Makine Öğrenmesi

Makine Öğrenmesine Giriş

Yazılım Mühendisliği Bölümü
Atılım Üniversitesi

Tolga Üstünkök

Ön Bilgilendirmeler

- Bazı İngilizce terimler oldukları gibi bırakılmışlardır. Bu terimleri slaytlar içerisinde *italic* yazılmış olarak görebilirsiniz.
- Makine öğrenmesi yöntemlerinin ve değerlendirme ölçütlerinin isimleri Türkçe'ye çevrilmeyip oldukları gibi bırakılmışlardır.

Ön Hazırlıklar

- Makine öğrenmesi[*Machine Learning*] (ML) dediğimiz zaman aslında basit nümerik yaklaşım yöntemlerinden tutun, yapay sinir ağlarına hatta bazı optimizasyon yöntemlerine kadar çok fazla alt konuyu kastederiz.
- Ancak makine öğrenmesinin detaylarına girmeden önce veri ile uğraşmak gerekir. Sonuçta veri öğrenmenin temelini oluşturmaktadır.
- Eğer veri çöpse, kullandığımız ML yöntemi de çöp üretir.

Agenda

- Keşifsel Veri Analizi[*Exploratory Data Analysis (EDA)*]
- Makine Öğrenmesi Yöntemleri
- Makine Öğrenmesi Problemleri
- Linear Regression
- Naïve Bayes
- Decision Tree
- Random Forest
- Artificial Neural Networks
- Değerlendirme Ölçütleri

Keşifsel Veri Analizi

- Makine öğrenmesiyle uğraşmadan önce veri üzerinde hangi aracın en iyi sonuçlar üreteceğini bilmek gerekir.
- Bunun için de veriyi anlamak önemlidir. Veriyi anlama işine Keşifsel Veri Analizi[*Exploratory Data Analysis (EDA)*] denir.

Keşifsel Veri Analizi

- EDA kullanmamızdaki bazı ana sebepler şu şekilde sıralanmıştır:
  - Verideki hataların tespit edilmesi
  - Veri hakkındaki varsayımların doğrulanması
  - Veri üzerinde kullanılacak metodların ve araçların seçimi
  - Değişkenler arasındaki ilişkilerin tespiti
  - Değişkenler arasındaki ilişkilerin yönünün ve büyüklüğünün tespiti

Keşifsel Veri Analizi

- Genelde veri yanda gördüğünüz şekilde yapılandırılmış olarak karşınıza çıkar.
- Burada her bir sütuna değişken[*variable*], her bir satıra da gözlem[*observation*] denir.

Değişken

Gözlem

outlook	temperature	humidity
sunny	hot	high
sunny	hot	high
overcast	hot	high
rainy	mild	high
rainy	cool	normal
rainy	cool	normal

Keşifsel Veri Analizi

- Eğer veriye bakıp kendinize “Neyi tahmin etmeye çalışıyorum?” sorusunu sorarsanız, bu sorunun cevabı size **bağımlı değişkenin[*dependent variable*]** ismini verir. Diğer değişkenlere de **bağımsız değişken[*independent variable*]** denir.

Bağımsız değişkenler

Bağımlı değişken

outlook	temperature	humidity	windy	play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

Keşifsel Veri Analizi

- *Data munging*, veriye gerekli işlemleri yaparak onu transform etme ve düzeltme görevlerine verilen genel addır.
- Bu görevler:
  - Değişkenlerin isimlerini değiştirmek
  - Veri tiplerini değiştirmek
  - Kodlamak[*encoding*], çözümlemek[*decoding*]
  - Veri kümelerini birleştirmek
  - Veriyi transform etmek
  - Kayıp verileri işlemek[*imputation*]
  - Aykırı verileri işlemek[*outlier detection*]

Keşifsel Veri Analizi

- Az önce bahsettiğimiz görevler sürekli olarak tekrar edilmek üzere tasarlanmışlardır.
- Bu görevlerin yerine getirilmesi bütün makine öğrenmesi görevlerinin en zaman alıcı ve zorlu kısmını oluşturur.
- Bu görevler yerine getirilirken verinin iki ana gruba ayrıldığı düşünülür. Bunlar:
  - Kategorik[*Categorical, Qualitative*]
  - Nümerik[*Numerical, Quantitative, Continuous*]

Keşifsel Veri Analizi

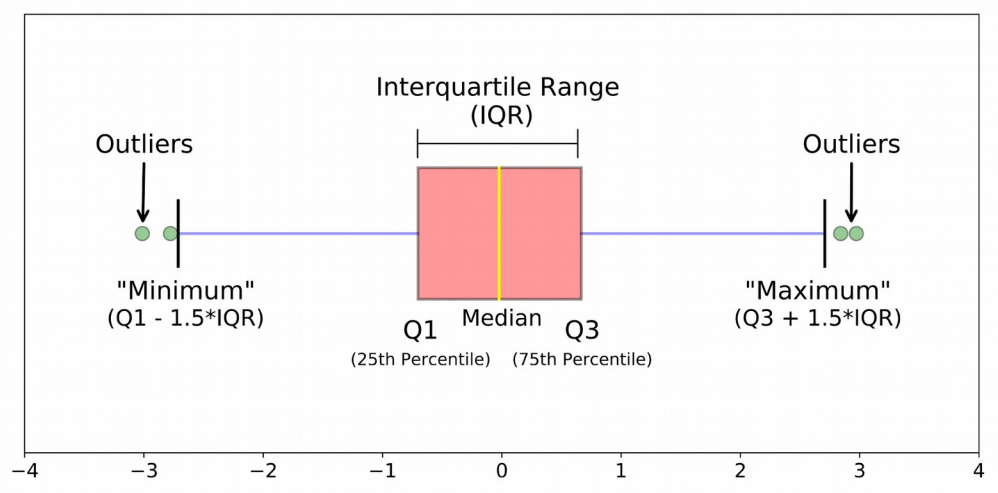
- Az önce bahsettiğimiz veri tipleri kendi alt gruplarına da ayrılırlar:
  - Kategorik veri
    - Nominal: ayrık birimler, sıralama önemsiz (i.e. male/female)
    - Ordinal: ayrık birimler, sıralama önemli (i.e. müşteri sınıfları)
  - Nümerik/Sürekli veri
    - Interval: mutlak 0'ı olmayan veriler (i.e. sıcaklık)
    - Ratio: mutalak 0'ı olan veriler (i.e. uzunluk ölçüleri)

Keşifsel Veri Analizi

- En basit seviyede genellikle iki tip tanımlayıcı istatistik kullanılır:
  - Merkezi eğilim[*central tendency*]
    - Ortalama[*mean*]
    - Medyan[*median*] (Q2)
    - Mod[*mode*]
  - Değişebilirlik
    - Yayılma aralığı[*range*] (max – min)
    - Standart sapma[*standard deviation*] (or variance)
    - Çeyrekler açıklığı[*inter-quartile range*] (Q3 - Q1)

Keşifsel Veri Analizi

- Bazen bütün bu değerleri sayısal olarak görmektense bir şekil herşeyi açıklayabilir.
- Yanda gördüğünüz grafiğe *“Boxplot”* denir.
- Az önce bahsettiğimiz bütün terimleri içinde bulunduran bir grafik türüdür ve çok sık kullanılır.



Keşifsel Veri Analizi

- EDA hakkında daha anlatılacak çok şey olmasına rağmen şimdilik bu kadarı yeterli olacaktır.
- Ancak bazı durumlarda bu konulara ekler yapmamız gerekcek.
- Yeri geldiğinde bu ekleri yapacağız.

Makine Öğrenmesi Yöntemleri

- Makine öğrenmesi yöntemleri doğrusal ve doğrusal olmayan olarak ikiye ayrılır.
- Doğrusal yöntemler, bağımlı ile bağımsız değişkenler arasındaki doğrusal ilişkiyi modellemeye çalışırlar.
- Doğrusal olmayan yöntemleri ise bağımlı ve bağımsız değişkenler arasında doğrusal olmayan bir ilişki olduğunu varsayarlar. Dolayısıyla doğrusal olmayan bir ilişki modellerler.

Makine Öğrenmesi Yöntemleri

- Burada aklınızda “O zaman doğrusal problemler için yalnızca doğrusal, doğrusal olmayan problemler için de yalnızca doğrusal olmayan yöntemleri kullanmak zorundayım.” şeklinde bir fikir oluşmuş olabilir.
- Bu fikir aslında yanlıştır. Her iki tip problem için de her iki tip yöntemi değişken olarak kullanabilirsiniz.
- Ancak doğrusal olmayan bir probleme doğrusal bir yöntem uygularsanız hata oranınız oldukça yüksek olacaktır.

Makine Öğrenmesi Problemleri

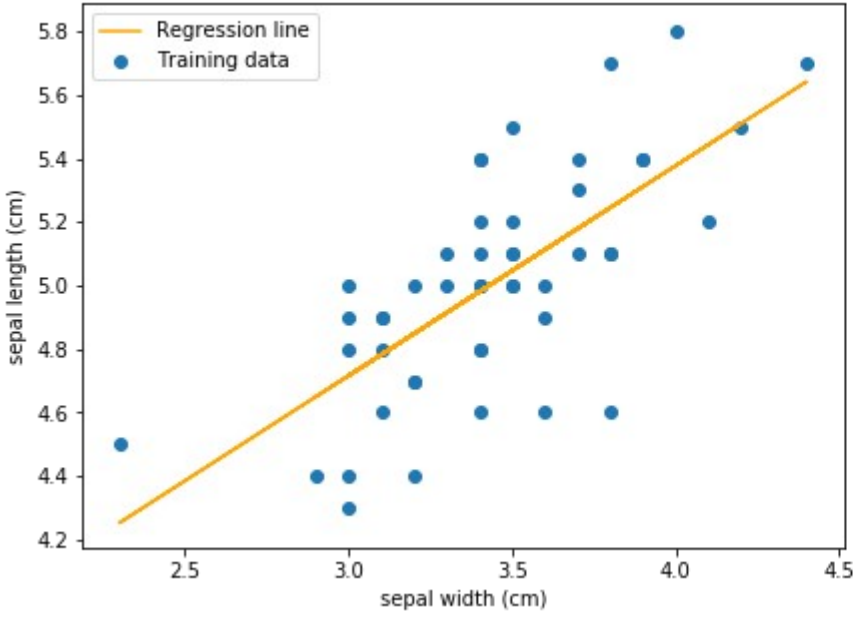
- Makine öğrenmesi problemleri sınıflandırma ve regresyon olarak ikiye ayrılırlar.
- Sınıflandırmaya örnek olarak:
  - Email: Spam / Spam Değil?
  - Tümör: Malignant / Benign?
- Regresyon ise sınıflandırma gibi ayrık değerlerden değil reel sayılardan oluşan tahminlere denir.

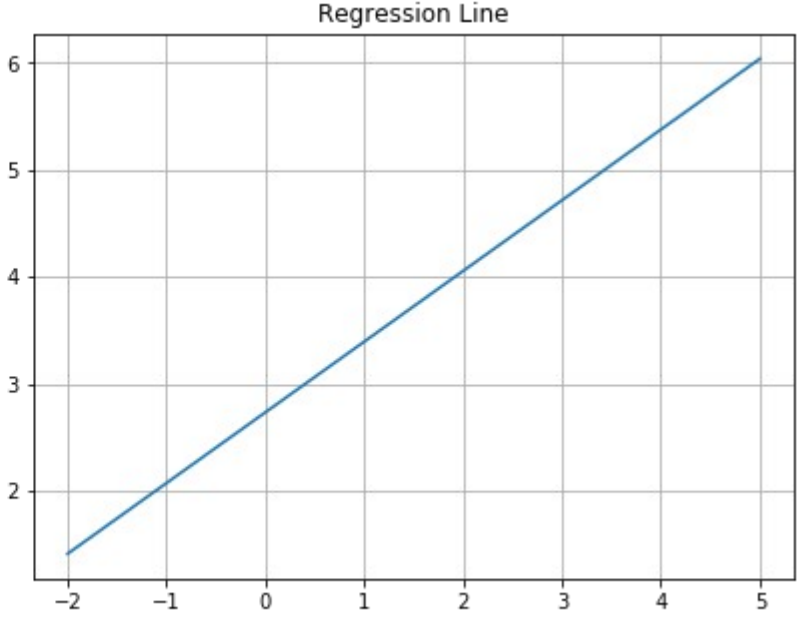
Simple Linear Regression

- Belki de en basit makine öğrenmesi yöntemlerinden biri *Simple Linear Regression (SLR)*'dır.
- SLR veri setindeki bağımsız değişken ile bağımlı değişken arasında doğrusal ilişkiyi modellemeye çalışır.
- Eğer birden fazla bağımsız değişken varsa modellenen doğru artık bir doğrudan ziyade bir hiper-düzlem[*hyperplane*] haline gelir. Bu durumda kullanılan modelin adı artık *Multiple Linear Regression (MLR)*'dır.

Simple Linear Regression

- Açıklamaya destek olması için aşağıdaki örneği inceleyelim:





Linear Regression

- *Linear Regression (LR)* adından da anlaşılacağı gibi birkaç limit vardır.
- Birincisi, yalnızca doğrusal[*linear*] modelleme yapabilir. Doğrusal olmayan problemler için iyi bir seçenek sunmaz. Ancak yine de kullanabilirsiniz (yüksek hata riskini göze alıyorsanız).
- İkincisi, yalnızca regresyon yapabilir. Yani bağımlı değişkeniniz kategorik ise, LR iyi bir çözüm olmaz.

Linear Regression

- Python için yazılmış *scikit-learn* paketinde MLR için bir modül belirlenmiştir.
- Dolayısıyla hesaplama detaylarına girmeden MLR'ı rahatlıkla kullanabilirsiniz. Örnek:

```
>>> import numpy as np
>>> from sklearn.linear_model import LinearRegression
>>> X = np.array([[1,1],[1,2],[2,2],[2,3]])
>>> # y = 1 * X_0 + 2 * x_1 + 3
>>> y = np.dot(X, np.array([1,2])) + 3
>>> reg = LinearRegression().fit(X, y)
>>> reg.coef_
array([1., 2.])
>>> reg.intercept_
3.0000...
```

Naïve Bayes Sınıflandırıcısı

- Naïve Bayes istatistik tabanlı bir sınıflandırıcıdır.
- Naïve Bayes adını 1812 yılında Thomas Bayes tarafından önerilen Bayes Teoremi'nden alır.
- Bayes Teoremi, bir olayın gerçekleşme olasılığını, o olayla ilişkisi bulunan harici olasılıkların yardımıyla hesaplar.
- Bayes Teoremi şu şekilde ifade edilmiştir:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Naïve Bayes Sınıflandırıcısı

- A ve B birer olaysa, P(A) ve P(B) bu olayların birbirlerinden bağımsız olarak meydana gelme olasılıklarıdır.
- P(A|B)'ye şartlı olasılık denir. B olayı meydana geldiği zaman A olayının meydana gelme olasılığını belirtir.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Naïve Bayes Sınıflandırıcısı

- Naïve Bayes'e naïve yani saf denmesinin sebebi, bağımsız değişkenlerin değerlerinin hedef sınıfa olan etkisinin bağımsız olduğunu varsaymasından kaynaklanmaktadır.
- Bu tip bir durum doğada nadiren bulunur. Dolayısıyla pek de gerçekçi değildir.

Naïve Bayes Sınıflandırıcısı

- Yine MLR'da olduğu gibi *scikit-learn* paketi içerisinde Naïve Bayes için tanımlamış birkaç modül bulundurur.
- Bunların içinde en sık kullanılanı Gaussian Naïve Bayes'tir.

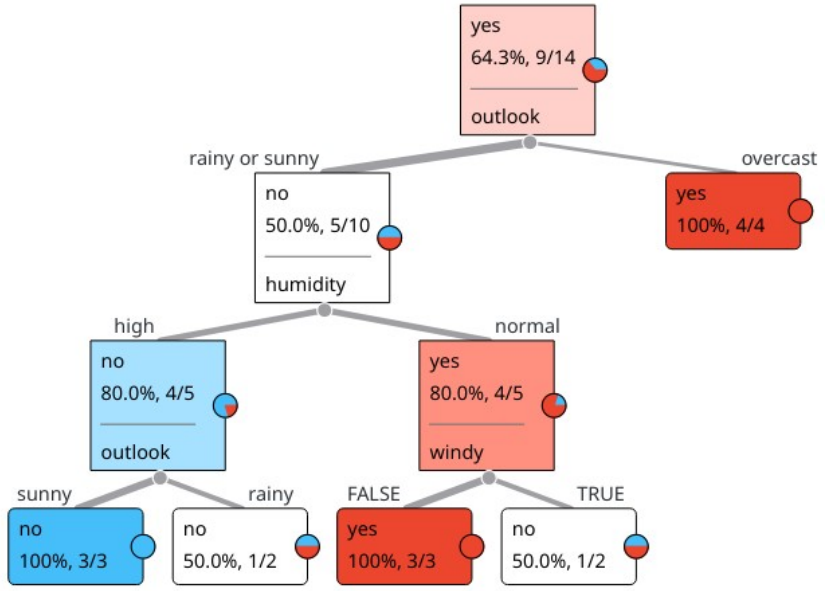
```
>>> from sklearn import datasets
>>> iris = datasets.load_iris()
>>> from sklearn.naive_bayes import GaussianNB
>>> gnb = GaussianNB()
>>> y_pred = gnb.fit(iris.data, iris.target).predict(iris.data)
>>> print("Number of mislabeled points out of a total %d points : %d"
...       % (iris.data.shape[0],(iris.target != y_pred).sum()))
Number of mislabeled points out of a total 150 points : 6
```

Decision Tree[Karar Ağacı]

- Decision Tree, kendisine verilen bağımsız değişkenlerden ürettiği kuralları kullanarak bağımlı değişkenleri tahmin eder.
- Hem sınıflandırma hem de regresyon için kullanılabilir.
- Decision Tree'ler Bilgi Teorisi'yle[*Information Theory*] çok yakından ilişkilidir.
- Dolayısyıyla çalışma şeklinin anlaşılması için *Entropy* fikrinin çok iyi oturmuş olması gereklidir.

Decision Tree[Karar Ağacı]

- Decision Tree'leri görselleştirmek oldukça kolaydır.
- Hatta bunu otomatik olarak yapan yazılımlar mevcuttur. (bkz. Orange3)
- Aşağıda (solda) sağdaki veri için Orange3 tarafından çizdirilmiş bir Decision Tree görmektesiniz.



outlook	temperature	humidity	windy	play
sunny	hot	high	FALSE	no
sunny	hot	high	TRUE	no
overcast	hot	high	FALSE	yes
rainy	mild	high	FALSE	yes
rainy	cool	normal	FALSE	yes
rainy	cool	normal	TRUE	no
overcast	cool	normal	TRUE	yes
sunny	mild	high	FALSE	no
sunny	cool	normal	FALSE	yes
rainy	mild	normal	FALSE	yes
sunny	mild	normal	TRUE	yes
overcast	mild	high	TRUE	yes
overcast	hot	normal	FALSE	yes
rainy	mild	high	TRUE	no

Decision Tree[Karar Ağacı]

- Yine *scikit-learn* paketi bizim için bir Decision Tree modülü tasarlamıştır.

```
>>> from sklearn import tree
>>> X = [[0, 0], [1, 1]]
>>> Y = [0, 1]
>>> clf = tree.DecisionTreeClassifier()
>>> clf = clf.fit(X, Y)
>>> clf.predict([[2., 2.]])
array([1])
```

Random Forest

- Random Forest birden fazla Decision Tree’nin biraraya getirilip sonuçlarının oylanarak veya aritmetik ortalamasının alınarak tek bir sonuca indirgendiği yöntemin ismidir.
- Görselleştirmesi zorlu olabilir. Ancak çok güçlü bir yöntemdir.
- Veriyi ezberlemekten kolayca kaçınabilir.

Random Forest

- Diğer yöntemlerde olduğu gibi *scikit-learn* paketi içerisinde Random Forest'ı kullanabileceğiniz bir modül mevcuttur.

```
>>> from sklearn.ensemble import RandomForestClassifier
>>> X = [[0, 0], [1, 1]]
>>> Y = [0, 1]
>>> clf = RandomForestClassifier(n_estimators=10)
>>> clf = clf.fit(X, Y)
```

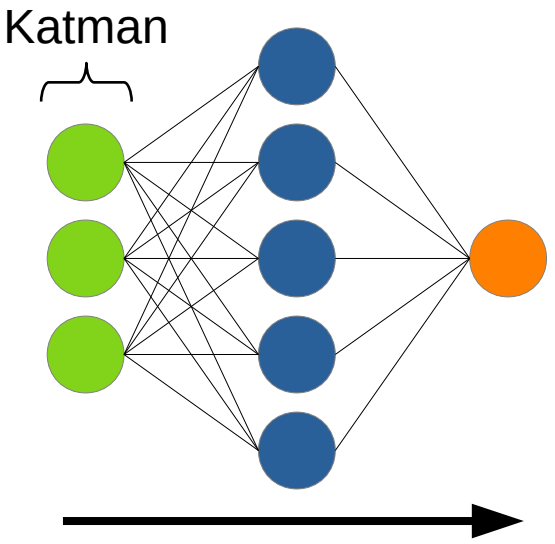
- Random Forest gördüğünüz üzere *ensemble* modülü içerisinde bulunmaktadır. Ensemble birden fazla metodun özel bir şekilde bir araya getirildiği metodlara verilen bir isimdir.

Yapay Sinir Ağları

- Belki de makine öğrenmesi yöntemleri arasında en popülerlerden biri Yapay Sinir Ağları[*Artificial Neural Networks (ANN)*]'dır.
- ANN'lerin bu kadar popüler olmasının sebeblerinden bir kaçı şu şekilde sıralanabilir:
  - Rahatlıkla başkalaşım geçirerek çok değişik mimarilerde biraraya getirilebilirler.
  - Hesaplamaları matrix çarpımlarından ibaret oldukları için bilgisayarlar (özellikle ekran kartları) tarafından çok hızlı yapılır.
  - Evrensel fonksiyon yaklaşımcıları[*universal function approximator*] oldukları için vereceğiniz her tür fonksiyonu modelleyebilirler.

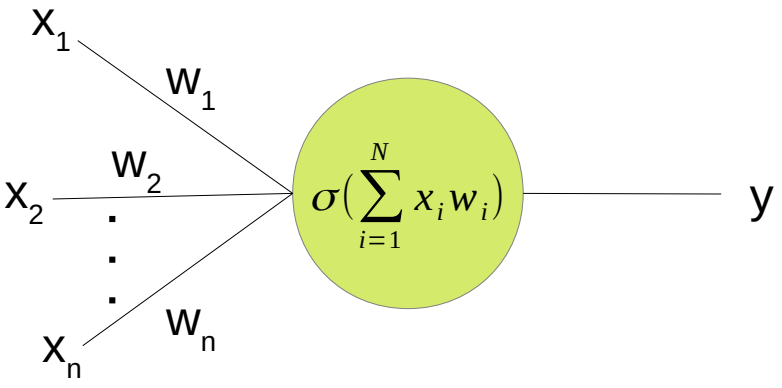
Yapay Sinir Ağları

- Bir ANN birçok katman haline getirilmiş düğümden ($N_{i,j}$) oluşur.
- Bütün bu düğümler şekilde gösterildiği gibi birbirine bağlıdır.
- Şekilde gösterilen her bir sütuna katman denir.
- İlk katman girdi katmanı[*input layer*] (yeşil), son katman ise çıktı katmanı[*output layer*] (turuncu) olarak adlandırılır.
- Ortadaki mavi katmana ise saklı katman[*hidden layer*] denir.
- Bir ANN'de veri yalnızca ok yönünde akar. Yani bu graph aslında bir *directed graph*'tır.
- Şimdi bir düğüme bir de yakından bakalım.



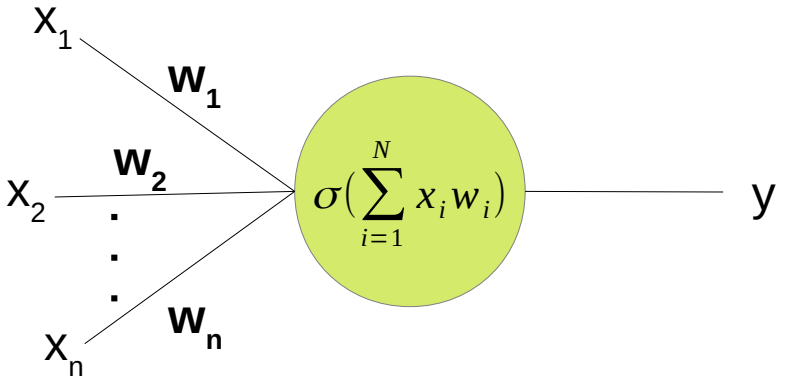
Yapay Sinir Ağları

- Yanda gördüğünüz şema yalnızca birtek düğüm/nöron için çizilmiştir.
- Bir nöron kendisine verilen bütün girdilerle (x_i), bu girdilerin geçtikleri (yolları (w_i) / ağırlıkları) çarpıp toplayarak doğrusal olmayan bir fonksiyona (σ) verir.
- İşte bu foksiyonun çıktısı nöronun da çıktısıdır.



Yapay Sinir Ağları

- Peki soru şu: “Öğrenme işlemi nerede yapılmakta?”
- Girdi zaten veriden geldiği için onu değiştiremeyiz. Foksiyon da zaten sabit ve daha önceden belirlenmiş bir fonksiyon.
- Dolayısıyla değiştirebileceğimiz geriye kalan değişkenler ağırlıklar yani w’lardır.
- Öğrenme işlemi doğru w değerlerini bulmaktan geçer.



Yapay Sinir Ağları

- Peki değerin “doğru” olduğunu nasıl anlarız?
- Denetimli öğrenme[*supervised learning*] yapılırken modele bağımlı değişkenlerle beraber bağımsız değişkenleri de veririz.
- Sonra modelden bağımsız değişkenleri verdiğimiz bir gözlem için bağımlı değişkeni tahmin etmesini isteriz.
- Modelin cevabının gerçek bağımlı değerden uzaklığı bize tahminin ne kadar doğru olduğunu söyler.

Yapay Sinir Ağları

- Bu uzaklık birçok farklı şekilde hesaplanabilir.
- O zaman w değerlerini pozitif ya da negatif yönde değiştirdiğimizde bu uzaklık azalıyor mu artıyor mu ona bakmamız gerekir.
- Bu işlemi bütün w değerleri için tekrarlayabiliriz.
- Ama sorun şu ki 1 000 000 tane w varsa o zaman ne olacak?

Yapay Sinir Ağları

- İşte bu oldukça çetin bir optimizasyon problemidir.
- Bu durumu çözmek için onlarca yöntem önerilmiştir.
- Açık ara en çok kullanılanları şu şekilde sıralayabiliriz:
  - Stochastic Gradient Descent (SGD)
  - RMSProp (Geoffrey Hinton)
  - Adam (Kingma, D. and Ba, J.)

Yapay Sinir Ağları

- Bu üç yöntemin dışında doğadan ilham almış bazı optimizasyon algoritmaları da vardır. Bunlar meta-heuristic'ler olarak bilinirler.
- Bu yöntemler yine ANN'leri eğitmek için kullanılabilirler ancak diğer üç yöntem kadar popüler olmamışlardır. Bazıları şu şekilde sıralanabilir:
  - Genetic Algorithms (GA)
  - Artificial Bee Colony Optimization (ABC)
  - Ant Colony Optimization (ACO)
  - Particle Swarm Optimization (PSO)

Değerlendirme Ölçütleri

- Şimdi az önce bahsettiğimiz tahminin gerçek değerden ne kadar uzak olduğunu ölçmenin yöntemlerine bir göz atalım.
- Bu uzaklığı problemin sınıflandırma mı yoksa regresyon mu olduğuna bakarak yapmamız gereklidir.
- Dolayısıyla değerlendirme ölçütlerini ikiye ayrılmış şekilde incelememiz gerekir.

Değerlendirme Ölçütleri: Regresyon

- Bir hatayı ya da doğru değerden olan uzaklığı ölçmenin birçok yolu vardır.
- Ama açık ara en çok kullanılanlardan biri *Mean Square Error (MSE)*'dir.
- MSE hatanın yönüne bakmaksızın karesini alarak biriktirir (toplar).
- Bu durum küçük hataların bile önemli olmasını sağlar.

Değerlendirme Ölçütleri: Regresyon

- MSE şu şekilde tanımlanabilir:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \hat{y}_i\right)^2$$

burada N tahmin edilen gözlem sayısını, y_i gerçek değeri, $\hat{y}_i$ de tahmin edilen değeri belirtmektedir.

- Kare alma işleminden dolayı sonuç hiçbir zaman negatif olamaz.

Değerlendirme Ölçütleri: Regresyon

- MSE'nin bir türevi olan *Root Mean Square Error (RMSE)* de MSE ile aynı özelliklere sahiptir.
- Tek farkı MSE değeri bulunduktan sonra karekökü alınmıştır.
- İstatistiksel olarak bakarsanız MSE varyansa bezerken, RMSE standart sapmaya benzer.

Değerlendirme Ölçütleri: Regresyon

- MSE’nin en önemli özelliği hatanın boyutunun gerçek verinin boyutundan farklılık göstermesidir (kare alma işleminden dolayı).
- Bu noktada *Mean Absolute Error (MAE)* kare alma işlemini yok eder ancak hala sonucu hiçbir zaman negatif vermez (mutlak değerden dolayı).

Değerlendirme Ölçütleri: Regresyon

- MAE şu şekilde tanımlanabilir:

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i|$$

  bu durumda *N* tahmin edilen gözlem sayısı, y_i tahmin edilen gözlemin gerçek değeri, $\hat{y}_i$ de tahmin edilen değeri belirtmektedir.

- Bu hata tipi daha çok finansal problemlerle uğraşılırken kullanılır.

Değerlendirme Ölçütleri: Sınıflandırma

- Şimdi biraz da sınıflandırma problemleri için tanımlanmış hata tiplerine bakalım.
- Sınıflandırma problemlerine tahmin edilen değer bir reel sayıdan ziyade genelde 1/0, var/yok, spam/spam değil şeklinde olduğu için daha farklı tipte hata ölçüm teknikleri geliştirilimiştir.
- Bu teknikler genel olarak istatistik tabanlıdır.

Değerlendirme Ölçütleri: Sınıflandırma

- Ancak bu ölçütlere geçmeden önce *“Confusion Matrix”* denen bir matrixten bahsetmemiz gerekmektedir.
- Bir örnek üzerinden *Confusion Matrix*’i tartışalım.
- Varsayalım kanser olduğundan şüphelenilen hücrelerin değişik özelliklerine bakarak hücrenin kanser tipinin *malignant* (kötü huylu) ya da *benign* (iyi huylu) olduğunu söyleyen bir model ile çalışıyoruz.

Değerlendirme Ölçütleri: Sınıflandırma

- Bu modelin bize verdiği cevapları aşağıdaki gibi bir tabloda toplayabiliriz.

		Predicted		
		Malignant	Not Malignant (Benign)	
Actual	Malignant	47	2	49
	Not Malignant (Benign)	5	85	90
		52	87	139

Değerlendirme Ölçütleri: Sınıflandırma

- Bir hücreyi malignant olarak tahmin etmeye pozitif bir tahmin malignant değil olarak tahmin etmeye de negatif tahmin diyelim.
- Malignanat'ı malignant, malignant değili de malignant değil olarak tahmin etmeye de *true(doğru)* tam tersine de *false* tahmin diyelim.

Değerlendirme Ölçütleri: Sınıflandırma

- O zaman yandaki tablonun kırmızı ile işaretli kısmına True Pozitif (TP) diyebiliriz.
- Mavi ile işaretli kısma da True Negatif (TN) diyebiliriz.

		Predicted		
		Malignant	Not Malignant (Benign)	
Actual	Malignant	47	2	49
	Not Malignant (Benign)	5	85	90
		52	87	139

Değerlendirme Ölçütleri: Sınıflandırma

- Sarı ile işaretli kısma False Pozitif (FP) diyebiliriz.
- Yeşil ile işaretli kısma da False Negatif (FN) diyebiliriz.

		Predicted: Malignant	Predicted: Not Malignant (Benign)	
Actual	Malignant	47	2	49
	Not Malignant (Benign)	5	85	90
		52	87	139

Değerlendirme Ölçütleri: Sınıflandırma

- Şimdi bu tanımları kullanarak hata tiplerini konuşalım.
- Hata tiplerinin bakacağımız ilki *accuracy*.
- Accuracy bütün tahminler arasından ne kadarının doğru (True) tahmin edildiğinin bir ölçüsüdür.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

- Accuracy genellikle veri dağılımının eşit olduğu durumlarda kullanışlıdır. Aksi takdirde accuracy'den alacağınız sonuç sizi yanıltabilir.

Değerlendirme Ölçütleri: Sınıflandırma

- İkinci olarak bakacağımız ölçüt *precision*.
- Precision bize malignant olarak tahmin ettiğimiz gözlemler arasında gerçekten ne kadarının malignant olduğunu söyler.

$$Precision = \frac{TP}{TP + FP}$$

- Bu ölçütü bağımlı değişkendeki bütün farklı değerler için yeniden hesaplayabilirsiniz.

Değerlendirme Ölçütleri: Sınıflandırma

- Üçüncü olarak *recall* ölçütüne bakacağız.
- Bu ölçüt bize gerçekte malignant olan gözlemlerin ne kadarını malignant olarak tahmin ettiğimizin oranını verir.

$$Recall = \frac{TP}{TP + FN}$$

- Precision'da olduğu gibi recall da bağımsız değişkendeki farklı değerler için yeniden hesaplanabilir.

Değerlendirme Ölçütleri: Sınıflandırma

- Son olarak bakacağımız ölçütün adı F1-score.
- F1-score aslında precision ve recall'un harmonik ortalamasıdır.

$$F1\,Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

- Bu durum değerlerden hangisi sayısal olarak diğerinden küçükse F1-Score'un ona daha yakın çıkacağı anlamına gelmektedir.

Değerlendirme Ölçütleri

- Burada bahsettiğimiz değerlendirme ölçütleri (ve daha fazlası) bir makine öğrenmesi görevinin hemen her aşamasında kullanılır.
- Bu aşamaları da üçe ayırabiliriz. Bunlar:
  - Training [Eğitim]
  - Validation [Doğrulama]
  - Testing

Değerlendirme Ölçütleri

- Eğitim[*Training*] ve test için veriler belli oranlarda birbirinden ayrılır. Bunun sebebi eğitim için kullanılan verinin test için kullanılmasını önlemektir.
- Test verisini eğitim bittikten sonra kalitesini ölçmek için kullanırız.
- Validation ise eğitim yapılırken modelin hiç görmediği bir veriyi tahmin etmeye çalıştığında ne durumda olduğunu görmek için kullanırız.
- Validation en çok eğitim sırasında modelin *overfit* olup olmadığını anlamak için faydalıdır.

Particle Swarm Optmization

- *Particle Swarm Optimization (PSO),* 1995 yılında Eberhart ve Kennedy tarafından önerilmiş bir optimizasyon tekniğidir.
- PSO'da amaç kuş ve balık sürülerinin yaptığı bazı davranışları taklit etmektir.
- Bu nedenden bu tarz algoritmalara doğadan ilham almış algoritmalar [*nature inspired algorithms*] deriz. Başka bir isim de meta-heuristic'tir.

Particle Swarm Optmization

- PSO genellikle optimize edilmesi zor (NP-hard) problemler için bazı durumlarda iyi bir çözüm olabilir.
- Ama hemen PSO kullanmadan önce diğer bazı meta-heuristic yöntemlerin de kontrol edilmesi gereklidir. Örneğin:
  - Genetic Algorithms (GA)
  - Artificial Bee Colony Optimization (ABC)
  - Ant Colony Optimization
  - Simulated Annealing
  - Tabu Search
  - ...

Particle Swarm Optmization

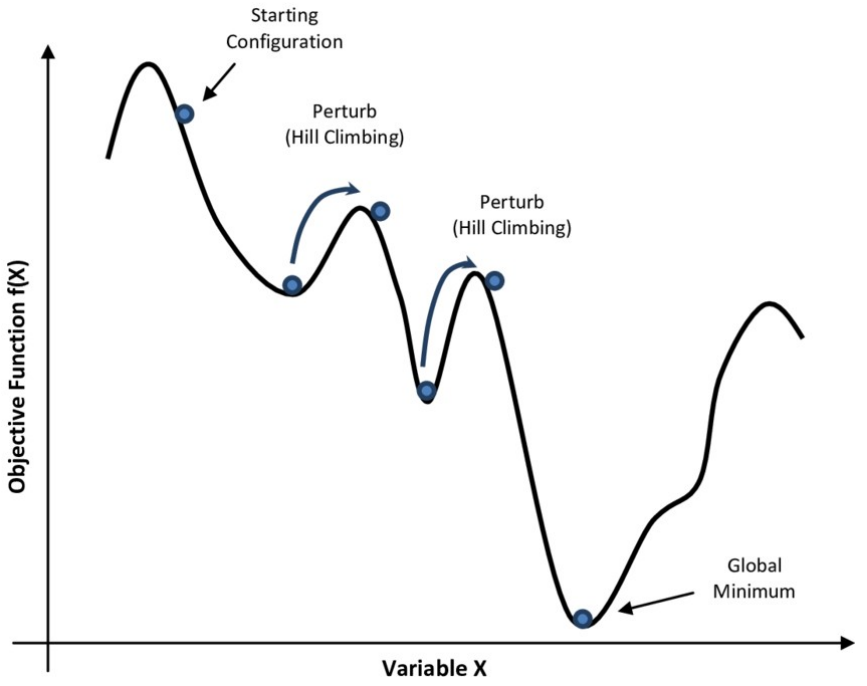
- PSO adından da anlaşılabilineceği gibi kuş ve balık sürülerini taklit etmek için pariküllerden yararlanır.
- Ancak algoritmanın nasıl çalıştığına geçmeden önce bazı terimleri öğrenelim.
- Üzerinde arama yaptığımız n boyutlu olası bütün çözümleri barındıran sayı kümesine ***search space*** denir.
- Bulunan ve bütün kısıtlamalara uyan bir sonuca ***feasible solution*** denir.
- Problemin yaklaşım olmaksızın tam doğru sonucuna ***exact solution*** denir.

Particle Swarm Optmization

- Problemin *search space*'indeki en derin olan vadi hariç bütün vadilere *local minimum* (çoğul. *Local minima*), en derin olana da global minimum denir.
- Aynı şekilde en yükse olan hariç bütün tepelere local maximum (çoğul. Local maxima), en derin olana da global maximum denir.
- Düzlüklere *plateau* denir.
- Şimdi bir örnekle bunlara bir göz atalım.

Particle Swarm Optmization

- Yanda gördüğünüz grafik az önce konuştuğumuz terimlerin çoğunu içermektedir.
- Grafikte bir tek *plateau* gözükmemektedir.



Particle Swarm Optmization

- Şimdi PSO’nun algoritmasının nasıl çalıştığına bir göz atalım.
- PSO’nun *initialization* adımı çok kolaydır ve aşağıdaki alt adımlardan oluşur:
  - *Search space,* sayısı daha önceden belirlenmiş partikülleri rastgele yerleştirerek doldur.
  - Her partiküle rastgele bir hız vektörü atanır.
- *Initialization*’dan sonra PSO ana iterasyon döngüsüne başlar.

Particle Swarm Optmization

- İterasyonun kaç kere yapılacağı bir *hyperparameter*'dır.
- İkinci adımda her partikülün o an bulunduğu yerdeki *fitness*'ları hesaplanır.
  - Fitness bulunan çözümün ne kadar kaliteli olduğunun bir ölçüsüdür.
- Ardından partiküllerin şu anki fitness'ları daha önceki fitness'larından daha iyiyse, daha önceki fitnesslarıyla değiştirilir ve şu anki pozisyonları kaydedilir.
- Daha sonra *fitness*'ı en iyi olan partikül bulunur ve bütün *swarm*'ın en iyisi olarak kaydedilir.

Particle Swarm Optmization

- Sonraki adım bütün partiküllere daha önceki hız değerleri de kullanarak yeni bir hız değeri atamaktır.
- Bu işlem aşağıdaki şekilde yapılır.

$$v_{id}(t+1) = w * v_{id}(t) +$$
$$c_1 * rand() * (pBest_{id} - x_{id}(t)) +$$
$$c_2 * rand() * (gBest_{id} - x_{id}(t))$$

Particle Swarm Optmization

- Daha sonra elde edilen bu yeni *v* değerleri kullanılarak her partikülün pozisyon vektörü güncellenir.
- Ardından da ikinci adıma tekrar dönülür ve bütün bu işlemler belli bir *stop condition*'ına uyana kadar tekrarlanır.
- Algoritmanın bu kadar basit olması implement edilmesini çok kolay bir hale getirir.
- Şimdi bir örneğine bakalım.

Epilog

- Burada gördüğümüz yöntemler birçok makine öğrenmesi yönteminin çok küçük bir alt kümesidir.
- Bu yöntemlerin eğitildikten sonra kalitesinin ölçülmesi için de daha önce bahsettiğimiz değerlendirme ölçütleri kullanılır.
- Bir sonraki konumuz olan Deep Learning[*Derin Öğrenme*] burada gördüğümüz çoğu konu ile yakından ilişkilidir.