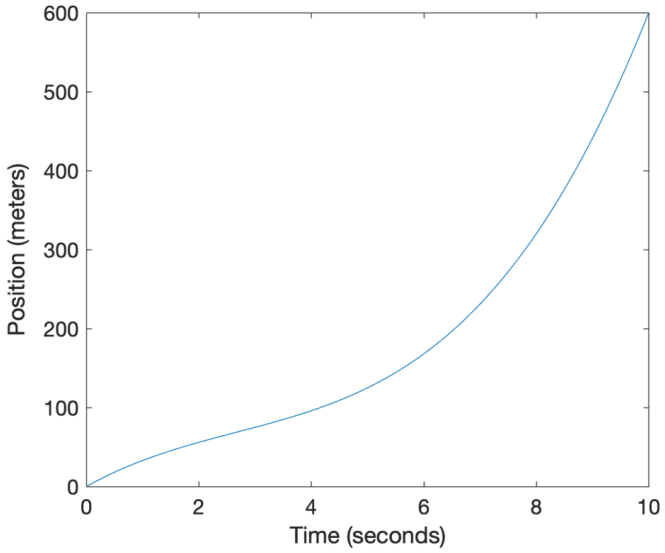
Hareketsel Hız = $\delta y / \delta t$

Hareketsel hızın arabanın konumundaki değişimi $\delta y$, $\delta t$ zaman aralığı içinde verdiği anlamına gelir. Başka bir deyişle, hareketsel hız, konumun zamana göre birinci türevidir.



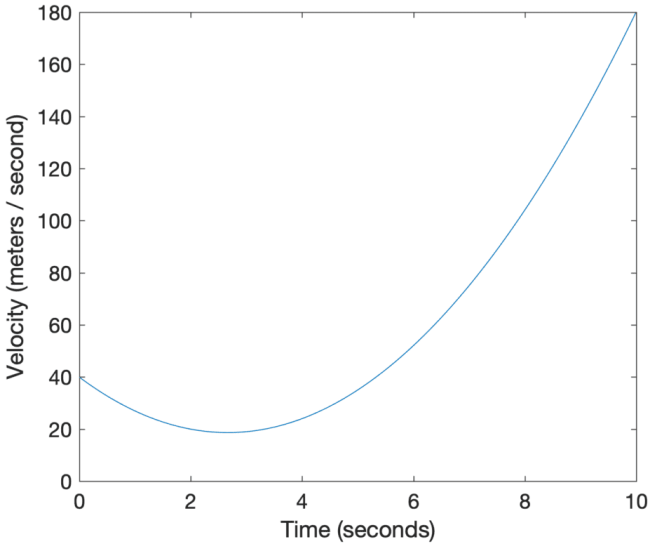

Arabanın hareketsel hızı, örneğin araba sürekli olarak saatte 100 kilometre hızla gidiyorsa sabit kalabilir veya zamanın bir fonksiyonu olarak da değişebilir. İkincisi durumunda, bu, hareketsel hız fonksiyonunun kendisinin zamanın bir fonksiyonu olarak değiştiği anlamına gelir veya daha basit bir ifadeyle, arabanın hızlandığı söylenebilir. İvme, hızın birinci türevi v ve konumun ikinci türevi y olarak zamana göre tanımlanır: ivme = $\delta v / \delta t = \delta^2 y / \delta t^2$

Daha iyi görselleştirmek için konum, hız ve ivme eğrilerini çizebiliriz. Arabanın konumunun zamanın bir fonksiyonu olarak y(t) = $t^3 - 8t^2 + 40t$ ile verildiğini varsayalım.




Arabanın zamana karşı konumunun çizgi grafiği

Türevini bulmak için kuvvet kuralını y(t)'ye uygulamak zorunda kalsaydık, hızın aşağıdaki fonksiyon tarafından tanımlandığını bulurduk: $v(t) = y'(t) = 3t^2 - 16t + 40$.
Bu, arabanın hız grafiğiyle gösterilir:



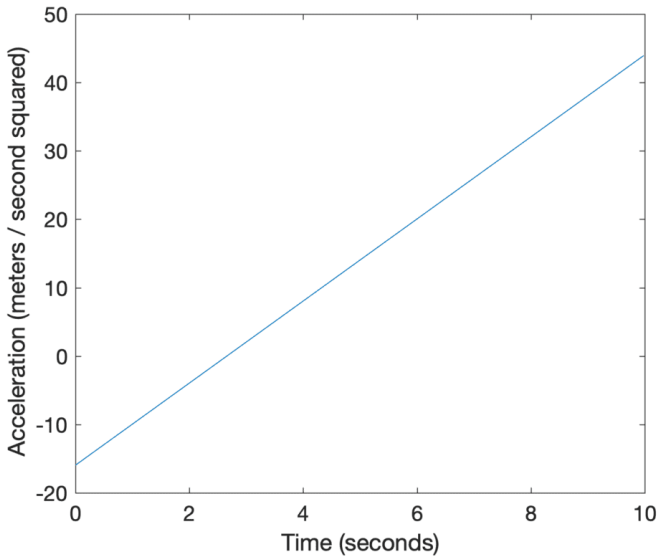

Arabanın zamana karşı hızının çizgi grafiği

Grafik, arabanın konumunun yolculuğun başında yavaşça değiştiğini, yaklaşık t = 2,7 saniyeye kadar hafifçe yavaşladığını, bu noktada değişim hızının arttığını ve yolculuğun sonuna kadar artmaya devam ettiğini gösteriyor.

Arabanın yolculuk boyunca pozitif bir hıza sahip olduğuna dikkat edin ve bunun nedeni asla yön değiştirmemesidir. Bu nedenle, kendimizi bu hareket halindeki arabada otururken hayal

etmemiz gerekseydi, hız göstergesi bize hız grafiğinde çizdiğimiz değerleri gösteriyor olurdu (çünkü hız baştan sona pozitif kalır, aksi takdirde hızı hesaplamak için hız grafiğinin mutlak değerini bulmamız gerekirdi).

Türevini bulmak için kuvvet kuralını v(t)'ye tekrar uygulamak zorunda kalsaydık, ivmenin aşağıdaki fonksiyon tarafından tanımlandığını bulurduk: $a(t) = v'(t) = 6t - 16$. İvme grafiğini de çizebiliriz:



Arabanın zamana karşı ivmesinin çizgi grafiği

Grafiğin artık t = [0, 2.7) saniye zaman aralığında negatif ivme ile karakterize edildiğini görüyoruz. Bunun nedeni, ivmenin hızın türevi olması ve bu zaman aralığında arabanın hızının azalmasıdır.

Tüm fonksiyonlar bir araya getirildiğinde aşağıdakiler yazılır:
$y(t) = t^3 - 8t^2 + 40t$
$v(t) = y'(t) = 3t^2 - 16t + 40$
$a(t) = v'(t) = 6t - 16$

t = 10s yerine koyarsak, yolculuğun sonunda arabanın 600 m yol aldığını, hızının 180 m/s olduğunu ve 44 $m/s^2$'de hızlandığını bulmak için bu üç işlevi kullanabiliriz. Tüm bu değerlerin az önce çizdiğimiz grafiklerle uyuştuğunu doğrulayabiliriz. Bu özel örneği, bir arabanın hızını ve ivmesini bulma bağlamında oluşturduk. Ancak zamanla (veya zamandan başka değişkenlerle) değişen ve bu özel örnek için az önce yaptığımız gibi türev kavramı uygulanarak incelenebilecek çok sayıda gerçek hayat olgusu vardır.

Birkaç örnek:

- Yakın gelecekte nüfus büyüklüğündeki değişiklikleri tahmin etmek için kullanılabilen, bir popülasyonun (bir insan topluluğu veya bir bakteri kolonisi olsun) zaman içindeki büyüme hızı.
- Hava tahmini için kullanılabilen konumun bir fonksiyonu olarak sıcaklıktaki değişiklikler.
- Gelecekteki borsa davranışını tahmin etmek için kullanılabilen borsanın zaman içindeki dalgalanmaları.

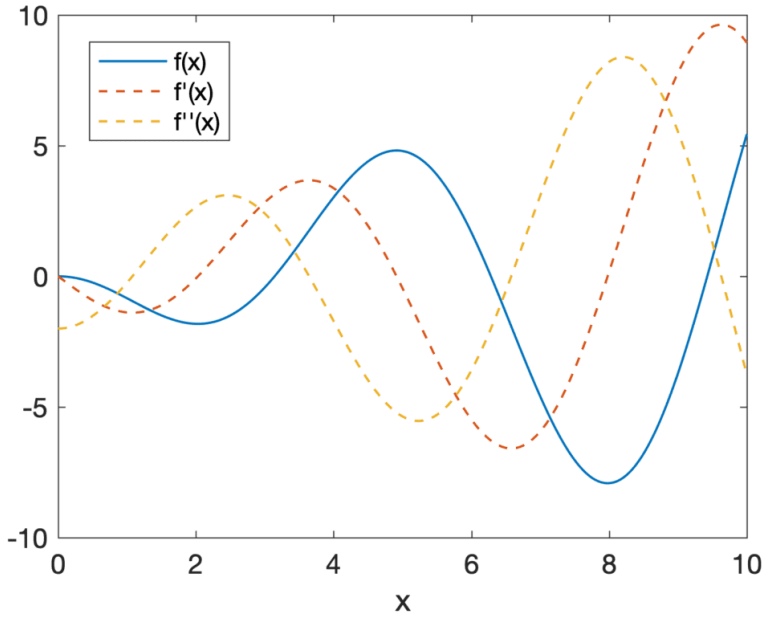
**Optimizasyon Algoritmalarında Türev Uygulamaları**

Türevler optimizasyon problemlerinin çözümünde önemli bilgiler sağlar. Araba örneği için yaptığımız alıştırmanın aynısını yaparak, türevlerin hata fonksiyonu hakkında ne söylediğine daha yakından bakalım.

Bu amaçla, fonksiyon optimizasyonu için aşağıdaki tek boyutlu test fonksiyonunu ele alalım:
$f(x) = -x \sin(x)$

f'(x) ile gösterilen birinci türevini bulmak için f(x)'e çarpım kuralı uygulanır ve sonra f''(x) ile gösterilen ikinci türevi bulmak için f'(x)'e çarpım kuralını tekrar uygulanır:
$f'(x) = -\sin(x) - x\cos(x)$
$f''(x) = x\sin(x) - 2\cos(x)$

Bu üç işlevi görselleştirmek için x'in farklı değerleri için çizebiliriz:




Fonksiyonun Doğru Grafiği, f(x), birinci türevi, f'(x) ve ikinci türevi, f''(x)

Daha önce araba örneğinde gözlemlediğimiz gibi, birinci türevin grafiği f(x)'in nasıl ve ne kadar değiştiğini gösterir. Örneğin, pozitif bir türev, f(x)'in artan bir fonksiyon olduğunu gösterirken, negatif bir türev bize f(x)'in artık azalmakta olduğunu söyler. Bu nedenle,

minimum fonksiyon arayışında optimizasyon algoritması, öğrenme hızı ε'ya dayalı olarak girdide küçük değişiklikler gerçekleştirirse:
$x\_new = x - \varepsilon\ f'(x)$

Daha sonra algoritma, türevin ters yönüne hareket ederek (işareti ters çevirerek) f(x)'i azaltabilir. İkinci türevi eğriliği ölçmek olarak düşünebiliriz.

Örneğin, algoritma birinci türevin sıfır olduğu kritik bir noktaya ulaşırsa, bu noktanın yerel maksimum, yerel minimum, eyer noktası veya yalnızca f'(x)'e dayalı düz bölge olduğunu ayırt edemez. Bununla birlikte, ikinci türev araya girdiğinde, algoritma, ikinci türev sıfırdan büyükse, söz konusu kritik noktanın yerel bir minimum olduğunu söyleyebilir. Bir yerel maksimum için, ikinci türev sıfırdan küçüktür. Dolayısıyla, ikinci türev, optimizasyon algoritmasına hangi yöne hareket etmesi gerektiği konusunda bilgi verebilir. Ne yazık ki, bu test, her iki durumda da ikinci türevi sıfır olan eyer noktaları ve düz bölgeler için sonuçsuz kalmaktadır.

Gradyan inişine dayalı optimizasyon algoritmaları, ikinci dereceden türevleri kullanmazlar ve bu nedenle birinci dereceden optimizasyon algoritmaları olarak bilinirler. İkinci türevlerin kullanımından yararlanan Newton yöntemi gibi optimizasyon algoritmalarına, aksi takdirde ikinci dereceden optimizasyon algoritmaları denir.

**Türev Formülleri**
Doğrusal, üstel ve logaritmik fonksiyonlar gibi bazı fonksiyonların türev formülleri aşağıda listelenmiştir:

- $d/dx\ (k) = 0$, burada k herhangi bir sabittir.
- $d/dx(x) = 1$
- $d/dx(x^n) = nx^{n-1}$
- $d/dx\ (kx) = k$, burada k herhangi bir sabittir.
- $d/dx\ (\sqrt{x}) = 1/2\sqrt{x}$
- $d/dx\ (1/x) = -1/x^2$
- $d/dx\ (\log x) = 1/x,\ x > 0$
- $d/dx\ (e^x) = e^x$
- $d/dx\ (a^x) = a^x \log a$

**Trigonometrik Fonksiyonların Türevleri:**

- $d/dx\ (\sin x) = \cos x$
- $d/dx\ (\cos x) = -\sin x$
- $d/dx\ (\tan x) = \sec^2 x$
- $d/dx\ (\text{cosec}\ x) = -\text{cosec}\ x \cot x$
- $d/dx\ (\sec x) = \sec x \tan x$
- $d/dx\ (\cot x) = -\text{cosec}^2 x$

**Türev Türleri:**
Türevler, birinci ve ikinci dereceden türevler gibi sıralarına göre farklı türlerde sınıflandırılabilir. Bunlar aşağıda verildiği gibi tanımlanabilir.

Birinci mertebeden türevler, fonksiyonun artan mı yoksa azalan mı olduğunu fonksiyonun yönü hakkında söyler. Birinci türev matematiği veya birinci dereceden türev, anlık bir değişim oranı olarak yorumlanabilir. Teğet doğrunun eğiminden de tahmin edilebilir.

İkinci dereceden türevler, verilen fonksiyon için grafiğin şekli hakkında bir fikir edinmek için kullanılır. Fonksiyonlar içbükeylik açısından sınıflandırılabilir. Verilen grafik fonksiyonunun içbükeyliği iki türe ayrılır:
• Yukarı İçbükey
• Aşağı İçbükey

Analiz-Türev Örneği: Burada f(x) bir fonksiyon olsun $f(x) = x^2$
$x^2$'nin türevi 2x'tir, yani x'teki her birim değişimde fonksiyonun değeri iki katına çıkar (2x).

**Limitler ve Türevler:**
dx o kadar küçük yapıldığında neredeyse hiç oluyor. Limitler ile, x'in sıfıra yaklaştığını ancak sıfır olmadığını söylemek istiyoruz.
Matematiksel olarak: tüm gerçek $\varepsilon > 0$ için gerçek bir $\delta > 0$ vardır, öyle ki $0 < |x - c| < \delta$, (burada $c \in R$) $|f(x) - L| < e$

**Anlık değişim hızı problemini çözmek için türev kurallarının uygulanması:**
Bir paraşütçü 2200 m yükseklikten uçaktan atlıyor. t saniye sonra paraşütçünün yerden metre cinsinden yüksekliği $h(t) = 2200 – 4,9t^2$ fonksiyonu ile temsil edilir (hava direncinin bir faktör olmadığı varsayılarak). Paraşütçü 4 saniye sonra ne kadar hızlı düşüyor?

Zamanın herhangi bir noktasında hava dalışçısının boyunun anlık değişim oranı, yükseklik fonksiyonunun türevi ile temsil edilir.
$h(t) = 2200 – 4,9t^2$
$h'(t) = 0 – 4,9\ (2t) = -9,8\ t$

4 s'deki anlık değişim oranını bulmak için türev fonksiyonunda t = 4'ü değiştirin.
$h'(t) = – 9,8\ (4) = -39,2$

4 saniye sonra paraşütçü 39.2 m/s hızla düşüyor.

**Derivative of tan x**

Let f(x) = tan x

We know that tan x = sin x/ cos x

Let us take u = sin x and v = cos x

As we know,

$d/dx\ (u/v) = [v(du/dx) – u(dv/dx)]/\ v^2$

$d/dx\ (\sin x/\cos x) = [\cos x(d/dx)\sin x – \sin x(d/dx)\cos x]/\ \cos^2 x$

$= [\cos x\ .\ \cos x – \sin x\ .\ (-\sin x)]/\cos^2 x = (\cos^2 x + \sin^2 x)/\cos^2 x$

Using the identity $\sin^2 A + \cos^2 A = 1$,

$= 1/\cos^2 x = \sec^2 x$ [since 1/cos x = sec x]

$d/dx\ (\tan x) = \sec^2 x$

Therefore, the derivative of tan x is $\sec^2 x$.

**Derivative of 1/x**

The derivative of 1/x can be derived as given below:

$d/dx\ (1/x) = d/dx\ (x^{-1})$

We know that $d/dx\ (x^n) = nx^{n-1}$

Here, n = -1

$d/dx(1/x) = d/dx\ (x^{-1}) = (-1)x^{(-1-1)}$

$= -x^{-2} = -1/x^2$

Hence, the derivative of 1/x is $-1/x^2$.

**Diferansiyel hesap**

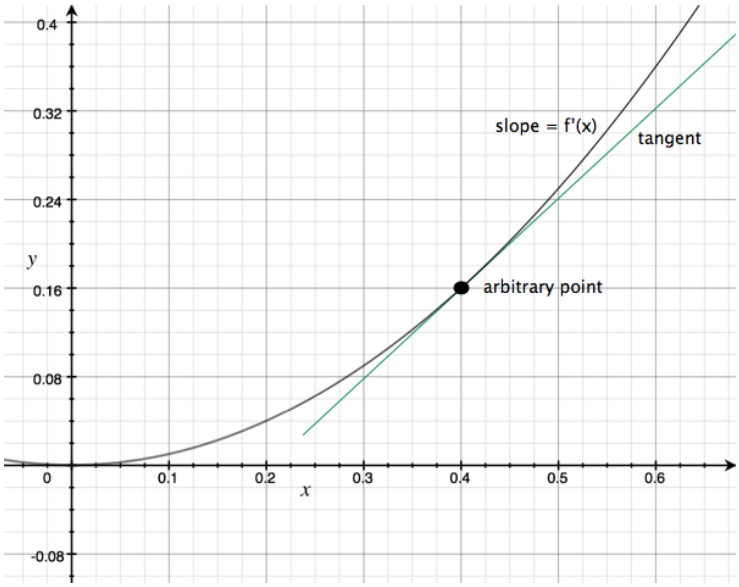
Diferansiyel Hesap, niceliklerin değişme oranları ile ilgilidir. Bir örnek, boyutsal uzayda bir noktanın yakınında değişiklik davranışını ölçmek için bir türev kullanmaktır. Eğer bir fonksiyonumuz varsa:

$$f\left(x\right) = x^2$$

daha sonra, diferansiyel hesabın kuvvet kuralı uygulanarak, türev işlevi her x noktasına göre y ekseni olarak da düşünülen f(x)'teki değişim olarak ifade edilir:

$$f'\left(x\right) = \frac{dy}{dx} = 2x$$

Bu aynı zamanda eğrinin x noktasındaki eğimini veya gradyanını da gösterir. Bir skaler alanın yön türevi (gradyan) artımın en çok olduğu yere doğru yönelmiş bir vektör alanını verir ve büyüklüğü değişimin en büyük değerine eşittir.




**Diferansiyel Hesap Kuralları**

Aşağıdaki kurallar, ortak fonksiyonların türevini hızlı bir şekilde belirlemenin bir yolunu sunar. Daha karmaşık işlevler, bu kurallar uygulanabilene kadar kademeli olarak daha küçük bileşenlere bölünebilir.

Using:

| | |
|---|---|
| constants | $c$ |
| radians | $r$ |
| variables | $a, u, n, x, y,$ |
| functions | $f, g, ln, log, sin, cos, tan, sec$ |

| | | |
|---|---|---|
| sin | $sin\,(r)$ | $cos\,(r)$ |
| cos | $cos\,(r)$ | $-\,sin\,(r)$ |
| tan | $tan\,(r)$ | $sec^2\,(r)$ |

| | original | derivative |
|---|---|---|
| constant | $c$ | $0$ |
| line | $x$ | $1$ |
| exponential | $a^x$ | $ln\,(a)\,a^x$ |
| exponential of e | $e^x$ | $e^x$ |
| square root | $\sqrt{x}$ | $(1/2)\,x^{1/2}$ |
| logarithm base $e$ | $ln\,(x)$ | $1/x$ |
| logarithm base $a$ | $log_a\,(x)$ | $1/\,(x \cdot ln\,(a))$ |

| | | |
|---|---|---|
| mult. by constant | $cf$ | $cf^{'}$ |
| power | $x^n$ | $nx^{n-1}$ |
| sum | $f+g$ | $f^{'}+g^{'}$ |
| difference | $f-g$ | $f^{'}-g^{'}$ |
| product | $f \cdot g$ | $fg^{'}+f^{'}g$ |
| quotient | $\frac{f}{g}$ | $\frac{\left(f^{'}g-g^{'}f\right)}{g^2}$ |
| reciprocal | $\frac{1}{f}$ | $\frac{-f^{'}}{f^2}$ |
| chain using $^{'}$ | $f\,(g\,(x))$ | $f^{'}\,(g\,(x))\,g^{'}\,(x)$ |
| chain using $\frac{d}{dx}$ | $\frac{dy}{dx}$ | $\frac{dy}{du}\frac{du}{dx}$ |

Matematikte Türev, bir fonksiyonun bir değişkene göre değişim oranıdır. Türevler, matematik ve diferansiyel denklemlerdeki problemlerin çözümü için temeldir. Genel olarak bilim adamları, ilgilenilen bazı değişkenlerin değişim oranını elde etmek için dinamik sistemleri gözlemler, bu bilgiyi bazı diferansiyel denklemlere dahil eder ve farklı koşullar altında orijinal sistemin davranışını tahmin etmek için kullanılabilecek bir fonksiyon elde etmek için entegrasyon tekniklerini kullanır.

Türev, belirli bir noktada fonksiyonda meydana gelen değişikliğin miktarı olan anlık değişim oranını göstermenin bir yoludur. Gerçek sayılar üzerinde etkili olan fonksiyonlar için, grafik üzerindeki bir noktadaki teğet doğrunun eğimidir. . Türev genellikle şu şekilde yazılır:

$\frac{dy}{dx}$.

Bu, y'deki farkın x'teki farka bölümüdür. Burada d değişken değildir, dolayısıyla iptal edilemez. y'nin x'e göre türevi, y'deki değişim bölü x'teki değişim olarak tanımlanır. Matematiksel olarak,

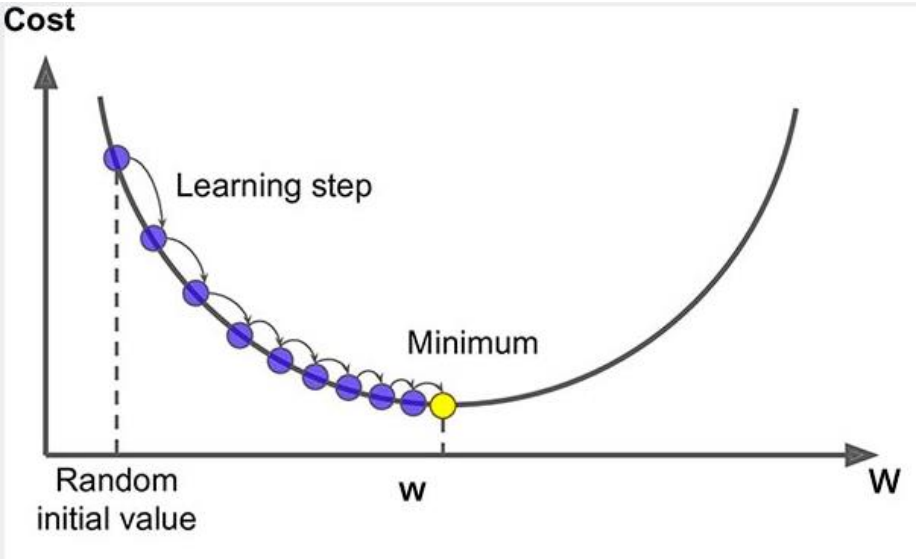
$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

Yani, iki x noktası arasındaki mesafe sıfıra yaklaştıkça, aralarındaki çizginin eğimi ilgili teğet çizgisine yaklaşır.

**Importance of derivatives in data science:**
Türevler, Makine öğrenimi tarafından optimizasyon problemlerini çözmek için kullanılır. Gradyan iniş gibi optimizasyon algoritmaları, herhangi bir amaç fonksiyonunu artırmak veya azaltmak için ağırlıkların artırılıp azaltılmayacağına karar vermek için türevleri kullanır.

Veri Bilim hemen hemen her model için hesabı kullanır ve Makine Öğreniminde temel ama çok mükemmel bir matematik örneği Gradient Descent'tir.

**Gradient Descent: -**
Gradyan, girişleri biraz değiştirirseniz bir fonksiyonun çıktısının ne kadar değiştiğini ölçer. Makine öğrenimi modelinde amacımız girdi verilerimizdeki maliyeti azaltmaktır. Maliyet fonksiyonu, bir ML modelinin tahminlerindeki hatayı izlemek için kullanılır. Bunu en aza indirmek, temelde mümkün olan en düşük hata değerine ulaşmak veya modelin doğruluğunu artırmak anlamına gelir. Bu nedenle, modelimizin parametrelerini değiştirirken bir eğitim veri kümesi üzerinde yineleme yaparak doğruluğu artırıyoruz.




Örneğin, bazı konulardaki notlarını ve mezuniyet için seçtikleri ileri çalışmaları içeren bir kullanıcı veri setimiz var. Amacımız, kişinin notlarını dikkate alarak mezuniyet akışını tahmin etmektir.

| | Mathematics | Physics | Chemistry | Biology | English | Stream |
|---|---|---|---|---|---|---|
| Swapnil | 65 | 67 | 58 | 48 | 69 | Arts |
| Priyanker | 75 | 70 | 87 | 87 | 88 | Science |
| Sarang | 86 | 85 | 86 | 78 | 87 | |

Bu veri setinde Swapnil ve Priyanker'a ait veriler var ve bu verileri kullanarak Sarang'ın mezuniyet akışını tahmin etmemiz gerekiyor.

Şimdi öznedeki işaretleri bir gradyan ve akış olarak alt hedef olarak ele alıyoruz. Altta tahmin ettiği sonucun doğru olması için modeli optimize etmeliyiz.

Swapnil'in ve Priyanker'in verilerini kullanarak, gradyan inişini oluşturacağız ve modelimizi öyle ayarlayacağız ki, eğer Sarang'ın işaretlerini verirsek, Science'ın gradyanın altındaki sonucunu ve Priyanker için aynısını tahmin etmesi gerekiyor. Bu bizim eğitimli modelimiz. Şimdi modelimize konu işaretleri verirsek akışı kolayca tahmin edebiliriz.

Bu modelde kullanabileceğimiz temel formül y = m*x + b'dir. Burada, y = yordayıcı, m = eğim, x = giriş, b = y- kesişme noktasıdır. Bu tür bir problem, belirli bir hattın ne kadar iyi olduğunu belirleyen bir maliyet fonksiyonu tanımlanarak çözülebilir. Maliyet fonksiyonu aşağıdaki denklem kullanılarak hesaplanabilir:

$$\text{Error}_{(m,b)} = \frac{1}{N}\sum_{i=1}^{N}(y_i - (mx_i + b))^2$$

Verilerimize daha iyi uyan satırlar, daha düşük hata değerleri ile sonuçlanacaktır. Bu fonksiyonu en aza indirirsek, verilerimiz için en iyi satırı elde ederiz. Eğimi hesaplamak için farklılaşmayı kullanırız.

Bu hata fonksiyonunda gradyan inişini çalıştırmak için gradyanını hesaplamamız gerekir. Gradyan hesaplamak için hata fonksiyonumuzun türevini almamız gerekir. Fonksiyonumuz iki parametre (m ve b) tarafından tanımlandığından, her biri için kısmi bir türev hesaplamamız gerekecek. Bu türevler aşağıdaki gibi hesaplanabilir:

$$\frac{\partial}{\partial \mathrm{m}} = \frac{2}{N}\sum_{i=1}^{N} - x_i(y_i - (mx_i + b))$$

$$\frac{\partial}{\partial \mathrm{b}} = \frac{2}{N}\sum_{i=1}^{N} - (y_i - (mx_i + b))$$

Aramamızı herhangi bir m ve b değeri çiftinde (yani herhangi bir satırda) başlayacak şekilde başlatabilir ve gradyan iniş algoritmasının hata fonksiyonumuzda yokuş aşağı en iyi satıra doğru ilerlemesine izin verebiliriz. Her yineleme, m ve b'yi önceki yinelemeden biraz daha düşük hata veren bir satıra güncelleyecektir. Her yineleme için hareket yönü, yukarıdan iki kısmi türev kullanılarak hesaplanır.

Öğrenme Hızı değişkeni, her yinelemede yokuş aşağı attığımız adımın ne kadar büyük olduğunu kontrol eder. Çok büyük bir adım atarsak, minimumun üzerine çıkabiliriz. Ancak, küçük adımlar atarsak, minimuma ulaşmak için birçok yineleme gerekecektir.

**Veri Biliminde Matematik ve kullanım alanları:**
Matematik, tamamen resmi bir şekilde geliştirilmiş soyut bir teoridir. Analiz, daha doğrusu analiz olarak adlandırılan, niceliklerin değişim oranını (eğrilerin eğimleri olarak yorumlanabilir) ve nesnelerin uzunluğunu, alanını ve hacmini inceleyen matematiğin dalıdır. Kalkülüs, diferansiyel ve integral hesap olarak ikiye ayrılır.

Türev,

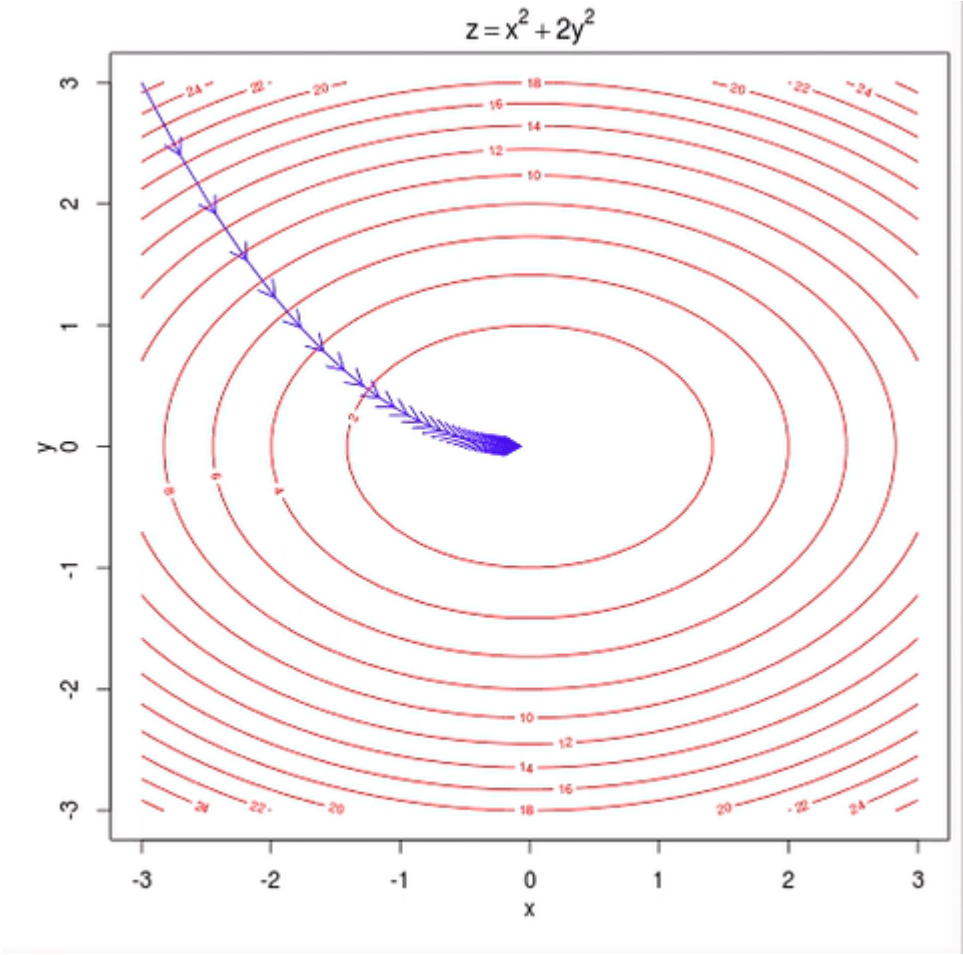
$$\frac{d}{dx}(x) = 1$$

İntegral,

$$\int 1\,dx = x + C$$

*Calculus kelimesi, "küçük taş" anlamına gelen Latince kökenlidir. Çünkü küçük parçalara bakarak bir şeyi anlamak gibi. Analiz, matematiğin kendine özgü bir alanıdır ve özellikle birçok makine öğrenimi algoritmasında, Veri Biliminin özünü öğrenmek için gereklidir. Diferansiyel Analiz, nasıl değiştiğini bulmak için bir şeyi küçük parçalara ayırır.*
*Integral Calculus, ne kadar olduğunu bulmak için küçük parçaları birleştirir (bütünleştirir).*

Dereceli alçalma (**Gradient Descent**):
Gradyan, girişleri biraz değiştirirseniz bir fonksiyonun çıktısının ne kadar değiştiğini ölçer. Bir topunuz ve bir kaseniz olduğunu varsayalım. Topu kasede nereye kaydırırsanız kaydırın, sonunda kasenin dibine düşecektir.




Gördüğünüz gibi bu top kasenin dibinde biten bir yol izliyor. Topun kasenin dibine indiğini de söyleyebiliriz. Resimden de görebileceğiniz gibi kırmızı çizgiler kasenin gradyanı ve mavi çizgi topun izlediği yoldur ve topun eğiminin yolu azaldıkça buna gradyan iniş denir.

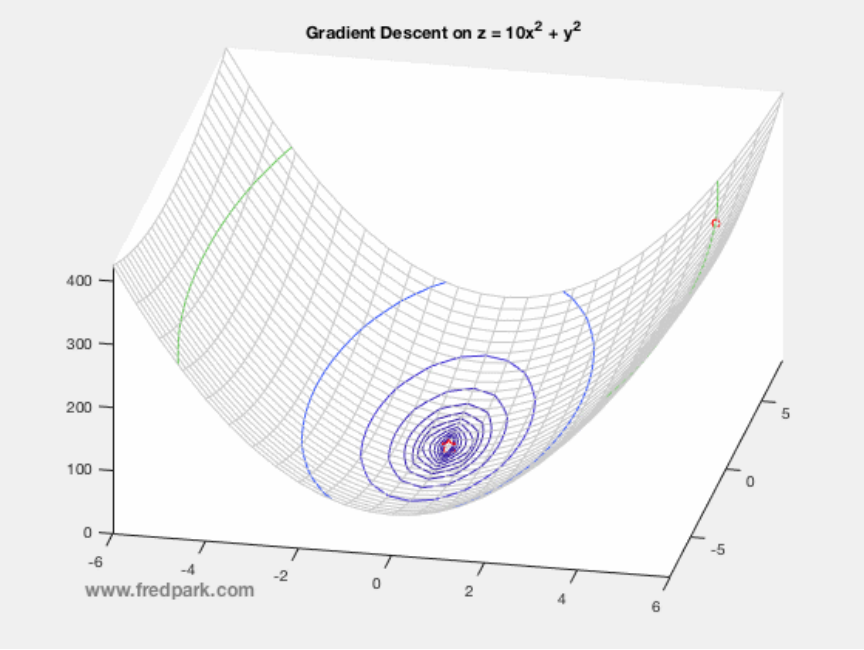
Makine öğrenimi modelimizde amacımız, girdi verilerimizdeki maliyeti azaltmaktır. maliyet fonksiyonu, bir ML modelinin tahminlerindeki hatayı izlemek için kullanılır. Dolayısıyla bunu en aza indirmek, temelde mümkün olan en düşük hata değerine ulaşmak veya modelin doğruluğunu artırmak anlamına gelir. Kısacası, modelimizin parametrelerini (ağırlıklar ve önyargılar) ince ayarlarken bir eğitim veri seti üzerinde yineleme yaparak doğruluğu

artırıyoruz. Bazı konularda notları ve meslekleri olan bir kullanıcı veri setimiz olduğunu düşünelim. Amacımız, kişinin işaretlerini dikkate alarak kişinin mesleğini tahmin etmektir.

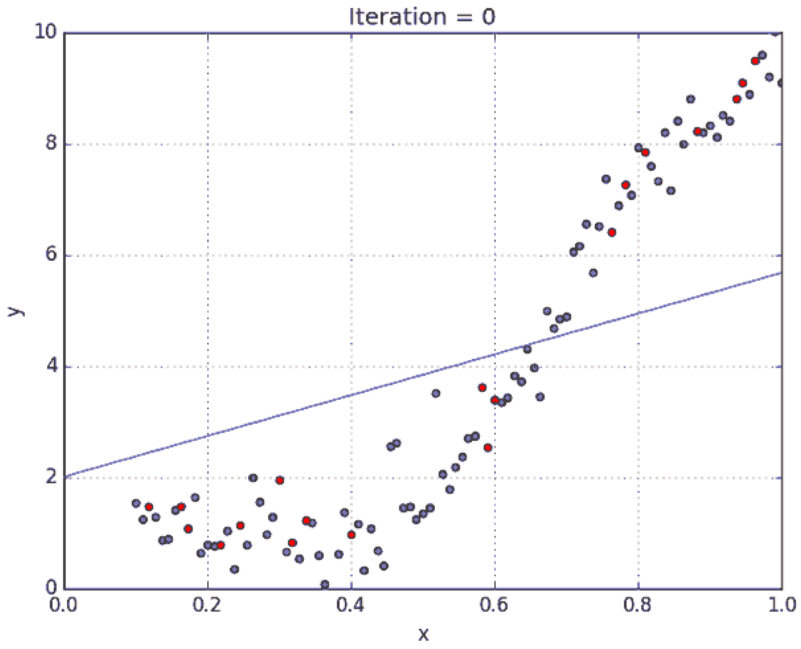
| | Mathematics | Physics | Chemistry | Biology | English | Profession |
|---|---|---|---|---|---|---|
| John | 78 | 56 | 65 | 83 | 66 | Doctor |
| Eve | 83 | 78 | 72 | 66 | 59 | Engineer |
| Adam | 75 | 67 | 79 | 55 | 70 | |

Bu veri setinde John ve havva verilerimiz var. Yuhanna ve Havva'nın referans verileriyle Adem'in mesleğini tahmin etmemiz gerekiyor.

Şimdi konudaki notları bir eğim olarak ve mesleği alt hedef olarak düşünün. Altta tahmin ettiği sonucun doğru olması için modelinizi optimize etmelisiniz. John's ve Eve'in verilerini kullanarak gradyan iniş oluşturacağız ve modelimizi öyle ayarlayacağız ki, John'un işaretlerini girersek Doktor'un gradyanın altındaki sonucunu ve Eve için aynı sonucu tahmin etmesi gerekiyor. Bu bizim eğitimli modelimiz. Şimdi modelimize konu işaretlerini verirsek mesleği kolayca tahmin edebiliriz.




Teorik olarak bu, gradyan inişi içindir, ancak gradyan inişi hesaplamak ve modellemek için matematik gerektirir ve artık makine öğreniminde analizin önemini görebiliriz.
Öncelikle şu ana kadar bildiğiniz konudan başlayalım. Lineer Cebir. Modelimiz için önce lineer cebiri ve formülünü kullanalım.

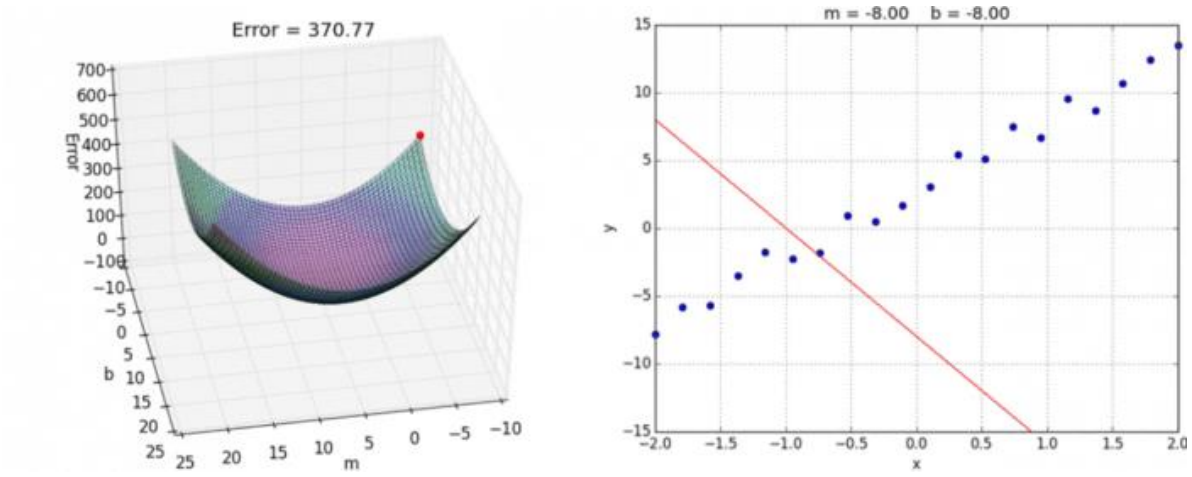



Bu modelde kullanabileceğimiz temel formül y = m*x +b'dir. burada, y = yordayıcı, m = eğim, x = girdi, b= y-kesme noktası.

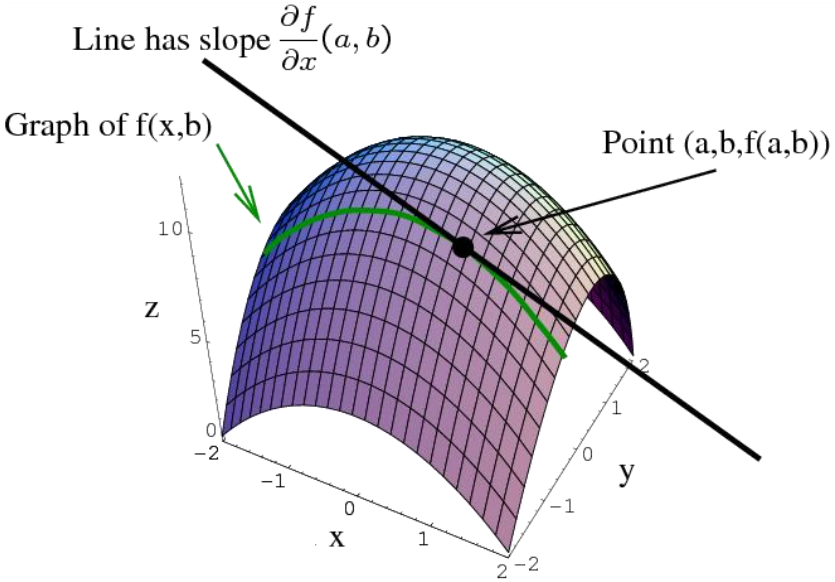
Bu tür bir sorunu çözmeye yönelik standart bir yaklaşım, belirli bir satırın ne kadar "iyi" olduğunu ölçen bir hata işlevi (maliyet işlevi olarak da adlandırılır) tanımlamaktır. Bu fonksiyon bir (m,b) çifti alacak ve çizginin verilerimize ne kadar iyi uyduğuna bağlı olarak bir hata değeri döndürecektir. Belirli bir doğru için bu hatayı hesaplamak için, veri kümemizdeki her bir (x,y) noktasını yineleyeceğiz ve her noktanın y değeri ile aday hattın y değeri (mx + b'de hesaplanan) arasındaki kare mesafeleri toplayacağız. Pozitif olmasını sağlamak ve hata fonksiyonumuzun türevlenebilir olmasını sağlamak için bu mesafenin karesini almak gelenekseldir.

$$\text{Error}_{(\text{m,b})} = \frac{1}{N}\sum_{i=1}^{N}(y_i - (mx_i + b))^2$$

Verilerimize daha iyi uyan satırlar (hata fonksiyonumuz tarafından daha iyi tanımlandığı yerde), daha düşük hata değerleri ile sonuçlanacaktır. Bu fonksiyonu en aza indirirsek, verilerimiz için en iyi satırı elde ederiz. Hata fonksiyonumuz iki parametreden (m ve b) oluştuğu için onu iki boyutlu bir yüzey olarak görselleştirebiliriz. Veri kümemiz için böyle görünüyor:




Bu iki boyutlu uzayda her nokta bir çizgiyi temsil eder. Fonksiyonun her noktadaki yüksekliği o satır için hata değeridir. Bazı satırların diğerlerinden daha küçük hata değerleri verdiğini görebilirsiniz (yani verilerimize daha iyi uyuyor). Gradyan iniş araması yaptığımızda, bu yüzeyde bir yerden başlayıp en düşük hataya sahip çizgiyi bulmak için yokuş aşağı ineceğiz.




z=f(x,y)z=f(x,y) fonksiyonunun grafiği bir yüzeydir ve y=by=b sabitlenmesi bir eğri verir (yeşille gösterilmiştir). ðfðx(a,b)ðfðx(a,b) kısmi türevi, bu eğriye x=ax=a noktasındaki teğet doğrunun eğimidir.

Bu hata fonksiyonunda gradyan inişini çalıştırmak için önce gradyanını hesaplamamız gerekir. Eğim bir pusula gibi davranacak ve bizi her zaman yokuş aşağı gösterecek. Bunu hesaplamak için hata fonksiyonumuzun türevini almamız gerekecek. Fonksiyonumuz iki parametre (m ve b) tarafından tanımlandığından, her biri için kısmi bir türev hesaplamamız gerekecek. Bu türevler şöyle çalışır:

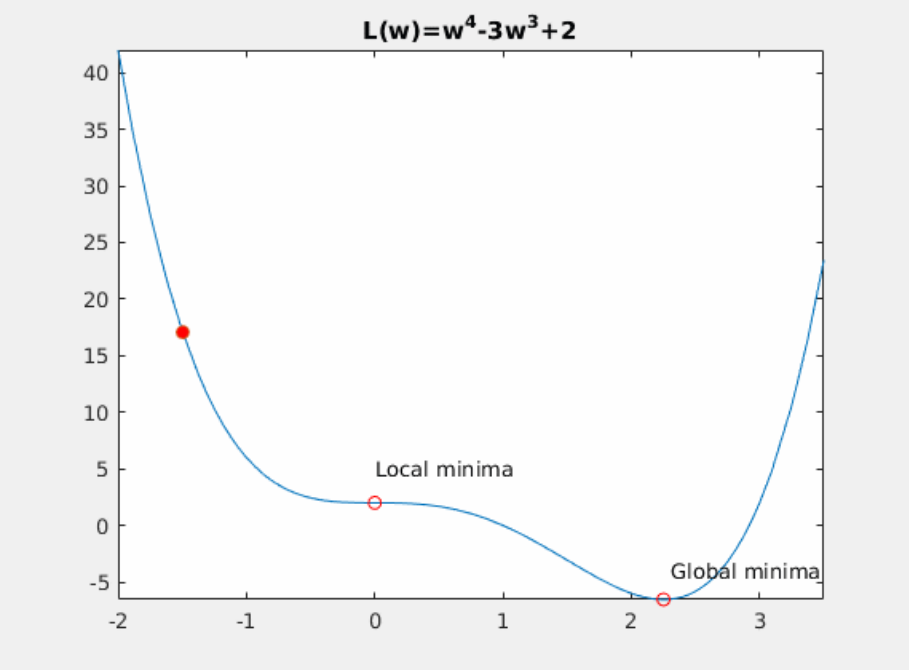
$$\frac{\partial}{\partial \mathrm{m}} = \frac{2}{N}\sum_{i=1}^{N} - x_i(y_i - (mx_i + b))$$

$$\frac{\partial}{\partial \mathrm{b}} = \frac{2}{N}\sum_{i=1}^{N} - (y_i - (mx_i + b))$$

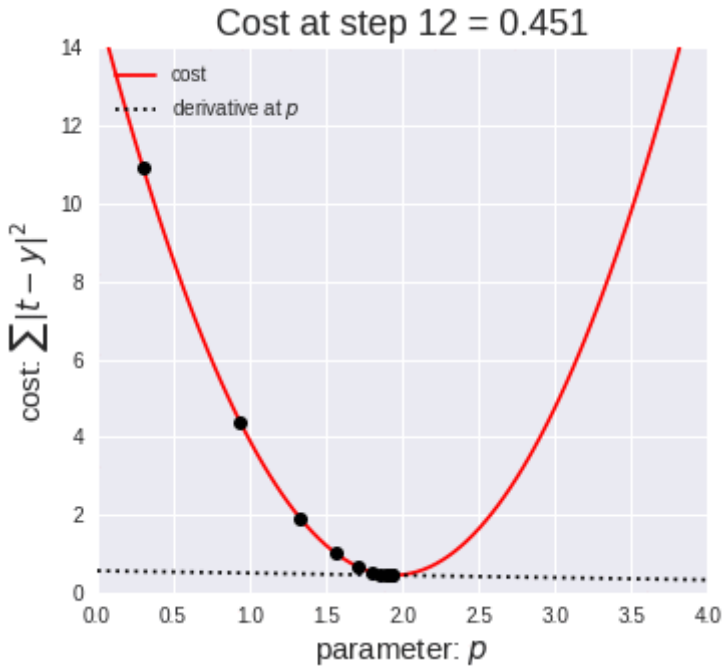
Artık gradyan inişini çalıştırmak için gereken tüm araçlara sahibiz. Aramamızı herhangi bir m ve b değeri çiftinde (yani herhangi bir satırda) başlayacak şekilde başlatabilir ve gradyan iniş algoritmasının hata fonksiyonumuzda yokuş aşağı en iyi satıra doğru ilerlemesine izin verebiliriz. Her yineleme, m ve b'yi önceki yinelemeden biraz daha düşük hata veren bir satıra güncelleyecektir. Her yineleme için hareket yönü, yukarıdan iki kısmi türev kullanılarak hesaplanır.

Öğrenme Hızı değişkeni, her yinelemede yokuş aşağı attığımız adımın ne kadar büyük olduğunu kontrol eder. Çok büyük bir adım atarsak, minimumun üzerine çıkabiliriz. Ancak, küçük adımlar atarsak, minimuma ulaşmak için birçok yineleme gerekecektir. Gradyan inişini öğrenmek için yüzeyi çizebilmiş olsak da, tartışamadığımızın farkında olmamızda fayda olan birkaç ek kavram var. Bunlardan birkaçı şunları içerir:

Dışbükeylik (Convexity=: Doğrusal regresyon sorunumuzda yalnızca bir minimum vardı. Hata yüzeyimiz dışbükeydi. Nereden başlarsak başlayalım, sonunda mutlak minimuma ulaşacağız. Genel olarak, durumun böyle olması gerekmez. Bir gradyan aramasının takılıp kalabileceği yerel minimumlarla ilgili bir sorun olması mümkündür. Bunu hafifletmek için birkaç yaklaşım vardır (örneğin, stokastik gradyan arama).



Yakınsama (Convergence): Aramanın bir çözüm bulduğunda nasıl belirleneceğinden bahsetmedik. Bu genellikle hata yinelemeden yinelemeye küçük değişiklikler aranarak yapılır (örneğin, gradyanın sıfıra yakın olduğu yer).




Çok Değişkenli Analiz ( Multivariate Calculus): Şimdi, sonunda gerçek hayatta elde edeceğimiz çok değişkenli verilerde hesabı öğretecek olan Çok Değişkenli analize derinlemesine dalalım.

## 4.2. Fonksiyon optimizasyonu için tek boyutlu (1d) test fonksiyonları

(One-Dimensional (1D) Test Functions for Function Optimization)

İşlev optimizasyonu, bir işleve, işlevin maksimum veya minimum çıktısıyla sonuçlanan bir girdi arayan bir çalışma alanıdır.

Çok sayıda optimizasyon algoritması vardır, basit ve görselleştirmesi kolay test fonksiyonları üzerinde optimizasyon algoritmaları için sezgileri incelemek ve geliştirmek önemlidir.
Tek boyutlu fonksiyonlar, tek bir girdi değeri alır ve girdinin tek bir değerlendirmesini verir. İşlev optimizasyonu çalışırken kullanılacak en basit test işlevi türü olabilirler.

Tek boyutlu fonksiyonların yararı, x ekseninde fonksiyona girişler ve y ekseninde fonksiyonun çıktıları ile iki boyutlu bir çizim olarak görselleştirilebilmeleridir. Fonksiyonun bilinen optimumları ve fonksiyonun herhangi bir örneklemesi de aynı çizim üzerinde çizilebilir.

Kullanabileceğimiz birçok farklı basit tek boyutlu test fonksiyonu vardır. Bununla birlikte, fonksiyon optimizasyonu alanında yaygın olarak kullanılan standart test fonksiyonları vardır. Farklı algoritmaları test ederken seçmek isteyebileceğimiz test fonksiyonlarının belirli özellikleri de vardır. Bu öğretide az sayıda basit tek boyutlu test işlevini keşfedeceğiz ve bunları beş farklı grupla özelliklerine göre düzenleyeceğiz; bunlar:

1. Konveks Tek Modlu Fonksiyonlar (Convex Unimodal Functions)
2. Konveks Olmayan Tek Modlu Fonksiyonlar (Non-Convex Unimodal Functions)
3. Çok Modlu İşlevler (Multimodal Functions)
4. Süreksiz Fonksiyonlar - Pürüzsüz Olmayan (Discontinuous Functions - Non-Smooth)
5. Gürültülü İşlevler (Noisy Functions)

Her işlev, Python kodu kullanılarak, hedef amaç işlevinin bir işlev uygulaması ve işlevin optimumunun açıkça işaretlendiği bir çizgi çizimi olarak gösterilen işlevin bir örneklemesi ile sunulacaktır. Tüm fonksiyonlar bir minimizasyon problemi olarak sunulur, örn. fonksiyonun minimum (en küçük değer) çıktısını veren girişi bulun. Herhangi bir maksimize etme işlevi, tüm çıktıya negatif bir işaret ekleyerek bir minimizasyon işlevi haline getirilebilir. Benzer şekilde, herhangi bir küçültme işlevi, aynı şekilde en büyük hale getirilebilir. Ardından, optimizasyon algoritmalarının davranışını incelemek veya karşılaştırmak için kendi projenizde kullanmak üzere bir veya daha fazla işlevin kodunu seçip kopyalayıp yapıştırabilirsiniz.

**Convex Unimodal Function**

A convex function is a function where a line can be drawn between any two points in the domain and the line remains in the domain.

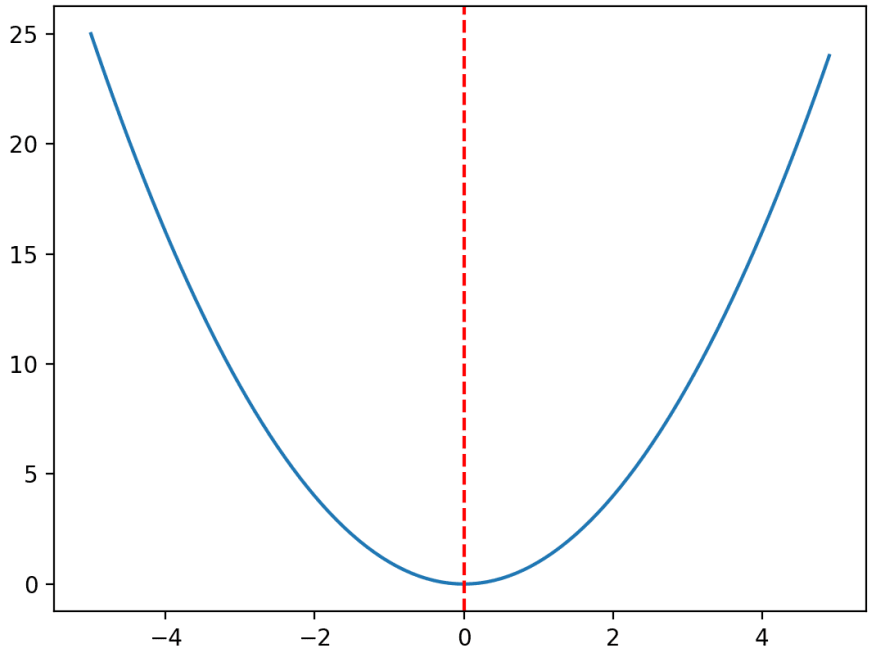
For a one-dimensional function shown as a two-dimensional plot, this means the function has a bowl shape and the line between two remains above the bowl.

Unimodal means that the function has a single optima. A convex function may or may not be unimodal; similarly, a unimodal function may or may not be convex.

The range for the function below is bounded to -5.0 and 5.0 and the optimal input value is 0.0.

```
# convex unimodal optimization function
from numpy import arange
from matplotlib import pyplot

# objective function
def objective(x):
 return x**2.0

# define range for input
r_min, r_max = -5.0, 5.0
# sample input range uniformly at 0.1 increments
inputs = arange(r_min, r_max, 0.1)
# compute targets
results = objective(inputs)
# create a line plot of input vs result
pyplot.plot(inputs, results)
# define optimal input value
x_optima = 0.0
# draw a vertical line at the optimal input
pyplot.axvline(x=x_optima, ls='--', color='red')
# show the plot
pyplot.show()
```

Running the example creates a line plot of the function and marks the optima with a red line.




Line Plot of Convex Unimodal Optimization Function

This function can be shifted forward or backward on the number line by adding or subtracting a constant value, e.g. 5 + x^2.
This can be useful if there is a desire to move the optimal input away from a value of 0.0.

**Non-Convex Unimodal Functions**

A function is non-convex if a line cannot be drawn between two points in the domain and the line remains in the domain.

This means it is possible to find two points in the domain where a line between them crosses a line plot of the function.

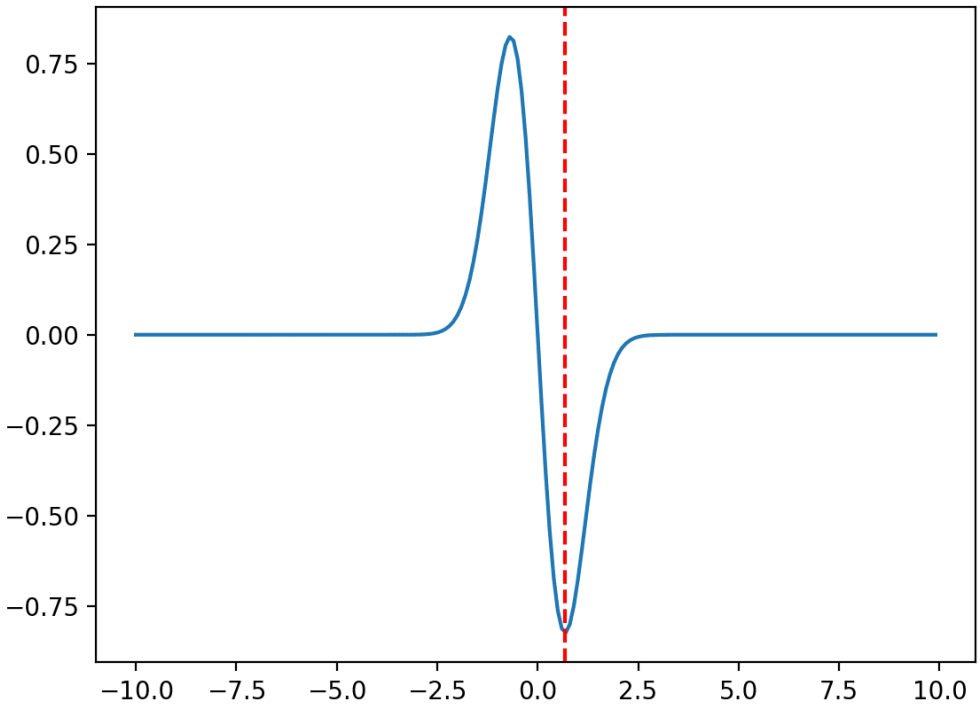
Typically, if a plot of a one-dimensional function has more than one hill or valley, then we know immediately that the function is non-convex. Nevertheless, a non-convex function may or may not be unimodal.

Most real functions that we're interested in optimizing are non-convex.

The range for the function below is bounded to -10.0 and 10.0 and the optimal input value is 0.67956.

```
# non-convex unimodal optimization function
from numpy import arange
from numpy import sin
from numpy import exp
from matplotlib import pyplot

# objective function
def objective(x):
 return -(x + sin(x)) * exp(-x**2.0)

# define range for input
r_min, r_max = -10.0, 10.0
# sample input range uniformly at 0.1 increments
inputs = arange(r_min, r_max, 0.1)
# compute targets
results = objective(inputs)
# create a line plot of input vs result
pyplot.plot(inputs, results)
# define optimal input value
x_optima = 0.67956
# draw a vertical line at the optimal input
pyplot.axvline(x=x_optima, ls='--', color='red')
# show the plot
pyplot.show()
```

Running the example creates a line plot of the function and marks the optima with a red line.




Line Plot of Non-Convex Unimodal Optimization Function

**Multimodal Functions**

A multi-modal function means a function with more than one "*mode*" or optima (e.g. valley).

Multimodal functions are non-convex.

There may be one global optima and one or more local or deceptive optima. Alternately, there may be multiple global optima, i.e. multiple different inputs that result in the same minimal output of the function.

Let's look at a few examples of multi-modal functions.

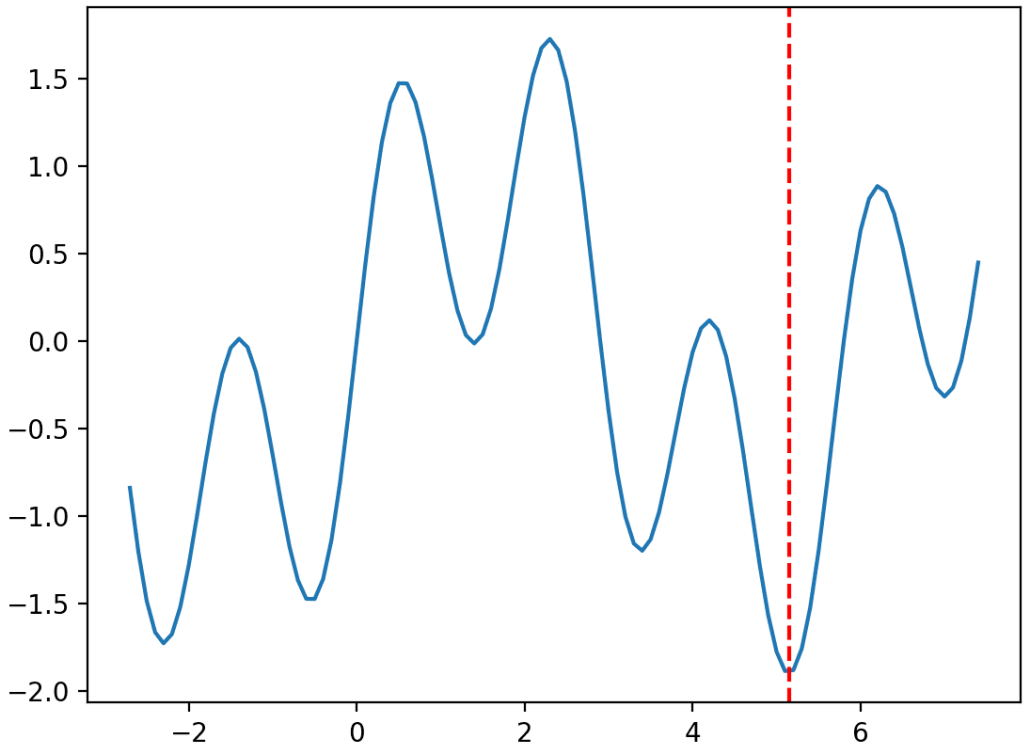
**Multimodal Function 1**

The range is bounded to -2.7 and 7.5 and the optimal input value is 5.145735.

```
1  # multimodal function
2  from numpy import sin
3  from numpy import arange
4  from matplotlib import pyplot
5
6  # objective function
7  def objective(x):
8   return sin(x) + sin((10.0 / 3.0) * x)
9
10 # define range for input
```

```
11 r_min, r_max = -2.7, 7.5
12 # sample input range uniformly at 0.1 increments
13 inputs = arange(r_min, r_max, 0.1)
14 # compute targets
15 results = objective(inputs)
16 # create a line plot of input vs result
17 pyplot.plot(inputs, results)
18 # define optimal input value
19 x_optima = 5.145735
20 # draw a vertical line at the optimal input
21 pyplot.axvline(x=x_optima, ls='--', color='red')
22 # show the plot
23 pyplot.show()
```

Running the example creates a line plot of the function and marks the optima with a red line.



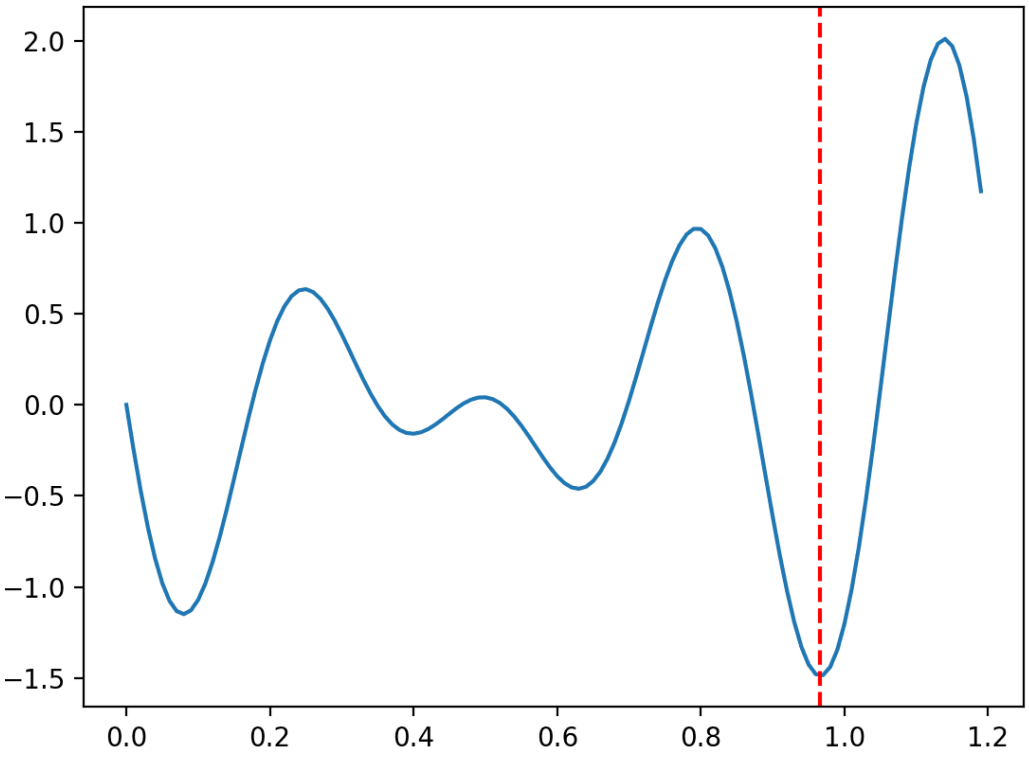

Line Plot of Multimodal Optimization Function 1

**Multimodal Function 2**

The range is bounded to 0.0 and 1.2 and the optimal input value is 0.96609.

```
# multimodal function
from numpy import sin
from numpy import arange
from matplotlib import pyplot

# objective function
def objective(x):
 return -(1.4 - 3.0 * x) * sin(18.0 * x)

# define range for input
r_min, r_max = 0.0, 1.2
# sample input range uniformly at 0.01 increments
inputs = arange(r_min, r_max, 0.01)
# compute targets
results = objective(inputs)
# create a line plot of input vs result
pyplot.plot(inputs, results)
# define optimal input value
x_optima = 0.96609
# draw a vertical line at the optimal input
pyplot.axvline(x=x_optima, ls='--', color='red')
# show the plot
pyplot.show()
```

Running the example creates a line plot of the function and marks the optima with a red line.



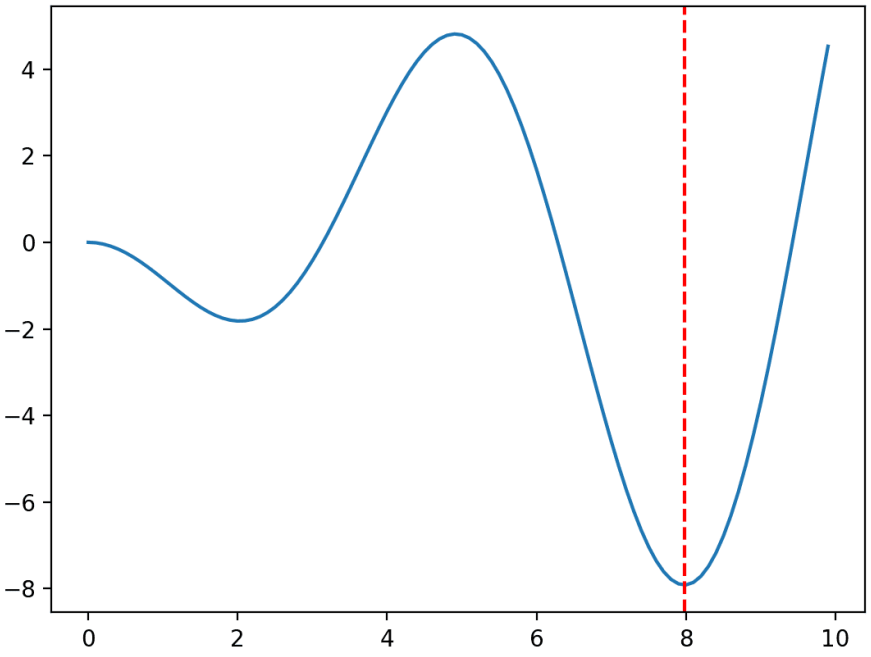

Line Plot of Multimodal Optimization Function 2

**Multimodal Function 3**

The range is bounded to 0.0 and 10.0 and the optimal input value is 7.9787.

```
1  # multimodal function
2  from numpy import sin
3  from numpy import arange
4  from matplotlib import pyplot
5
6  # objective function
7  def objective(x):
8   return -x * sin(x)
9
10 # define range for input
11 r_min, r_max = 0.0, 10.0
12 # sample input range uniformly at 0.1 increments
13 inputs = arange(r_min, r_max, 0.1)
14 # compute targets
15 results = objective(inputs)
16 # create a line plot of input vs result
17 pyplot.plot(inputs, results)
18 # define optimal input value
19 x_optima = 7.9787
20 # draw a vertical line at the optimal input
21 pyplot.axvline(x=x_optima, ls='--', color='red')
22 # show the plot
23 pyplot.show()
```

Running the example creates a line plot of the function and marks the optima with a red line.




Line Plot of Multimodal Optimization Function 3

**Discontinuous Functions (Non-Smooth)**

A function may have a discontinuity, meaning that the smooth change in inputs to the function may result in non-smooth changes in the output.

We might refer to functions with this property as non-smooth functions or discontinuous functions.

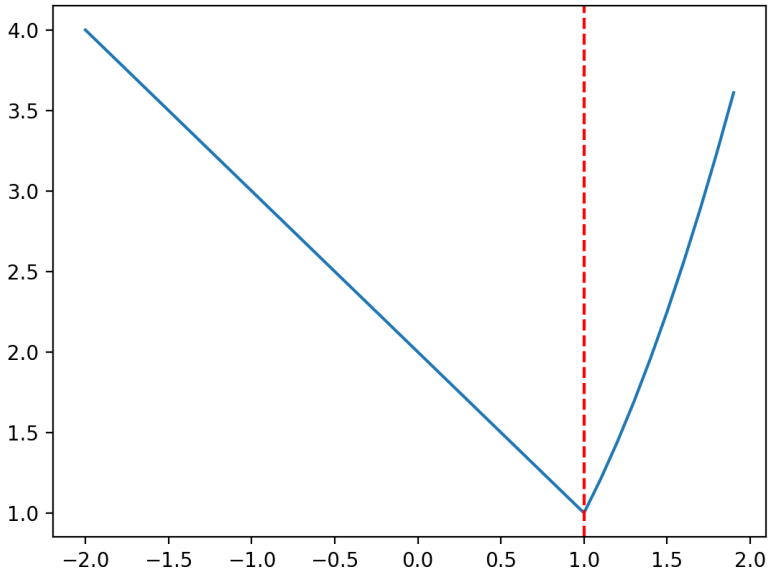
There are many different types of discontinuity, although one common example is a jump or acute change in direction in the output values of the function, which is easy to see in a plot of the function.

Discontinuous Function

The range is bounded to -2.0 and 2.0 and the optimal input value is 1.0.

```
# non-smooth optimization function
from numpy import arange
from matplotlib import pyplot

# objective function
def objective(x):
 if x > 1.0:
 return x**2.0
 elif x == 1.0:
 return 0.0
 return 2.0 - x

# define range for input
r_min, r_max = -2.0, 2.0
# sample input range uniformly at 0.1 increments
inputs = arange(r_min, r_max, 0.1)
# compute targets
results = [objective(x) for x in inputs]
# create a line plot of input vs result
pyplot.plot(inputs, results)
# define optimal input value
x_optima = 1.0
# draw a vertical line at the optimal input
pyplot.axvline(x=x_optima, ls='--', color='red')
# show the plot
pyplot.show()
```

Running the example creates a line plot of the function and marks the optima with a red line.




Line Plot of Discontinuous Optimization Function

**Noisy Functions**

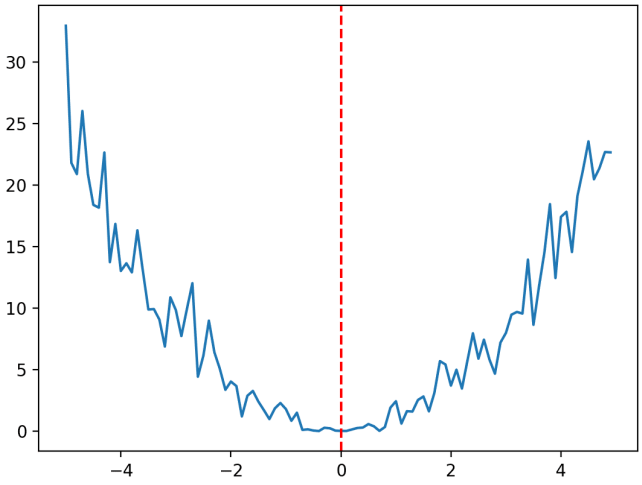
A function may have noise, meaning that each evaluation may have a stochastic component, which changes the output of the function slightly each time.

Any non-noisy function can be made noisy by adding small Gaussian random numbers to the input values.

The range for the function below is bounded to -5.0 and 5.0 and the optimal input value is 0.0.

```
# noisy optimization function
from numpy import arange
from numpy.random import randn
from matplotlib import pyplot

# objective function
def objective(x):
 return (x + randn(len(x))*0.3)**2.0

# define range for input
r_min, r_max = -5.0, 5.0
# sample input range uniformly at 0.1 increments
inputs = arange(r_min, r_max, 0.1)
# compute targets
results = objective(inputs)
# create a line plot of input vs result
pyplot.plot(inputs, results)
# define optimal input value
x_optima = 0.0
# draw a vertical line at the optimal input
pyplot.axvline(x=x_optima, ls='--', color='red')
# show the plot
pyplot.show()
```

Running the example creates a line plot of the function and marks the optima with a red line.




Line Plot of Noisy Optimization Function

## 4.3. Two-Dimensional (2D) Test Functions for Function Optimization

Function optimization is a field of study that seeks an input to a function that results in the maximum or minimum output of the function.

There are a large number of optimization algorithms and it is important to study and develop intuitions for optimization algorithms on simple and easy-to-visualize test functions. **Two-dimensional functions** take two input values (x and y) and output a single evaluation of the input. They are among the simplest types of test functions to use when studying function optimization. The benefit of two-dimensional functions is that they can be visualized as a contour plot or surface plot that shows the topography of the problem domain with the optima and samples of the domain marked with points.
In this tutorial, you will discover standard two-dimensional functions you can use when studying function optimization.

**Tutorial Overview**
A two-dimensional function is a function that takes two input variables and computes the objective value.

We can think of the two input variables as two axes on a graph, x and y. Each input to the function is a single point on the graph and the outcome of the function can be taken as the height on the graph.

This allows the function to be conceptualized as a surface and we can characterize the function based on the structure of the surface. For example, hills for input points that result in large relative outcomes of the objective function and valleys for input points that result in small relative outcomes of the objective function.

A surface may have one major feature or global optima, or it may have many with lots of places for an optimization to get stuck. The surface may be smooth, noisy, convex, and all manner of other properties that we may care about when testing optimization algorithms. There are many different types of simple two-dimensional test functions we could use. Nevertheless, there are standard test functions that are commonly used in the field of function optimization. There are also specific properties of test functions that we may wish to select when testing different algorithms.

We will explore a small number of simple two-dimensional test functions in this tutorial and organize them by their properties with two different groups; they are:

1. Unimodal Functions

    1. Unimodal Function 1
    2. Unimodal Function 2
    3. Unimodal Function 3
2. Multimodal Functions
    1. Multimodal Function 1
    2. Multimodal Function 2
    3. Multimodal Function 3

Each function will be presented using Python code with a function implementation of the target objective function and a sampling of the function that is shown as a surface plot.
All functions are presented as a minimization function, e.g. find the input that results in the minimum (smallest value) output of the function. Any maximizing function can be made a minimization function by adding a negative sign to all output. Similarly, any minimizing function can be made maximizing in the same way.
I did not invent these functions; they are taken from the literature. See the further reading section for references.

You can then choose and copy-paste the code one or more functions to use in your own project to study or compare the behavior of optimization algorithms.

**Unimodal Functions**

Unimodal means that the function has a single global optima.

A unimodal function may or may not be convex. A convex function is a function where a line can be drawn between any two points in the domain and the line remains in the domain. For a two-dimensional function shown as a contour or surface plot, this means the function has a bowl shape and the line between two remains above or in the bowl.

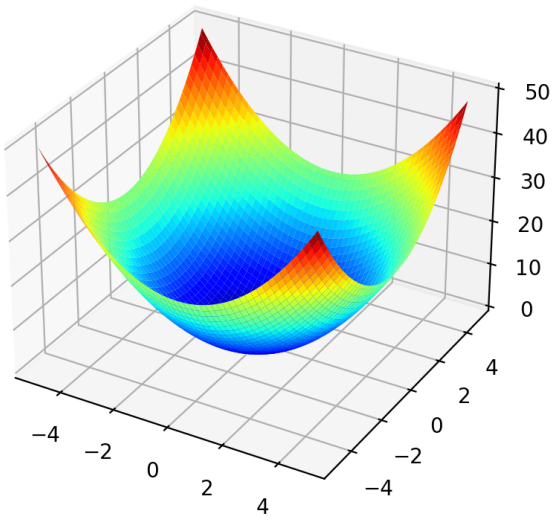
Let's look at a few examples of unimodal functions.

**Unimodal Function 1**

The range is bounded to -5.0 and 5.0 and one global optimal at [0.0, 0.0].

```
# unimodal test function
from numpy import arange
from numpy import meshgrid
from matplotlib import pyplot
from mpl_toolkits.mplot3d import Axes3D

# objective function
def objective(x, y):
 return x**2.0 + y**2.0

# define range for input
r_min, r_max = -5.0, 5.0
# sample input range uniformly at 0.1 increments
xaxis = arange(r_min, r_max, 0.1)
yaxis = arange(r_min, r_max, 0.1)
# create a mesh from the axis
x, y = meshgrid(xaxis, yaxis)
# compute targets
results = objective(x, y)
# create a surface plot with the jet color scheme
figure = pyplot.figure()
axis = figure.gca(projection='3d')
axis.plot_surface(x, y, results, cmap='jet')
# show the plot
pyplot.show()
```

Running the example creates a surface plot of the function.



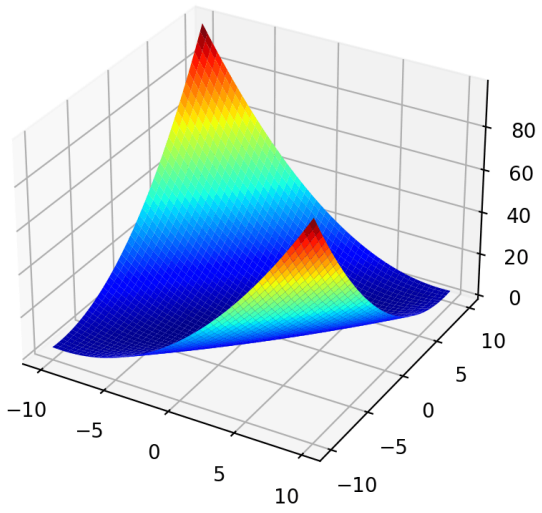

Surface Plot of Unimodal Optimization Function 1

**Unimodal Function 2**

The range is bounded to -10.0 and 10.0 and one global optimal at [0.0, 0.0].

```
# unimodal test function
from numpy import arange
from numpy import meshgrid
from matplotlib import pyplot
from mpl_toolkits.mplot3d import Axes3D

# objective function
def objective(x, y):
 return 0.26 * (x**2 + y**2) - 0.48 * x * y

# define range for input
r_min, r_max = -10.0, 10.0
# sample input range uniformly at 0.1 increments
xaxis = arange(r_min, r_max, 0.1)
yaxis = arange(r_min, r_max, 0.1)
# create a mesh from the axis
x, y = meshgrid(xaxis, yaxis)
# compute targets
results = objective(x, y)
# create a surface plot with the jet color scheme
figure = pyplot.figure()
axis = figure.gca(projection='3d')
axis.plot_surface(x, y, results, cmap='jet')
# show the plot
pyplot.show()
```

Running the example creates a surface plot of the function.



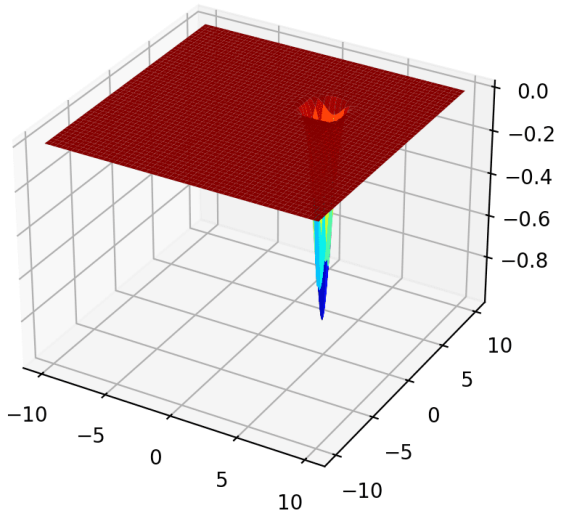

Surface Plot of Unimodal Optimization Function 2

**Unimodal Function 3**

The range is bounded to -10.0 and 10.0 and one global optimal at [pi, pi]. This function is known as Easom's function.

```
# unimodal test function
from numpy import cos
from numpy import exp
from numpy import pi
from numpy import arange
from numpy import meshgrid
from matplotlib import pyplot
from mpl_toolkits.mplot3d import Axes3D

# objective function
def objective(x, y):
 return -cos(x) * cos(y) * exp(-((x - pi)**2 + (y - pi)**2))

# define range for input
r_min, r_max = -10, 10
# sample input range uniformly at 0.01 increments
xaxis = arange(r_min, r_max, 0.01)
yaxis = arange(r_min, r_max, 0.01)
# create a mesh from the axis
x, y = meshgrid(xaxis, yaxis)
# compute targets
results = objective(x, y)
# create a surface plot with the jet color scheme
figure = pyplot.figure()
axis = figure.gca(projection='3d')
axis.plot_surface(x, y, results, cmap='jet')
# show the plot
pyplot.show()
```

Running the example creates a surface plot of the function.




Surface Plot of Unimodal Optimization Function 3

**Multimodal Functions**

A multi-modal function means a function with more than one "*mode*" or optima (e.g. valley). Multimodal functions are non-convex. There may be one global optima and one or more local or deceptive optima. Alternately, there may be multiple global optima, i.e. multiple different inputs that result in the same minimal output of the function.

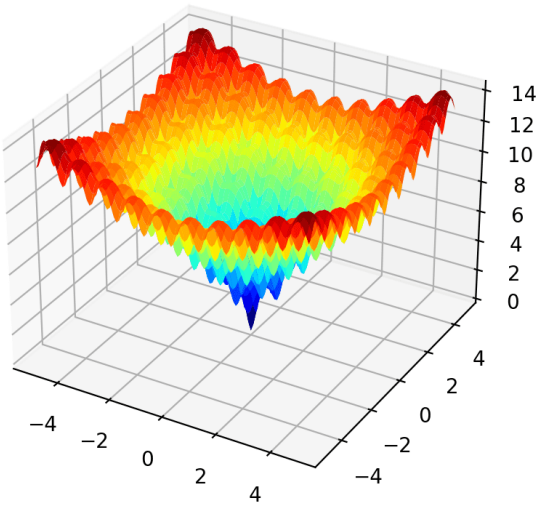
Let's look at a few examples of multimodal functions.

**Multimodal Function 1**

The range is bounded to -5.0 and 5.0 and one global optimal at [0.0, 0.0]. This function is known as Ackley's function.

```
# multimodal test function
from numpy import arange
from numpy import exp
from numpy import sqrt
from numpy import cos
from numpy import e
from numpy import pi
from numpy import meshgrid
from matplotlib import pyplot
from mpl_toolkits.mplot3d import Axes3D

# objective function
def objective(x, y):
 return -20.0 * exp(-0.2 * sqrt(0.5 * (x**2 + y**2))) - exp(0.5 * (cos(2 * pi * x) + cos(2 * pi * y))) + e + 20

# define range for input
r_min, r_max = -5.0, 5.0
# sample input range uniformly at 0.1 increments
xaxis = arange(r_min, r_max, 0.1)
yaxis = arange(r_min, r_max, 0.1)
# create a mesh from the axis
x, y = meshgrid(xaxis, yaxis)
# compute targets
results = objective(x, y)
# create a surface plot with the jet color scheme
figure = pyplot.figure()
axis = figure.gca(projection='3d')
axis.plot_surface(x, y, results, cmap='jet')
# show the plot
pyplot.show()
```

Running the example creates a surface plot of the function.



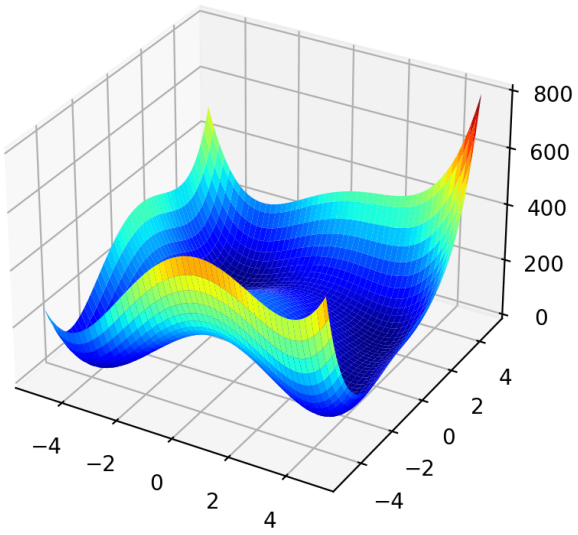

Surface Plot of Multimodal Optimization Function 1

**Multimodal Function 2**

The range is bounded to -5.0 and 5.0 and the function as four global optima at [3.0, 2.0], [-2.805118, 3.131312], [-3.779310, -3.283186], [3.584428, -1.848126]. This function is known as Himmelblau's function.

```
# multimodal test function
from numpy import arange
from numpy import meshgrid
from matplotlib import pyplot
from mpl_toolkits.mplot3d import Axes3D

# objective function
def objective(x, y):
 return (x**2 + y - 11)**2 + (x + y**2 -7)**2

# define range for input
r_min, r_max = -5.0, 5.0
# sample input range uniformly at 0.1 increments
xaxis = arange(r_min, r_max, 0.1)
yaxis = arange(r_min, r_max, 0.1)
# create a mesh from the axis
x, y = meshgrid(xaxis, yaxis)
# compute targets
results = objective(x, y)
# create a surface plot with the jet color scheme
figure = pyplot.figure()
axis = figure.gca(projection='3d')
axis.plot_surface(x, y, results, cmap='jet')
# show the plot
pyplot.show()
```

Running the example creates a surface plot of the function.




Surface Plot of Multimodal Optimization Function 2

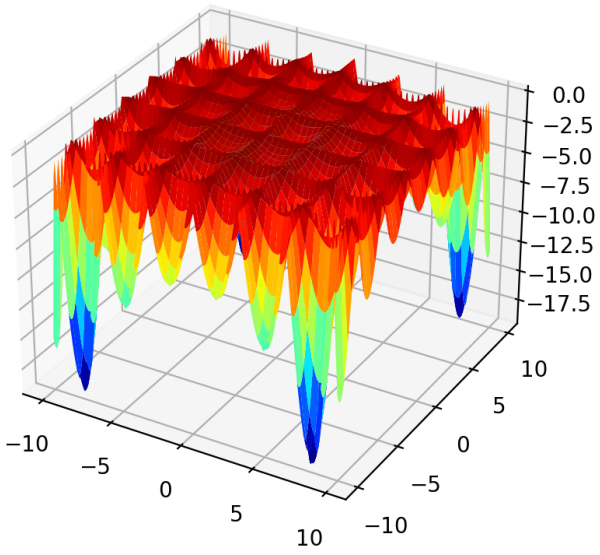
**Multimodal Function 3**

The range is bounded to -10.0 and 10.0 and the function as four global optima at [8.05502, 9.66459], [-8.05502, 9.66459], [8.05502, -9.66459], [-8.05502, -9.66459]. This function is known as Holder's table function.

```
    # multimodal test function
    from numpy import arange
1   from numpy import exp
2   from numpy import sqrt
3   from numpy import cos
4   from numpy import sin
5   from numpy import e
6   from numpy import pi
7   from numpy import absolute
8   from numpy import meshgrid
9   from matplotlib import pyplot
10 from mpl_toolkits.mplot3d import
11 Axes3D
12
13 # objective function
14 def objective(x, y):
15  return -absolute(sin(x) * cos(y) *
16 exp(absolute(1 - (sqrt(x**2 +
17 y**2)/pi))))
18
19 # define range for input
20 r_min, r_max = -10.0, 10.0
21 # sample input range uniformly at 0.1
22 increments
23 xaxis = arange(r_min, r_max, 0.1)
24 yaxis = arange(r_min, r_max, 0.1)
25 # create a mesh from the axis
26 x, y = meshgrid(xaxis, yaxis)
27 # compute targets
28 results = objective(x, y)
29 # create a surface plot with the jet
30 color scheme
31 figure = pyplot.figure()
32 axis = figure.gca(projection='3d')
    axis.plot_surface(x, y, results,
    cmap='jet')
```

```
    # show the plot
    pyplot.show()
```

Running the example creates a surface plot of the function.




Surface Plot of Multimodal Optimization Function 3

## 4.4. Optimization algorithms: the Newton Method

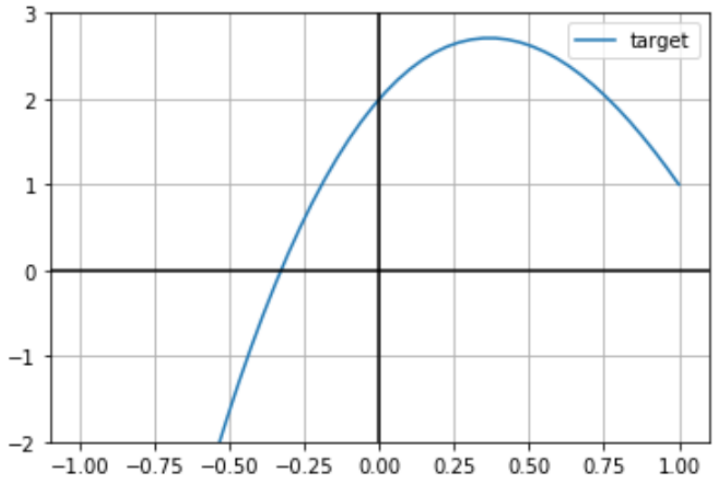
Predictive Statistics and Machine Learning aim at building models with parameters such that the final output/prediction is as close as possible to the actual value. This implies the optimization of an objective function, which might be either minimized (like loss functions) or maximized (like Maximum Likelihood function).

The idea behind optimization routine is starting from an initial random value of our target function's variable, and then updating this value according to a given rule. The iteration stops when the distance between two values is approaching zero, or maybe after a maximum number of iterations arbitrarily decided.

In this article, I'm going to provide an intuitive explanation of the Newton method, proposing a geometric interpretation. The assumption behind this method is that our target function f(x), the one we want to optimize, is twice differentiable and f"(x) is not equal to zero. Here, we will be working with a smooth and regular polynomial:

$$f(x) = x^3 - 6x^2 + 4x + 2$$

```
import numpy as np
import math
import matplotlib.pyplot as plt
def f(x):
   return x**3-6*x**2+4*x+2
x = np.linspace(-1, 1)
fig, ax = plt.subplots()
ax.plot(x, f(x), label='target')
ax.grid()
```




Now, the idea of Newton method is that, in order to find the optimum (in our case, the maximum) of our target, we have to start by approximating our curve in a random starting point with a second order Taylor extension, then compute the maximum of that extension

(that means, taking its first derivative and setting it equal to zero). The maximum of the quadratic approximation will be the new value of our x. Then, with an iterative procedure, this is repeated and the resulting series of x, xn, tends to converge towards the maximum of the target, x*, as n tends towards infinite.
However, let's proceed step by step, and let's first compute the Taylor expansion of our function at a given point x0:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2}f''(x_0)(x - x_0)^2$$

Now, we want the first derivative of this quantity to be equal to 0, since we want to compute the optimum of this new curve. More specifically, we want to differentiate with respect to (x-x0), which is the increment needed to reach the maximum of the approximation. Hence:

$$\frac{\partial\left[f(x_0) + f'(x_0)(x - x_0) + \frac{1}{2}f''(x_0)(x - x_0)^2\right]}{\partial(x - x_0)} = 0$$

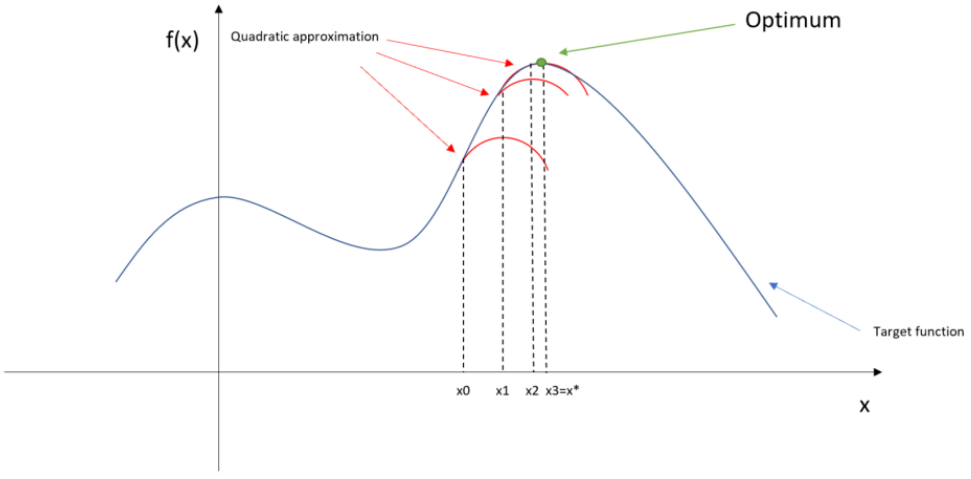
$$f'(x_0) + f''(x_0)(x - x_0) = 0$$

$$x = x_0 - \frac{f'(x_0)}{f''(x_0)}$$

This formula can be generalized with the following updating rule:

$$x_{t+1} = x_t - \frac{f'(x_0)}{f''(x_0)}$$

Let's see it geometrically:



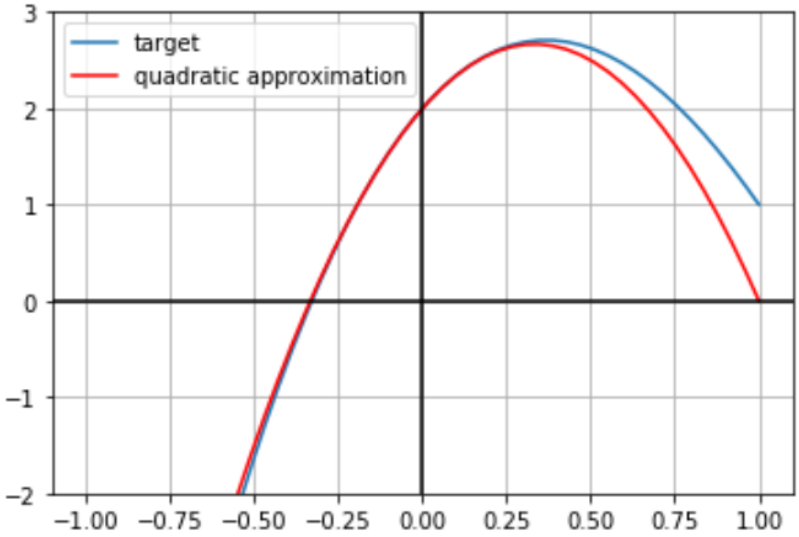
As you can see, as xn approaches x*, the distance between two x values approaches towards zero. As anticipated, since it might be possible that xn and x* never perfectly match, one might impose a stopping criterion when, for example, the updating factor is less than an arbitrarily small epsilon.

Now let's implement it in Python, using as target the function we already defined. To proceed with Newton method, we have to define three further elements: first and second order derivative, and the quadratic approximation:

```
1       def fprime(x):
2           return 3*x**2-12*x+4
3       def fsecond(x):
4           return 6*x - 12
5       def quadratic_approx(x, x0, f, fprime, fsecond):
6           return f(x0)+fprime(x0)*(x-x0)+0.5*fsecond(x0)*(x-x0)**2
```

Let's first have a look at the plot of the quadratic approximation in x0=0:

```
x = np.linspace(-1, 1)
fig, ax = plt.subplots()
ax.plot(x, f(x), label='target')
ax.grid()
ax.plot(x, quadratic_approx(x, 0, f, fprime, fsecond), color='red', label='quadratic
approximation')
ax.set_ylim([-2,3])
ax.axhline(y=0, color='k')
ax.axvline(x=0, color='k')
plt.legend()
```



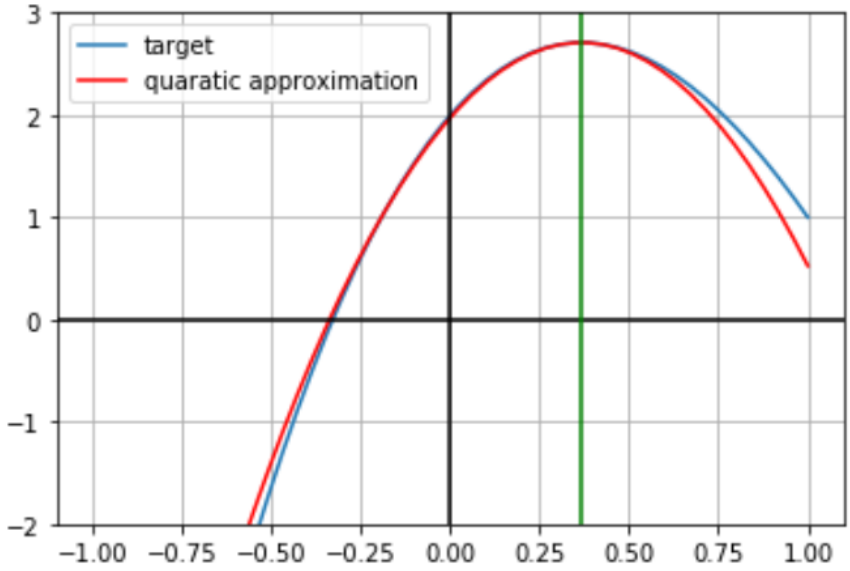

Now let's define our Newton method:

```
1       def newton(x0, fprime, fsecond, maxiter=100, eps=0.0001):
2           x=x0
```

```
3           for i in range(maxiter):
4               xnew=x-(fprime(x)/fsecond(x))
5               if xnew-x<eps:
6                   return xnew
7                   print('converged')
8                   break
9               x = xnew
10          return x
```

Let's plot the result with the x_star:

```
x_star=newton(0, fprime, fsecond)
fig, ax = plt.subplots()
ax.plot(x, f(x), label='target')
ax.grid()
ax.plot(x, quadratic_approx(x, x_star , f, fprime, fsecond), color='red', label='quaratic approximation')
ax.set_ylim([-2,3])
ax.axhline(y=0, color='k')
ax.axvline(x=0, color='k')
ax.axvline(x = x_star, color='green')
plt.legend()
```




As you can see, now the maximum of the linear approximation coincides (approximately) with the maximum of our target. Of course, the iteration didn't last a lot, since we started from a x0 pretty close to the real maximum of the function. Nevertheless, this is the exact procedure whatever the starting point is.

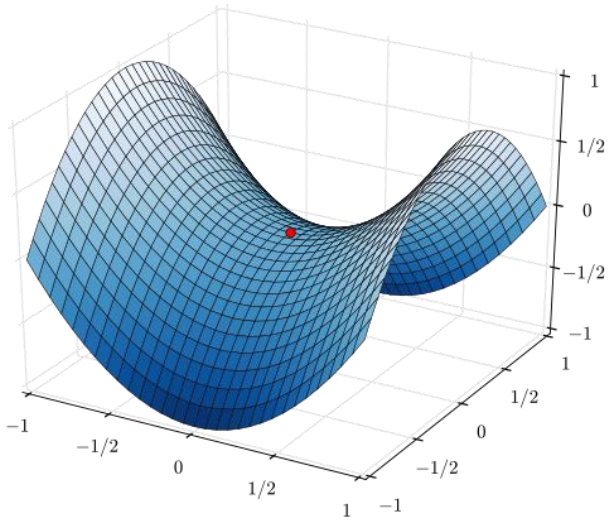
Newton method is extremely powerful, yet it can be applied only under the assumption of twice differentiability, differently from other optimization methods as gradient descent

(which only requires first differentiability). Hence, you might want to use this method only in case of 'well-behaving' function, like our polynomial of the example above.

**Newton method attracts to saddle points**

saddle points are common in machine learning, or in fact any multivariable optimization.

Look at the function: $f=x^2-y^2$




If you apply multivariate Newton method, you get the following.

$$\mathbf{x}_{n+1} = \mathbf{x}_n - [\mathbf{H}f(\mathbf{x}_n)]^{-1}\nabla f(\mathbf{x}_n)$$

Let's get the Hessian:

$$\mathbf{H} = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

$$\mathbf{H} = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}$$

Invert it:

$$[\mathbf{H}f]^{-1} = \begin{bmatrix} 1/2 & 0 \\ 0 & -1/2 \end{bmatrix}$$

Get the gradient:

$$\nabla f = \begin{bmatrix} 2x \\ -2y \end{bmatrix}$$

Get the final equation:

$$\begin{bmatrix} x \\ y \end{bmatrix}_{n+1} = \begin{bmatrix} x \\ y \end{bmatrix}_n - \begin{bmatrix} 1/2 & 0 \\ 0 & -1/2 \end{bmatrix} \begin{bmatrix} 2x_n \\ -2y_n \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix}_n - \begin{bmatrix} x \\ y \end{bmatrix}_n = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

So, you see how the Newton method led you to the saddle point at x=0,y=0.

In contrast, the gradient descent method will not lead to the saddle point. The gradient is zero at the saddle point, but a tiny step out would pull the optimization away as you can see from the gradient above - its gradient on y-variable is negative.

# 5. Fourier Transform

Fourier dönüşümü, sadece sinüs ve kosinüs temel işlevleri kullanılarak sentezlenebilen sinyalleri analiz etmek için yeterlidir.

Dünyadaki birçok şey, bir dalga formu aracılığıyla tanımlanabilir - zaman, mekan veya başka bir değişkenin fonksiyonu. Örneğin, ses dalgaları, elektromanyetik alanlar, radyodan dinlediğiniz müzik, hisse senetlerini zamana göre fiyatı, nefesinizin sıklığı vb.

Fourier Dönüşümü, bize bu dalga formlarını doğrudan görüntülemenin benzersiz ve güçlü bir yolunu sunduğu için önemli bir rol oynamaktadır. Fourier Transform, görüntü analizi, görüntü filtreleme, görüntü rekonstrüksiyonu ve görüntü sıkıştırması gibi çok çeşitli uygulamalarda kullanılır.

Bir Fransız matematikçi ve fizikçi Jean Baptiste Joseph Fourier, Fourier analizini geliştirdi. Periyodik sinyalin uygun seçilmiş sinüzoidal dalgaların toplamı olarak temsil edilebileceği konusunda tartışmalı bir iddiaya sahipti. Bu yazının bir gözden geçiricisi olan matematikçi Lagrange, süreksiz eğimler gibi köşeleri olan sinyalleri temsil etmek için bir yaklaşımın kullanılamayacağı konusunda ısrar etti. Lagrange'ın görüşü doğruydu ama tam olarak değil, çünkü sıfır enerjiye sahip iki sinüzoidal işaret arasındaki fark çok yakındı. Makale sonunda Lagrange öldükten sonra yayınlandı. Fourier'in genelleme iddiasının biraz kuvvetli olduğu ortaya çıksa da, sonuçları günümüze kadar devam eden önemli bir araştırma selini harekete geçirdi.

Fourier dönüşümü fizik ve mühendislik alanındaki birçok uygulama ile matematiksel bir dönüşümdür. Fourier serileri denilen trigonometrik serileri kullanarak kısmi diferansiyel denklemleri içeren birçok önemli problemi çözebiliriz.

Evrende gözlediğiniz tüm dalga formları farklı frekans ve genliklere sahip sinüs fonksiyonlarının toplamından ibarettir!

## The Fourier transform

we'll be interested in signals defined for all $t$

the **Fourier transform** of a signal $f$ is the function

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt$$

- $F$ is a function of a *real* variable $\omega$; the function value $F(\omega)$ is (in general) a complex number

$$F(\omega) = \int_{-\infty}^{\infty} f(t)\cos\omega t\,dt - j\int_{-\infty}^{\infty} f(t)\sin\omega t\,dt$$

- $|F(\omega)|$ is called the *amplitude spectrum* of $f$; $\angle F(\omega)$ is the *phase spectrum* of $f$
- notation: $F = \mathcal{F}(f)$ means $F$ is the Fourier transform of $f$; as for Laplace transforms we usually use uppercase letters for the transforms (*e.g.*, $x(t)$ and $X(\omega)$, $h(t)$ and $H(\omega)$, etc.)

## Fourier transform and Laplace transform

Laplace transform of $f$

$$F(s) = \int_{0}^{\infty} f(t)e^{-st}\,dt$$

Fourier transform of $f$

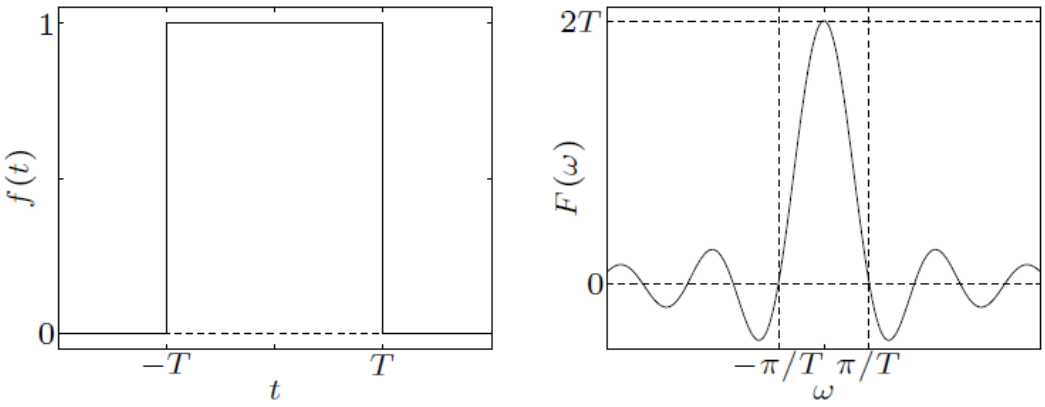
$$G(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}\,dt$$

very similar definitions, with two differences:

- Laplace transform integral is over $0 \leq t < \infty$; Fourier transform integral is over $-\infty < t < \infty$
- Laplace transform: $s$ can be any complex number in the region of convergence (ROC); Fourier transform: $j\omega$ lies on the imaginary axis

**rectangular pulse:** $f(t) = \begin{cases} 1 & -T \le t \le T \\ 0 & |t| > T \end{cases}$

$$F(\omega) = \int_{-T}^{T} e^{-j\omega t}\, dt = \frac{-1}{j\omega}\left(e^{-j\omega T} - e^{j\omega T}\right) = \frac{2\sin \omega T}{\omega}$$




**unit impulse:** $f(t) = \delta(t)$

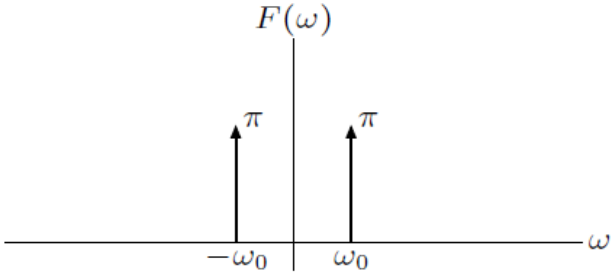
$$F(\omega) = \int_{-\infty}^{\infty} \delta(t) e^{-j\omega t}\, dt = 1$$

## Fourier transform of periodic signals

similarly, by allowing impulses in $\mathcal{F}(f)$, we can define the Fourier transform of a periodic signal

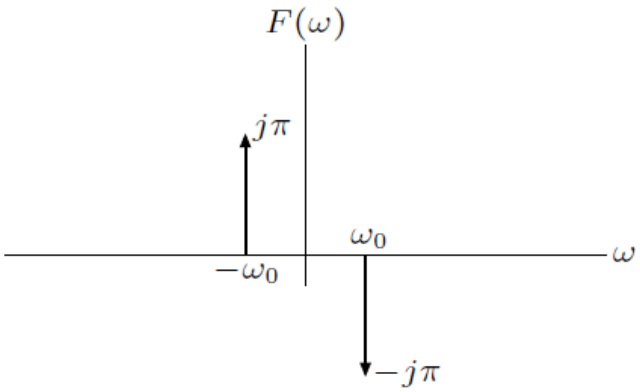
**sinusoidal signals:** Fourier transform of $f(t) = \cos \omega_0 t$

$$\begin{aligned} F(\omega) &= \frac{1}{2}\int_{-\infty}^{\infty} \left(e^{j\omega_0 t} + e^{-j\omega_0 t}\right) e^{-j\omega t}\, dt \\ &= \frac{1}{2}\int_{-\infty}^{\infty} e^{-j(\omega-\omega_0)t} dt + \frac{1}{2}\int_{-\infty}^{\infty} e^{-j(\omega+\omega_0)t} dt \\ &= \pi\delta(\omega-\omega_0) + \pi\delta(\omega+\omega_0) \end{aligned}$$



Fourier transform of $f(t) = \sin \omega_0 t$

$$\begin{aligned} F(\omega) &= \frac{1}{2j} \int_{-\infty}^{\infty} \left( e^{j\omega_0 t} - e^{-j\omega_0 t} \right) e^{-j\omega t} \, dt \\ &= \frac{1}{2j} \int_{-\infty}^{\infty} e^{-j(\omega-\omega_0)t} dt + -\frac{1}{2j} \int_{-\infty}^{\infty} e^{-j(\omega_0+\omega)t} dt \\ &= -j\pi\delta(\omega - \omega_0) + j\pi\delta(\omega + \omega_0) \end{aligned}$$




## Examples

**sign function:** $f(t) = \begin{cases} 1 & t \geq 0 \\ -1 & t < 0 \end{cases}$

write $f$ as $f(t) = -1 + 2g(t)$, where $g$ is a unit step at $t = 0$, and apply linearity

$$F(\omega) = -2\pi\delta(\omega) + 2\pi\delta(\omega) + \frac{2}{j\omega} = \frac{2}{j\omega}$$

**sinusoidal signal:** $f(t) = \cos(\omega_0 t + \phi)$

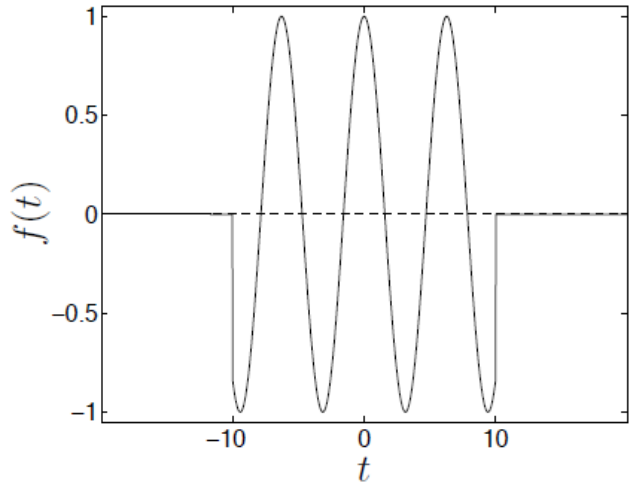
write $f$ as

$$f(t) = \cos(\omega_0(t + \phi/\omega_0))$$

and apply time shift property:

$$F(\omega) = \pi e^{j\omega\phi/\omega_0} \left( \delta(\omega - \omega_0) + \delta(\omega + \omega_0) \right)$$

**pulsed cosine:** $f(t) = \begin{cases} 0 & |t| > 10 \\ \cos t & -10 \le t \le 10 \end{cases}$




write $f$ as a product $f(t) = g(t)\cos t$ where $g$ is a rectangular pulse of width 20 (see page 12-7)

$$\mathcal{F}(\cos t) = \pi\delta(\omega - 1) + \pi\delta(\omega + 1), \quad \mathcal{F}(g(t)) = \frac{2\sin 10\omega}{\omega}$$

## The inverse Fourier transform

if $F(\omega)$ is the Fourier transform of $f(t)$, *i.e.*,

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}\,dt$$

then

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{j\omega t}\,d\omega$$

let's check

$$\begin{aligned} \frac{1}{2\pi}\int_{\omega=-\infty}^{\infty} F(\omega)e^{j\omega t}\,d\omega &= \frac{1}{2\pi}\int_{\omega=-\infty}^{\infty}\left(\int_{\tau=-\infty}^{\infty} f(\tau)e^{-j\omega\tau}\right)e^{j\omega t}d\omega \\ &= \frac{1}{2\pi}\int_{\tau=-\infty}^{\infty} f(\tau)\left(\int_{\omega=-\infty}^{\infty} e^{-j\omega(\tau-t)}d\omega\right)d\tau \\ &= \int_{-\infty}^{\infty} f(\tau)\delta(\tau - t)d\tau \\ &= f(t) \end{aligned}$$

## Fourier Transform:

$$F(j\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt$$

$$\int e^{-j\omega t}dt = \frac{1}{-j\omega}e^{-j\omega t}$$

1. Find the Fourier series for (periodic extension of)

$$f(t) = \begin{cases} 1, & t \in [0,2); \\ -1, & t \in [2,4). \end{cases}$$

Determine the sum of this series.

2. Find the Fourier series for (periodic extension of)

$$f(t) = \begin{cases} t-1, & t \in [0,2); \\ 3-t, & t \in [2,4). \end{cases}$$

Determine the sum of this series.

3. Find the sine Fourier series for (periodic extension of)

$$f(t) = \begin{cases} t-1, & t \in [0,2); \\ 3-t, & t \in [2,4). \end{cases}$$

Determine the sum of this series.

4. Find the cosine Fourier series for (periodic extension of)

$$f(t) = \begin{cases} 1, & t \in [0,1); \\ 0, & t \in [1,4). \end{cases}$$

Determine the sum of this series.

5. Find the Fourier series for (periodic extension of)

$$f(t) = 1 - t^2,\ t \in [-1,1).$$

Determine the sum of this series.

**Soru:**

The Fourier series for the function $f(x) = \sin^2 x$ is

$$f(x) = \sin^2 x = \frac{1 - \cos 2x}{2}$$

$$= 0.5 - 0.5 \cos 2x$$

$$f(x) = A_0 + \sum_{n=1}^{\infty} a_n \cos n\omega_0 x + b_n \sin n\omega_0 x$$

$f(x) = \sin^2 x$ is an even function so $b_n = 0$

$$A_0 = 0.5$$

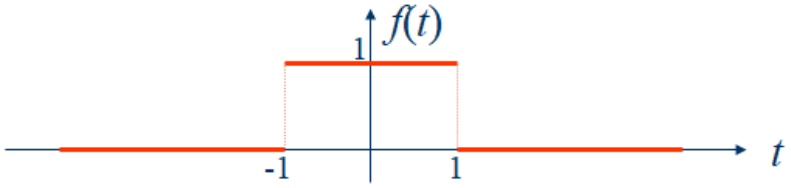
$$a_n = \begin{cases} -0.5, & n = 1 \\ 0 & , \text{ otherwise} \end{cases}$$

$$\omega_0 = \frac{2\pi}{T_0} = \frac{2\pi}{T} = 2$$

**Soru-15:**

Fourier Transform:

$$F(j\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt$$




Şekilde verilmiş olan *f(t)'nin fourier dönüşümünü bulunuz.*

$$\int_{-1}^{1} e^{-j\omega t}dt = \frac{1}{-j\omega}e^{-j\omega t}\Big|_{-1}^{1}$$

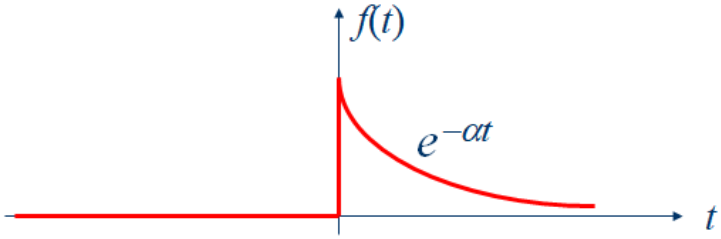
$$F(j\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt = \int_{-1}^{1} e^{-j\omega t}dt = \frac{1}{-j\omega}e^{-j\omega t}\Big|_{-1}^{1}$$

$$= \frac{j}{\omega}(e^{-j\omega} - e^{j\omega}) = \frac{2\sin\omega}{\omega}$$

**Soru-16:**

Fourier Transform:

$$F(j\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt$$




Şekilde verilmiş olan *f(t)'nin fourier dönüşümünü bulunuz.*

$$\int e^{-j\omega t} dt = \frac{1}{-j\omega} e^{-j\omega t}$$

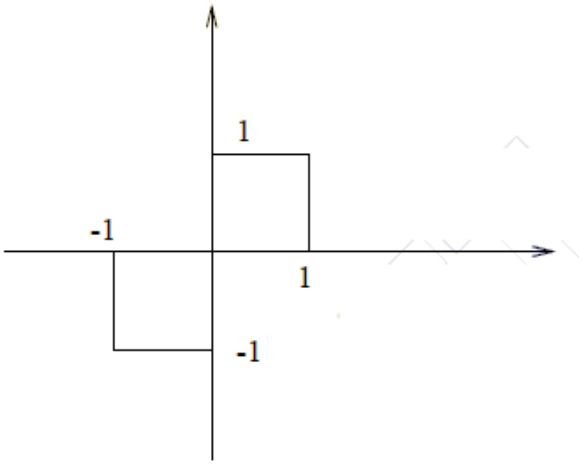
$$F(j\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt = \int_{0}^{\infty} e^{-\alpha t} e^{-j\omega t} dt$$

$$= \int_{0}^{\infty} e^{-(\alpha + j\omega)t} dt = \frac{1}{\alpha + j\omega}$$

**Soru-3:**

$$x(t) = \begin{cases} -1 & -1 \le t < 0 \\ 1 & 0 \le t < 1 \\ 0 & \text{elsewhere} \end{cases}$$

a) X(t) fonksiyonunu çiziniz




b) Fourier dönüşümünü bulunuz

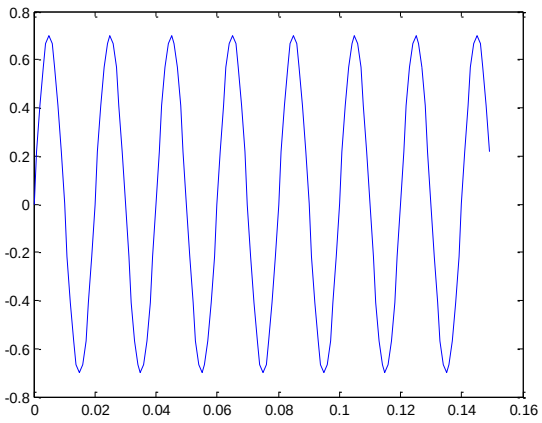
$$\begin{aligned} X(\omega) &= -\int_{-1}^{0} e^{-j\omega t}dt + \int_{0}^{1} e^{-j\omega t}dt \\ &= -\int_{0}^{1} e^{j\omega t}dt + \int_{0}^{1} e^{-j\omega t}dt \\ &= -2j\int_{0}^{1} \sin(\omega t)\,dt \\ &= 2j\frac{1}{\omega}\cos(\omega t)\Big|_{0}^{1} \\ &= 2j\frac{1}{\omega}(\cos(\omega) - 1) \end{aligned}$$

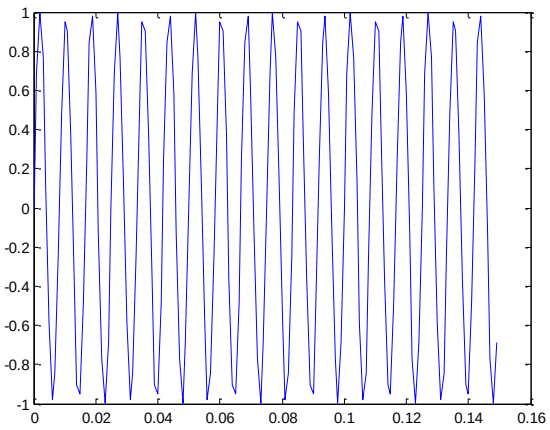
## 5.1. Matlab ile Sinyal Analizi

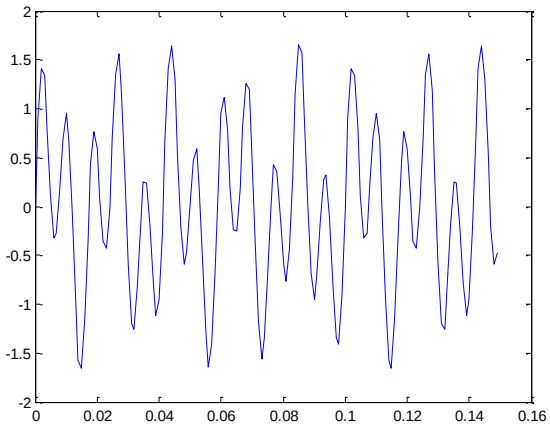
Y = fft(X) computes the discrete Fourier transform (DFT) of X using a fast Fourier transform (FFT) algorithm.
Y = fft(X,n) returns the n-point DFT. If no value is specified, Y is the same size as X.

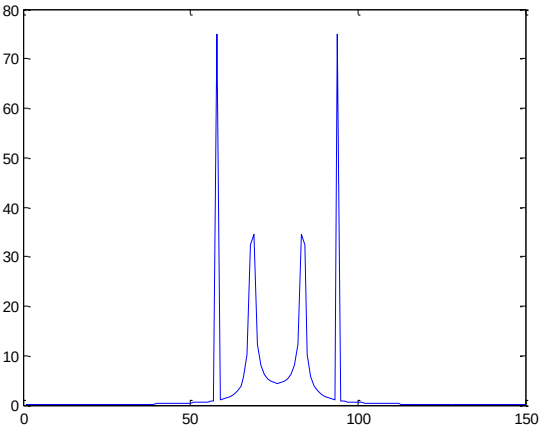
**Örnek:**
Gürültülü Sinyal
Use Fourier transforms to find the frequency components of a signal buried in noise. Specify the parameters of a signal with a sampling frequency of 1000Hz and a signal duration of 150 ms.

```
clear all
close all
Fs = 1000;          % Sampling frequency
T = 1/Fs;           % Sampling period
L = 150;            % Length of signal
t = (0:L-1)*T;      % Time vector

S1 = 0.7*sin(2*pi*50*t);
S2 = sin(2*pi*120*t);
figure, plot(t,S1)
figure, plot(t,S2)

S=S1+S2;
figure, plot(t,S)
```

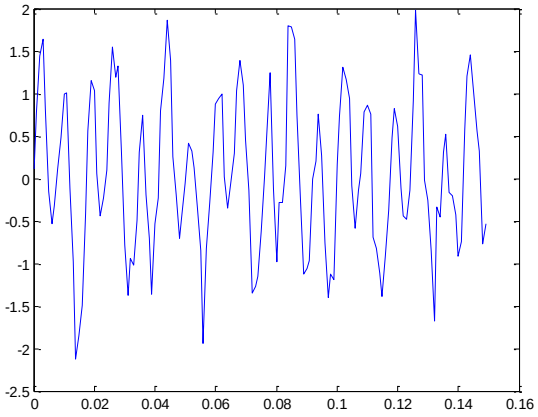


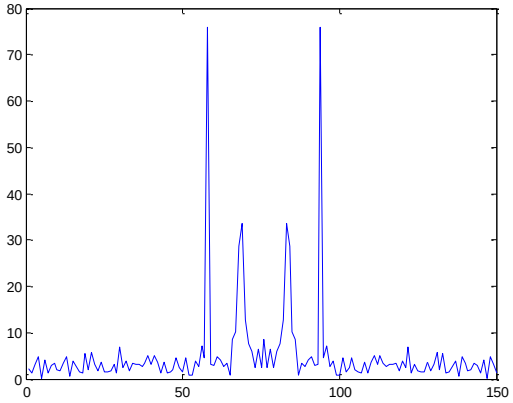





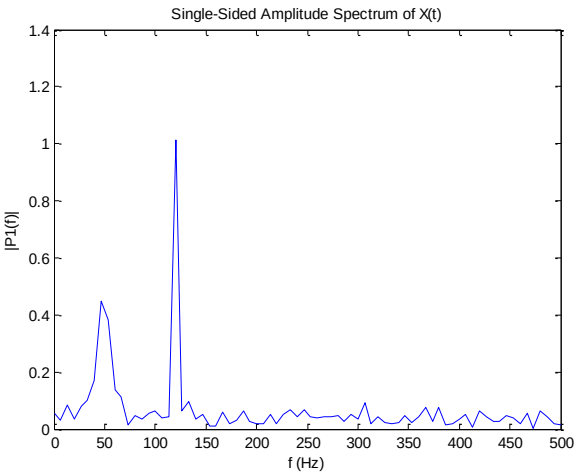




```
Y = fft(S);
figure, plot(fftshift(abs(Y)))

G = S + 0.25*randn(size(t));
figure, plot(t,G)
FG=fft(G)
figure, plot(fftshift(abs(FG)))
```



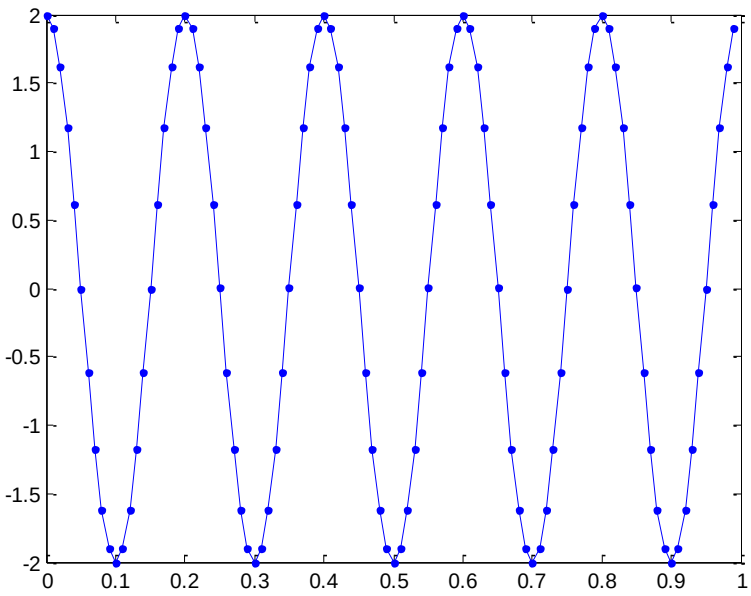

```
P2 = abs(FG/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
f = Fs*(0:(L/2))/L;
figure, plot(f,P1)
title('Single-Sided Amplitude Spectrum of G(t)')
xlabel('f (Hz)')
ylabel('|P1(f)|')
```



## 5.2. Fourier analysis and Matlab

Başlangıç olarak, bazı basit 1D sinyalleri oluşturalım ve Fourier dönüşümlerini inceleyelim. Bu ilk örnekte, basit bir periyodik sinyalin frekans içeriğini kontrol edeceğiz. Verilen büyüklük, frekans, örnekleme oranı ve süreye sahip bir kosinüs sinyali oluşturun:

```
clear all
 close all
  mag   = 2;      % magnitude (arbitrary units)
  f = 5;      % frequency in Hz
  Ps=50;         % number of sampling on a periot
  samp = f*Ps;    % sampling rate in Hz
  t         = 0:1/samp:1-1/samp;  % time (1s of data)
  N         = length(t)
  x = mag*cos(2*pi*f*t);     % the signal equation
  figure
  plot(t,x,'.-');
```




Örnekleme hızı (saniyedeki örnek sayısı) ile sinyalin frekansı arasındaki ilişkinin iyi anlaşılması gerekir. Süresi 1 saniye olan bir sinyal üretildiğinden, saniyedeki döngü sayısı kolayca hesaplanabilir:

Frekans=5Hz ise bir sinyalin peryodu=T=1/5=0.2 sec.

1 saniyede üretilecek sinyal sayısı=1 saniye * frekans=5 adettir.

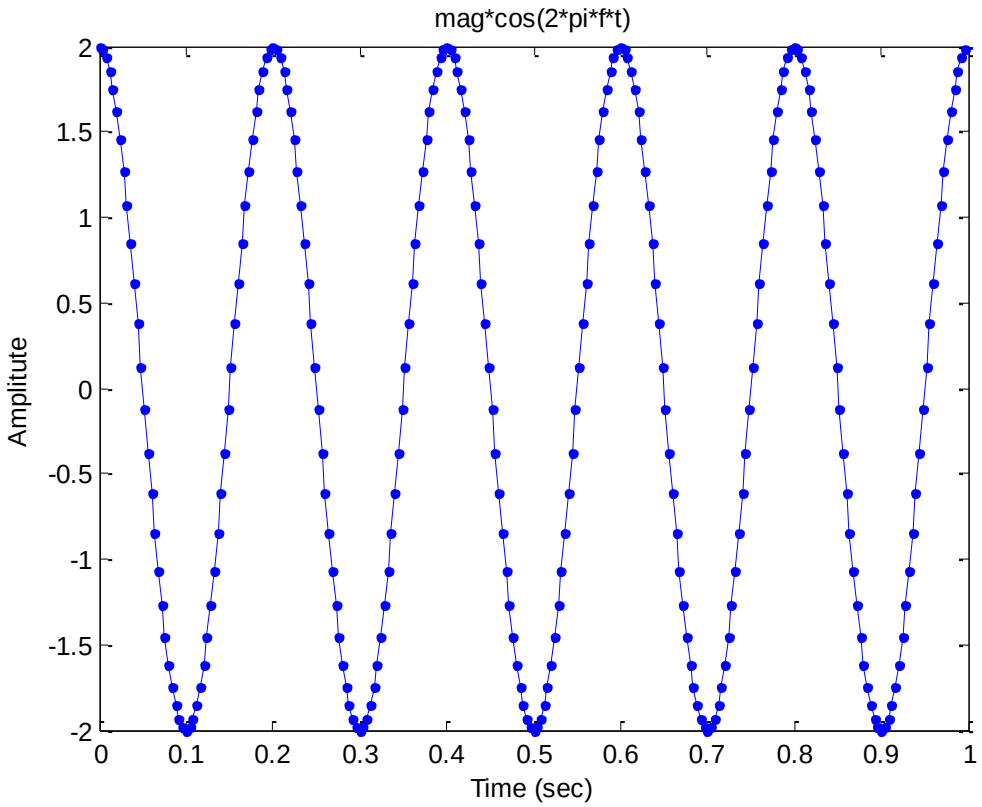
Bir peryotluk sinyaldeki örnek sayısı=toplam örnek sayısı/sinyal sayısı=100/5=20 adettir.

**Örnek:**

1 saniyede üretilecek sinyal sayısı=2 adet ise T=1/2=0.5sec, f=1/T=2Hz,

Toplam örnek sayısı=Bir peryotluk sinyaldeki örnek sayısı * sinyal sayısı=40*2=80 adettir.

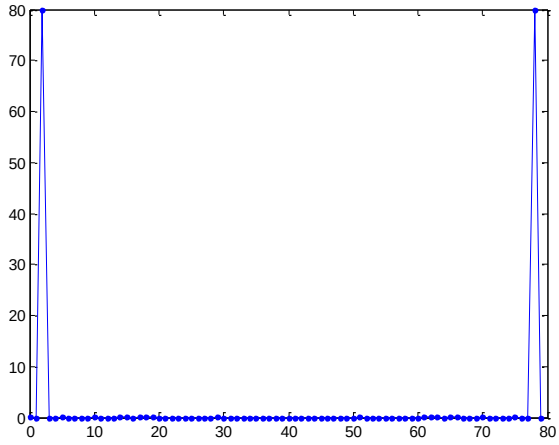
```
clear all
 close all
  mag   = 2;      % magnitude (arbitrary units)
  f = 5;      % frequency in Hz
  Ps=50;         % number of sampling on a periot
  samp = f*Ps;    % sampling rate in Hz
  t         = 0:1/samp:1-1/samp;  % time (1s of data)
  N         = length(t)
  x = mag*cos(2*pi*f*t);     % the signal equation
  figure
  plot(t,x,'.-');
  xlabel('Time (sec)');
  ylabel('Amplitute');
  title('mag*cos(2*pi*f*t)')
```




Son zaman noktasının kaldırılma nedeni, FFT yaparken, matlab, sinyali periyodik varsayar ve kendisini süresiz olarak tekrarlar. Matlab'ın sinyalin saf sonsuz bir kosinüs fonksiyonu olduğunu görmesi gerekir.

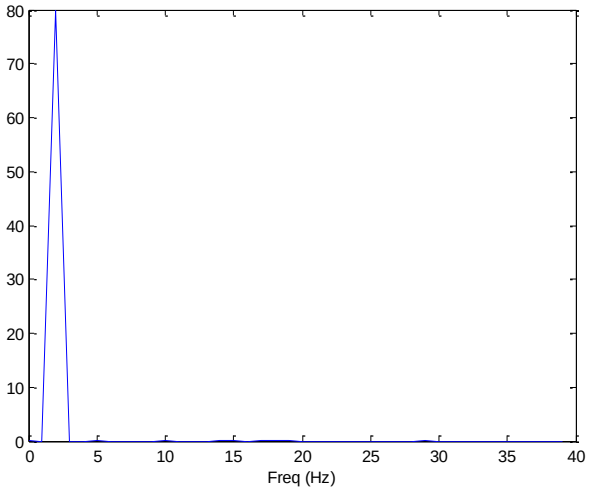
Bu sinyalin Fourier dönüşümünü alalım,

```
clear all
close all
  mag  = 2;      % magnitude (arbitrary units)
  freq = 2;      % frequency in Hz
  samp = 80;    % sampling rate in Hz
  t        = 0:1/samp:1-1/samp;  % time (1s of data)
  N        = length(t)
  x = mag*cos(2*pi*freq*t);     % the signal equation
  figure,   plot(t,x,'.-');

  y  = fft(x);              % do Fast Fourier Transform
  f  = linspace(0,N-1,N);  % vector of frequencies for plotting
  figure,  plot(f,abs(y),'.-');     % plotting the magnitude of the FFT
  figure,  plot(f(1:N/2),abs(y(1:N/2))); % plotting half of the fft results
  xlabel('Freq (Hz)');           % labelling x-axis
```




Bu çizime **güç spektrumu** denir.

İlk yükselti kosinüs fonksiyonunun frekansı ile çakışıri. İkinci zirve simetriğidir. Bu nedenle, fourier dönüşümünün sadece ilk yarısını gösterilir:




**Matlab'da (ifft) ters FFT işlevi:**

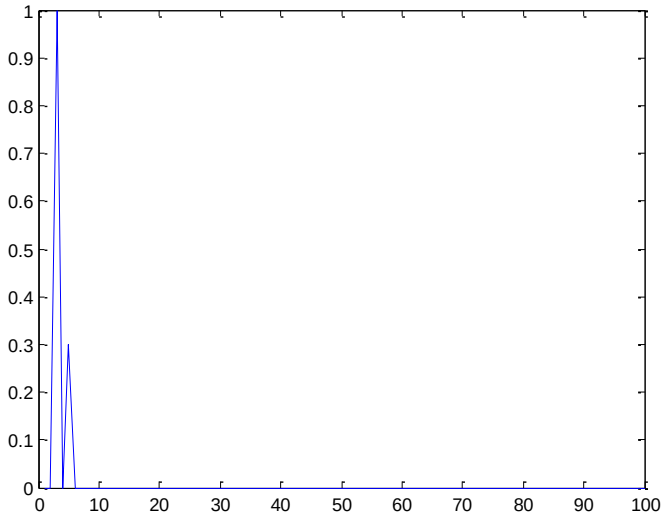
Biri büyüklük ve diğeri faz olan iki sinyalin ters fourier dönüşümü

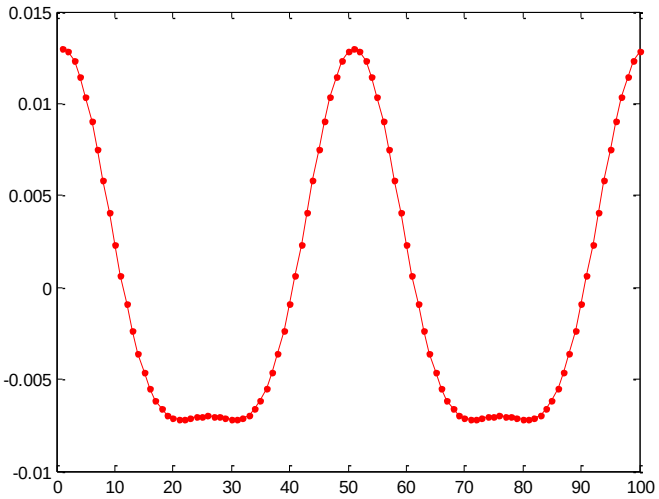
Note: Compute the value of $e^{i\pi}$. The notation 1i is Matlab's code for the famous imaginary number sqrt(-1).

Y = exp(1i*pi)

Y = -1.0000 + 0.0000i

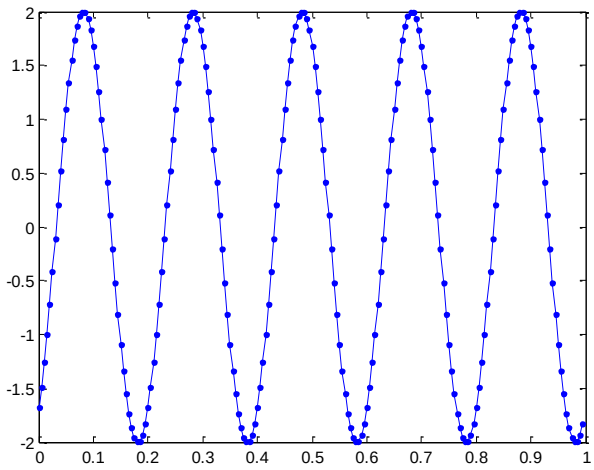
```
clear all
close all
  mag = zeros(1,100);
  mag(3)=1;         % spike in the magnitude at freq=3
  mag(5)=.3;        % spike in the magnitude at freq=5
  ph=zeros(1,100); % zero phase throughout
  y = mag.*exp(1i*ph);  % the complex signal (fft of some real signal)
  figure, plot(abs(y))

  x = ifft(y);          % the inverse fft (x is in general complex too)
  x1 = real(x);  % here we imagine that we can "measure" the real part of x

  z = ifft(mag);        % here we just use the magnitude to do the ifft
  z1 = real(z);

  figure, plot(x1,'.-r')    % plot the real part of the signal
  figure,plot(z1,'.-b')    % plot the ifft of the magnitude
```

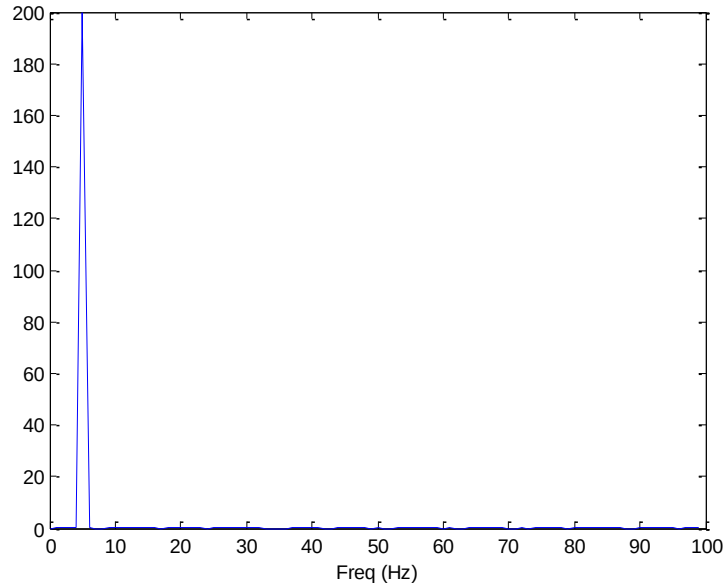

1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50
60
70
80
90
100


0.015
0.01
0.005
0
-0.005
-0.01
0
10
20
30
40
50
60
70
80
90
100

**Inverse FFT**

```
clear all
close all
  mag  = 2;      % magnitude (arbitrary units)
  freq = 5;      % frequency in Hz
  samp = 200;    % sampling rate in Hz

  t = 0:1/samp:1-1/samp;  % time (1s of data)
  N = length(t);   % store the number of time points
  x = mag*cos(2*pi*freq*t + 10);    % the signal equation (+phase)
  figure,  plot(t,x,'.-');
  y  = fft(x);            % do Fast Fourier Transform
  f  = linspace(0,N-1,N);  % vector of frequencies for plotting
  figure,  plot(f(1:N/2),abs(y(1:N/2))); % plotting half of the fft results
  xlabel('Freq (Hz)');           % labelling x-axis
  z  = ifft(y);
  plot(t,z,'r--');
```



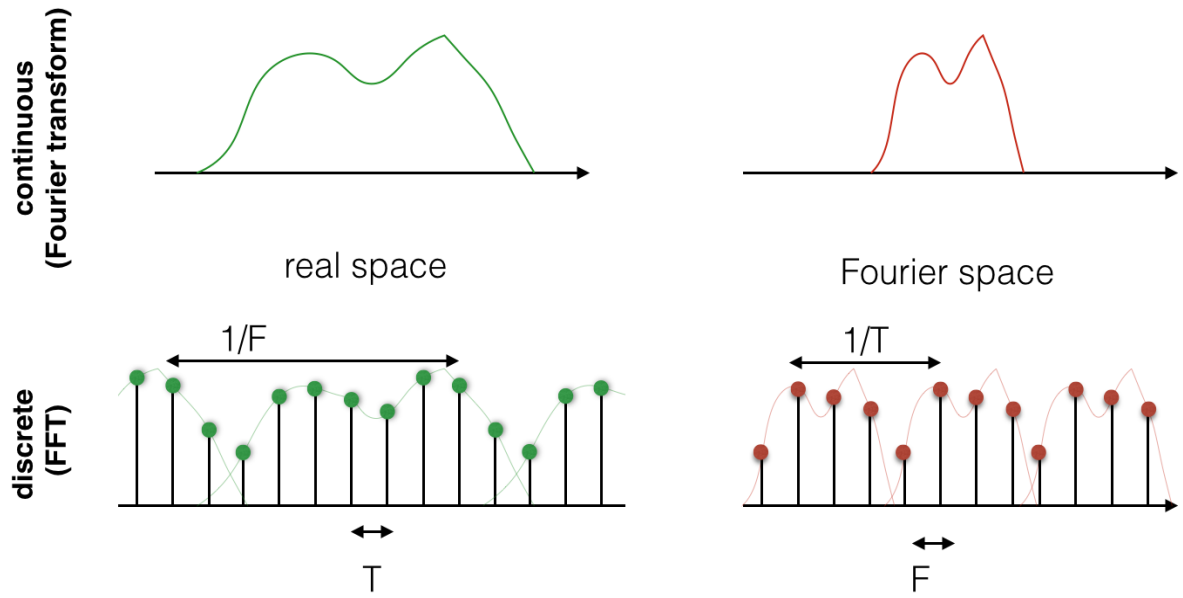
**2D Fourier transform:**

Matlab'daki FFT'nin 2 boyutlu versiyonuna FFT2 denir. Şimdi bir resme 2D Fourier dönüşümünü uygulayacağız.

```
clear all
close all
  im=imread('Apricot.png');
  figure, imshow(im)
  im1=im(:,:,1);
  figure, imshow(im1)

    y=fft2(im1);

    clim=quantile(abs(y(:)),[.01 .99]);
    figure
    imagesc(fftshift(abs(y)),clim);colormap gray
    title('magnitude');

    clim=quantile(angle(y(:)),[.01 .99]);
    figure
    imagesc(fftshift(angle(y)),clim);colormap gray
    title('phase')
```

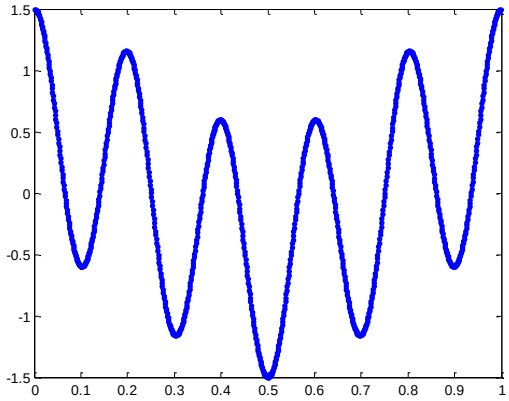
**Sampling and aliasing**

Verilerin ayrık ya da kesiktir. Örnekleme, gerçek uzayda Fourier dönüşümünü periyodik olarak tekrarlamakla aynıdır ve tekrar süresi, örnekleme süresinin tersidir.




İki kosinüs fonksiyonunun süperpozisyonundan oluşan basit bir sinyali ele alalım. Bu sinyali örnekleme hızında (sürekli gibi) simüle edelim.

```
clear all
close all
  freq1 = 5;       % freq in Hz
  freq2 = 1;       % freq in Hz
  samp  = 1000;    % sampling rate in Hz
  t         = 0:1/samp:1;  % time (1s of data)
  t         = t(1:end-1);  % remove last time point
  N         = length(t);   % store the number of time points
  x = cos(2*pi*freq1*t) + .5*cos(2*pi*freq2*t);    % the signal equation
  figure
  plot(t,x,'.-');
```




Bu sinyal matlab tarafından periyodik olduğu varsayılır, bu nedenle sınırlı bir süreye sahip olsa da, FFT tarafından tekrarlanıyormuş gibi ve dolayısıyla gerçek bir kosinüs fonksiyonu gibi muamele görür. Bu sinyalden alt örnekler aldığımızda ne olacağını görelim.

```
clear all
close all
  freq1 = 5;       % freq in Hz
  freq2 = 1;       % freq in Hz
  samp  = 1000;    % sampling rate in Hz
  t        = 0:1/samp:1;  % time (1s of data)
  t        = t(1:end-1);  % remove last time point
  N        = length(t);   % store the number of time points
  x = cos(2*pi*freq1*t) + .5*cos(2*pi*freq2*t);    % the signal equation
  figure
  plot(t,x,'.-');
  F = 50;     % sub-sampling frequency (Hz)
  T = 1/F;    % sub-sampling period (sec)
  n = N*T;    % sub-sampling period (in samples)
  z = x(1:n:end);     % sub-sampled signal
  y = fft(z);         % fft of sub-sampled signal
  subN = length(z);   % length of sub-sampled signal
  f  = linspace(0,subN-1,subN);  % vector of frequencies for plotting
  figure
  plot(f(1:subN/2),abs(y(1:subN/2))) % plot powerspectrum as before
```

Şimdi alt örneklemenin sıklığını değiştirin ve fourier dönüşümüne ne olduğunu görün. 10Hz altına düştüğünüzde, powerspectrum'da zirveyi 5Hz'de kaybettiğinizi görebilmelisiniz. Sinyali fft'den yeniden yapılandırmak ve neredeyse sürekli sürümle karşılaştırmak için ifft işlevini kullanmaya çalışın.  F=10 alın.

**Sıfır dolgu:**

Sıfır dolguyu anlamak için 2D FFT ve Apricot resmini kullanacağız. İlk önce görüntüyü yükleyin ve diğer pikselleri kaldırarak alt örnekleyin:

```
clear all
close all
  im=imread('Apricot.png');
  x=im(1:2:end,1:2:end,1);
  figure
  imshow(x)
  y=fft2(x);
  clim=quantile(abs(y(:)),[.01 .99]);
  figure,  imagesc(fftshift(abs(y)),clim);colormap gray
```

Fourier dönüşümünde tekrarlama süresi örnekleme periyodu ile ters orantılıdır. Sıfır doldurma, sinyali yeniden oluşturmadan önce FFT'nin her iki tarafına sıfırların eklenmesi anlamına gelir. Matlab FFT, sinyalin periyodik olduğunu varsaydığından, sıfırlama sinyali, daha yüksek tekrarlama periyodu ile sinyali tekrarlamak gibidir, bu da gerçek alanda daha yüksek örnekleme oranı anlamına gelir. Bu nedenle sıfır dolgusu, bir görüntüyü frekans alanının bize izin verdiğinden daha yüksek çözünürlükte yeniden yapılandırmamızı sağlar ve enterpolasyon kavramıyla yakından ilişkilidir.

```
y = fftshift(y); % we need this before zero padding
z = ifft2(y,size(im,1),size(im,2)); % zero padded ifft
clim=quantile(abs(z(:)),[.01 .99]);
figure,imshow(abs(z),clim);colormap gray
```

Orijinal görüntüyle karşılaştırıldığında, enterpolasyonun neden olduğu salınımlara dikkat edilmelidir.

**Filtering:**

Burada evrişim ve Fourier dönüşümünü kullanarak basit bir doğrusal filtre uygulayacağız. Gerçek uzayda evrişimin Fourier uzayda çarpma ile aynı olduğunu ve tam tersini hatırlayın. Ancak evrişimin çoğaltılmasından daha zordur, bu nedenle genellikle sinyalin ve filtrenin Fourier dönüşümünü yapmak, sonra iki FFT'yi birlikte çoğaltmak ve sonra ters bir FFT yapmak iyi bir fikirdir. Görüntünün ve filtrenin Fourier dönüşümünü hesaplayalım:

```
siz = [3 3];   % 3x3 box. You can play with changing this
box = ones(siz)/9;
x=im(:,:,1);
y=fft2(x);
f=fft2(box,size(x,1),size(x,2));  % use zeropadding
z=ifft2(y.*f);  % inverse FFT of the product
figure,
clim=quantile(abs(z(:)),[.01 .99]);
imagesc(abs(z),clim);colormap gray
```

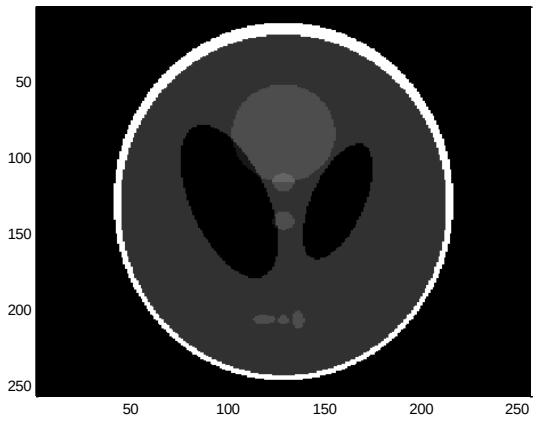
Gauss filtresi ortalama filtreye benzer, ancak kutudaki değerlerin sabit olması yerine, kutunun ortasında ve verilen standart sapmayla ortalanmış bir Gauss fonksiyonunu takip etmeleri dışında. Şimdi başka bir filtre tipi tasarlayalım: bir kenar algılama filtresi. Bu tür en basit filtre, ayrı ayrı terimlerle, bitişik pikseller arasındaki fark olan bir türev yapmaktır.

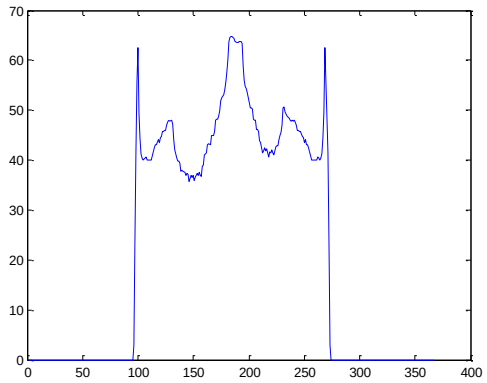
```
 box = [-1 1];    % along x-dimension
box = [-1;1];    % along y-dimension
box = [-1 0 1];  % a slightly better filter along x
box = [-1;0;1];  % a slightly better filter along y
x=im(:,:,1);
y=fft2(x);
box = [-1 0 1]; % try with different ones if you have time
f=fft2(box,size(x,1),size(x,2));  % use zeropadding
z=ifft2(y.*f);  % inverse FFT of the product
figure,
clim=quantile(abs(z(:)),[.01 .99]);
imagesc(abs(z),clim);colormap gray
```

**Image reconstruction - Computed Tomography**

İlk olarak, bir nesneye ihtiyacımız var. İki boyutta kalacağız ve tıbbi görüntülemede en ünlü görüntüyü kullanacağız: Shepp-Logan fantomu. Bu resim o kadar popüler ki Matlab'ın görüntüleme araç kutusuna dahil edildi, bu yüzden onu oluşturalım:

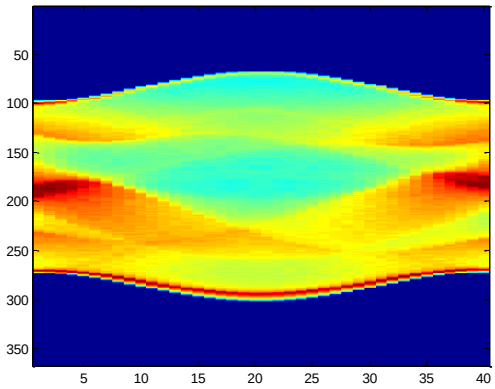
```
clear all
close all
  im=phantom;
  figure, imagesc(im), colormap gray
  R = radon(im,0);
  figure, plot(R);
```



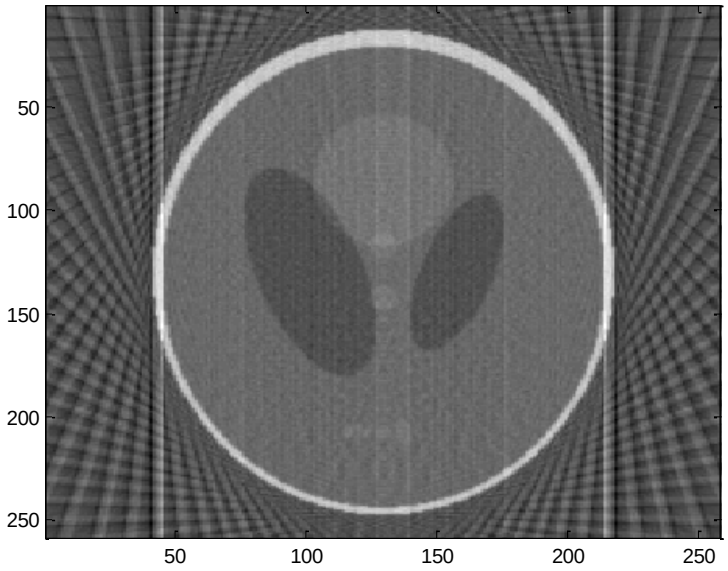

Şimdi daha fazla açı boyunca Radon dönüşümüne bakalım ve sonuçları çizelim:

```
R = radon(im,linspace(0,180,40));    % 40 angles from 0 to 180 degrees
  figure, imagesc(R);
```




Şimdi ters bir Radon dönüşümü hesaplayalım ve nesnenin neye benzediğini görelim:

```
  x = iradon(R,linspace(0,180,40));
  figure,imagesc(x),colormap gray
```




Ters Radon transfromuna geri projeksiyon da denir. Nesneyi yeniden yapılandırmaya çalıştığınızda bu arka izdüşümün geride "izler" bıraktığını açıkça görebilirsiniz. Kaç tanesinin yeterli olduğunu anlamak için farklı projeksiyonlarla yukarıdakileri deneyin. Ayrıca, nesnedeki küçük ayrıntıların düzgün bir şekilde kurtarılması için daha fazla açı gerektirdiğine dikkat edin.

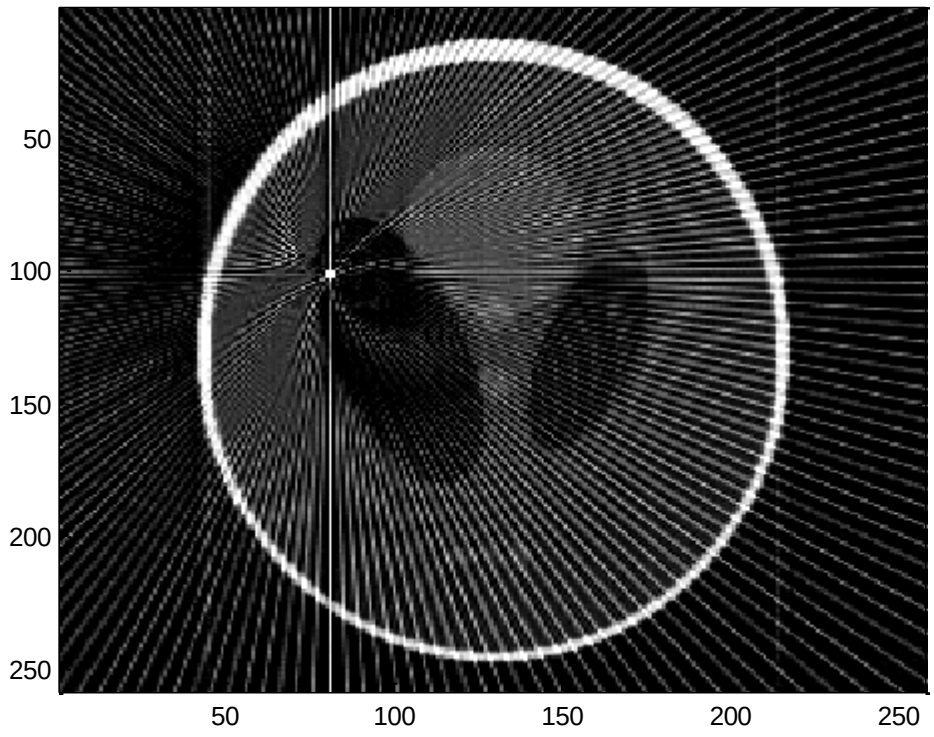
Bilgisayarlı Tomografide ölçüm aslında nesnenin X-ışını huzmesi boyunca tam olarak integrali değil, ışınlar boyunca "emilim" in integralidir. Ama bunu görmezden geleceğiz ve nesnenin

kütlesinin integralini ölçüyormuşuz gibi davranacağız. Sonra bir "metal" arayacağız.Hastanın vücudunun içinde bir implant olduğunda olur. Orijinal fantom görüntüsünde bir ani yükselterek bunu modelleyebiliriz:

Şimdi radon dönüşümünü (ölçüm işlemini simüle etmek için) ve sonra radon dönüşümünü (nesnenin ölçümlerden yeniden yapılandırılması) çalıştırın. Simüle edilmiş metal implantın neden olduğu bazı çarpıcı etkileri görebilmelisiniz.

```
im(100,80)=100;
R = radon(im,linspace(0,180,100));
x = iradon(R,linspace(0,180,100));
figure,imagesc(x,[0 1]);colormap gray
```

Radon dönüşümü, projeksiyon-dilim teoremi aracılığıyla Fourier dönüşümü ile ilgilidir. Bu teorem, bir nesneniz varsa (örneğin 2B'de), nesneyi bir çizgiye yansıtmanın, nesnenin 2B Fourier dönüşümünden bir dilim almakla aynı olduğunu belirtir.




Not: metal olduğu bir önceki resimdeki sdairenin dışındaki saçılmalardan anlaşılmaktadır.

# 6. Laplace Transformation

Laplace dönüşümü diferansiyel denklemlerin çözümünde ortaya çıkmaktadır:

- Bir diferansiyel denklemi cebirsel bir denkleme dönüştürür. Böylece, daha basit cebirsel denklemleri çözerek ve sonra dönüştürmeyi tersine çevirerek, diferansiyel denkleme bir çözüm üretebiliriz.
- Bir rastgele değişkenin moment üretme fonksiyonu, yoğunluğunun Laplace dönüşümü ile ilgilidir (-s ile değiştirerek) ve toplamın yoğunluğunu dikkate alırken yararlı olabilir çünkü bir konvolüsyonun Laplace dönüşümü onların Laplace dönüşümlerinin ürünüdür.

Fourier dönüşümü, sadece sinüs ve kosinüs temel işlevleri kullanılarak sentezlenebilen sinyalleri analiz etmek için yeterlidir. Ancak, sinyalin üstel bileşenleri olduğunu bulduğumuzda, örneğin, zaman içinde üssel olarak değişkenlik gösteren sinüs dalgası için, bu, fourier'in verdiği bize sadece yarım bilgi. Böyle durumlarda kaybolur. Bu tür durumlarda laplace dönüşümü zorunludur.

Laplace, filtre tasarımında ve analog devrelerin analizinde yaygın olarak kullanılır.
Laplace, hangi üssel koşulların kullanılması gerektiğine karar verebilir, böylece sistemimiz birleşir ve aynı zamanda sabit kalır.

Aniden başlayan veya biten herhangi bir şeyi analiz etmek isterseniz, Laplace dönüşümünden büyük ölçüde faydalanırsınız. Bu geçişlere sistem cevabı gibi gerçek uygulamalarda çok karşılaşılır. Aksine, Fourier dönüşümü, bir Laplace dönüşümünün problemi büyük ölçüde basitleştirdiği asimetrik fenomenlerle uğraşırken çok verimsizdir. Bu, Laplace'ın çok önemli bir analitik aracı dönüştürmesini sağlar.

Laplace tansformasyonu doğrusal sistemleri tanımlamak için yaygın olarak kullanılmaktadır. Türevleri içeren dağınık denklemleri cebirsel denklemlere dönüştürebilir. Eğer sistem bir girişe ve bir çıkışa sahipse, çıkışın girişin bir fonksiyonu olarak tanımlandığı durumlarda, onu s etki alanına (laplace kullanarak) dönüştürebilir ve sistemi bir transfer fonksiyonu ile tanımlayabilirsiniz. X (s) 'in girdi olduğunu, Y (s)' nin çıkış olduğunu ve H (s) 'nin transfer fonksiyonu olarak bilinen şey olduğunu varsayalım. Sistem çıkışı Y (s) = H (s) * X (s) olarak tanımlanabilir. Transfer fonksiyonu Y (s) / X (s) ile eşit olacaktır. Bu önemli görünmeyebilir, ancak bir sistemin transfer fonksiyonu bulunduğunda, bize söylenen sistemin çok fazla bir kısmını söyleyebilir. Transfer fonksiyonunun pay ve paydası faktörize edilirse, bir kutup-sıfır tanımı yapılabilir. Sisteminiz yüksek frekansları söndürmek için yapılmış bir elektrik devresi ise, düşük geçiş filtresi, sadece kutup-sıfır çizimi filtrenin belirli frekanslara nasıl tepki verdiğini size söyleyebilir. Bu, laplace dönüşümünün sadece küçük bir uygulamasıdır, ancak

birçok mühendislik dalında kullanışlı olabilen transfer fonksiyonları ile lineer sistemlerin tanımlanmasını sağlar.

- Sistemlerin, sadece bir frekans veya zaman alanı verileri yerine, sinusoidler ve üsteller gibi doğal bileşenler açısından analiz edilmesine yardımcı olur.
- Yoğun aktarım işlevini, kutuplar ve sıfırlar cinsinden tanımlanabildiği ve sezgisel olarak sistemin kararlılığını, aşılmasını veya gürültüsünü tahmin etmede yardımcı olan uygun bir alana (S-etki alanı) çevirir.
- Fourier Dönüşümü, şifreli bir dönüşümü çok daha kolay bir çarpıma dönüştürürken, bir Laplace Dönüşümü, S-etki alanında basit bir polinom cebiri ile yoğun bir diferansiyel denklemin çözülmesine yardımcı olur.
- Yinelemeli filtrelerin kararlılığını analiz etmek için DSP'de temel olan Z-Dönüşümü, Laplace Dönüşümü'nün yakın akrabasıdır.

$$F(s) = \mathcal{L}\{f(t)\} = \int_0^{+\infty} e^{-st} f(t) dt \qquad s > 0$$

$F(s) = \mathrm{L}[f(t)] = \int_0^{\infty} f(t) e^{-st} dt$

as the Laplace transform of $f(t)$, and the inverse Laplace transform is

$f(t) = \mathrm{L}^{-1}[F(s)] = \frac{1}{2\pi i} \int_{a-i\infty}^{a+i\infty} F(s) e^{st} ds$

Theorem: The conditions for the existence of Laplace transform of $f(t)$ are
(1) $f(t)$ is piecewise continuous in every finite interval.
(2) $f(t)$ is an exponential order function.

**Laplace Dönüşümleri**

1) Dinamik ve kontrolde standart gösterimi (kısa gösterim) sağlar.
2) Matematik cebirsel işlemlere dönüştürür.
3) Blok diyagram analizi için avantajlıdır.

**Laplace dönüşümleri proses kontrolünde kullanılabilir:**

1) Diferansiyel denklemlerin çözümü (doğrusal)
2) Doğrusal kontrol sistemlerinin analizi (frekans cevabı)
3) Farklı girdiler için geçici yanıtın tahmini

# Laplace Transform of Basic Functions

| Original function | Transformed function |
|---|---|
| $1$ | $\frac{1}{s}$ |
| $t^n$ | $\frac{n!}{s^{n+1}}$ |
| $t^a$ | $\frac{\Gamma(a+1)}{s^{a+1}}$ |
| $e^{at}$ | $\frac{1}{s-a}$ |

| Original function | Transformed function |
|---|---|
| $\sin at$ | $\frac{a}{s^2+a^2}$ |
| $\cos at$ | $\frac{s}{s^2+a^2}$ |
| $\sinh at$ | $\frac{a}{s^2-a^2}$ |
| $\cosh at$ | $\frac{s}{s^2-a^2}$ |

1. $\mathsf{L}\ [1] = \int_0^\infty e^{-st}dt = -\frac{1}{s}e^{-st}\Big|_0^\infty = \frac{1}{s}$

2. $\mathsf{L}\ [t^a] = \int_0^\infty t^a e^{-st}dt = \int_0^\infty (\frac{u}{s})^a e^{-u}\frac{du}{s} = \frac{1}{s^{a+1}}\int_0^\infty u^a e^{-u}du = \frac{\Gamma(a+1)}{s^{a+1}}$

3. $\mathsf{L}\ [e^{at}] = \int_0^\infty e^{at}e^{-st}dt = \frac{e^{-(s-a)t}}{-(s-a)}\Big|_0^\infty = \frac{1}{s-a}$

4. $\mathsf{L}\ [e^{iat}] = \frac{1}{s-ia} \Rightarrow \mathsf{L}\ [\cos at + i\sin at] = \frac{s}{s^2+a^2} + i\frac{a}{s^2+a^2}$

   $\therefore \mathsf{L}\ [\cos at] = \frac{s}{s^2+a^2}$, and $\mathsf{L}\ [\sin at] = \frac{a}{s^2+a^2}$

5. $\mathsf{L}\ [\sinh at] = \mathsf{L}\ [\frac{e^{at}-e^{-at}}{2}] = \frac{1}{2}(\frac{1}{s-a} - \frac{1}{s+a}) = \frac{a}{s^2-a^2}$

   $\mathsf{L}\ [\cosh at] = \mathsf{L}\ [\frac{e^{at}+e^{-at}}{2}] = \frac{1}{2}(\frac{1}{s-a} + \frac{1}{s+a}) = \frac{s}{s^2-a^2}$

**$\Gamma(x)=\int_0^{\infty} e^{-u}u^{x-1}du$ is called Gamma function.**

The properties of Gamma function

$\Gamma(a+1)=a\Gamma(a)$.

$\Gamma(1)=1$.

$\Gamma(n+1)=n!$, $n$ is a natural number.

$\Gamma(\frac{1}{2})=\sqrt{\pi}$.

$$\Gamma(a+1)=\int_0^{\infty} e^{-u}u^{(a+1)-1}du=\int_0^{\infty} e^{-u}u^{a}du=-(e^{-u}u^{a}\Big|_0^{\infty}-a\int_0^{\infty} e^{-u}u^{a-1}du)=a\int_0^{\infty} e^{-u}u^{a-1}du=a\Gamma(a)$$

$$\Gamma(1)=\int_0^{\infty} e^{-u}u^{1-1}du=\int_0^{\infty} e^{-u}du=-e^{-u}\Big|_0^{\infty}=1$$

When $n$ is a natural number, then

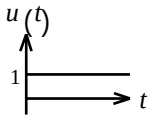
$$\Gamma(n+1)=n\Gamma(n)=n(n-1)\Gamma(n-1)=n(n-1)(n-2)\Gamma(n-2)$$
$$=\cdots=n(n-1)(n-2)\cdots\times 2\times 1\times\Gamma(1)=n!$$

$$\Gamma(\frac{1}{2})=\int_0^{\infty} e^{-u}u^{\frac{1}{2}-1}du=\int_0^{\infty} e^{-u}u^{-\frac{1}{2}}du=2\int_0^{\infty} e^{-u}d\sqrt{u}=2\int_0^{\infty} e^{-x^2}dx=2\sqrt{\int_0^{\infty} e^{-x^2}dx\cdot\int_0^{\infty} e^{-y^2}dy}$$
$$=2\sqrt{\int_0^{\infty}\int_0^{\infty} e^{-(x^2+y^2)}dxdy}=2\sqrt{\int_0^{\frac{\pi}{2}}\int_0^{\infty} e^{-r^2}rdrd\theta}=2\sqrt{\frac{1}{2}\int_0^{\frac{\pi}{2}}(-e^{-r^2})_0^{\infty}d\theta}=2\sqrt{\frac{1}{2}\int_0^{\frac{\pi}{2}}1d\theta}$$
$$=2\sqrt{\frac{1}{2}\cdot\frac{\pi}{2}}=\sqrt{\pi}$$

# Laplace Transform of Special Functions

## 1. Unit step function (Heaviside function)

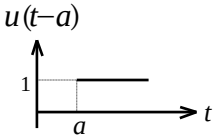
$$u(t)=\begin{cases}1 & t>0\\ 0 & t<0\end{cases}$$




$$\mathcal{L}[u(t)]=\int_0^\infty u(t)e^{-st}dt$$

$$=\int_0^\infty e^{-st}dt=\frac{1}{s}$$

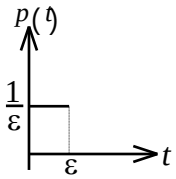
$$\Rightarrow u(t-a)=\begin{cases}1 & t>a\\ 0 & t<a\end{cases}$$




$$\mathcal{L}[u(t-a)]=\frac{e^{-as}}{s}$$

## 2. Unit impulse function (Dirac delta function)

(1) square wave function $p(t)=\begin{cases}\frac{1}{\varepsilon} & 0\le t\le\varepsilon\\ 0 & t>\varepsilon\end{cases}$




$$\mathcal{L}[p(t)]=\int_0^\infty p(t)e^{-st}dt=\int_0^\infty\frac{1}{\varepsilon}[u(t)-u(t-\varepsilon)]e^{-st}dt$$

$$=\frac{1}{\varepsilon}\left(\frac{1}{s}-\frac{e^{-\varepsilon s}}{s}\right)=\frac{1-e^{-\varepsilon s}}{s\varepsilon}$$

(2) unit impulse function $\delta(t)=\lim_{\varepsilon\to0}p(t)$ ( also called singular function, Dirac delta function)

$$\mathcal{L}[\delta(t)]=\mathcal{L}[\lim_{\varepsilon\to0}p(t)]=\lim_{\varepsilon\to0}\{\mathcal{L}[p(t)]\}=\lim_{\varepsilon\to0}\frac{1-e^{-\varepsilon s}}{s\varepsilon}=\lim_{\varepsilon\to0}\frac{se^{-\varepsilon s}}{s}=1$$

Properties:

(i) $\int_0^\infty\delta(t)dt=u(t)$

(ii) $\int_0^\infty g(t)\delta(t-a)dt=g(a)$

(iii) $\int_0^\infty g(t)\delta(t)dt=g(0)$

$\mathcal{L}[\delta(t-a)]=e^{-as}$

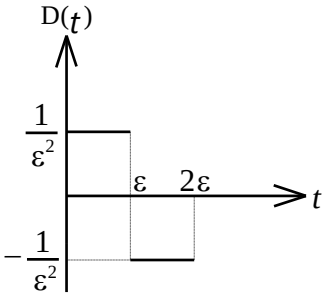
3. Unit double function

$$D(t)=\lim_{\varepsilon\to 0}\frac{1}{\varepsilon^2}[u(t)-2u(t-\varepsilon)+u(t-2\varepsilon)]$$

$$\mathcal{L}[D(t)]=\lim_{\varepsilon\to 0}\frac{1}{\varepsilon^2}\left(\frac{1}{s}-\frac{2e^{-\varepsilon s}}{s}+\frac{e^{-2\varepsilon s}}{s}\right)$$

$$=\lim_{\varepsilon\to 0}\frac{1-2e^{-\varepsilon s}+e^{-2\varepsilon s}}{\varepsilon^2 s}=\lim_{\varepsilon\to 0}\frac{2se^{-\varepsilon s}-2se^{-2\varepsilon s}}{2\varepsilon s}$$

$$=\lim_{\varepsilon\to 0}\frac{-2s^2e^{-\varepsilon s}+4s^2e^{-2\varepsilon s}}{2s}=s$$




## Question:

Let $x(t)=e^{-3t}[H(t)*\delta(t)]$, where $H(t)$ is unit step function, and $\delta(t)$ is Dirac delta function, and $*$ represents convolution, find $(a)\dfrac{d^2x}{dt^2}$ $(b)\mathcal{L}[x(t)]$

Solution : $(a)\ H(t)*\delta(t)=\int_0^t H(\tau)\delta(t-\tau)d\tau=H(t)$

$$\therefore x(t)=e^{-3t}H(t)$$

$$\frac{dx}{dt}=e^{-3t}[-3H(t)+\delta(t)]$$

$$\frac{d^2x}{dt^2}=e^{-3t}\{-3[-3H(t)+\delta(t)]-3\delta(t)+\delta'(t)\}$$

$$=e^{-3t}[9H(t)-6\delta(t)+\delta'(t)]$$

$$(b)\ \mathcal{L}[x(t)]=\mathcal{L}[e^{-3t}H(t)]=\frac{1}{s+3}$$

# Properties of Laplace Transform

| Property | Original Function | Transformed Function |
|---|---|---|
| Linearity | $af(t)+bg(t)$ | $aF(s)+bG(s)$ |
| Shifting | $f(t-a)\,u(t-a)$ | $e^{-as}F(s)$ |
| | $e^{at}f(t)$ | $F(s-a)$ |
| Scaling | $f(at)$ | $\frac{1}{a}F(\frac{s}{a})$ |
| Differentiation | $f^{(n)}(t)$ | $s^nF(s)-s^{n-1}f(0)-s^{n-2}f'(0)-s^{n-3}f''(0)-\ldots-f^{(n-1)}(0)$ |
| | $(-t)^nf(t)$ | $\frac{d^nF(s)}{ds^n}$ |
| Integration | $\int_0^t f(\tau)d\tau$ | $\frac{1}{s}F(s)$ |
| | $\frac{1}{t}f(t)$ | $\int_s^\infty F(s)ds$ |
| Convolution | $\int_0^t f(\tau)g(t-\tau)d\tau$ | $F(s)G(s)$ |
| Periodic Function | $f(t)=f(t+T)$ | $\frac{1}{1-e^{-sT}}\int_0^T f(t)e^{-st}dt$ |
| Initial Value Theorem | $\lim_{t\to0} f(t)=\lim_{s\to\infty} sF(s)$ | |
| | $\lim_{t\to0}\frac{f(t)}{g(t)}=\lim_{s\to\infty}\frac{F(s)}{G(s)}$ | |
| Final Value Theorem | $\lim_{t\to\infty} f(t)=\lim_{s\to0} sF(s)$ | |
| | $\lim_{t\to\infty}\frac{f(t)}{g(t)}=\lim_{s\to0}\frac{F(s)}{G(s)}$ | |

# 7. Tanımlayıcı İstatistik

Tanımlayıcı istatistik, yorumlamak amacı ile bir veri kümesindeki bilgileri özetleme ve düzenleme yöntemlerini ifade eder. Bazı istatistiksel kavramları açıklamak için aşağıdaki tabloyu kullanacağız.

**Characteristics of 10 loan applicants**

| Applicant | Marital Status | Mortgage | Income ($) | Income Rank | Year | Risk |
|---|---|---|---|---|---|---|
| 1 | Single | y | 38,000 | 2 | 2009 | Good |
| 2 | Married | y | 32,000 | 7 | 2010 | Good |
| 3 | Other | n | 25,000 | 9 | 2011 | Good |
| 4 | Other | n | 36,000 | 3 | 2009 | Good |
| 5 | Other | y | 33,000 | 4 | 2010 | Good |
| 6 | Other | n | 24,000 | 10 | 2008 | Bad |
| 7 | Married | y | 25,100 | 8 | 2010 | Good |
| 8 | Married | y | 48,000 | 1 | 2007 | Good |
| 9 | Married | y | 32,100 | 6 | 2009 | Bad |
| 10 | Married | y | 32,200 | 5 | 2010 | Good |

**Öğeler:** Bilgilerin toplandığı varlıklara öğeler denir. Yukarıdaki tabloda, unsurlar 10 başvurandır. Öğelere ayrıca vakalar veya özneler de denir.

**Değişkenler:** Bir elemanın özelliğine değişken denir. Medeni durum, ipotek, gelir, rütbe, yıl ve risk gibi farklı unsurlar için farklı değerler alabilir. Değişkenlere nitelikler de denir. Değişkenler niteliksel veya niceliksel olabilir.

**Niteliksel:** Niteliksel bir değişken, öğelerin bazı özelliklere göre sınıflandırılmasını veya kategorilere ayrılmasını sağlar. Niteliksel değişkenler medeni durum, ipotek, rütbe ve risktir. Nitel değişkenlere kategorik değişkenler de denir.

**Nicel:** Nicel bir değişken sayısal değerler alır ve üzerinde aritmetiğin anlamlı bir şekilde gerçekleştirilmesine izin verir. Nicel değişkenler gelir ve yıldır. Nicel değişkenlere sayısal değişkenler de denir.

**Ayrık Değişken:** Sonlu veya sayılabilir sayıda değer alabilen sayısal bir değişken, her bir değerin her nokta arasında boşluk olacak şekilde ayrı bir nokta olarak grafiklendirilebildiği ayrı bir değişkendir. 'yıl' ayrık bir değişkene örnektir.

**Sürekli Değişken:** Sonsuz sayıda değer alabilen sayısal değişken, olası değerleri sayı doğrusunda bir aralık oluşturan, noktalar arasında boşluk olmayan sürekli bir değişkendir. 'gelir' sürekli değişkene bir örnektir.

**Popülasyon:** Popülasyon, belirli bir problemle ilgili tüm unsurların kümesidir. Parametre, popülasyonun bir özelliğidir.

**Örnek:** Bir örnek, popülasyonun bir alt kümesinden oluşur. Bir örneğin bir özelliğine istatistik denir.

**Rastgele örnek:** Her bir elemanın seçilme şansının eşit olduğu bir örnek aldığımızda.

**Merkez Ölçüleri:** Ortalama, Medyan, Mod, Orta Aralık. Verinin orta kısmının sayı doğrusunda nerede olduğunu belirtir. Bir grup sayıya bakarak ne öğrenebiliriz? Makine Öğreniminde (ve matematikte) bizi ilgilendiren genellikle üç değer vardır:

1) Ortalama - Ortalama değer
2) Medyan - Orta nokta değeri
3) Mod - En yaygın değer

**Ortalama:**
Ortalama, bir veri setinin aritmetik ortalamasıdır. Ortalamayı hesaplamak için değerleri toplayın ve değer sayısına bölün. Örnek ortalaması, bir örneğin aritmetik ortalamasıdır ve $\bar{x}$ ("x-bar") ile gösterilir. Popülasyon ortalaması, bir popülasyonun aritmetik ortalamasıdır ve $\mu$ (“myu”, m için Yunanca harf) ile gösterilir.

Örnek: 13 arabanın hızını kaydetmiş olalılım:
hız = [109, 96, 97, 98, 121, 96, 113, 97, 104, 88, 87, 95, 96]
Ortalama değeri nedir?

Ortalamayı hesaplamak için tüm değerlerin toplamı bulunur ve toplamı değer sayısına bölünür: (109+96+97+98+121+96+113+97+104+88+87+95+96) / 13 =99.77

Örnek: Ortalama hızı bulmak için NumPy mean() yöntemini kullanılır.

```
import numpy
speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]
x = numpy.mean(speed)
print(x)
```

Sonuç: 89.76923076923077

**Median (Orta nokta değeri):**
Medyan, tek sayıda veri değeri olduğunda ve veriler artan düzende sıralandığında ortadaki veri değeridir. Çift sayı varsa, medyan, ortadaki iki veri değerinin ortalamasıdır. Gelir verileri artan düzende sıralandığında, ortadaki iki değer 32.100 ABD Doları ve 32.200 ABD Doları olup, bunun ortalaması medyan gelir olan 32.150 ABD Dolarıdır.

Örnek: 13 arabanın hızını kaydetmiş olalılım:
hız = [109, 96, 97, 98, 121, 96, 113, 97, 104, 88, 87, 95, 96]
medyan değeri nedir?

Medyan değer, tüm değerleri sıraladıktan sonra ortadaki değerdir: 87, 88, 95, 96, 96, 96, 97, 97, 98, 104, 99, 113, 121
Medyanı bulmadan önce sayıların sıralanması önemlidir. NumPy modülünün bunun için bir yöntemi vardır.

Örnek:  Orta değeri bulmak için NumPy median() yöntemini kullanılır.

```
import numpy
speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]
x = numpy.median(speed)
print(x)
```

Ortada iki sayı varsa, bu sayıların toplamını ikiye bölünür.
77, 78, 85, 86, 86, 86, 87, 87, 94, 98, 99, 103
(86 + 87) / 2 = 86.5

Example:
```
import numpy
speed = [99,86,87,88,86,103,87,94,78,77,85,86]
x = numpy.median(speed)
print(x)
```

**Mode (En yaygın değer):**
Mod, en büyük sıklıkta (frekansta) oluşan veri değeridir. Hem nicel hem de kategorik değişkenlerin modları olabilir, ancak yalnızca nicel değişkenlerin ortalamaları veya medyanları olabilir. Her gelir değeri yalnızca bir kez oluşur, bu nedenle mod yoktur. Yıl için mod, 4 frekansla 2010'dur.

Mod değeri dizide en çok görünen değerdir.
99, 86, 87, 88, 111, 86, 103, 87, 94, 78, 77, 85, 86 = 86
SciPy modülünün bunun için bir yöntemi vardır.

Örnek
En çok görünen sayıyı bulmak için SciPy mode() yöntemini kullanın:

```
from scipy import stats
speed = [99,86,87,88,111,86,103,87,94,78,77,85,86]
x = stats.mode(speed)
print(x)
```

Sonuç: ModeResult(mode=array([86]), count=array([3]))

**Mid-range**
The mid-range is the average of the maximum and minimum values in a data set. The mid-range income is:
mid-range(income) = (max(income) + min(income))/2 = (48000 + 24000)/2 = $36000

**Measures of Variability: Range, Variance, Standard Deviation**
*Quantify the amount of variation, spread or dispersion present in the data.*

**Range**
The range of a variable equals the difference between the maximum and minimum values. The range of income is:
range(income) = max (income) − min (income) = 48,000 − 24,000 =$24000

Range only reflects the difference between largest and smallest observation, but it fails to reflect how data is centralized.

**Standart Sapma:**
Population variance is defined as the average of the squared differences from the Mean, denoted as $\sigma^2$ (“sigma-squared”):

$$\sigma^2 = \frac{\sum (x - \mu)^2}{N}$$

Larger Variance means the data are more spread out.
The sample variance $s^2$ is approximately the mean of the squared deviations,
with N replaced by n-1. This difference occurs because the sample mean is used as an approximation of the true population mean.

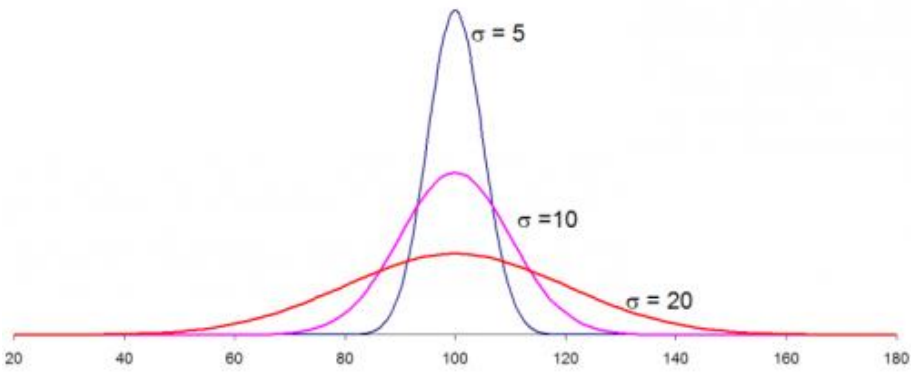
$$s^2 = \frac{\sum (x - \bar{x})^2}{n - 1}$$

Standart sapma, değerlerin aritmetik ortalamaya göre ne kadar yayıldığını açıklayan bir sayıdır.  Düşük bir standart sapma, sayıların çoğunun ortalama (ortalama) değere yakın yayıldığı anlamına gelir.  Yüksek standart sapma, değerlerin daha geniş bir aralığa yayıldığı anlamına gelir.

The *standard deviation* or sd of a bunch of numbers tells you how much the individual numbers tend to differ from the mean.

The sample standard deviation is the square root of the sample variance: sd = √ $s^2$. For example, incomes deviate from their mean by $7201.

The population standard deviation is the square root of the population variance: sd= √ $\sigma^2$.




Three different data distributions with same mean (100) and different standard deviation (5,10,20)

The smaller the standard deviation, narrower the peak, the data points are closer to the mean. The further the data points are from the mean, the greater the standard deviation.

Örnek: Bu sefer 7 arabanın hızını kaydettik:
hız = [86,87,88,86,87,85,86]
Standart sapma:  0.9
Bu, değerlerin çoğunun, 86.4 olan ortalama değerden 0.9 aralığında olduğu anlamına gelir.

Aynı şeyi daha geniş bir aralıkta bir sayı seçimi ile yapalım:
hız = [32,111,138,28,59,77,97]
Standart sapma: 37.85
Yani değerlerin çoğu, ortalama değer olan 77,4'ten 37,85 aralığındadır.
Gördüğünüz gibi, daha yüksek bir standart sapma, değerlerin daha geniş bir aralığa yayıldığını gösterir.

Example
Use the NumPy std() method to find the standard deviation:

```
import numpy
speed = [86,87,88,86,87,85,86]
x = numpy.std(speed)
print(x)
```

**Varyans**

Varyans, değerlerin ne kadar yayılmış olduğunu gösteren başka bir sayıdır. Aslında varyansın karekökünü alırsanız standart sapmayı elde edersiniz! Ya da tam tersi, standart sapmayı kendisiyle çarparsanız varyansı elde edersiniz!

Varyansı hesaplamak için aşağıdaki gibi yapmanız gerekir:

1. Ortalamayı bulunur:

(32+111+138+28+59+77+97) / 7 = 77,4

2. Her değer için: ortalamadan farkı bulunur:

32 - 77,4 = -45,4
111 - 77,4 = 33,6
138 - 77,4 = 60,6
28 - 77,4 = -49,4
59 - 77,4 = -18,4
77 - 77,4 = - 0,4
97 - 77,4 = 19,6

3. Her bir fark için kare değeri bulunur:

(-45.4)2 = 2061.16
(33.6)2 = 1128.96
(60.6)2 = 3672.36
(-49.4)2 = 2440.36
(-18.4)2 = 338.56
(- 0.4)2 = 0.16
(19.6)2 = 384.16

4. Varyans, bu kare farkların ortalamasıdır:

(2061.16+1128.96+3672.36+2440.36+338.56+0.16+384.16) / 7 = 1432,2

Example: Use the NumPy var() method to find the variance.

```
import numpy
speed = [32,111,138,28,59,77,97]
x = numpy.var(speed)
print(x)
```

Standart sapma
Öğrendiğimiz gibi, standart sapmayı bulma formülü varyansın karekökü̈dür: √1432.25 = 37.85

Example: Use the NumPy std() method to find the standard deviation.

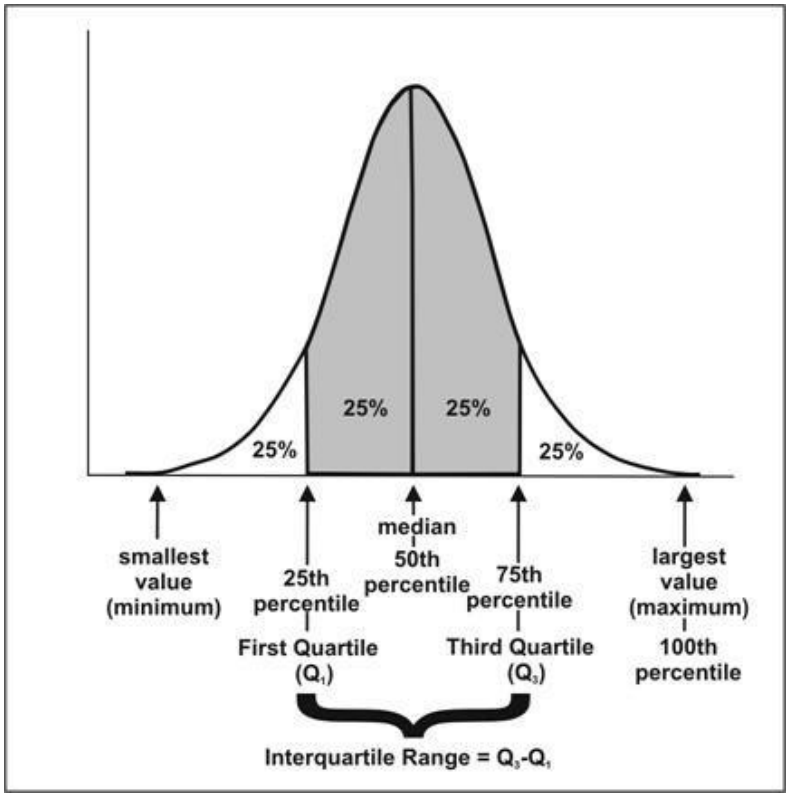
```
import numpy
speed = [32,111,138,28,59,77,97]
x = numpy.std(speed)
print(x)
```

Semboller:
Standart Sapma genellikle Sigma sembolü ile temsil edilir: σ
Varyans genellikle Sigma Karesi sembolü ile temsil edilir: $\sigma^2$

**Measures of Position: Percentile, Z-score, Quartiles**
*Indicate the relative position of a particular data value in the data distribution.*

**Yüzdelikler (Percentile):**
The pth percentile of a data set is the data value such that p percent of the values in the data set are at or below this value. The 50th percentile is the median. For example, the median income is $32,150, and 50% of the data values lie at or below this value.

**Percentile rank**
The percentile rank of a data value equals the percentage of values in the data set that are at or below that value. For example, the percentile rank. of Applicant 1's income of $38,000 is 90%, since that is the percentage of incomes equal to or less than $38,000.




Yüzdelikler, istatistiklerde, değerlerin belirli bir yüzdesinin daha düşük olduğu değeri tanımlayan bir sayı vermek için kullanılır.

Örnek: Diyelim ki bir sokakta yaşayan tüm insanların yaşlarından oluşan bir dizimiz var.
yaş = [5,31,43,48,50,41,7,11,15,39,80,82,32,2,8,6,25,36,27,61,31]

75. yüzdelik nedir?
Dizi küçükten büyüğe sıralanır: 2, 5, 6, 7, 8, 11, 15, 25, 27, 31, 31, 32, 36, 39, 41, 43, 48, 50, 61, 80, 82
Kaçıncı değer olduğu, (75/100)*21=15.75 den n=16 olarak hesaplanır. Cevap 43, yani insanların %75'i 43 veya daha genç.

Kod öreneği:
```
import numpy
ages = [5,31,43,48,50,41,7,11,15,39,80,82,32,2,8,6,25,36,27,61,31]
x = numpy.percentile(ages, 75)
print(x)
```

Example: What is the age that 90% of the people are younger than?

```
import numpy
ages = [5,31,43,48,50,41,7,11,15,39,80,82,32,2,8,6,25,36,27,61,31]
x = numpy.percentile(ages, 90)
print(x)
```

**Interquartile Range (IQR)**

The first quartile (Q1) is the 25th percentile of a data set; the second quartile (Q2) is the 50th percentile (median); and the third quartile (Q3) is the 75th percentile.

The **IQR** measures the difference between 75th and 25th observation using the formula: IQR = Q3 − Q1.

A data value x is an outlier if either x ≤ Q1 − 1.5(IQR), or x ≥ Q3 + 1.5(IQR).

**Z-score**

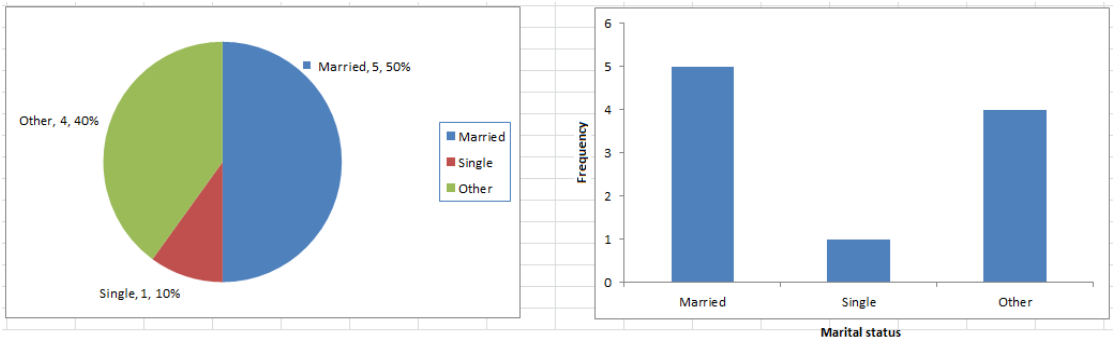
The Z-score for a particular data value represents how many standard deviations the data value lies above or below the mean.

$$\text{Z-score} = \frac{x - \bar{x}}{s}$$

So, If z is positive, it means that the value is above the average. For Applicant 6, the Z-score is (24,000 − 32,540)/ 7201 ≈ −1.2, which means the income of Applicant 6 lies 1.2 standard deviations below the mean.

**Uni-variate Descriptive Statistics**

Different ways you can describe patterns found in uni-variate data include central tendency : mean, mode and median and dispersion: range, variance, maximum, minimum, quartiles , and standard deviation.




Pie chart [left] & Bar chart [right] of Marital status from loan applicants table.

The various plots used to visualize uni-variate data typically are Bar Charts, Histograms, Pie Charts. etc.

**Bi-variate Descriptive Statistics**

Bi-variate analysis involves the analysis of two variables for the purpose of determining the empirical relationship between them. The various plots used to visualize bi-variate data typically are scatter-plot, box-plot.

**Scatter Plots**

The simplest way to visualize the relationship between two quantitative variables , x and y. For two continuous variables, a *scatter-plot* is a common graph. Each (x, y) point is graphed on a Cartesian plane, with the x axis on the horizontal and the y axis on the vertical. Scatter plots are sometimes called correlation plots because they show how two variables are correlated.

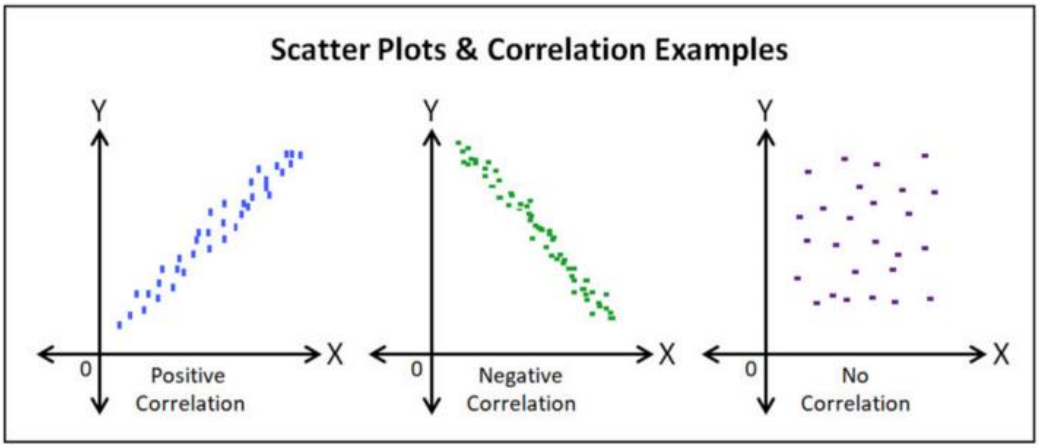
**Correlation**

A correlation is a statistic intended to quantify the strength of the relationship between two variables. The **correlation coefficient** r quantifies the strength and direction of the linear relationship between two quantitative variables. The correlation coefficient is defined as:

$$r = \frac{\sum(x - \bar{x})(y - \bar{y})}{(n-1)\, s_x s_y}$$

where sx and sy represent the standard deviation of the x-variable and the y-variable, respectively. $-1 \le r \le 1$.

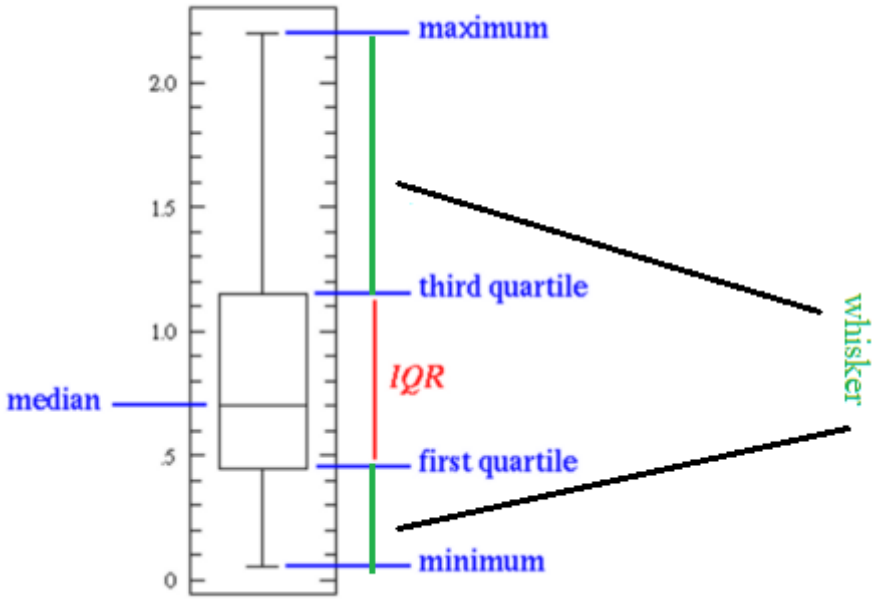
*If r is positive and significant, we say that x and y are **positively correlated**. An increase in x is associated with an increase in y.*
*If r is negative and significant, we say that x and y are **negatively correlated**. An increase in x is associated with a decrease in y.*




Positive correlation (r > 0), Negative correlation (r < 0), No correlation (r = 0)

**Box Plots**

A box plot is also called a box and whisker plot and it's used to picture the distribution of values. When one variable is categorical and the other continuous, a box-plot is commonly used. When you use a box plot you divide the data values into four parts called quartiles. You start by finding the median or middle value. The median splits the data values into halves. Finding the median of each half splits the data values into four parts, the quartiles. Each box on the plot shows the range of values from the median of the lower half of the values at the bottom of the box to the median of the upper half of the values at the top of the box. A line in the middle of the box occurs at the **median** of all the data values.
The **whiskers** then point to the largest and smallest values in the data.




The five-number summary of a data set consists of the minimum, Q1, the median, Q3, and the maximum.

Kutu grafikleri, bir dağılımın çarpık olup olmadığını ve veri setinde olası olağandışı gözlemler (aykırı değerler) olup olmadığını belirtmek için özellikle yararlıdır.

Sol bıyık, aykırı olmayan minimum değere kadar uzanır. Sağ bıyık, aykırı olmayan maksimum değere kadar uzanır. Sol yatay çizgi sağ çizgiden daha uzun olduğunda, dağılım sola çarpıktır ve bunun tersi de geçerlidir. Bıyıkların uzunluğu yaklaşık olarak eşit olduğunda, dağılım simetriktir.

# 8. Olasılık

Makine Öğrenimi, verilerden öğrenmek için istatistik, olasılık, algoritmalar kullanan ve akıllı uygulamalar oluşturmak için kullanılabilecek içgörüler sağlayan disiplinler arası bir alandır. Olasılık ve istatistik, olayların göreceli sıklığını analiz etmekle ilgilenen matematiğin ilgili alanlarıdır. Olasılık gelecekteki olayların olasılığını tahmin etmekle ilgilenirken, istatistikler geçmiş olayların sıklığının analizini içerir.

Çoğu insan olasılık dereceleri hakkında sezgisel bir anlayışa sahiptir, bu nedenle günlük konuşmamızda "muhtemelen" ve "olası değil" gibi kelimeler kullanırız, ancak bu dereceler hakkında nicel iddialarda bulunma hakkında konuşacağız.

Olasılık teorisinde, bir olay, bir olasılığın atandığı bir deneyin sonuçları kümesidir. E bir olayı temsil ediyorsa, P(E), E'nin meydana gelme olasılığını temsil eder. E'nin olabileceği (başarı) veya olmayabileceği (başarısızlık) bir duruma deneme denir. Bu olay yazı tura atmak, zar atmak veya torbadan renkli bir top çekmek gibi herhangi bir şey olabilir. Bu örneklerde olayın sonucu rastgeledir, dolayısıyla bu olayların sonucunu temsil eden değişkene rastgele değişken denir.

Yazı tura atmanın basit bir örneğini ele alalım. Madeni para adilse, tura gelmesi kadar tura gelmesi de olasıdır. Başka bir deyişle, madeni parayı defalarca atacak olsaydık, atışların yaklaşık yarısının yazı ve yarısının yazı olmasını beklerdik. Bu durumda tura gelme olasılığının 1/2 veya 0,5 olduğunu söylüyoruz.

Bir olayın ampirik olasılığı, olayın meydana gelme sayısının gözlemlenen toplam olay sayısına bölünmesiyle verilir. Ön denemeler ve başarıları gözlemlersek, başarı olasılığı s/n'dir. Yukarıdaki örnekte. herhangi bir yazı tura dizisi tam olarak %50'den fazla veya daha az turaya sahip olabilir.

Öte yandan teorik olasılık, belirli bir olayın meydana gelebileceği yol sayısının toplam olası sonuç sayısına bölünmesiyle verilir. Böylece bir kafa bir kez ortaya çıkabilir ve olası sonuçlar ikidir (kafa, kuyruk). Bir kafanın gerçek (teorik) olasılığı 1/2'dir.

Makine öğrenmesi algoritmaları uygulanırken, kullanılacak algoritmanın bulunduğu ortamın deterministik olmadığı, yani aynı girdi için her zaman aynı çıktının garanti edilemediği durumlarla karşılaşılmış olunabilir. Benzer şekilde, gerçek dünyada, girdinin aynı kalmasına rağmen davranışın değişebileceği, bunun gibi senaryolar vardır. Belirsizlik ne olursa olsun vardır. Makine öğrenimi çok büyük miktarda veri, birden çok hiperparametre ve karmaşık bir ortam içerdiğinden, belirsizliklerin olması zorunludur. Eksik değişkenler, eksik modelleme veya verilerin olasılıklı olması şeklinde olabilir.

Olasılığı anlamadan makine öğrenimi algoritmalarına dalmak ilk başta iyi görünebilir, ancak yine de karmaşık algoritmalara derinlemesine dalmak sizi temelleri tekrar gözden geçirmeye zorlayabilir. Bu kavramlar, matematiğin makine öğrenimini nasıl etkilediğini tam olarak anlamada karmaşık görünebilir. Naive Bayes gibi bir algoritmanın nasıl tamamen ona bağlı olduğu da merak edilmelidir!

Kavramları basitleştirmek ve makine öğrenimi dünyasında olasılığın önemini anlaşılmasına yardımcı olmak için sıradan olmayan terimlerle açıklanmalıdır.

Olasılık, kesinlik faktörünü temsil eder. Kesinlik, gerçekleşecek bir olaya atayacağınız orandır. Diyelim ki bir zar atıyorsunuz ve zarlarda 6'nın ortaya çıkma kesinliğinin ⅙ olduğunu söylüyorsunuz. Bu, zarlarda 6 gelme ihtimalinin %16.67 olduğu anlamına gelir. Bu, belirli bir olaya ayırdığınız kesinlik budur. Bu da olasılık olarak bilinir veya tam olarak bizim durumumuzda buna sıklık olasılığı denir.

Sıklık olasılığı, olayın birçok deneme/olay arasında meydana gelme sıklığını belirtir. Zar atmak sık görülür, çünkü ⅙ sonsuz sayıda zar atma denemesinden 6'nın ortaya çıkma olasılığının 1/6 olduğu anlamına gelir.

Tüm senaryolar, önceki varsayımımızdaki gibi sıklıkla ilgili değildir. Yakıt fiyatının enflasyon veya deflasyon olasılığını tahmin ettiğimiz bir makine öğrenimi problemini ele alırsak, bunu frekans olasılık senaryosunda görüldüğü gibi tekrar perspektifinde düşünmüyoruz. Bunun yerine, bu olayın belirli bir olasılık/kesinlikle gerçekleşebileceğini söylüyoruz.

İkinci fenomene Bayes olasılığı denir. Bir olayın tekrarlanma sıklığını düşünmek yerine, inancımızı ölçüyoruz. Şu ifadeyi düşünün - diyabetik bir hastanın kalp yetmezliği geliştirme olasılığı %32. Bu ifade, hastanın semptomlarının sonsuz kopyalarını yarattığımız yerde tekrar etmeye eğilimli değildir. Bunun yerine kalp yetmezliğinin olabileceğini %32 kesinlik ile ölçüyoruz.

Altogether, **probability** measures the extent of certainty pertaining to an uncertain event. As machine learning revolves around probable yet not mandatory situations, probability plays a crucial role in approximating the analysis.

*Formulating an easy and uncertain rule is better in comparison to formulating a complex and certain rule — it's cheaper to generate and analyze. Moreover, a "certain" rule doesn't guarantee generating the right and required output always. An "uncertain" rule, on the other hand, though non-deterministic, helps in reaching a generalized conclusion.*

**Discrete and Continuous Variables**

A discrete variable takes a finite set of values whereas a continuous variable takes an infinite number of values. If you have a dataset with two attributes — age group and profession, age group is continuous and profession is discrete. In probability, we define the probability of discrete variables using **probability mass function (PMF)**.

$$\sum_{x \epsilon X} P(x) = 1$$

Sum of probabilities is 1

PMF assigns probability to every possible variable specific to the data attribute. The probability for all possible variables shouldn't exceed 1, as to how certainty shouldn't exceed 100%. Each probability concerning a variable has to lie between (included) 0 and 1. If we consider the profession attribute, we can define its probability by stating that every profession's probability has to be in between 0 and 1, and all of them have to add up to 1.

$$\int p(x)dx = 1$$

Integral of probabilities is 1

The probability of continuous variables can be defined using **probability density function (PDF)**. As continuous variables are not finite, we use an integral to define PDF. The probability of every possible continuous value has to be greater than or equal to zero but not preferably less than or equal to 1 as a continuous value isn't finite. Although, the integration of all probabilities has to be equal to 1.

**Probability Distribution**

*Probability distribution defines the likelihood of possible values that a random variable can take.* PMF and PDF that have been described earlier for discrete and continuous variables respectively are probability distributions.

If we want to determine the probability distribution on two or more random variables, we use **joint probability distribution**. For a typical data attribute in machine learning, we have multiple possible values. Computing probability of all values falls under joint probability.

If we want to define the probability distribution only on a subset of variables, we use **marginal probability distribution**. This is useful if we want to estimate the probability on only a specific set of input variables (concerning *x* attribute) when given the other input values (concerning *y* attribute).

$$P(x = X) = \sum_{y} P(x = X, y = Y)$$

For Discrete Variable

There are cases where we want to compute the probability of an event when a different event happens. This probability distribution is termed as **conditional probability distribution**.

$$P(x = X|y = Y) = \frac{P(x = X, y = Y)}{P(y = Y)}$$

Probability of x given y

Joint probability distribution can be decomposed into conditional distributions as follows:

$$P(x_1, ..., x_n) = P(x_1) \prod_{i=2}^{n} P(x_i|x_1, ..., x_{i-1})$$

Example: P(z, y, x) = P(x | y, z) * P(y | z) * P(z)

In this case, if the events are disjoint or independent, then the probability can be expressed as the product of all events' probabilities.

$$P(x = X, y = Y) = P(x = X) * P(y = Y)$$

If the events are conditionally independent, the probability is given as follows:

$$P(x = X, y = Y|z = Z) = P(x = X|z = Z) * P(y = Y|z = Z)$$

**Bayes Rule**

Taking a cue from the joint probability distribution, we define Bayes Rule as the *probability of an event computed using the prior knowledge of its related events*.

$$P(y|x) = \frac{P(x|y) * P(y)}{P(x)}$$

Computing the probability of y given x using the prior probability of x given y

Bayes Rule plays a significant role in **Bayesian statistics** where probability is believed to be a *degree of belief* in an event.

**Expected Value**

*The expectation, or expected value is the mean of many repetitions of an event.* Practically, it helps in ascertaining if one has to engage in a given event.

A variable can take several possible values (probabilities) each of which has a likelihood attached. Summing up these details into a single variable gives expectation.

It's given by the following formula:

$$E_{x \sim P}[f(x)] = \sum_{x} P(x) f(x)$$

For Discrete Variable

We also have conditional expected value which defines the expectation of a variable (x) given another variable (y) as follows:

$$E_{x \sim P}[f(x)] = \sum_{x} P(x|y) f(x)$$

**Variance**

Variance defines how the output of an event varies as values (influencing the event) are picked from a probability distribution. It defines how the one value differs from the other values, or in simple terms, the variability of the dataset.

$$Var(f(x)) = E[(f(x) - E[f(x)])^2]$$

**Standard Deviation**

Standard deviation gives the spread of the dataset, as to how far the values are from the mean (expected value). It's given by the square root of variance.

$$Std(f(x)) = \sqrt{Var(f(x))}$$

**Covariance**

Covariance defines the linear relation between two variables. If it's positive, both the variables tend to take higher values and if it's negative, when one variable takes a higher value, the other takes a lower value.

$$Cov(f(x_1), f(x_2)) = E[(f(x_1) - E(f(x_1))(f(x_2) - E(f(x_2)))]$$

**Types of Probability Distributions**

Here are the distributions that we usually come across in machine learning:

**1. Bernoulli Distribution**

Bernoulli distribution is the probability distribution of a random variable — is *1* with a probability of *p* and 0 with a probability of *1-p*. This is typically related to a True/False or a classification scenario.

**2. Binomial Distribution**

Multiple Bernoulli trials constitute a binomial distribution. It's the probability distribution constituting True/False questions in *n* trials.

**3. Multinoulli Distribution**

Multinoulli distribution is the case where a single variable can have multiple outcomes. It's the transition from binary to several categories. When it's a multi-classification problem, this distribution comes into the picture.

**4. Multinomial Distribution**

Multiple Multinoulli trials constitute Multinomial distribution.

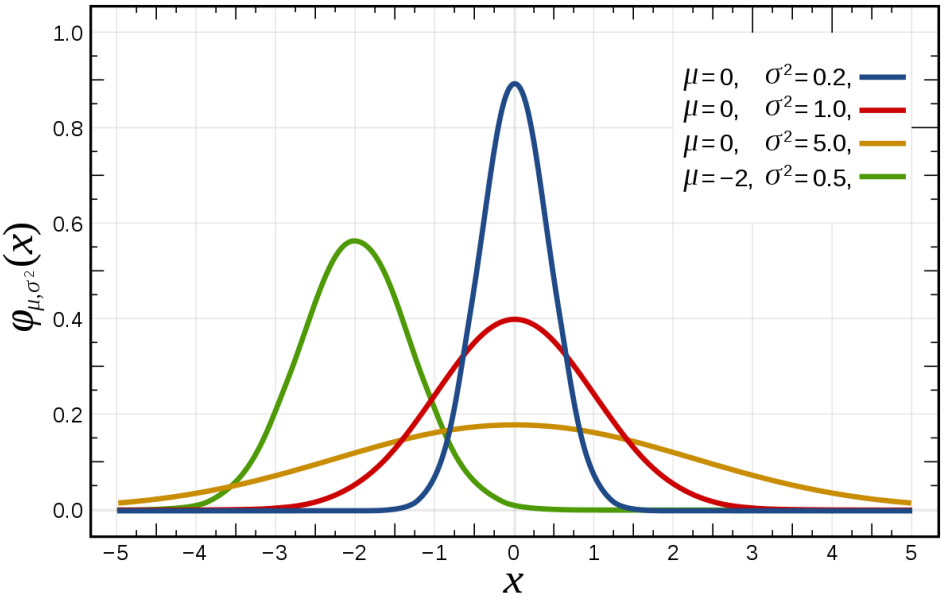
**5. Gaussian Distribution**

Gaussian or *Normal* Distribution is a commonly used distribution in machine learning. Several processes in our nature take the form of Gaussian distribution. In fact, there's a *Central Limit Theorem* which states that the normalized sum of several independent variables is inclined towards Gaussian distribution irrespective of the distribution that each variable takes.

Moreover, this distribution induces maximum uncertainty into the data and requires minimum prior knowledge as it can solely be defined using the mean and variance of data.

$$N(x;\mu,\sigma^2) = \sqrt{\frac{1}{2\pi\sigma^2}}e^{(-\frac{1}{2\sigma^2}(x-\mu)^2)}$$

mu is the mean, and sigma², the variance



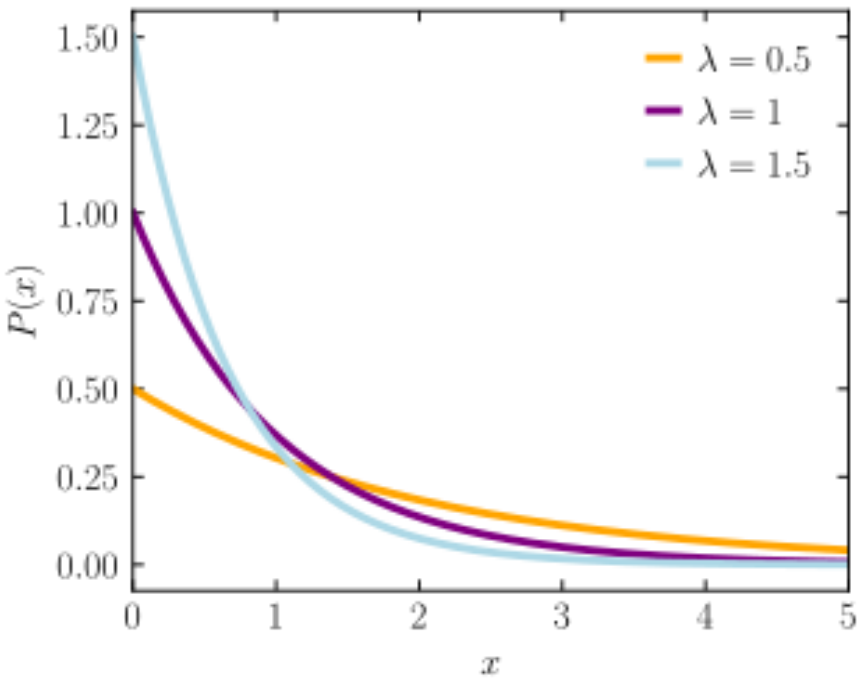

Gaussian Distribution with varying means and variances

**6. Exponential Distribution**

Exponential distribution is concerned about the time until an event occurs. Mathematically, it has a sharp point at *x = 0*.

$$p(x;\lambda) = \lambda 1_{x\geq 0} e^{-\lambda x}$$

Probability is 0 when x < 0




Exponential Distribution at varying lambdas which is the rate parameter

**Bileşik olasılık**

P(A ve B) veya P(A ∩ B) ile gösterilen A ve B olaylarının olasılığı, A ve B olaylarının her ikisinin de meydana gelme olasılığıdır. P(A ∩ B) = P(A). P(B) . Bu, yalnızca A ve B birbirinden bağımsız olduğunda geçerlidir, yani A oluşursa, B olma olasılığını değiştirmez ve bunun tersi de geçerlidir.

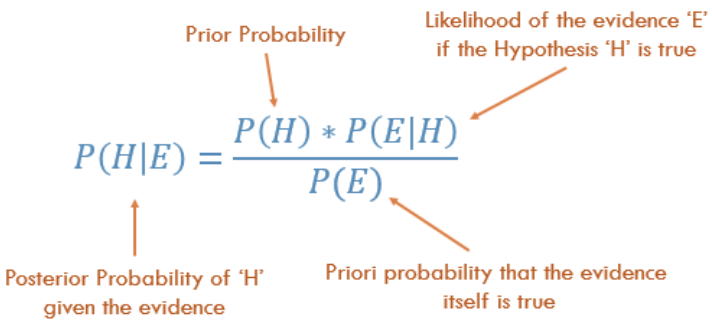
**Şartlı olasılık**

A ve B'nin bağımsız olmadığını düşünelim, çünkü A meydana geldiyse, B'nin olasılığı daha yüksektir. A ve B bağımsız olmadığında, B'nin meydana geldiği verilerde A'nın olma olasılığı olan koşullu olasılığı, P (A|B) hesaplamak genellikle yararlıdır: P(A|B) = P(A ∩ B)/ P(B).

Bir B olayına koşullanmış bir A olayının olasılığı gösterilir ve tanımlanır P(A|B) = P(A∩B)/P(B) Benzer şekilde, P(B|A) = P(A ∩ B)/ P(A) . A ve B'nin ortak olasılığını P(A ∩ B)= P(A).P(B|A) şeklinde yazabiliriz, bu şu anlama gelir: “Her ikisinin de olma olasılığı, birincisinin olma olasılığıdır, ve sonra birincisi verilen ikincisi oldu.”

**Bayes’ Theorem**

Bayes teoremi, iki olayın koşullu olasılıkları arasındaki bir ilişkidir. Örneğin, sıcak ve güneşli bir günde dondurma satma olasılığını bulmak istiyorsak, Bayes teoremi bize başka herhangi bir günde (yağmurlu, rüzgarlı, karlı vb.) dondurma satma olasılığı hakkında ön bilgileri kullanmamız için gerekli araçları verir.




Burada H ve E olaylardır, P(H|E), E olayının zaten gerçekleşmiş olması koşuluyla, H olayının gerçekleşmesinin koşullu olasılığıdır. Denklemdeki olasılık P(H) temelde frekans analizidir; önceki verilerimiz göz önüne alındığında, olayın meydana gelme olasılığı nedir. Denklemdeki P(E|H) olasılık olarak adlandırılır ve frekans analizinden elde edilen bilgiler göz önüne alındığında esasen kanıtın doğru olma olasılığıdır. P(E), gerçek kanıtın doğru olma olasılığıdır.

Dondurma sattığımız olayı H, hava durumu olayı E ile gösterilsin. O halde, hava durumuna göre herhangi bir günde dondurma satma olasılığının ne olduğunu sorabiliriz. Matematiksel olarak bu, denklemin sol tarafına eşdeğer olan P(H=dondurma satışı | E= hava durumu türü)

olarak yazılır. Sağ taraftaki P(H), dondurma satışının marjinal olasılığını zaten bildiğimiz için önsel olarak bilinen ifadedir. Örneğimizde bu, P(H = dondurma satışı), yani dışarıdaki havanın türünden bağımsız olarak dondurma satma olasılığıdır. Örneğin, potansiyel 100 kişiden 30'unun bir yerlerdeki bir dükkandan gerçekten dondurma aldığını söyleyen verilere bakabilirim. Yani benim P(H = dondurma satışı) = 30/100 = 0,3, hava durumu hakkında bir şey bilmeden önce. Bayes Teoremi bu şekilde önceki bilgileri birleştirmemize izin verir.

Bayes teoreminin klasik bir kullanımı klinik testlerin yorumlanmasındadır. Diyelim ki rutin bir tıbbi muayene sırasında doktorunuz size nadir görülen bir hastalık için pozitif test ettiğinizi bildiriyor. Ayrıca bu testlerin sonuçlarında bazı belirsizliklerin olduğunun da farkındasınız. Hastalığı olan hastaların %95'i için bir Duyarlılık (gerçek pozitif oran olarak da adlandırılır) sonucumuz ve sağlıklı hastaların %95'i için bir Özgüllük (gerçek negatif oran olarak da adlandırılır) sonucumuz olduğunu varsayarsak.

"+" ve "−" sırasıyla pozitif ve negatif bir test sonucunu gösterirse, test doğrulukları koşullu olasılıklardır: P(+|hastalık) = 0.95, P(-|sağlıklı) = 0.95,
Bayes terimleriyle, pozitif bir test olan P(hastalık|+) verilen hastalık olasılığını hesaplamak istiyoruz.

*P*(disease|+) = *P(*+|disease)* P*(disease)*/P*(+)

**How to evaluate** *P(+)***, all positive cases ?** We have to consider two possibilities, *P*(+|disease) and *P*(+|healthy). The probability of a false positive, *P*(+|healthy), is the complement of the *P*(-|healthy). Thus *P*(+|healthy) = 0.05.

$$P(\text{disease}|+) = \frac{P(+|\text{disease})P(\text{disease})}{P(+|\text{disease})P(\text{disease}) + P(+|\text{healthy})P(\text{healthy})}$$

Importantly, Bayes' theorem reveals that in order to compute the conditional probability that you have the disease given the test was positive, you need to know the "prior" probability you have the disease*P*(disease), given no information at all. That is, you need to know the overall incidence of the disease in the population to which you belong. Assuming these tests are applied to a population where the actual disease is found to be 0.5%, *P*(disease)= *0.005* which means *P*(healthy) *= 0.995.*
So, *P(disease|+) = 0.95 * 0.005 /(0.95 * 0.005 + 0.05 * 0.995) = 0.088*

In other words, despite the apparent reliability of the test, the probability that you actually have the disease is still less than 9%. Getting a positive result increases the probability you have the disease. But it is incorrect to interpret the 95 % test accuracy as the probability you have the disease.

# 8.1. Hipotez Testi

Makine öğrenimi konusunda uzmanlaşan herkes hipotez testinden bahsedildiğini duyar. Hipotez testi, istatistiksel kararlar vermede kullanılan istatistiksel bir yöntemdir. Hipotez Testi temel olarak popülasyon parametresi hakkında yaptığımız bir varsayımdır.

Makine öğrenimi modelleri, genellikle k-kat çapraz doğrulama kullanılarak hesaplanan ortalama performanslarına göre seçilir.

En iyi ortalama performansa sahip algoritmanın, daha kötü ortalama performansa sahip algoritmalardan daha iyi olması beklenir. Peki ya ortalama performanstaki fark istatistiksel bir tesadüften kaynaklanıyorsa?

Çözüm, herhangi iki algoritma arasındaki ortalama performans farkının gerçek olup olmadığını değerlendirmek için istatistiksel bir hipotez testi kullanmaktır.

Ortalama model performansına göre model seçimi yapmak yanıltıcı olabilir. Bir hipotez testi, hangi ifadenin örnek veriler tarafından en iyi şekilde desteklendiğini belirlemek için bir popülasyon hakkında birbirini dışlayan iki ifadeyi değerlendirir. Bir bulgunun istatistiksel olarak anlamlı olduğunu söylediğimizde, bu bir hipotez testi sayesindedir.

Modele güvenmek ve tahminlerde bulunmak için hipotez testi kullanılır. Modeli eğitmek için örnek veriler kullanılacağı zaman, popülasyon hakkında varsayımlarda bulunulur. Hipotez testi yapılarak, bu varsayımlar istenen bir önem düzeyi için doğrulanır.

Makine öğrenimi modelleri, genellikle k-kat çapraz doğrulama kullanılarak hesaplanan ortalama performanslarına göre seçilir. En iyi ortalama performansa sahip algoritmanın, daha kötü ortalama performansa sahip algoritmalardan daha iyi olması beklenir. Peki ya ortalama performanstaki fark istatistiksel bir tesadüften kaynaklanıyorsa? Çözüm, herhangi iki algoritma arasındaki ortalama performans farkının gerçek olup olmadığını değerlendirmek için istatistiksel bir hipotez testi kullanmaktır.

Model seçimi, bir dizi farklı makine öğrenimi algoritmasını değerlendirmeyi veya işlem hatlarını modellemeyi ve performanslarına göre karşılaştırmayı içerir. Performans metriğine göre en iyi performansı elde eden model veya modelleme hattı, daha sonra yeni veriler üzerinde tahminler yapmaya başlamak için kullanabilecek son model olarak seçilir. Bu, klasik makine öğrenimi algoritmaları ve derin öğrenme ile regresyon ve sınıflandırma tahmine dayalı modelleme görevleri için geçerlidir.

Kusursuz olmasa da, istatistiksel hipotez testi, model seçimi sırasında hem yorumlamaya hem de sonuçların sunumuna olan güveni artırabilir. İstatistiksel hipotez testleri, makine öğrenimi modellerini karşılaştırmaya ve nihai bir model seçmeye yardımcı olabilir. İstatistiksel hipotez testlerinin safça uygulanması yanıltıcı sonuçlara yol açabilir. İstatistiksel testlerin doğru kullanımı zordur ve McNemar testini veya değiştirilmiş eşleştirilmiş Student t testi ile 5×2 çapraz doğrulamayı kullanmak için bazı fikir birliği vardır.

Makine öğrenimi modellerini istatistiksel anlamlılık testleri aracılığıyla karşılaştırmak, kullanılabilecek istatistiksel test türlerini etkileyecek bazı beklentiler getirir; örneğin:
**Beceri Tahmini:** Model becerisinin belirli bir ölçüsü seçilmelidir. Bu, kullanılabilecek testlerin türünü sınırlayacak sınıflandırma doğruluğu (bir orantı) veya ortalama mutlak hata (özet istatistik) olabilir.
**Tekrarlanan Tahminler**: İstatistikleri hesaplamak için bir beceri puanı örneği gereklidir. Belirli bir modelin aynı veya farklı veriler üzerinde tekrarlanan eğitimi ve test edilmesi, kullanılabilecek test türünü etkileyecektir.
**Tahminlerin Dağılımı:** Beceri puanı tahminleri örneğinin dağılımı, belki Gauss ya da olmayabilir. Bu, parametrik veya parametrik olmayan testlerin kullanılıp kullanılamayacağını belirleyecektir.
**Merkezi Eğilim:** Model beceri genellikle, beceri puanlarının dağılımına bağlı olarak ortalama veya medyan gibi bir özet istatistik kullanılarak açıklanacak ve karşılaştırılacaktır. Test bunu doğrudan hesaba katabilir veya almayabilir.

İstatistiksel bir testin sonuçları genellikle bir test istatistiği ve bir p değeridir ve her ikisi de modeller arasındaki farkın güven veya anlamlılık düzeyini ölçmek için sonuçların sunumunda yorumlanabilir ve kullanılabilir. Bu, istatistiksel hipotez testleri kullanmamaktan ziyade model seçiminin bir parçası olarak daha güçlü iddialarda bulunulmasına izin verir. Model seçiminin bir parçası olarak istatistiksel hipotez testlerinin kullanılmasının istendiği göz önüne alındığında, özel kullanım durumunuza uygun bir testi seçilmelidir.

Regresyon modellerini ele alalım: Doğrusal bir regresyon modeli aracılığıyla düz bir çizgi uydurulduğunda, doğrunun eğimi ve kesişimi elde edilir. Bir lineer regresyon modelinde beta katsayılarının anlamlı olup olmadığını doğrulamak için hipotez testi kullanılır. Doğrusal regresyon modeli her çalıştırıldığında, katsayının anlamlı olup olmadığı kontrol edilerek doğrunun anlamlı olup olmadığı test edilir.

Hipotez testi gerçekleştirmek için temel adımlar aşağıdaki gibidir:

- Bir hipotez formüle edilir.
- Önem düzeyi belirlenir.
- Testin türünü belirlenir.
- Teste göre istatistik değerleri ve p değerleri hesaplanır.
- Karar verilir.

**Hipotez testinin türünü seçimi:**
Tahmin değişkenine göre test istatistiği türü seçilir: nicel veya kategorik. Aşağıda nicel veriler için yaygın olarak kullanılan test istatistiklerinden birkaçı verilmiştir.

| Tahmin değişkeni tipi | Dağılım tipi | İstenen Test | Nitelikler |
|---|---|---|---|
| Nicel | Normal dağılım | Z – Test | • Büyük numune boyutu<br>• Bilinen popülasyon standart sapması |
| Nicel | T Dağılımı | T-Test | • Örnek boyutu 30'dan az<br>• Popülasyon standart sapması bilinmiyor |
| Nicel | Pozitif çarpık dağılım | F – Test | • 3 veya daha fazla değişkeni karşılaştırmak istediğinizde |
| Nicel | Negatif çarpık dağılım | NA | • Bir hipotez testi gerçekleştirmek için özellik dönüşümü gerektirir |
| Kategorik | NA | Chi-Square test | • Bağımsızlık testi<br>• Formda olmanın güzelliği |

**Z-istatistiği – Z Testi:**
Örnek normal bir dağılım izlediğinde Z-istatistiği kullanılır. Ortalama ve standart sapma gibi popülasyon parametrelerine göre hesaplanır. Bir örneklem ortalamasını bir popülasyon ortalaması ile karşılaştırmak istediğimizde bir örnek Z testi kullanılır. İki örneğin ortalamasını karşılaştırmak istediğimizde iki örnek Z testi kullanılır.

**T-istatistiği – T-Testi:**
Örnek bir T dağılımını takip ettiğinde ve popülasyon parametreleri bilinmediğinde T istatistiği kullanılır. T dağılımı normal dağılıma benzer, normal dağılımdan daha kısadır ve daha düz bir kuyruğa sahiptir. Örneklem büyüklüğü 30'dan küçükse ve popülasyon parametreleri bilinmiyorsa T dağılımını kullanırız.

**F-istatistiği – F testi:**
Üç veya daha fazla grup içeren numuneler için F Testini tercih ederiz. Birden fazla grup üzerinde t testi yapmak, Tip-1 hata olasılığını artırır. Bu gibi durumlarda ANOVA kullanılır.

Varyans analizi (ANOVA), üç veya daha fazla grubun ortalamalarının farklı olup olmadığını belirleyebilir. ANOVA, araçların eşitliğini istatistiksel olarak test etmek için F-testlerini kullanır.

F-istatistiği, veriler pozitif olarak çarpık olduğunda ve bir F dağılımını takip ettiğinde kullanılır. F dağılımları her zaman pozitiftir ve sağa çarpıktır.

F = Numune ortalamaları arasındaki varyasyon/numuneler içindeki varyasyon
Negatif çarpık veriler için özellik dönüşümü yapmamız gerekir

**Ki-kare testi:**
Kategorik değişkenler için ki-kare testi yapıyor olacağız.

İki tür ki-kare testi şunlardır:
Ki-kare bağımsızlık testi – İki kategorik değişken arasında anlamlı bir ilişki olup olmadığını belirlemek için Ki-Kare testini kullanırız.
Ki-kare Uyum iyiliği, örnek verilerin popülasyonu doğru bir şekilde temsil edip etmediğini belirlememize yardımcı olur.

Hipotezin temeli normalleştirme ve standart normalleştirmedir, tüm hipotez bu 2 terimin temeli etrafında dönüyor.

**Normal dağılım:**
Bir değişkenin dağılımı normal bir eğri - özel bir çan şeklindeki eğri - şeklindeyse, bir değişkenin normal olarak dağıldığı veya normal bir dağılıma sahip olduğu söylenir. Normal dağılımın grafiği, aşağıdaki özelliklerin tümüne sahip olan normal eğri olarak adlandırılır: Ortalama, medyan ve mod.

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

**Standartlaştırılmış Normal Dağılım:**
Standart normal dağılım, ortalaması 0 ve standart sapması olan normal bir dağılımdır.

$$x_{new} = \frac{x - \mu}{\sigma}$$

## 8.1.1. T-Testi

T- testi, gruplar arasındaki farkların ne kadar önemli olduğunu size söyler; Başka bir deyişle, bu farklılıkların (araçlarla ölçülür) tesadüfen meydana gelip gelemeyeceğini size bildirir.
Çok basit bir örnek: Diyelim ki üşüttünüz ve natüropatik bir ilaç denediniz. Soğuk algınlığınız birkaç gün sürer. Bir daha nezle olduğunuzda, reçetesiz satılan bir ilaç alırsınız ve soğuk algınlığı bir hafta sürer. Arkadaşlarınızı araştırıyorsunuz ve hepsi size homeopatik ilacı aldıklarında soğuk algınlığının daha kısa sürdüğünü (ortalama 3 gün) söylüyorlar. Gerçekten bilmek istediğiniz şey, bu sonuçlar tekrarlanabilir mi? Bir t testi, iki grubun ortalamalarını karşılaştırarak ve bu sonuçların tesadüfen olma olasılığını size bildirerek size söyleyebilir.

Başka bir örnek: Student T testleri, ortalamaları karşılaştırmak için gerçek hayatta kullanılabilir. Örneğin, bir ilaç şirketi, yaşam beklentisini iyileştirip iyileştirmediğini öğrenmek için yeni bir kanser ilacını test etmek isteyebilir. Bir deneyde her zaman bir kontrol grubu (plasebo veya "şeker hapı" verilen bir grup) vardır. Kontrol grubu ortalama +5 yıllık bir yaşam beklentisi gösterebilirken, yeni ilacı alan grup +6 yıllık bir yaşam beklentisine sahip olabilir. Görünüşe göre ilaç işe yarayabilir. Ama bir uğultudan kaynaklanıyor olabilir. Bunu test etmek için araştırmacılar, sonuçların tüm popülasyon için tekrarlanabilir olup olmadığını öğrenmek için bir Student t-testi kullanacaklardı.

**T skoru**, iki grup arasındaki fark ile gruplar arasındaki fark arasındaki orandır. t puanı ne kadar büyükse, gruplar arasındaki fark o kadar fazladır. t puanı ne kadar küçükse, gruplar arasındaki benzerlik o kadar fazladır. 3 t puanı, grupların kendi içlerindekinden üç kat daha farklı olduğu anlamına gelir. Bir t testi yaptığınızda, t değeri ne kadar büyük olursa, sonuçların tekrarlanabilir olma olasılığı o kadar artar.

Büyük bir t-skoru size grupların farklı olduğunu söyler.
Küçük bir t-skoru size grupların benzer olduğunu söyler.

**T-Değerleri ve P-değerleri**

"Yeterince büyük" ne kadar büyük? Her t-değerinin onunla birlikte gidecek bir p-değeri vardır. Bir p değeri, örnek verilerinizden elde edilen sonuçların tesadüfen meydana gelme olasılığıdır. P değerleri %0 ila %100 arasındadır. Genellikle ondalık olarak yazılırlar. Örneğin, %5'lik bir p değeri 0,05'tir. Düşük p değerleri iyidir; Verilerinizin tesadüfen oluşmadığını belirtirler. Örneğin, .01'lik bir p değeri, bir deneyden elde edilen sonuçların tesadüfen meydana gelme olasılığının yalnızca %1 olduğu anlamına gelir. Çoğu durumda, verilerin geçerli olduğu anlamına gelen 0,05 (%5) p değeri kabul edilir.

Add up all of the squared differences

| Subject # | Score 1 | Score 2 | X-Y | (X-Y)^2 |
|---|---|---|---|---|
| 1 | 3 | 20 | -17 | 289 |
| 2 | 3 | 13 | -10 | 100 |
| 3 | 3 | 13 | -10 | 100 |
| 4 | 12 | 20 | -8 | 64 |
| 5 | 15 | 29 | -14 | 196 |
| 6 | 16 | 32 | -16 | 256 |
| 7 | 17 | 23 | -6 | 36 |
| 8 | 19 | 20 | -1 | 1 |
| 9 | 23 | 25 | -2 | 4 |
| 10 | 24 | 15 | 9 | 81 |
| 11 | 32 | 30 | 2 | 4 |
| | | **SUM:** | **-73** | **1131** |

$$t = \frac{(\sum D)/N}{\sqrt{\frac{\sum D^2 - \left(\frac{(\sum D)^2}{N}\right)}{(N-1)(N)}}}$$

$$t = \frac{-73/11}{\sqrt{\frac{1131 - \left(\frac{(-73)^2}{11}\right)}{(11-1)(11)}}}$$

$$t = \frac{-73/11}{\sqrt{\frac{1131 - \left(\frac{5329}{11}\right)}{110}}}$$

$$t = -2.74$$

Serbestlik derecelerini elde etmek için örneklem boyutundan 1 çıkarın. 11 öğemiz var, yani 11-1 = 10.

Serbestlik derecelerini kullanarak t-tablosunda p değerini bulun. Belirtilen bir alfa seviyeniz yoksa, 0,05 (%5) kullanın. Bu örnek problem için, df = 10 ile t değeri 2.228'dir.

(2.228) t-tablo değerinizi hesapladığınız t-değeriniz (-2.74) ile karşılaştırın. Hesaplanan t değeri, .05'lik bir alfa düzeyinde tablo değerinden daha büyüktür. p değeri, alfa seviyesinden küçüktür: p <.05. Ortalamalar arasında bir fark olmadığı sıfır hipotezini reddedebiliriz.

## 8.1.2. Chi-Square Bağımsızlık Testi

İki kategorik değişkenin bağımsızlığı testi Chi-kare bağımsızlık testi kullanılarak gerçekleştirilir.

r = satır sayısı, c = sütun sayısı olmak üzere r × c beklenmedik durum tablosu olarak da adlandırılan iki yönlü bir tabloda iki kategorik değişken özetleyebilir. İlgilenilen soru "İki değişken bağımsız mı?"

Sıfır hipotezi: İki kategorik değişken bağımsızdır
Alternatif hipotez: İki kategorik değişken bağımlıdır

**Chi-Kare Testi İstatistik**

$$X^2 = \sum_{i=1}^{r}\sum_{j=1}^{c}\frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

Burada O, gözlemlenen frekansı temsil eder. E, sıfır hipotezi altında beklenen frekanstır ve şu şekilde hesaplanır:

$$E = \frac{(Satır\ Toplamı) * (Sütun\ Toplamı)}{Örnek\ Boyutu}$$

Serbestlik derecesi = (r - 1) (c - 1) olan $X_\alpha^2$ kritik değeriyle test istatistiğinin değeri, karşılaştırılır ve eğer $X^2 > X_\alpha^2$ varsa boş hipotez reddedilir.

**Örnek:**
Cinsiyet eğitim seviyesinden bağımsız mıdır? Rastgele 395 kişiden oluşan bir örneklemle anket yapıldı ve her bir kişiden aldıkları en yüksek eğitim düzeyini bildirmeleri istendi. Anket sonucunda elde edilen veriler aşağıdaki tabloda özetlenmiştir:

| | High School | Bachelors | Masters | Ph.d. | Total |
|---|---|---|---|---|---|
| Female | 60 | 54 | 46 | 41 | 201 |
| Male | 40 | 44 | 53 | 57 | 194 |
| Total | 100 | 98 | 99 | 98 | 395 |

Cinsiyet ve eğitim düzeyi %5 önem düzeyinde bağımlı mı? Başka bir deyişle, yukarıda toplanan verilere göre, bireyin cinsiyeti ile almış olduğu eğitim düzeyi arasında bir ilişki var mıdır?

İşte beklenen sayıların tablosu:
60 için beklenen değer için E= (201*100)/395=50.886
54 için beklenen değer için E= (201*98)/395=49.868
46 için beklenen değer için E= (201*99)/395=50.377
41 için beklenen değer için E= (201*98)/395=49.868

40 için beklenen değer için E= (194*100)/395=49.114
44 için beklenen değer için E= (194*98)/395=48.132
53 için beklenen değer için E= (194*99)/395=48.623
57 için beklenen değer için E= (194*98)/395=48.132

| | High School | Bachelors | Masters | Ph.d. | Total |
|---|---|---|---|---|---|
| Female | 50.886 | 49.868 | 50.377 | 49.868 | 201 |
| Male | 49.114 | 48.132 | 48.623 | 48.132 | 194 |
| Total | 100 | 98 | 99 | 98 | 395 |

$$X^2 = \frac{(60 - 50.886)^2}{50.886} + \cdots + \frac{(57 - 48.132)^2}{48.132} = 8.006$$

Serbestlik derecesi = (r - 1) (c - 1) olan $X_\alpha^2$ kritik değeriyle test istatistiğinin değeri, $X^2$ karşılaştırılır ve eğer $X^2 > X_\alpha^2$ varsa boş hipotez reddedilir.

Serbestlik derecesi = (r - 1) (c - 1) = (4-1)(2-1)=3

Talodan 3 serbestlik dereceli kritik değeri 7.815'tir. 8.006 > 7.815 olduğundan, sıfır hipotezini reddedilir ve eğitim düzeyinin %5 anlamlılık düzeyinde cinsiyete bağlı olduğu sonucuna varılır.

**Percentage Points of the Chi-Square Distribution**

| Degrees of Freedom | Probability of a larger value of $x^2$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 0.99 | 0.95 | 0.90 | 0.75 | 0.50 | 0.25 | 0.10 | 0.05 | 0.01 |
| 1 | 0.000 | 0.004 | 0.016 | 0.102 | 0.455 | 1.32 | 2.71 | 3.84 | 6.63 |
| 2 | 0.020 | 0.103 | 0.211 | 0.575 | 1.386 | 2.77 | 4.61 | 5.99 | 9.21 |
| 3 | 0.115 | 0.352 | 0.584 | 1.212 | 2.366 | 4.11 | 6.25 | 7.81 | 11.34 |
| 4 | 0.297 | 0.711 | 1.064 | 1.923 | 3.357 | 5.39 | 7.78 | 9.49 | 13.28 |
| 5 | 0.554 | 1.145 | 1.610 | 2.675 | 4.351 | 6.63 | 9.24 | 11.07 | 15.09 |
| 6 | 0.872 | 1.635 | 2.204 | 3.455 | 5.348 | 7.84 | 10.64 | 12.59 | 16.81 |
| 7 | 1.239 | 2.167 | 2.833 | 4.255 | 6.346 | 9.04 | 12.02 | 14.07 | 18.48 |
| 8 | 1.647 | 2.733 | 3.490 | 5.071 | 7.344 | 10.22 | 13.36 | 15.51 | 20.09 |
| 9 | 2.088 | 3.325 | 4.168 | 5.899 | 8.343 | 11.39 | 14.68 | 16.92 | 21.67 |
| 10 | 2.558 | 3.940 | 4.865 | 6.737 | 9.342 | 12.55 | 15.99 | 18.31 | 23.21 |
| 11 | 3.053 | 4.575 | 5.578 | 7.584 | 10.341 | 13.70 | 17.28 | 19.68 | 24.72 |
| 12 | 3.571 | 5.226 | 6.304 | 8.438 | 11.340 | 14.85 | 18.55 | 21.03 | 26.22 |
| 13 | 4.107 | 5.892 | 7.042 | 9.299 | 12.340 | 15.98 | 19.81 | 22.36 | 27.69 |
| 14 | 4.660 | 6.571 | 7.790 | 10.165 | 13.339 | 17.12 | 21.06 | 23.68 | 29.14 |
| 15 | 5.229 | 7.261 | 8.547 | 11.037 | 14.339 | 18.25 | 22.31 | 25.00 | 30.58 |
| 16 | 5.812 | 7.962 | 9.312 | 11.912 | 15.338 | 19.37 | 23.54 | 26.30 | 32.00 |
| 17 | 6.408 | 8.672 | 10.085 | 12.792 | 16.338 | 20.49 | 24.77 | 27.59 | 33.41 |
| 18 | 7.015 | 9.390 | 10.865 | 13.675 | 17.338 | 21.60 | 25.99 | 28.87 | 34.80 |
| 19 | 7.633 | 10.117 | 11.651 | 14.562 | 18.338 | 22.72 | 27.20 | 30.14 | 36.19 |
| 20 | 8.260 | 10.851 | 12.443 | 15.452 | 19.337 | 23.83 | 28.41 | 31.41 | 37.57 |
| 22 | 9.542 | 12.338 | 14.041 | 17.240 | 21.337 | 26.04 | 30.81 | 33.92 | 40.29 |
| 24 | 10.856 | 13.848 | 15.659 | 19.037 | 23.337 | 28.24 | 33.20 | 36.42 | 42.98 |
| 26 | 12.198 | 15.379 | 17.292 | 20.843 | 25.336 | 30.43 | 35.56 | 38.89 | 45.64 |
| 28 | 13.565 | 16.928 | 18.939 | 22.657 | 27.336 | 32.62 | 37.92 | 41.34 | 48.28 |
| 30 | 14.953 | 18.493 | 20.599 | 24.478 | 29.336 | 34.80 | 40.26 | 43.77 | 50.89 |
| 40 | 22.164 | 26.509 | 29.051 | 33.660 | 39.335 | 45.62 | 51.80 | 55.76 | 63.69 |
| 50 | 27.707 | 34.764 | 37.689 | 42.942 | 49.335 | 56.33 | 63.17 | 67.50 | 76.15 |
| 60 | 37.485 | 43.188 | 46.459 | 52.294 | 59.335 | 66.98 | 74.40 | 79.08 | 88.38 |

Örnek:

Bir üniversitenin bilgisayar mühendisliğinde okuyan öğrencilerin cinsiyetleri ders seçiminden bağımsız mıdır?

| | Bilgisayar Organizasyonu | Yapay Zeka | Sinyaller ve Sistemler | Gömülü Sistemler | Quantum Hesaplama | Toplam |
|---|---|---|---|---|---|---|
| Erkek Öğrenci | 25 | 10 | 20 | 15 | 10 | **80** |
| Kız Öğrenci | 10 | 5 | 10 | 5 | 5 | **35** |
| Toplam | **35** | **15** | **30** | **20** | **15** | **115** |

Beklenen sayıların tablosu:

| | Bilgisayar Organizasyonu | Yapay Zeka | Sinyaller ve Sistemler | Gömülü Sistemler | Quantum Hesaplama | Toplam |
|---|---|---|---|---|---|---|
| Erkek Öğrenci | 24.35 | 10.43 | 20.87 | 13.91 | 10.43 | **80.00** |
| Kız Öğrenci | 10.65 | 4.57 | 9.13 | 6.09 | 4.57 | **35.00** |
| Toplam | **35.00** | **15.00** | **30.00** | **20.00** | **15.00** | **115.00** |

$X^2 = 0.57$

Serbestlik derecesi = (r - 1) (c - 1) = (2-1)(5-1)=4
%5 anlamlılık düzeyinde 4 serbestlik dereceli kritik değeri, tablodan, $X_a^2$ = 9.49'ur.

$X^2$ karşılaştırılır ve eğer $X^2 < X_\alpha^2$ varsa boş hipotez ret edilmez.

## 8.1.3. Güç Analizi

Verilerden hesaplanan p-değerinin 0.12 olduğu bir araştırma deneyi düşünün. Sonuç olarak, bu p değeri α= 0,05'ten büyük olduğu için boş hipotez reddedilemez. Bununla birlikte, sıfır hipotezini reddedemediğimiz iki olası durum vardır:

1) sıfır hipotezi makul bir sonuçtur,
2) örneklem boyutu, sıfır hipotezini kabul edecek veya reddedecek kadar büyük değil, yani ek örnekler ek kanıt sağlayabilir.

Güç analizi, araştırmacıların, testin makul bir sonuca varmak için yeterli gücü içerip içermediğini belirlemek için kullanabilecekleri prosedürdür. Başka bir perspektiften, belirli bir güç seviyesine ulaşmak için gereken numune sayısını hesaplamak için güç analizi de kullanılabilir.

Örnek:
X, rastgele bir üniversite öğrencisinin boyunu göstersin. X'in bilinmeyen ortalama değeri μ ve standart sapması 9 ile normal olarak dağıldığını varsayın. n = 25 öğrenciden rastgele bir örnek alın, böylece α=0.05 durumunda I. Tip hata yapma olasılığını ayarladıktan sonra sıfır hipotezini Ho, μ=170 değerini alternatif hipoteze Ha, μ>175 karşı test edebiliriz.

Gerçek popülasyon ortalaması μ=175 ise, hipotez testinin gücü nedir?

$$z = \frac{\bar{x} - \mu}{\sigma/\sqrt{n}}$$

$$\bar{x} = \mu + z\left(\frac{\sigma}{\sqrt{n}}\right)$$

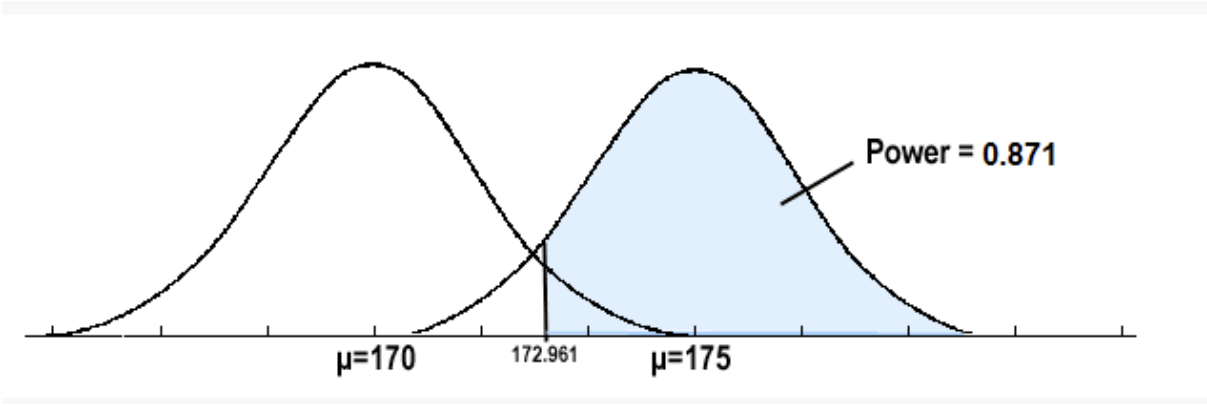
$$\bar{x} = 170 + 1.645\left(\frac{9}{\sqrt{25}}\right)$$

$$= 172.961$$

Bu nedenle, gözlemlenen örnek ortalaması 172.961 veya daha büyük olduğunda boş hipotezi reddetmeliyiz:

$$\begin{aligned}\text{Power} &= P(\bar{x} \geq 172.961 \text{ when } \mu = 175) \\ &= P\left(z \geq \frac{172.961 - 175}{9/\sqrt{25}}\right) \\ &= P(z \geq -1.133) \\ &= 0.8713\end{aligned}$$

and illustrated below:




Özetle, gerçek bilinmeyen popülasyon ortalaması gerçekte ise, Ha, μ>175 alternatif hipotez lehine Ho, μ=170 sıfır hipotezini reddetme şansımızın %87,13 olduğu belirlenir.

**Numune Boyutu Bölümünün Hesaplanması**

Örneklem büyüklüğü sabitse, Tip I α hatasının azaltılması Tip II β hatasını da artıracaktır. Her ikisinin de azalması isteniyorsa, örneklem büyüklüğü artırılmalıdır.

Belirtilen α, β, $\mu_a$ değerleri için gereken en küçük örnek boyutunu hesaplamak için, μa, gücü değerlendirmek istediğiniz μ'nin olası değeridir.

Tek Kuyruklu Test için Örnek Boyutu:

$$n = \frac{\sigma^2(Z_\alpha + Z_\beta)^2}{(\mu_0 - \mu_a)^2}$$

İki Kuyruklu Test için Örnek Boyutu:

$$n = \frac{\sigma^2(Z_{\alpha/2} + Z_\beta)^2}{(\mu_0 - \mu_a)^2}$$

**Örnek:**
X, rastgele bir üniversite öğrencisinin boyunu göstersin. X'in bilinmeyen μ ortalama ve standart sapma 9 ile normal olarak dağıldığını varsayın.

α=0.05 durumunda I. sıfır hipotezi Ho, μ=170 değerini alternatif hipoteze Ha, μ>175 karşı test edebiliriz. μ = 175 alternatifinde 0.90 güce ulaşmak için gerekli olan numune boyutunu n bulun.

$$\begin{aligned} n &= \frac{\sigma^2(Z_\alpha + Z_\beta)^2}{(\mu_0 - \mu_a)^2} \\ &= \frac{9^2(1.645 + 1.28)^2}{(170 - 175)^2} \\ &= 27.72 \\ n &= 28 \end{aligned}$$

Özetle, veriler sıfır hipotezinin reddedilemeyeceğini gösterdiğinde doğru kararın verebilebilmesi için güç analizinin ne kadar önemli olduğu görülmelidir. Ayrıca, araştırmanın ihtiyaçlarını karşılayan bir farkı tespit etmek için gereken minimum örnek boyutunu hesaplamak için güç analizinin nasıl kullanılabileceği de görilmelidir.

## 8.2. Markov Zinciri

Markov Zincirleri, zamana bağlı, uzaya bağlı stokastik süreçleri modellemek için basit ve çok kullanışlı araçlardan biridir. Finans (hisse fiyatı hareketi), satışlar (satış miktarı bilgisi), NLP algoritmaları (sonlu durum dönüştürücüler, POS Etiketleme için Gizli Markov Modeli), hava durumu tahmini vb. gibi birçok alan, tahminlerini kolay ve doğru bir şekilde yapmak için Markov zincirini kullanır.

Markov zinciri, geleceğin geçmişe bağlı olmadığı, şimdiye bağlı olduğu bir stokastik süreçler sınıfını temsil eder. Bir stokastik süreç, eğer süreç geleceği işleyecek olan Markovyen özelliklerden oluşuyorsa, Markov zinciri olarak düşünülebilir. Şimdiki durum ve geçmiş süreçten bağımsız bilgilere ihtiyacımız var.

Xn durumunun n zamanda kaydedildiği bir durumu düşünün. n+1 anındaki gelecek durum, n anındaki duruma bağlıdır. n zamanında vaka sayısının Xn olduğu ve n+1 zamanında vaka sayısının Xn+1 olduğu korona vakalarına bir örnek verelim. Dolayısıyla, Markov zinciri tanımını izliyorsak, n+1 zamanındaki vakaların sayısı, n zamanındaki vakaların sayısına bağlı olacaktır (Xn+1, Xn'ye bağlı olacaktır), geçmişte {Xn−1, Xn olan vakaların sayısına bağlı olacaktır. -2, . . . , X0}. Markov zincirini anlamak için temelde Markov zinciri kavramı tarafından kullanılan bazı terimleri anlamamız gerekebilir. Bu terimler aşağıda açıklanmıştır.

**Durum uzayı:**
Markov zinciri için durum uzayı S = {1,2,3….., n} olduğunda S ile verilebilirse, sürecin durumu Xn değeri ile verilebilir. Örneğin, Xn = 8 ise, sürecin durumu 8'dir. Dolayısıyla, herhangi bir n anında, sürecin Xn değeri tarafından verildiği durumu söyleyebiliriz.

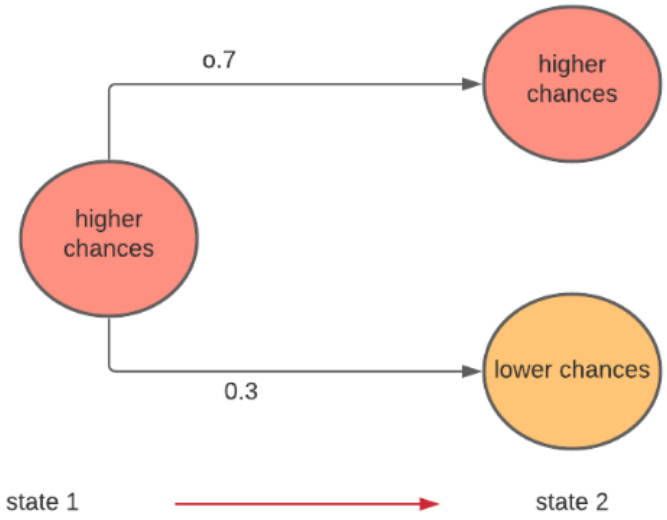
Örneğin, bir öğrenci sınıfında, eski başarısız kaydı olan öğrencilerin başarısız olarak nihai bir sonuç geliştirme olasılığı daha yüksektir ve önceki sınavda daha düşük not alan öğrencilerin sonucu başarısız olarak alma olasılığı daha yüksektir. Dolayısıyla bu durumda, eski başarısız kaydı olan öğrencilerin sınavdan kalma şanslarının daha yüksek, notu düşük olan öğrencilerin ise daha düşük olduğunu söyleyebiliriz. Bu senaryoda iki durumumuz var: daha düşük şans ve daha yüksek şans. Ve S={1,2}.

***Yörünge:***
Markov zincirinin yörüngesi, stokastik sürecin başından beri var olduğu durumların sırası olarak düşünülebilir.

Yani yörünge değerlerini s0,s1,s2…….sn olarak gösterebilirsek, durum X0=s0, X1 = s1…….Xn=sn olarak değerler alacaktır.

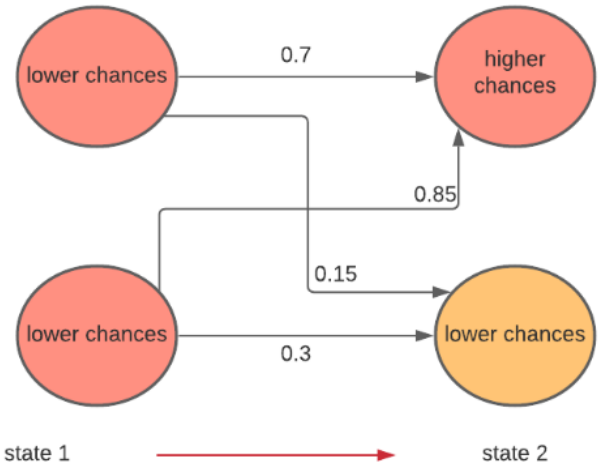
**Geçiş olasılığı**

Markov zincirleri belirli bir zamanda kayıtsız durumlar olamaz, ancak zamanla durumlarını değiştirebilirler. Durumdaki değişiklik, durum geçişi olarak adlandırılabilir. Yukarıda verilen örnekten, örneğin Markov zinciri ya daha düşük ya da daha yüksek şansa sahip olabilir.




Yukarıdaki resim, durumların geçişinin bir temsilidir. Durum 1'de zincir, devam etmekte olan sınavın başarısızlık şansının daha yüksek olduğu durumda olduğunu söyleyebiliriz. Bir sonraki sınavın başarısızlık şansı daha yüksek olan bir duruma girme olasılığı 0.7'dir ve durumun daha düşük şansa geçme olasılığı 0.3'tür. Öğrencilerin mevcut sınavdan başka bir sınava daha düşük şans durumuna geçme olasılığı 0,3'tür.

Sistemin daha düşük şans durumunda olduğunu ve benzer bir geçiş diyagramının çizildiğini varsayalım. Burada geçiş olasılıkları 0.85 ve 0.15'tir. Her iki diyagramı da kullanarak tam bir süreç çizebiliriz.



Yukarıdaki görüntü, Durum 1'den Durum 2'ye Birleşik-durum geçiş diyagramının bir temsilidir. Bir zaman örneği için, bu süreçler geriye doğru gidemezler, ancak bir sonraki zaman örneğinde geriye gidebilirler.

**Durum geçiş matrisi**

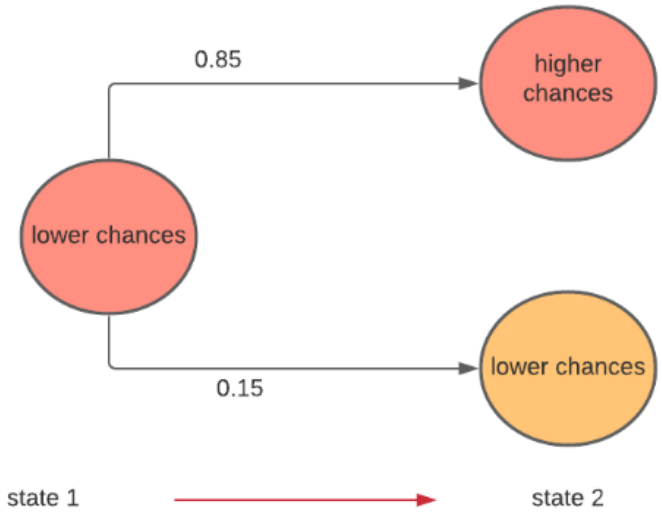
Tüm geçiş olasılıklarının matrisine geçiş matrisi denir. Satırların başlangıç noktası ve sütunların bitiş noktası olduğu yer.

| | lower chances | higher chances |
|---|---|---|
| lower chances | 0.15 | 0.85 |
| higher chances | 0.3 | 0.7 |

Yukarıdaki matris, verilen örneğin geçiş matrisinin bir temsilidir. sürecin düşük şans durumundan düşük riske geçişi 0.15 olasılığa sahiptir. Düşük riskten yüksek şansa geçişin olasılığı 0.85'tir.

**Markov Zinciri Kullanarak Tahmin**

Markov zinciri, gelecekteki değer için tahminler yapmak için çok güçlü bir araçtır. Çeşitli faydalı içgörüler sağladığından, içgörüleri anlamak için geçiş olasılıklarını, geçiş matrisini, durum uzayını ve yörüngeyi bilmek çok gerekli hale gelir.



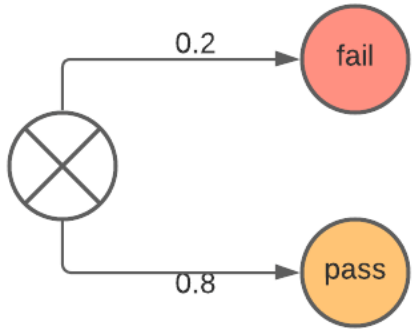
Ayrıca süreç hakkında ön bilgi sahibi olunması gereken temel şeylerden biri de sürecin başlangıç durumudur. Tahmin sürecini açıklamak için, birkaç değişikliğin uygulandığı yukarıdaki öğrenci başarısızlık şansı örneğine bir göz atalım.

**İlk Durum ve Tek Adımlı Tahmin**

Bu sefer bir mühendislik sınavı ve gözlem şu ki, öğrenciler ilk yıl matematik sınavlarında başarısız olurlarsa, temel derslerinde üç kez başarısız olurlar ve ilk yıl matematiği geçerlerse, çekirdek derslerini geçme olasılıkları daha yüksektir. dört kez konu sınavı. Yani örneğin geçiş matrisi

| | fail | pass |
|---|---|---|
| fail | 0.75 | 0.25 |
| pass | 0.2 | 0.8 |

**Sürecin ilk hali şu şekilde olacaktır:**




Yukarıdaki başlangıç durumundan, öğrencilerin sadece başlangıç durumu ve geçiş matrisini çarparak çekirdek konuyu geçme olasılığı şeklinde bir gelecek tahmini yapabiliriz.

Verilen örnek için bir sonraki adım için tahmin olacaktır.

$$\begin{Bmatrix} 0.2 & 0.8 \end{Bmatrix} \begin{Bmatrix} 0.75 & 0.25 \\ 0.2 & 0.8 \end{Bmatrix} = \begin{Bmatrix} 0.31 & 0.69 \end{Bmatrix}$$

Yukarıdaki sezgiden yola çıkarak, ilk durumdan sonraki ilk durum için tahminin aşağıdaki formülle verilebileceğini söyleyebiliriz.

İlk Durum X Geçiş Matrisi = Tahmin

**Uzun Dönem Olasılık**

Uzun dönem olasılığı, kararlı durum olasılığı olarak kabul edilebilir. Çünkü prosedürdeki bir durum kararlı olduğunda kararlı durum olasılığını hesaplayabiliriz. Burada Markov zincirinde, eğer ilk aşama kararlı ise, yani bir kez sabit hale geldiğinde, kararlı durum olasılığını hesaplayabiliriz.

Diyelim ki V0 ilk durum olasılık vektörü ve T geçiş matrisi, yani tek seferlik adım tahmini şu şekilde gösterilebilir:

$V1 = V0 \times T$

Burada dikkate değer ve çok basit bir matematik olan bir şey, bir vektördeki vektör ve matrisin nokta çarpımıdır ve bu sezgiyle, bir kerelik adımı tahmin etme sürecinde tekrar bir vektörle karşılaştığımızı söyleyebiliriz. başlangıç durumu olarak kabul edilir. Veya daha resmi olarak gelecekte tahmin edilen her bir tek seferlik adımın yalnızca bir sonraki adımından sorumlu olacağını söylemek.

Dolayısıyla, ikinci adımı tahmin etmek istiyorsak, tahmin formülü şu şekilde olacaktır:

$V2 = V1 \times T$

Ve burada bir adımın tahmininden V1'in değerini biliyoruz. V1 değerini koyarak

$V2 = (V0 \times T) \times T$

$V2 = V0 \times T^2$

Benzer şekilde, üçüncü adım için tahmin şu şekilde olacaktır:

$V3 = V2 \times T = (V0 \times T^2) \times T$

$V3 = V0 \times T^3$

Bu nedenle, n'inci zaman adımı tahmininden bahsederken, tahmin aşağıdaki formülle hesaplanabilir.

$V_n = V_{n-1} \times T = V_0 \times T^n$

Dolayısıyla, yukarıda verilen yinelemeli süreç, uzun süreçlerin gelecekteki durum olasılığının tahmininde bu şekilde yardımcı olur. Burada uzun dönem olasılık şu şekilde yazılabilir:

$V_\infty = V_0 \times T^\infty$.

Yukarıdaki uzun dönem olasılık formülünden, geçiş matrisi tarafından yapılan hiçbir çarpma miktarının uzun dönem olasılık vektöründe değişikliklere yol açmadığını söyleyebiliriz.

**Markov Zincirinin Avantajları**

- Yukarıda gördüğümüz gibi, Markov zincirini ardışık bir veriden türetmek çok kolaydır.
- Dinamik değişim mekanizmasının derinliklerine dalmamıza gerek yok.
- Markov zinciri çok anlayışlı. Herhangi bir sürecin eksik olduğumuz alanını söyleyebilir ve ayrıca iyileştirmeye göre değişiklikler yapabiliriz.
- Çok düşük veya mütevazı hesaplama gereksinimleri, sistemin herhangi bir boyutu tarafından kolayca hesaplanabilir.

**Markov Zincirinin Uygulanması**

Markov zincirleri, hava durumu, sıcaklık, satış vb. her türlü tahminde bulunabilen tahmin için kullanılabilir. Bu, müşteri davranışını tahmin etmek için kullanılabilir. Bildiğimiz gibi, sıralı verilerle iyidir, bu nedenle POS etiketleme gibi birçok NLP problem çözümü ile birleştirilebilir. Marka sadakati ve tüketici davranışı analiz edilebilir.
Oyun alanında şans oyununda çeşitli modeller geliştirilebilir.

Görildüğü gibi, Markov zinciri kavramını anlamak ve hesaplamak zor değil, bu nedenle her boyutta hesaplama için de çok kolay olduğunu söyleyebiliriz. Markov zincirinin kolaylığı ve doğruluğu nedeniyle daha birçok kullanım alanı olabilir, çok popüler bir araştırma konusu haline gelmiştir.

# 9. Standardization and Normalization

standardization,
normalization,
regularization,
normalization

Four common normalization techniques may be useful:

- scaling to a range
- clipping
- log scaling
- z-score

Normalleştirme, verilerinizin dağılımının bir Gauss dağılımını takip etmediğini bildiğiniz zaman kullanmak için iyidir. Bu, K-En Yakın Komşular ve Sinir Ağları gibi herhangi bir veri dağılımını varsaymayan algoritmalarda faydalı olabilir.
Öte yandan standardizasyon, verilerin Gauss dağılımını takip ettiği durumlarda yardımcı olabilir. Ancak, bu mutlaka doğru olmak zorunda değildir. Ayrıca, normalleştirmeden farklı olarak standardizasyonun sınırlayıcı bir aralığı yoktur. Bu nedenle, verilerinizde aykırı değerler olsa bile bunlar standardizasyondan etkilenmeyecektir.
Ancak günün sonunda, normalleştirme veya standardizasyon kullanma seçimi, probleminize ve kullandığınız makine öğrenme algoritmasına bağlı olacaktır. Verilerinizi ne zaman normalleştirmeniz veya standartlaştırmanız gerektiğini size söyleyecek kesin ve hızlı bir kural yoktur. Her zaman modelinizi ham, normalleştirilmiş ve standartlaştırılmış verilere uydurarak başlayabilir ve en iyi sonuçlar için performansı karşılaştırabilirsiniz.
Ölçekleyiciyi eğitim verilerine sığdırmak ve ardından test verilerini dönüştürmek için kullanmak iyi bir uygulamadır. Bu, model test süreci sırasında herhangi bir veri sızıntısını önleyecektir. Ayrıca, hedef değerlerin ölçeklendirilmesi genellikle gerekli değildir.
Giriş veri setinin özellikleri, aralıkları arasında büyük farklılıklara sahip olduğunda veya basitçe farklı ölçüm birimlerinde ölçüldüğünde (örneğin, Pound, Metre, Miles… vb.)
Başlangıç özelliklerinin aralıklarındaki bu farklılıklar, birçok makine öğrenimi modelinde sorunlara neden olur. Örneğin, mesafe hesaplamasına dayalı modeller için, özelliklerden biri geniş bir değer aralığına sahipse, mesafe bu özel özellik tarafından yönetilecektir.
Bunu bir örnekle açıklamak gerekirse: Diyelim ki sırasıyla [1 ila 2] Metre ve [10 ila 200] Pound arasında değişen Metre cinsinden Yükseklik ve Pound cinsinden Ağırlık olmak üzere iki özelliğe sahip 2 boyutlu bir veri setimiz var. Bu veri setinde hangi mesafe tabanlı modeli uygularsanız uygulayın, Ağırlık özelliği Yükseklik özelliğine hakim olacak ve Yüksekliğe göre daha büyük değerlere sahip olduğu için mesafe hesaplamasına daha fazla katkısı olacaktır. Bu nedenle, bu sorunu önlemek için, standartlaştırma kullanarak özellikleri karşılaştırılabilir ölçeklere dönüştürmek çözümdür.

**Scaling to a range:**
Recall from MLCC that **scaling** means converting floating-point feature values from their natural range (for example, 100 to 900) into a standard range—usually 0 and 1 (or sometimes -1 to +1). Use the following simple formula to scale to a range:

x′=(x−xmin)/(xmax−xmin)

Scaling to a range is a good choice when both of the following conditions are met:

- You know the approximate upper and lower bounds on your data with few or no outliers.
- Your data is approximately uniformly distributed across that range.

A good example is age. Most age values falls between 0 and 90, and every part of the range has a substantial number of people.

In contrast, you would *not* use scaling on income, because only a few people have very high incomes. The upper bound of the linear scale for income would be very high, and most people would be squeezed into a small part of the scale.

**Bir aralığa ölçekleme:**
Ölçeklemenin, kayan nokta özellik değerlerini doğal aralıklarından (örneğin, 100 ila 900) standart bir aralığa - genellikle 0 ve 1 (veya bazen -1 ila +1) dönüştürmek anlamına geldiğini hatırlayın. Bir aralığa ölçeklendirmek için aşağıdaki basit formülü kullanılır:

$$x' = (x - x_{min})/(x_{max} - x_{min})$$

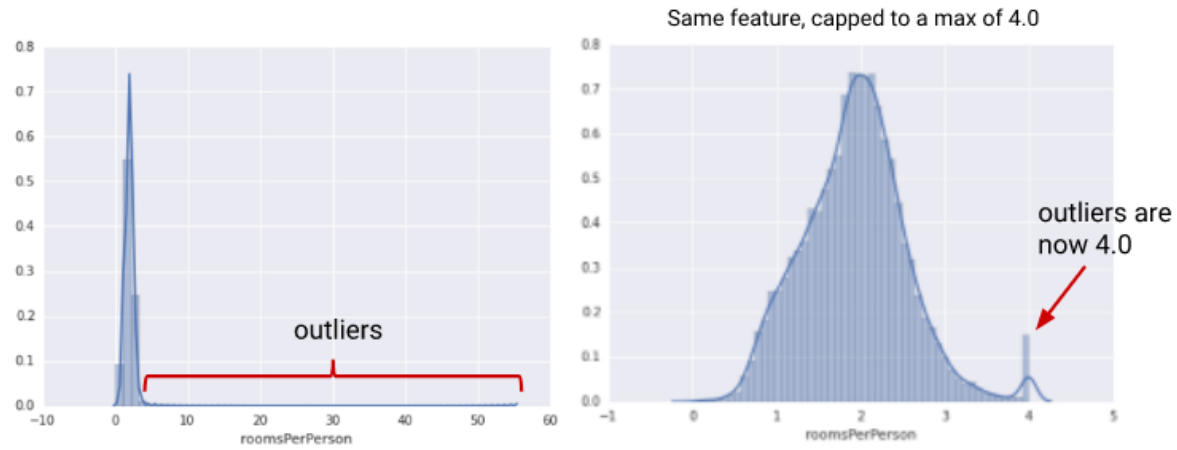
Bir aralığa ölçekleme, aşağıdaki koşulların her ikisi de karşılandığında iyi bir seçimdir:

- Çok az veya hiç aykırı değer olmadan verilerinizdeki yaklaşık üst ve alt sınırları bilirsiniz.
- Verileriniz bu aralıkta yaklaşık olarak eşit olarak dağıtılır.

İyi bir örnek yaştır. Çoğu yaş değeri 0 ile 90 arasındadır ve aralığın her bölümünde önemli sayıda insan bulunur.

Buna karşılık, gelir üzerinde ölçeklendirmeyi kullanmazsınız, çünkü yalnızca birkaç kişi çok yüksek gelire sahiptir. Doğrusal gelir ölçeğinin üst sınırı çok yüksek olacaktır ve çoğu insan ölçeğin küçük bir kısmına sıkıştırılacaktır.

**Feature Clipping:**



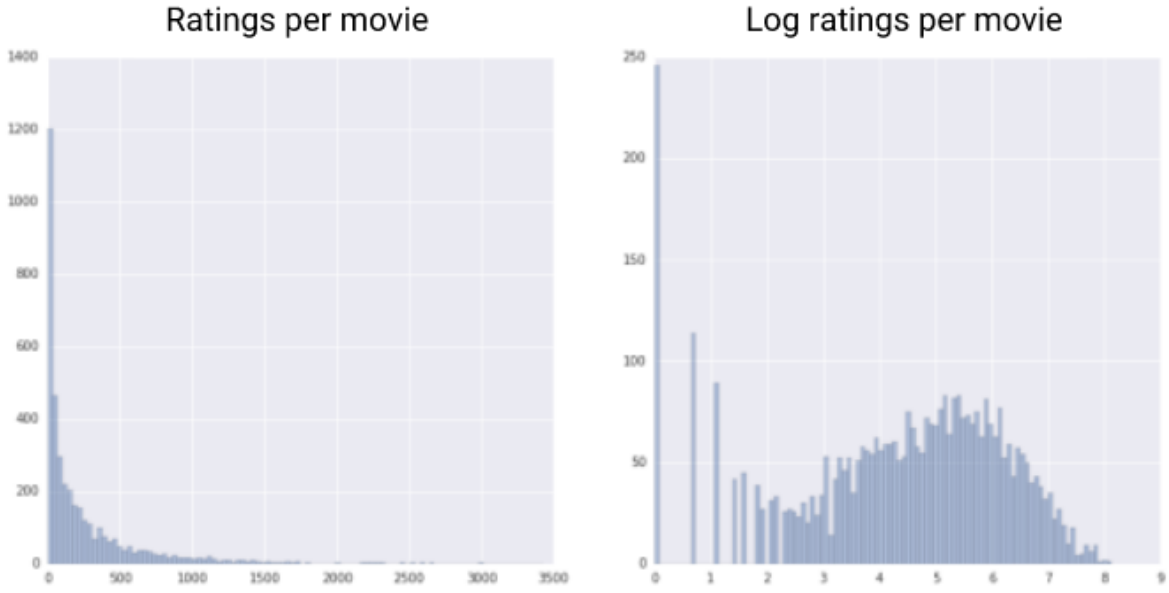

if x > max, then x' = max. if x < min, then x' = min

**Log Scaling:**

Log scaling computes the log of your values to compress a wide range to a narrow range.

x'=log(x)

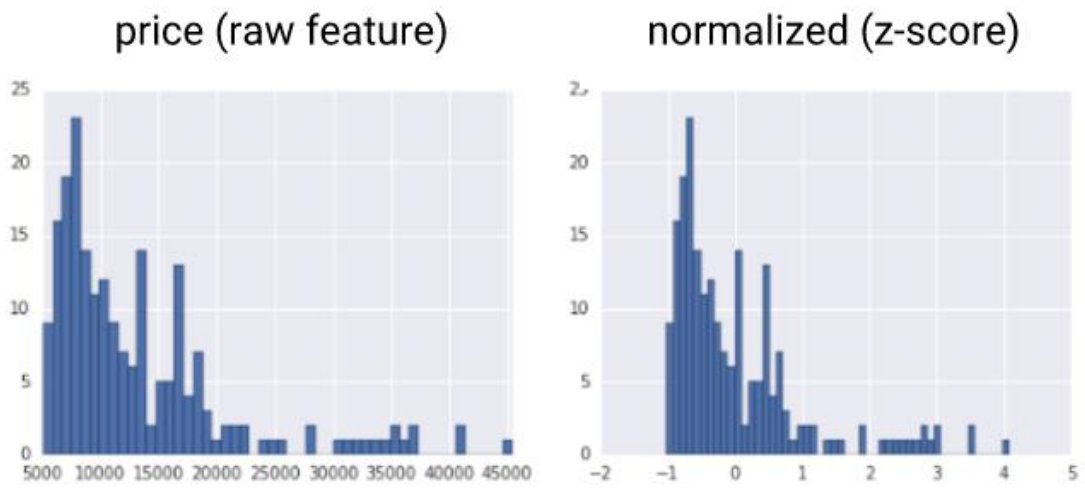
Log scaling is helpful when a handful of your values have many points, while most other values have few points. This data distribution is known as the *power law* distribution. Movie ratings are a good example. In the chart below, most movies have very few ratings (the data in the tail), while a few have lots of ratings (the data in the head). Log scaling changes the distribution, helping to improve linear model performance.




**Z-Score:**

Z-score is a variation of scaling that represents the number of standard deviations away from the mean. You would use z-score to ensure your feature distributions have mean = 0

and std = 1. It's useful when there are a few outliers, but not so extreme that you need clipping.
The formula for calculating the z-score of a point, *x*, is as follows:
x'=(x−μ)/σ
**Note:** $\mu$ is the mean and $\sigma$ is the standard deviation.




**Standard Normalization**
The most commonly applied type of normalization transforms all features to have a mean 0 and standard deviation of 1.

*In machine learning, we usually operate under the assumption that features are distributed according to a Gaussian distribution. The standard Gaussian bell curve, also known as the standard normal distribution, has a mean of 0 and a standard deviation of 1.*

Setting the mean to 0 is achieved by calculating the current mean for each variable x (each of which has n entries).

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Then you subsequently subtract the mean from each variable x to obtain your rescaled x with a new mean of 0.

$$x = x - \mu$$

Next, you need to calculate the standard deviation.

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$$

*Note that since the mean is already zero, we do not need to subtract the standard deviation anymore.*Lastly, you divide your variables (features) by the standard deviation. If you have a background in statistics, you might recognize that this process is equivalent to calculating z scores that are commonly used for constructing confidence intervals.

$$z = \frac{x - \mu}{\sigma}$$

It is, therefore, sometimes referred to as z-score normalization.

**Z-skoru,** verileri standartlaştırmanın en popüler yöntemlerinden biridir ve her özelliğin her değeri için ortalamanın çıkarılması ve standart sapmaya bölünmesiyle yapılabilir.

$$z = \frac{value - mean}{standard\ deviation}$$

Standardizasyon yapıldıktan sonra, tüm özelliklerin ortalaması sıfır, standart sapması bir ve dolayısıyla aynı ölçeğe sahip olacaktır.

**Özellik Kırpma**

Veri kümeniz aşırı uç değerler içeriyorsa, belirli bir değerin üzerindeki (veya altındaki) tüm özellik değerlerini sabit bir değerle sınırlayan özellik kırpmayı deneyebilirsiniz. Örneğin, 40'ın üzerindeki tüm sıcaklık değerlerini tam olarak 40 olacak şekilde kırpabilirsiniz.

Özellik kırpmayı diğer normalleştirmelerden önce veya sonra uygulayabilirsiniz.

| Normalization Technique | Formula | When to Use |
|---|---|---|
| Linear Scaling | $x' = (x - x_{min})/(x_{max} - x_{min})$ | When the feature is more-or-less uniformly distributed across a fixed range. |
| Clipping | if x > max, then x' = max. if x < min, then x' = min | When the feature contains some extreme outliers. |
| Log Scaling | x' = log(x) | When the feature conforms to the power law. |
| Z-score | x' = (x - μ) / σ | When the feature distribution does not contain extreme outliers. |

The most commonly applied type of normalization transforms all features to have a mean 0 and standard deviation of 1.

*In machine learning, we usually operate under the assumption that features are distributed according to a Gaussian distribution. The standard Gaussian bell curve, also known as the standard normal distribution, has a mean of 0 and a standard deviation of 1.*

Setting the mean to 0 is achieved by calculating the current mean for each variable x (each of which has n entries).

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

Then you subsequently subtract the mean from each variable x to obtain your rescaled x with a new mean of 0.

$$x = x - \mu$$

Next, you need to calculate the standard deviation.

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$$

*Note that since the mean is already zero, we do not need to subtract the standard deviation anymore.*
Lastly, you divide your variables (features) by the standard deviation.

$$x = \frac{x}{\sigma}$$

If you have a background in statistics, you might recognize that this process is equivalent to calculating z scores that are commonly used for constructing confidence intervals.

$$z = \frac{x - \mu}{\sigma}$$

It is, therefore, sometimes referred to as z-score normalization.

# 10. Information Theory

Araştırmacılar, 1900'lerin başından beri bilgiyi nicelleştirme üzerine kafa yordular ve 1948'de Claude Shannon, "A Mathematical Theory of Communication" adlı olağanüstü bir makale yayınladı. Bu makale Bilgi Teorisi alanını doğurdu. Bilgi Teorisi, tanım olarak, bilginin nicelenmesi, depolanması ve iletişiminin incelenmesidir. Ama bundan çok daha fazlası. İstatistiksel Fizik, Bilgisayar Bilimleri, Ekonomi vb. Alanlara önemli katkılar sağlamıştır.

Shannon'ın makalesinin ana odak noktası, makaleyi yayınladığında Bell Laboratuvarlarında çalıştığı için genel iletişim sistemiydi. Bilgi entropisi ve fazlalık gibi birkaç önemli kavram oluşturdu. Günümüzde temel temelleri kayıpsız veri sıkıştırma, kayıplı veri sıkıştırma ve kanal kodlama alanlarında uygulanmaktadır.

Bilgi Teorisinde kullanılan teknikler doğası gereği olasılıklıdır ve genellikle 2 spesifik nicelik ile ilgilenir, yani. Entropi ve Karşılıklı Bilgi. Bu iki terime daha derin bir dalış yapalım.

**Shannon Entropy (or just Entropy)**

Entropy is the measure of uncertainty of a random variable or the amount of information required to describe a variable. Suppose $x$ is a discrete random variable, and it can take any value defined in the set, $\chi$. Let's assume the set is finite in this scenario. The probability distribution for $x$ will be $p(x) = \Pr\{\chi = x\}, x \in \chi$. With this in mind, entropy can be defined as

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x).$$

**Entropy**

The unit of entropy is bit. If you observe the formula, entropy is entirely dependent on the **probability of the random variable** and not on the value of $x$ itself. There is a negative sign in front of the formula, making it perpetually positive or 0. If entropy is 0, there is no new information to be gained. I will demonstrate the implementation of this formula through an example.

Consider the scenario of a coin toss. There are two probable outcomes, heads or tails, with equal probabilities. If we quantify that, $x \in$ {heads,tails}, and $p$(heads) = 0.5 and $p$(tails) = 0.5. If we plug in these values in the formula:

$$H(x) = -\left[\left(p(x_{heads}) * \log\left(p(x_{heads})\right)\right) + \left(p(x_{tails}) * \log\left(p(x_{tails})\right)\right)\right]$$
$$= -[(0.5 * log_2 0.5) + (0.5 * log_2 0.5)] = -[(-0.5) + (-0.5)] = 1$$

**Calculating Entropy in a coin toss event**

Therefore, entropy is 1 bit, i.e., the coin toss's outcome can be expressed completely in 1 bit. So, to intuitively express Shannon entropy's concept, it is understood as "how long does a message need to be to convey its value completely". I want to dive a little deeper and discuss the concepts of Joint Entropy, Conditional Entropy and Relative Entropy.

**Joint and Conditional Entropy**

Previously, I defined Entropy for a single random variable, but now I will extend it to a pair of random variables. It is a simple aggregation as we can define the pair of variables (*X*, *Y*) as a single vector-valued random variable.

The joint entropy *H*(*X*, *Y*) of a pair of discrete random variables (*X*, *Y*) with a joint distribution *p*(*x*, *y*) is defined as

$$H(X, Y) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log p(x, y)$$

**Joint Entropy**

This can also be represented in terms of expected value.

$$H(X, Y) = -E \log p(X, Y)$$

**Joint Entropy (Expected Value form)**

Similarly, for conditional entropy, *H*(*Y*|X) is defined as:

$$H(Y|X) = \sum_{x \in \mathcal{X}} p(x) H(Y|X = x)$$

**Conditional Entropy**

Intuitively, this is the average of the entropy of *Y* given *X* over all possible values of *X*. Considering the fact that (*X*, *Y*) ~ *p*(*x*, *y*), the conditional entropy can also be expressed in terms of expected value.

$$H(Y|X) = -E \log p(Y|X).$$

**Conditional Entropy (Expected Value form)**

Let's try an example to understand Conditional Entropy better. Consider a study where subjects were asked:

I) if they smoked, drank or didn't do either.

II) if they had any form of cancer

Now, I will represent these questions' response as two different discrete variables belonging to a joint distribution.

| Activity | Cancer |
|---|---|
| Smoking | Yes |
| Smoking | Yes |
| Smoking | Yes |
| Alcohol | No |
| Neither | Yes |
| Alcohol | Yes |
| Alcohol | Yes |
| Smoking | No |
| Alcohol | Yes |
| Neither | No |

**Data Table**

On the left, you can see the data table where 10 subjects answered the questions. We have three different possibilities for the variable Activity (I will call it X). The second column represents whether the subject has/had cancer (variable Y). There are two possibilities here, i.e. Yes or No. Since we are not dealing with continuous variables yet, I have kept these variables discrete. Let's create a probability table that will make the scenario clearer.

| | Yes | No |
|---|---|---|
| Smoking | $^{3}/_{10}$ | $^{1}/_{10}$ |
| Alcohol | $^{3}/_{10}$ | $^{1}/_{10}$ |
| Neither | $^{1}/_{10}$ | $^{1}/_{10}$ |

**Probability Table for the above example**

Next, I will calculate the value of the marginal probability $p(x)$ for all the possible value of X.

$$p(X = \text{"Smoking"}) = \frac{4}{10}$$

$$p(X = \text{"}Alcohol\text{"}) = \frac{4}{10}$$

$$p(X = \text{"}Neither\text{"}) = \frac{2}{10}$$

**Marginal Probability of X**

Based on the probability table, we can plug in the value in the conditional entropy formula.

$$
\begin{aligned}
H(Y|X) &= p(\text{"Smoking"})H(Y|X=\text{"Smoking"}) + p(\text{"Alcohol"})H(Y|X=\text{"Alcohol"}) + p(\text{"Neither"})H(Y|X=\text{"Neither"}) \\
&= -\frac{4}{10}\left\{\frac{3}{10}\log\left(\frac{3}{10}\right)+\frac{1}{10}\log\left(\frac{1}{10}\right)\right\} - \frac{4}{10}\left\{\frac{3}{10}\log\left(\frac{3}{10}\right)+\frac{1}{10}\log\left(\frac{1}{10}\right)\right\} - \frac{2}{10}\left\{\frac{1}{10}\log\left(\frac{1}{10}\right)+\frac{1}{10}\log\left(\frac{1}{10}\right)\right\} \\
&= 0.4 * 0.857 + 0.4 * 0.857 + 0.2 * 0.664 \\
&= 0.8184\ bits
\end{aligned}
$$

**Conditional Probability in the above example**

**Relative Entropy**

Relative Entropy is somewhat different as it moves on from random variables to distributions. It is a measure of the distance between two distributions. **A more instinctive way to put it would be: Relative entropy or KL-Divergence, denoted by *D*(*p*||*q*), is a measure of the inefficiency of assuming that the distribution is *q* when the true distribution is *p*.** It can be defined as:

$$D(p||q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}$$

Relative entropy is always non-negative and can be 0 only if *p* = *q*. Although a point to note here is that it is not a true distance since it is not symmetric in nature. But it is often considered as a "distance" between distributions.

Lets take an example to solidify this concept! Let *X = {0,1}* and consider two distributions *p* and *q* on *X.* Let *p*(0) = 1 — *r*, *p*(1) = *r* and let *q*(0) = 1 — *s*, *q*(1) = *s.* Then,

$$D(p||q) = (1-r)\log\left[\frac{1-r}{1-s}\right] + r log \frac{r}{s}$$

Relative Entropy for p||q

I would also like to demonstrate the non-symmetric property, so I will also calculate *D*(*q*||*p*).

$$D(q||p) = (1-s)\log\left[\frac{1-s}{1-r}\right] + s log \frac{s}{r}$$

Relative Entropy for q||p

If r = s, *D*(*p*||*q*) = *D*(*q*||*p*) = 0. But I will take some different values, for instance, *r* = 1/2 and *s* = 1/4.

$$D(p||q) = \left(1-\frac{1}{2}\right)\log\left[\frac{1-\frac{1}{2}}{1-\frac{1}{4}}\right] + \frac{1}{2} log \frac{\frac{1}{2}}{\frac{1}{4}} = 0.2075 \text{ bit}$$

$$D(q||p) = \left(1 - \frac{1}{4}\right) \log \left[\frac{1 - \frac{1}{4}}{1 - \frac{1}{2}}\right] + \frac{1}{4} log \frac{\frac{1}{4}}{\frac{1}{2}} = 0.1887 \text{ bit}$$

As you can see, $D(p||q) \neq D(q||p)$.

Now, since we have discussed the different types of entropies, we can move onto Mutual Information.

**Mutual Information**

Mutual Information is a measure of the amount of information that one random variable contains about another random variable. **Alternatively, it can be defined as the reduction in uncertainty of one variable due to the knowledge of the other.** The technical definition for it would be as follows:

Consider two random variables $X$ and $Y$ with a joint probability mass function $p(x, y)$ and marginal probability mass functions $p(x)$ and $p(y)$. The mutual information $I(X; Y)$ is the relative entropy between the joint distribution and the product distribution $p(x)p(y)$.

$$I(X; Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x, y) \log \frac{p(x, y)}{p(x)\, p(y)}$$
$$= D(p(x, y)||p(x)\, p(y))$$

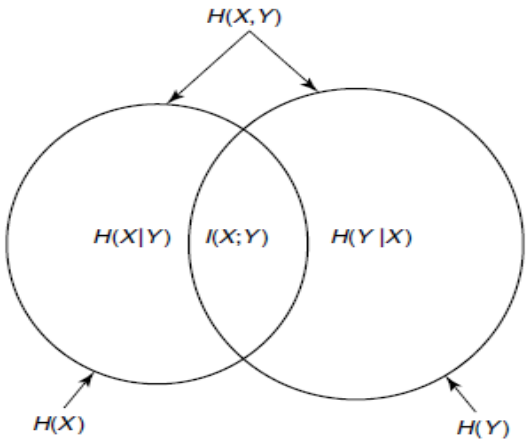
Mutual Information

Mutual Information can also be expressed in terms of Entropy. The derivation is quite fun, but I will refrain myself as it might clutter the article.

$$I(X; Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

Mutual Information w.r.t. Entropy

From the above equation, we see that mutual information is the reduction in the uncertainty of $X$ due to the knowledge of $Y$. There is a Venn diagram perfectly describing the relationship.




Relationship between Mutual Information and Entropy

Let's take an example to understand it better. I can use the example of relating Smoking, Drinking, Neither with having cancer that I used while explaining entropy. We had seen that the $H(Y|X) = 0.8184$ bits. To calculate Mutual Information, I require one more term $H(Y)$. $H(Y)$, in this case, will be:

$$\begin{aligned} H(Y) &= -\sum_{y \in Y} p(y) \log p(y) \\ &= \frac{7}{10} \log \frac{7}{10} + \frac{3}{10} \log \frac{3}{10} \\ &= 0.876 \end{aligned}$$

Therefore, Mutual Information is defined by:

$$\begin{aligned} I(X;Y) &= H(Y) - H(Y|X) \\ &= 0.876 - 0.8184 \\ &= 0.0576 \end{aligned}$$

Rassal bir değişkenin belirsizlik ölçütü olarak bilinen Entropi, bir süreç için tüm örnekler tarafından içerilen enformasyonun beklenen değeridir. ***Enformasyon ise rassal bir olayın gerçeklesmesine ilişkin bir bilgi ölçütüdür.*** Eşit olasılıklı durumlar yüksek belirsizliği temsil eder. Shannon'a göre bir sistemdeki durum değiştiğinde entropideki değişim kazanılan enformasyonu tanımlar. Buna göre maksimum belirsizlik durumundaki değişim muhtemelen maksimum enformasyonu sağlayacaktır. Shannon bilgiyi bitlerle temsil ettiği için logartimayı iki tabanında kullanımıştır.

$$I(x) = \log_2 \frac{1}{P(x)} = -\log_2 P(x)$$

***Shannon'a göre entropi, iletilen bir mesajın tasıdığı enformasyonun beklenen değeridir.***

Shannon Entropisi (H) adıyla anılan terim, tüm $x_i$ durumlarına ait $P(x_i)$ olasılıklarına bağlı bir değerdir.

$$H(X) = E(I(X)) = \sum_{1 \le i \le n} P(x_i).I(x_i) = \sum_{i=1}^{n} P(x_i) \log_2 \frac{1}{P(x_i)} = -\sum_{i=1}^{n} P_i \log_2 P_i$$

$$Log_2(P) = \frac{10}{3} Log_{10}(P)$$

$$H(X) = -\frac{10}{3} \sum_{i=1}^{n} P_i \, Log_{10} P_i$$

**10 tabanında logaritmik ifadeler:**

- Log1=0, Log 2 ≈ 0.3, Log 3 ≈ 0.477, Log 5 ≈ 0.7, Log 7 ≈ 0.845, Log10=1
- $\log(a*b)=\log a + \log b$; $\log a^n = n*\log a$

**Örnek:**

Bir paranın havaya atılması olayı, rassal X sürecini temsil etsin. Yazı (1/2) ve tura (1/2) gelme olasılıkları eşit olduğu için X sürecinin entropisi 1 olan para atma olayı (X) gerçeklestiğinde 1 bitlik bilgi kazanılacaktır.

$$H(X) = -\sum_{i=1}^{2} p_i \log_2 p_i = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1$$

**Örnek:**

Hava, nem, rüzgar, su sıcaklığı gibi değerlere göre pikniğe gidip gitmeme kararı verilmiş 4 olay sonucunda pikniğe gidildi mi? sorusunun iki yanıtı var. Evet yanıtının olasılığı: ¾, hayır yanıtının olasılığı:1/4.

| Olay No | Hava | Nem | Rüzgar | Su sıcaklığı | Pikniğe gidildi mi? |
|---|---|---|---|---|---|
| 1 | güneşli | normal | güçlü | ılık | Evet |
| 2 | güneşli | yüksek | güçlü | ılık | Evet |
| 3 | yağmurlu | yüksek | güçlü | ılık | Hayır |
| 4 | güneşli | yüksek | güçlü | soğuk | Evet |

$$H(X) = -\sum_{i=1}^{2} p_i \log_2 p_i$$

Entropy(S)=H(x)=-(3/4)$\log_2$(3/4)-(1/4)$\log_2$(1/4)=0.811

**Örnek:**

Makine öğrenmesi algoritmasının olasılık hesaplanması sonucunda karar vermesi gerekmektedir. İki durum söz konusudur. Birinci durumun olma olasılığı, P1=0.6, olmama olasığı, P2=0.4 ise entropisini hesaplayınız.

$Log_2(0.6) = -0.743$

$Log_2(0.4) = -4/3$

Entropi, H(x)=0.6*0.743+0.4*4/3=0.979

## Calculation of Mutual Information

Alternatively, I can also use *H*(*X*) and *H*(*X*|*Y*) to calculate Mutual Information, and it will yield the same result. We can see how knowing X means so little for the uncertainty of variable Y. Let me change the passage of this example and lead you to how it all makes sense in Machine Learning. Suppose X is a predictor variable and Y is the predicted variable. Mutual Information between them can be a great precursor to check how useful the feature will be for predictions. Let us discuss the implications of Information Theory in Machine Learning.

**Applications of Information Theory in Machine Learning**

There are quite a few applications around, but I will stick with a few popular ones.

## Decision Trees

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. The core algorithm used here is called ID3, which was developed by Ross Quinlan. It employs a top-down greedy search approach and involves partitioning the data into subsets with homogeneous data. **The ID3 algorithms decide the partition by calculating the homogeneity of the sample using entropy.** If the sample is homogenous, entropy is 0, and if the sample is uniformly divided, it has maximum entropy. But entropy does not have direct implication on the construction of the trees. The algorithm relies on Information Gain, which is based on the decrease in entropy after a dataset is split on an attribute. If you think intuitively, you will see that this is actually Mutual Information that I had mentioned above. Mutual Information decreases the uncertainty of one variable given the value of the other. In DT, we calculate the entropy of the predicted variable. Then, the dataset is split based on entropy, and the entropy of the resultant variable is subtracted from the previous entropy value. This is Information Gain and, obviously, Mutual Information in play.

## Cross-Entropy

Cross entropy is a concept very similar to Relative Entropy. Relative entropy is when a random variable compares true distribution *p* with how the approximated distribution *q* differs from *p* at each sample point (divergence or difference). Whereas cross-entropy directly compares true distribution *p* with approximated distribution *q*. Now, cross-entropy is a term heavily used in the field of deep learning. **It is used as a loss function that measures the performance of a classification model whose output is a probability value between 0 and 1.** Cross-entropy loss increases as the predicted probability diverge from the actual label.

## KL-Divergence

K-L Divergence or Relative Entropy is also a topic embedded in the deep learning literature, specifically in VAE. Variational Autoencoders take in input in the form of Gaussian Distributions rather than discrete data points. It is optimal for the distributions of the VAE to be regularized to increase the amount of overlap within the latent space. **K-L divergence measures this and is added to the loss function.**

K-L Divergence is also used in t-SNE. tSNE is a dimensionality reduction technique that is mainly used to visualize data in high dimensions. It converts similarities between data points to joint probabilities and **tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data.**
**Calculating imbalance in target class distribution**

Entropy can be used to calculate target class imbalances. If we consider the predicted feature as a random variable with two classes, a balanced set (50/50 split) should have the maximum entropy as we saw in the case of the coin toss. But if the split is skewed and one class has a 90% prevalence, then there's lesser knowledge to be gained, hence a lower entropy. Implementing the chain rule for calculating entropy, we can check whether a multiclass target variable is balanced in a single quantified value, albeit an average that masks the individual probabilities.

# 11. Robotiğin Arkasındaki Matematik

Robotik araştırmaları katlanarak artıyor ve yeni bir endüstriyel devrime işaret ediyor. Bugün dünya çapında bir milyonun üzerinde robot çalışıyor ve bu sayı zamanla artıyor.

Bugün, insan zekası, fiziksel yetenek, algı ve davranışla karşılaştırılabilir ve hatta ötesinde beyin gücüne sahip birçok robotumuz var. Ve bilgisayar destekli cerrahi gibi bazı alanlarda, bu akıllı makineler insan yeteneklerini bile aşabilir.

Tehlikeli maddelerin taşınmasından bileşenlerin kaynaklanmasına, müşteriler için belgelerin dosyalanması veya halıların süpürülmesi gibi idari görevlerin yerine getirilmesine kadar, Robotlar toplumda birçok önemli rol oynamaktadır.

Ama onlara tüm dahice şeyleri yaptıran nedir? Matematik, mühendislik ve fiziğin çeşitli yönleri sihrin gerçekleşmesini sağlar.

**Robotlar Dünyaya Ne Zaman Tanıtıldı?**

Robotik çalışmaları, en eski matematiksel çalışmalar kadar eskidir. “Robot” kelimesi, 1920 yılında Çek yazar Karel ÄŒapek'in “Rossum’un Evrensel Robotları” adlı oyunuyla dünyaya tanıtılmıştır.

18. yüzyılda, Leonardo da Vinci uzuvlarını hareket ettirebilen, bükülebilen ve oturabilen programlanabilir bir robot tasarladı. 19. yüzyıl, Charles Babbage (İngiliz matematikçi ve programlanabilir bilgisayarın mucidi) ve Ada Lovelace'in (Babbage'ın analitik motoru için programlar yarattı) çalışmalarıyla robotiğe önemli katkılar getirdi.

20. yüzyılda Norbert Wiener, Alan Turing, John von Neumann ve Claude Shannon, robotik inovasyonunda bir patlamaya yol açtı.

Artık Robot Teknolojisi endüstride yaygın olarak kullanılmaktadır. Bir odada kendi yolunu bulabilen, ışığı, kokuları algılayabilen, uzanıp tutabilen, bir şeyleri ve hatta ifadeleri öğrenip tanıyabilen, insan beynini taklit edebilen ve çok daha fazlasını yapabilen Mobil robotik, endüstriyel robotik, saha robotiği ve gelişmiş robotiklerimiz var..

**Robotik Sistem Nedir?**

Matematiksel açıdan robotik sistem, çoklu alt sistemlerdeki fonksiyonlarla temsil edilen karmaşık bir sistemdir. Algı ve eylem arasındaki akıllı bağlantıdır ve hareketlilik özelliklerine göre sınıflandırılır. Matematiksel kavramlar ne kadar zorsa, robotik bir sistemde o kadar fazla esneklik ortaya çıkar.

**Robot modelleme, planlama ve Kontrolde matematiğin uygulanması**
Robotik okumak isteyenler matematiği anlamak için ellerinden geleni yapmalıdır çünkü robotik araştırmalarında bir zorunluluktur. Doğru boyuttaki parçaları bulmak, görevleri gerçekleştirmek için gereken ölçümleri yapmak, performansı test etmek veya hız ve güç veya tekerlek çapı ve kat edilen mesafe arasındaki kalıpları ve ilişkileri tespit etmek için bazı temel matematiksel kavramları anlamanız gerekir.

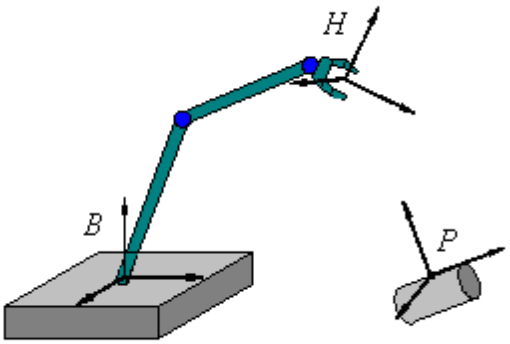
**Robot modelleme, planlama ve Kontrolde matematiğin uygulanması**
Robotikte matematiği anlamak araştırmalarında bir zorunluluktur. Doğru boyuttaki parçaları bulmak, görevleri gerçekleştirmek için gereken ölçümleri yapmak, performansı test etmek veya hız ve güç veya tekerlek çapı ve kat edilen mesafe arasındaki kalıpları ve ilişkileri tespit etmek için bazı temel matematiksel kavramları anlamanız gerekir.

Robotik için temel matematik önkoşulları şunlardır:

- Matematik
- Adi diferansiyel denklemler
- Gelişmiş Lineer cebir
- Geometri
- Sayısal analiz

İşte bir robotun tüm işlemleri verimli bir şekilde gerçekleştirmesine yardımcı olan matematiksel kavramların bazı yönleri.

**Dönüşüm Matrisleri, Robotikte Koordinat Sistemlerini Değiştirmeye yardımcı olur.**




Belirli bir eylemi gerçekleştirmek için robot koordinat sistemini değiştirir. Bu nedenle robotik problemlerin birden fazla koordinat sistemi vardır. Yukarıda gösterilen resimde, bağlı üç koordinat sistemi vardır; robotun tabanı **B**, eli **H** ve robotun tutması gereken parça **P.**
Sistemler, B tabanını, H elini veya P parçasını hareket ettirirseniz, ilgili koordinat sistemlerinin onunla birlikte hareket etmesi anlamında birbirine bağlıdır.
Koordinat sistemi P, silindir üzerindeki noktaları bulur.
Koordinat sistemi B, elin konumunu tanımlar.

Koordinat sistemi H elden olan mesafeyi ölçer.
Ana odak noktası, elin parçayı kavramak için doğru şekilde hareket etmesi için robotun eline H göre P parçasının konumunu ve yönünü belirlemektir.

Bunu hesaplamak için elin tabana göre pozisyonunu ve yönünü bilmek hayati önem taşır, böylece el parçayı kavramak için doğru bir şekilde yönlendirilir.

Bu durumda, ikinci koordinat sisteminin P'nin birinci koordinat sistemine H göre konumunu ve yönünü açıklamak için bir dönüşüm matrisi kullanılabilir.

Bir dönüşüm matrisi düşünün,

$$\mathbf{T}=\left[\begin{array}{cccc} 0 & -1 & 0 & -4 \\ 1 & 0 & 0 & 5 \\ 0 & 0 & 1 & 2 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

Bu dönüşüm matrisi, birinci koordinat sistemine H göre ikinci bir koordinat sisteminin P yönünü ve konumunu tanımlamak için kullanılabilir. T matrisi, birinci koordinat sisteminin birim vektörlerinin uç noktalarına ve orijinine uygulanır,
Birim vektörler i=1, j=1 ve k=1'dir,

İlk koordinat sisteminin i, j ve k birim vektörleri ve orijinli dönüşüm matrisi şu şekilde temsil edilir:

$$\left[\begin{array}{cccc} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline 1 & 1 & 1 & 1 \end{array}\right]$$

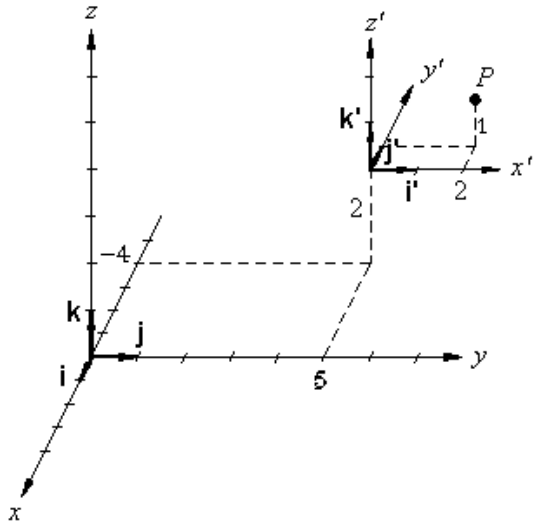
origin, unit vector **i**, unit vector **j**, unit vector **k**

İkinci koordinat sisteminin birim vektörlerinin (i', j', k') başlangıç ve bitiş noktalarını elde etmek için dönüşüm matrisi T'yi birinci koordinat sistemiyle çarpın.

$$\left[\begin{array}{ccc|c} 0 & -1 & 0 & -4 \\ 1 & 0 & 0 & 5 \\ 0 & 0 & 1 & 2 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \left[\begin{array}{c|c|c|c} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline 1 & 1 & 1 & 1 \end{array}\right] = \left[\begin{array}{c|c|c|c} -4 & -4 & -5 & -4 \\ 5 & 6 & 5 & 5 \\ 2 & 2 & 2 & 3 \\ \hline 1 & 1 & 1 & 1 \end{array}\right]$$

| transformation matrix | origin | unit vector **i** | unit vector **j** | unit vector **k** | origin | unit vector **i'** | unit vector **j'** | unit vector **k'** |
|---|---|---|---|---|---|---|---|---|

**T** · {first coordinate system} = {second coordinate system}




Resim, P noktasının ikinci koordinat sisteminde (x' = 2, y' = 1, z' = 1) ve birinci koordinat sisteminde (x = -5, y = 7, z = 3) konumunda olduğunu göstermektedir.

**Robotun Elinin bir parça aldığını düşünelim.**

Parçaya iliştirilen koordinat sistemi, P, koordinat sistemleri dünyasına göre konumlandırılır, buna W diyelim ve dönüşüm matrisi ile gösterilir,

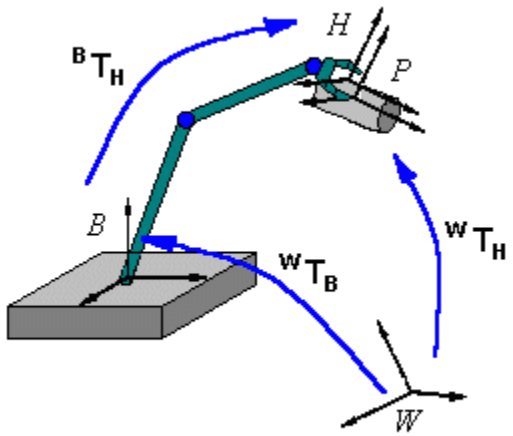
$$^{W}T_{P} = \left[\begin{array}{ccc|c} 0 & 1 & 0 & -1 \\ 0 & 0 & -1 & 2 \\ -1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

Robotun tabanı, B, dünya koordinat sistemlerine göre konumlandırılırken, dönüşüm matrisi ile gösterilir,

$$ {}^{W}T_{B} = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 9 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] $$

Parçayı kavramak için el çerçevesi ve parça çerçevesi hizalanmalıdır ve amaç hizalamayı temsil eden dönüşüm matrisini bulmaktır.

**Dönüşüm matrisini bulalım.**




Parçayı tutarken bir robotun hareketini gösteren şekle bakın. Robotun eli P parçasını kavradığında, bot çerçeveleri hizalanır, yani,

$$ {}^{W}T_{H} = {}^{W}T_{P} $$

Şimdi, W dünyasından taban B'ye dönüşümün ve B tabanından H ele dönüşümün, W dünyasından ele, H'ye tek bir dönüşümde birleştirilebileceğini düşünün. Temsil edilmesi durumunda,

$$ {}^{W}T_{H} = {}^{W}T_{B}\,{}^{B}T_{H} $$

Putting the known matrices,

$$ \left[\begin{array}{ccc|c} 0 & 1 & 0 & -1 \\ 0 & 0 & -1 & 2 \\ -1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 9 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \times {}^{B}T_{H} $$

Tabandan ele dönüşüm için bu matris denkleminin çözülmesi şu şekilde ifade edilir:

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 9 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]^{-1} \times \left[\begin{array}{ccc|c} 0 & 1 & 0 & -1 \\ 0 & 0 & -1 & 2 \\ -1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = {}^{B}\mathbf{T}_{H}$$

Or,

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -5 \\ 0 & 0 & 1 & -9 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \times \left[\begin{array}{ccc|c} 0 & 1 & 0 & -1 \\ 0 & 0 & -1 & 2 \\ -1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = {}^{B}\mathbf{T}_{H}$$

Resulting in the final transformation matrix,

$$\left[\begin{array}{ccc|c} 0 & 1 & 0 & -2 \\ 0 & 0 & -1 & -3 \\ -1 & 0 & 0 & -9 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] = {}^{B}\mathbf{T}_{H}$$

3x3 alt matrisinin üç sütununun, robotun temel koordinat sistemine göre el koordinat sisteminin yönünü temsil ettiğini unutmayın. Son sütun, taban koordinat sisteminin orijini ile ilgili el koordinat sisteminin orijinini gösterirken.

**Bir dizi adi diferansiyel denklemle Robotik Köpeği kontrol etmek için bir çerçeve**
Doğrusal olmayan dinamik sistemler, dörtlü veya bacaklı robotlarda hareketi kontrol etmek için yaratıcı olanaklar sağlar. Doğrusal olmayan dinamik sistem, canlıların sinirsel yapısı ile benzerliği temsil eder. Dinamik bir sistemin kendisi, birbiriyle ve robotun aktüatörleri ve sensörleri ile bağlantılı çoklu sistemlerin bir koleksiyonu olsa da, bu dinamik sistem koleksiyonunu bir beyindeki nöronlar topluluğu olarak düşünebilirsiniz.

Beyin bir duyusal girdiye sahip olduğunda, sinyal kalıpları üretir. Bu yaklaşımı takiben, hayvanların sinir ağı elde edilebilir ve öğrenme yetenekleri ve uyarlanabilir hareket içeren hayvan benzeri robotun (bu durumda Robotik Köpek) davranışını bilmek faydalıdır.

Robotik köpeği kontrol etmek için, bir dizi dinamik sistemle bir çerçeve oluşturalım. Robotik Köpeğin bacaklarını yürümeden ve yere değdirmeden hareket ettirelim.

Bunu yapmak için Robotik köpeği dikdörtgen bir kutuya koyun, böylece bacaklar aşağıdaki resimde gösterildiği gibi yere değmeden serbestçe hareket eder.

Amaç, iki bacağın hareketinin senkronizasyonunu sağlamaktır ve yalnızca bir bacakta insan tarafından yürütülen hareketin frekansı, diğer bacaktaki ACPO tarafından tahrik edilen gerçek frekansa yeterince yakınsa elde edilebilir. Bu senaryo, Robotik Köpeğin bir diferansiyel sistem tarafından kontrol edildiğini gösterir, çünkü bir insan kullanıcı robotik köpeğin salınan bir bacağını kontrol ettiğinde, bir pertürbasyon girdisi verir.

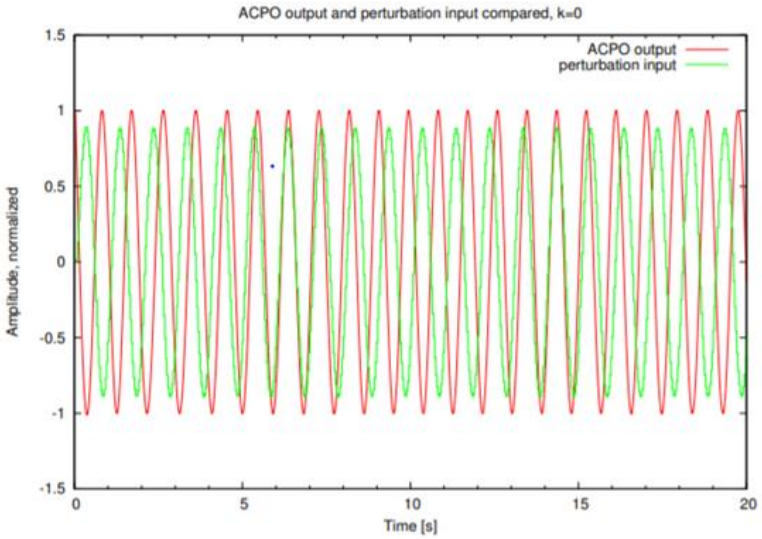
Consider,
x = the position of the left front leg which is controlled by ACPO
$\dot{x}$ = the position of the right front leg
y = the position of the left back leg
$\dot{y}$ = the position of the right back leg
k= the coupling constant

The oscillator is represented as,

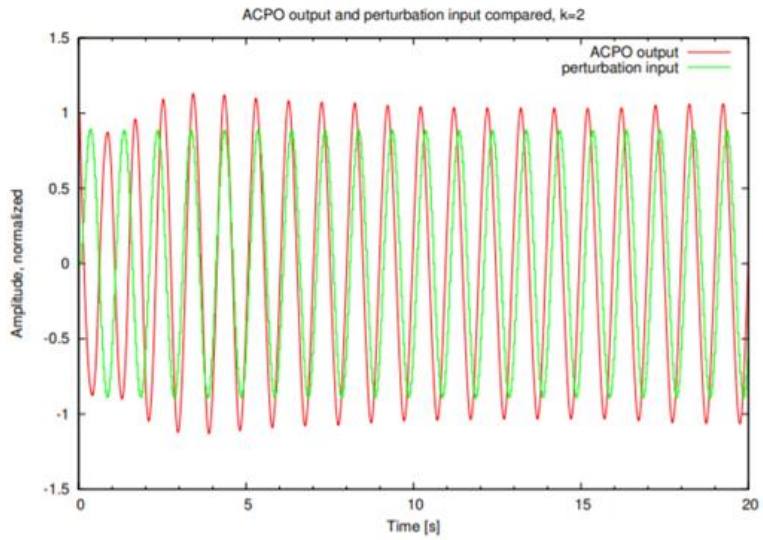
$$\begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} g\left(\frac{r_0}{\sqrt{x^2+y^2}} - 1\right)x - yw \\ g\left(\frac{r_0}{\sqrt{x^2+y^2}} - 1\right)y + xw \end{bmatrix} + k\begin{bmatrix} p \\ 0 \end{bmatrix}$$

Where:
• p is the right fore leg's position input;
• parameters: — g = 10 is the oscillator's gain;
– r0 = 1 is the oscillator's radius;
– w = 2π is the oscillator's intrinsic frequency.

The trails are made with different values of k and the higher the value of k, the stronger the coupling between the legs. At certain value of k, the coupling will be chaotic. Here are the graphs that show synchronization of legs at different values of coupling constant k, In this graph, For k = 0, no synchronization is observed.




For a run with k = 2 the best synchronization after a few seconds is being observed as shown in the graph below,




Matematik, robotik modelleme, planlama ve yürütmede önemli bir rol oynar. Robotik bilim adamları, bu karmaşık denklemlerle nasıl başa çıkacaklarını biliyorlar ve yüzyılın daha gelişmiş ve işlevsel Robotiklerini yaratmak için bir yazılım çerçevesi oluşturuyorlar. Ekonomide en fazla büyümeye sahip olmak için robotikte araştırma ve geliştirmeye yatırım yapmak çok önemlidir, çünkü bundan 10 ila 15 yıl sonra dünyanın tamamen Robotik olacağı ve bu akıllı makineler arasında kendinizin oturduğunu göreceksiniz.

## 12. Mühendislikte sistemlerin matematiksel modellenmesi

Gelişen küresel pazarlar, benzeri görülmemiş teknolojik yetenekler, artan tüketici beklentileri ve sürekli değişen sosyal gereksinimler, genellikle geleneksel mühendislik paradigmasının ötesine geçen zorlu tasarım zorlukları ortaya çıkardıkça, mühendislik sistemlerinin kapsamı ve karmaşıklığı sürekli artmaktadır. Bu zorluklar, mühendislerin ve teknik yöneticilerin teknolojik sistemleri daha büyük bir bütünün parçası olarak ele almalarını gerektirmektedir. Mevcut sistem modelleme çerçevelerinin kapsamı, mühendislik sistemleri hakkındaki bilgileri temsil etme konusunda sınırlıdır. Mühendislik sistemleri için kavramsal bir çerçeve ve geliştirilmiş bir modelleme çerçevesinin sunulması, mühendislere ve yöneticilere bilgiyi görsel olarak düzenlemek ve söylemi daha iyi sistem mühendisliğini kolaylaştıracak şekilde yapılandırmak için geliştirilmiş bir araç sağlamasıdır. Bir mühendislik sistemi için açık ve özlü bir çerçeve sağlayarak mevcut literatürü zenginleştirir ve mühendislerin sistem mühendisliği verilerinin daha iyi toplanmasına, depolanmasına, işlenmesine ve analizine izin verecek şekilde sistem bilgilerini etiketlemesi ve düzenlemesi için bir metodoloji sağlar.