[1]

VERİ MADENCİLİĞİ

0) INTRO (GİRİŞ)

- Predictive ⇒ Geçmişe ya da geleceğe yönelik tahminsel (öngörüsel) } ANA YÖNTEMLER
  - Classification, supervised learning, regression ⇒ örnekler
- Descriptive ⇒ Veriden tanımlayıcı, yorumlayıcı
  - Clustering (unsupervised), association rules ⇒ örnekler
- Data → data preprocessing → data mining → post processing → information } süreç
  - data preprocessing ⇒ feature selection, dimensional reduction, normalization, data subsetting
  - post processing ⇒ visualization, filtering patterns, pattern interpretation

1) DATA PREPROCESSING (ÖN İŞLEME)

Yapılma sebebi ⇒ Veri noisy (gürültülü), incomplete (tamamlanmamış), inconsistent (tutarsız), eksik, tekrarlanan, çelişkili olabildiği için

* Verinin önemli karakteristikleri ⇒ Dimensionality, Sparsity, Resolution, Size
* Veri önişleme işlemleri:
  - Data cleaning → Fill the missing value (eksik, yanlış değerleri doldurma) → naive-bayes, karar ağacı
    - remove outlier → (gürültü yumuşatma)
    - smoothing noisy data → binning metodları, clustering, regresyon
  - Data integration → farklı veri kaynaklarının entegre olması
  - Data transformation → normalizasyon (z-score, min-max, decimal scaling)
    - agregasyon → birleştirme
  - Data reduction (verinin küçültülmesi) → dimensionality reduction (boyutsal)
    - numerosity reduction (sayısal)
    - data compression (verinin sıkılaştırılması)
  - Data discretization → veri ayrıklaştırılması
* Data Quality (Veri Kalitesi) Kriterleri
  - Kesinlik (Accuracy) → doğru/yanlış
  - Tamamlanmışlık (Completeness) → kaydedilmemiş/ulaşılamayan
  - Yorumlanabilirlik (Interpretability) → verinin ne kadar kolay anlaşılacağı
  - Tutarlılık (Consistency) → verilerin tutarsızlığı/sallantıda olması
  - İnandırıcılık (Believability) — Güncellik (Timeliness)

2) BINNING METODLARI (KUTULAMA)

⇒ Mean, Median, Boundaries, Mode

SORU: 4, 8, 15, 21, 21, 24, 25, 28, 34 verisindeki gürültüyü temizle.

1→ Küçükten büyüğe sırala:

Bin1: 4, 8, 15
Bin2: 21, 21, 24
Bin3: 25, 28, 34

} binning] Kaç parça olacağı soruda verilir

2→ Means → Parçaların ortalamaları bulunur, her eleman o değeri alır.

Median → Her eleman, ortadaki elemanın değerini alır. (Veri sayısı çiftse → ortadaki iki değerin ort.

Boundaries → Her eleman, min ve max olan iki değerden hangisine yakınsa onun değerini alır. Eşit mesafedeyse, min.'u alır.

(means)	(median)	(boundaries)	(mode)
Bin1: 9,9,9	Bin1: 8,8,8	Bin1: 4,4,15	Bin1: 4,8,15
Bin2: 22,22,22	Bin2: 21,21,21	Bin2: 21,21,24	Bin2: 21,21,21
Bin3: 29,29,29	Bin3: 28,28,28	Bin3: 25,25,34	Bin3: 25,28,34

3) NORMALİZASYON

a) Z-score Normalizasyonu:

$$v' = \frac{v - \bar{x}}{S_x}$$

v': yeni değer
v: eski "
$\bar{x}$: ortalama
S_x: standart sapma

* Standart sapma:
1) Aritmetik ortalamayı bul
2) Her veriden A.O'yi çıkar
3) Sonuçların karelerini al topla
4) Veri sayısının 1 eksiğine böl, karekökü al.

b) Min-Max Normalizasyonu:

$$v' = \frac{v - min(A)}{max(A) - min(A)} \cdot [newmax(A) - newmin(A)] + newmin(A)$$

A: ilgili sütun
v: yeni değer
v: eski değer
Max(A): sütundaki max
Min(A): sütundaki min
newMin(A) > soruda verilen
newmin(A) aralıkların değeri

c) Decimal Scaling Normalizasyonu:

1) Soruda normalizasyon aralığı verilir.
2) Mutlak değerce en büyük olan değerin, verilen normalizasyon aralığında olacağı şekilde, bu sayı j'nin mümkün olan en küçük değerinde olduğu 10^j sayısı bulunur ve verideki her değer bu sayıya bölünür.

SORU: Veri seti: 10, 20, -52, -700 ⇒ [-1, 1] aralığında DS ile normalize et.

-Mutlak değerce en büyük: -700

10^j : 1000

$v_1 = 0,01$ $v_3 = -0,052$
$v_2 = 0,02$ $v_4 = -0,7$

4) CLASSIFICATION TEKNİKLERİ (SINIFLANDIRMA)

□ Temel Sınıflandırıcılar:

- Decision Tree: Karar Ağacı
- Nearest-neighbor: KNN
- Neural Networks, Deep Neural Nets (ANN, YSA, Multi Layer Perceptron)
- Naive-Bayes
- Support Vector Machines (SVM)

□ Topluluk Sınıflandırıcılar

- Boosting
- Bagging
- Random Forests

5) PRECISION-RECALL-F SCORE (ALGORİTMA PERFORMANSI)

SORU: Varsayalım ki bir döküman koleksiyonunda toplam 50 tane veri madencilği ile ilişkili döküman yer almaktadır. Koleksiyon toplam 200 dökümandan oluşmaktadır. Otomatik olarak döküman sınıflayan bir veri madenciliği algoritması sonucunda elde edilen durum şu şekildedir. Veri madenciliği ile ilişkili olan 45 adet döküman doğru keşfedilmiştir. Gerçekte veri madenciliği ile ilişkili olmayan 45 döküman ilişkili olarak yakalanmıştır. Bu sınıflama algoritması ile ilgili performans değerleri nedir?

Sınıf ⇒ veri madenciliği ile ilişkili kitaplar

$$\text{Precision} = \frac{\text{keşfedilen veri madenciliği kitapları}}{\text{keşfedilen tüm kitaplar}} = \frac{45}{90} = \%50 \quad (0,5)$$

$$\text{Recall} = \frac{\text{keşfedilen veri madenciliği kitapları}}{\text{tüm veri madenciliği kitapları}} = \frac{45}{50} = \%90 \quad (0,9)$$

$$\text{F-score} = \frac{\text{Precision ve Recall harmanik ortalaması}}{} = \frac{2 \cdot P \cdot R}{P+R} = \frac{2 \cdot 0,5 \cdot 0,9}{0,5+0,9} = \frac{0,9}{1,4} = 0,64$$

6) ENTROPY (KARAR AĞAÇLARI-1)

⇒ Entropi rastgeleliği, belirsizliği ve beklenmeyen durumun ortaya çıkma olasılığını gösterir.

⇒ Decision tree çizimi ve information gain (bilgi kazancı) hesaplamasında kullanılır.

⇒ $p_1, p_2, \dots p_n$ toplamları 1 olan olasılıklardır.

$$\text{Entropy} = E(p_1, p_2, \dots p_n) = -\sum_{i=1}^{n} p_i \cdot \log_2(p_i)$$

⇒ S veri kümesinde 14 örnek: 9'u A sınıfına, 5'i B sınıfına ait.

$$E(p_1, p_2, \dots p_n) = -\sum_{i=1}^{n} p_i \cdot \log_2(p_i) \Rightarrow E(p_1, p_2) = -\frac{9}{14} \cdot \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \cdot \log_2\left(\frac{5}{14}\right) = 0,940$$

4

⇒ Sınıflandırmada; olayın olması beklenen bir durumda entropi = 0

⇒ Örnekler aynı sınıfa aitse, entropi = 0

⇒ Örnekler sınıflar arasında eşit dağıtılmışsa, entropi = 1

⇒ Örnekler sınıflar arasında rastgele dağılmışsa, 0 < entropi < 1

7) BİLGİ KAZANCI (KARAR AĞAÇLARI-2)

⇒ Ağaç oluşturmada çoklu dallanma yaklaşımıdır.

⇒ Bilgi kazancı en büyük olan node, root (kök) düğümdür. (Bilgi kazancına göre nitelik seçilir)

⇒ Sınıflandırma sonucu için en az sayıda karşılaştırma yapmayı hedefler.

⇒ Bilgi Kazanç = Genel Entropi − Sütun Entropi

⇒ Genel Entropi, en sağdaki sütun ile hesaplanır.

⇒ Sütun Entropi = $E_1 \cdot p_1 + E_2 \cdot p_2 + \dots + E_n \cdot p_n$

Örnek:

A	B	Label
T	F	+
T	T	+
T	T	+
T	F	−
T	T	+
F	F	−
F	F	−
F	F	−
T	T	−
T	F	−

→ Genel Entropi $= -\frac{4}{10} \cdot \log_2\left(\frac{4}{10}\right) - \frac{6}{10} \cdot \log_2\left(\frac{6}{10}\right) = 0,9709$

		(+)	(−)			(+)	(−)
A ⇒	T	4	3	B ⇒	T	3	1
	F	0	3		F	1	5

$E(A, label) = E(T) \cdot p(T) + E(F) \cdot p(F)$

$= \left[\left(-\frac{4}{7} \cdot \log_2 \frac{4}{7} - \frac{3}{7} \cdot \log_2 \frac{3}{7}\right) \cdot \frac{7}{10}\right] + \left[\left(-\frac{\overbrace{0}^{0}}{3} \cdot \log_2 \frac{0}{3} - \frac{\overbrace{3}^{0}}{3} \cdot \log_2 \frac{3}{3}\right) \cdot \frac{3}{10}\right.$

$= 0,6897$

A'nın bilgi kazancı = 0,9709 − 0,6897 = 0,2812

B için de aynısı yapılır ve B.K en büyük olan root seçilir.

8) GINI İNDEKSİ (KARAR AĞAÇLARI-3)

⇒ Ağaç oluşturmada, ikiye dallanma yaklaşımıdır.

⇒ Gini index değeri en küçük olan nitelik seçilir.

⇒ $GINI(S) = 1 - \sum [p_i]^2$

Örnek:

C_1	2
C_2	4

$p(C1) = \frac{2}{6}$, $p(C2) = \frac{4}{6}$ ⇒ $Gini = 1 - \left(\frac{2}{6}\right)^2 - \left(\frac{4}{6}\right)^2 = 0,444$

9) KARAR AĞAÇLARI (4)

⇒ Akış diyagramı şeklinde bir ağaç yapısıdır.

⇒ Veri kümesinden, belli durumlarda nasıl davranacağı belirtilen bir ağaç çıkartılır.

⇒ Decision tree (karar ağacı) oluşturma algoritmaları; ağacı daha küçük, kompakt, sıkıştırılmış ve verimli hale getirmeyi hedefler.

⇒ Karar ağacı oluşturma algoritmaları:
- Hunt's Algorithm
- CART
- ID3, C4.5
- SLIQ, SPRINT

⇒ Karar ağacı modelleri, eğitim verilerindeki detayları öğrenir ve uygular.

⇒ Hem kategorik hem de sayısal verileri işleyen tek yöntemdir.

⇒ En iyi bölme belirlenmesinde "Greedy yaklaşımı" kullanılır (çoğunlukla aynı sınıfa ait örneklerin bulunduğu düğümler tercih edilir. (homojen sınıf dağılımına sahip olan düğümler)

⇒ Düğüm saflığı ölçümü 3 yolla yapılabilir:

- Entropy ⇒ information gain (bilgi kazanç) ile → çoklu dallanma
- Gini index ⇒ ikiye dallanma (binary)
- Misclassification error

⇒ Ağaç Oluşturma Yaklaşımları:

- Kökten yapraklara ulaşılır (parçala ve çöz = divide and conquer)
- Düğümü dallara ayırmada Greedy yaklaşımı kullanılır. Her adımda en iyi bölüm bulunur. Her düğümde dallanmak için en iyi nitelik seçilir. Her dal için aynı yaklaşım uygulanır.
- Yaprak düğümler en çok örneği olan sınıfla etiketlenir.

⇒ Karar ağaçlarının uygulanması kolaydır.

⇒ Karar ağaçları, bilinmeyen kayıtları sınıflandırmada son derece hızlıdır.

⇒ Küçük boyutlu ağaçların yorumlanması kolaydır.

10) KNN (K-Nearest Neighbor)

⇒ k=3, 5... gibi tek sayılar kullanılarak belirlenir. Verilen verinin, eğitim verisindeki değerlerle öklid uzaklığı ölçülür ve en yakın k komşuya göre benzerlik tayin edilir.

⇒ Supervised bir tekniktir ⇒ label (etiketler) olduğu için

⇒ Feature similarity measure yöntemi kullanır ⇒ parameter tuning

⇒ Avantajları:
- Uygulanması basittir.
- Gürültülü verilere karşı etkilidir
- Eğitim dokümanlarının sayısı çoğa ise daha etkindir.

⇒ Dezavantajları:
- Algoritma başlangıçta k parametresine ihtiyaç duyar.
- En iyi sonucun alınabilmesi için hangi uzaklık ölçümünün uygulanacağı ve hangi özelliklerin alınacağı bilgisi açık değildir.
- Hesaplama maliyeti yüksektir.

⇒ Lazy learner (tembel) bir tekniktir Decision tree ise, eager learner tekniktir.

⇒ Tüm eğitim datasını kullanır ⇒ eğitim adımı yok

⇒ Generalized model oluşturmaz.

11) NAIVE BAYES

⇒ Bayes teoremini dayanak alan bir yöntemdir. ⇒ $P(Y|X) = \frac{P(X|Y)\,P(Y)}{P(X)}$

⇒ Class labeller ile sınıflandırılacak olasılıklar birbirleri ile çarpılır ve son olarak class label olasılığı ile çarpılarak son durum bulunur.

⇒ Örneklerin hangi sınıfa, hangi olasılıkla ait olduklarını belirtir.

⇒ Nitelikler birbirinden bağımsızdır ve hepsi aynı derecede önemlidir.

⇒ Koşullu olasılıklardan birinin 0 olduğu büyük veri setlerinde, ilgili koşullu olasılığın payına 1 eklenerek sonuç 0'dan kurtarılır ⇒ Laplace düzeltmesi

⇒ Avantajları:
- Gerçeklemesi kolaydır.
- Çoğu durumda iyi sonuçlar verir.
- Alakasız niteliklere ve izole gürültü noktalarına dirençlidir.

⇒ Dezavantajları:
- Niteliklerin bağımsız olduğunu kabul eder ancak gerçek hayatta değişkenler birbirine bağımlıdır. Bu sebepten dolayı değişkenler arası ilişki modellenemez.
  ↳ Çözüm ⇒ bayes ağları

12) REGRESYON

⇒ Classification ⇒ kesikli veriler
Regresyon ⇒ sürekli veriler

⇒ Regresyon, verilerden yola çıkarak ortaya bir takım ilişkiler koymakta kullanılır. Bağımsız bir değişkenin değerleri için, bağımlı değişken tahmin edilmek istenir.

⇒ Lineer Regresyon: Bağımlı bir değişken (Y) ile bağımsız bir değişken (X) serisinin arasındaki ilişkinin doğrusal bir fonksiyonel biçimidir ve veriyi lineer bir çizgiye uydurma çalışmasıdır. (Lineer modelleme)

⇒ Regresyon Amaçları:
- Y'nin x'e bağlı ilişkisinin ortaya çıkarılması ⇒ istatiksel olarak anlamlı olmalı
- Yeni x değerleri için Y'nin tahmin edilmesi ⇒ daha önce verilmemiş veriler verildiğinde, ortaya veya geleceğe yönelik tahmin ortaya koyma

⇒ Lojistik Regresyon: Bağımlı=hedef=çıktı değişkeninin binary (0,1) olması söz konusuysa kullanılır.
Classification problemlerinde kullanılmak üzere geliştirilmiş ve tüm kesikli değerler ile çalışabilmektedir.

⇒ Lineer regresyon, en sıkça kullanılan ve en basit regresyon çeşididir. Bağımlı ve bağımsız değişken arasında düz bir çizgi çizerek aradaki bağıntıyı ortaya çıkarır. Sadece sürekli verilerde çalışır. Kategorik verilerde çalışmaz.

⇒ Lojistik regresyon ⇒ Kesikli (belirli sayıda değer alabilme) ve sürekli (bir aralıktaki tüm değerleri alabilme) verilerle çalışabilir.

13) SVM (Destek Vektör Makineleri)

⇒ Sınıflandırmayı doğrusal veya doğrusal olmayan bir fonksiyon yardımıyla yerine getirir.

⇒ Veriyi birbirinden ayırmak için en uygun fonksiyonun tahmin edilmesi esasına dayanır.

⇒ SVM; böl, parçala, yönet metoduyla çalışır ⇒ Her problemi ikili (binary) şekilde böler.

⇒ Kümelendirme ve regresyon analizinde de kullanılırlar.

⇒ Sınırlar lineer olmadığında, kernel trick yoluyla problem bir boyut yukarı taşınarak çözülür.

⇒ Doğrusal olmayan bir işlem ile n boyutlu veri kümesi $n>0$ olacak şekilde dönüştürülür.

⇒ Yüksek boyutta doğrusal sınıflandırma işlemi yapabilir.

⇒ Uygun bir dönüşüm ile her zaman veri bir hiper düzlem ile iki sınıfa ayrılabilir.

⇒ Hiper düzleme en yakın öğrenme verileri destek vektörleri olarak adlandırılır.
⇒ Overfitting durumunda C parametresi küçültülür.
⇒ Öğrenme problemi dışbükey bir optimizasyon problemi olarak formüle edilmiştir.

14) YSA (Yapay Sinir Ağları)

⇒ Lineer olmayan veriler arasındaki ilişkileri bulabilen güçlü bir yapıdır.

⇒ Perceptron: Yapay sinir hücresi ⇒ en temeli (1 giriş, 1 çıkış katmanı)

⇒ Girilen tüm değerlerin, istenilen formata dönüştürülmesi için bir aktivasyon fonksiyonu gerekir.

⇒ Aktivasyon fonksiyonları kullanılarak, çıktı değerinin istenen sınıflara ait olması sağlanır.

⇒ Sık kullanılan aktivasyon fonksiyonları: Sigmoid, Tanh, ReLU, Lineer, Siga

⇒ Sigmoid, Tanh, ReLU aktivasyon fonksiyonları; doğrusal olmayan (nonlineer) karar düzlemlerinin öğrenilmesine yardım eder.

⇒ Öğrenme katsayısı (λ=learning rate) gözetim altında tutulması gereken bir değerdir.

⇒ YSA, genellikle birden fazla gizli (ara) katmandan oluşur.

⇒ Epoch: Öğrenmenin gerçekleştiği döngüye denir.
⇒ Her bir epochta öğrenmenin kalitesi veya hata(loss) miktarı ölçülür.

⇒ Hata ve kayıpları azaltmak için iterasyon yapılır. (Öğrenme, süreçlerin sürekli tekrarlandığı bir döngüdür)

⇒ Optimizasyonda gradient descent algoritması kullanılır.

⇒ YSA ile öğrenme aşamaları:
1) Yapay sinir ağı oluşturma
   - giriş verisini modelleme; gizli katman sayısını ve bu katmanlardaki nöron sayısını belirleme
2) Yapay sinir ağını eğitme
3) Yapay sinir ağını küçültme
4) Sonucu yorumlama

⇒ YSA oluşturma
- giriş nöron sayısı = öğrenme kümesindeki verilerin nitelik sayısı
- gizli nöron sayısı ⇒ öğrenme sırasında ayarlanır.
- çıkış nöron sayısı = sınıf sayısı

⇒ YSA eğitme: Amaç, veri kümesindeki örneklerin hepsini doğru sınıflandıracak ağırlıkları belirlemektir.
1) Ağırlıklara rastgele değerler ata
2) Öğrenme kümesindeki giriş değerlerini teker teker sinir ağına uygula
3) Çıkışı hesapla
4) Hata değerini hesapla
5) Ağırlıkları hata fonksiyonunu en küçültecek şekilde yap.

⇒ YSA küçültme ⇒ Ağırlıklardan bazıları sınıflandırma sonucunu etkilemeyecek şekilde silinir.

⇒ Avantajları:
- Doğru sınıflandırma oranı genelde yüksek
- Kararlı - öğrenme kümesinde hata olduğu durumda da çalışır.
- Çıkış ayrık, sürekli ya da ayrık veya sürekli değişkenlerden oluşan bir vektör olabilir.

⇒ Dezavantajları:
- Öğrenme süresi uzun
- Öğrenilen fonksiyonun anlaşılması zor

⇒ Perceptron ⇒ ara katman içermez ⇒ doğrusal olmayan ayırma yapamaz.

⇒ Multi-layer Neural Network (Çok katmanlı Sinir Ağları) ⇒ En az bir ara katman içerirler ve doğrusal olmayan karar yüzeylerini içeren her türlü sınıflandırma görevini çözebilir.

⇒ Gizli katmandaki her düğüm, önceki katmandan gelen aktivasyonlar üzerinde çalışır ve aktivasyonları bir sonraki katmanın düğümlerine iletir. İleri beslemeli sinir ağları olarak da adlandırılırlar.

⇒ Her gizli katman bir soyutlama düzeyini temsil eder. Karmaşık özellikler, daha basit özelliklerin birleşimleridir. Katman sayısı, YSA derinliği olarak bilinir. Daha derin ağlar, karmaşık özellikler hiyerarşisini ifade eder.

⇒ YSA, kendisine örnekler gösterildikçe bu ağırlık değerlerini değiştirir. Hedef, ağa gösterilen örnekler için doğru çıkışları verecek ağırlıkları bulmaktır.

⇒ Biyolojik sinir hücresi → Yapay sinir hücreleri

Aksan → Çıktı
Dendrit → Toplama fonksiyonu
Çekirdek → Aktivasyon fonksiyonu
Sinaps → Ağırlıklar

⇒ YSA'nın öğrenme süresi uzundur fakat öğrenilen durumun yeni örneklerde test edilmesi yani değerlendirilmesi hızlıdır.

⇒ YSA'da bilgi:
- Bilgi ağın bağlantı ağırlıkları ile temsil edilir.
- YSA zekası ağırlıklardır.
- Ağın sahip olduğu ağırlık değerlerinin doğru olduğu ölçüde ağın performansı fazladır.
- Bilgi dağıtılmıştır, ağırlık değerlerinin bazıları kaybolsa dahi ağ çalışmasını sürdürebilir.

⇒ YSA'daki her hata, çıkış değerlerine ve bias değerinin çıkışa yansımasıdır.

⇒ YSA eğitimi batch processing (çevrimiçi) modda yapılır.

⇒ Büyük öğrenme oranında; sistem veriyi çok hızlı öğrenir, toplam hata artar. Düşük öğrenme oranında sistem çok yavaş öğrenir, eğitim zamanı artar ancak hata azalır.

⇒ YSA için lokal minimum hata tuzağına düşmek doğaldır ve bu durum global minimuma ulaşmasına engel olabilir. Momentum, öğrenme algoritmasını lokal minimumdan kaçacak şekilde önceki yönü tutmaya çalışır.

⇒ En iyi sonuçlar alınmadan birçok deneme yapılmalıdır. 0,6-0,8 aralığındaki değerler önerilir.

⇒ YSA durdurma kriterleri:
- Zaman eşik değerine ulaştığında bitebilir.
- Önceden tanımlı epoch değerine ulaştığında bitebilir.
- Önceden tanımlı hata değerine ulaştığında bitebilir.
- İki epoch arasındaki hata değeri azaldığında eğitim bitebilir.

⇒ Örnekler eğitim kümesinden rastgele seçilirler.

⇒ Bias:
- Sistem performansını etkiler.
- Öğrenmede bir ağırlık gibi alınır.
- Giriş sinyallerinin toplamı 0 olduğunda öğrenme gerçekleşmez. Çıkış değerleri hep 1 olan bir nöron olur, nöronların giriş sinyallerinin sürekli sıfırdan farklı olmasını sağlar.
- Öğrenmeyi hızlandırırken, yerel optimum değerlere takılmayı güçleştirirler.

⇒ YSA yapılarına göre ileri beslemeli ve geri beslemeli olarak ikiye ayrılır.

⇒ Backpropagation: İleri beslemeli YSA'nı eğitmek için yaygın olarak kullanılan bir algoritma. Geri yayılım algoritması, zincir kuralı doğrultusunda her ağırlığa göre kayıp fonksiyonunun gradyanını hesaplayarak, gradyanı tek bir seferde bir katman olarak hesaplar. Zincir kuralındaki ara terimlerin gereksiz hesaplanmalarını önlemek için son katmandan geriye doğru ilerleyerek çalışan bir dinamik programlama örneğidir.

⇒ Backpropagation'daki ağırlık güncelleme döngüsü binlerce kez iteratif olarak devam edebilir. Belirli bir iterasyon sayısında döngü sonlandırılabilir ya da hata değerine göre sonlandırılabilir. Bu seçim kritiktir; çünkü az sayıda iterasyon hatayı yeterince düşürmez, çok sayıda iterasyon ise overfitting'e sebep olabilir.

15) MODEL SEÇİMİ, DEĞERLENDİRMESİ VE ENSEMBLE

⇒ Model başarısı değerlendirilirken, validation test dikkate alınır (training = eğitim seti değil)

⇒ Modelin başarısını ölçmede kullanılan metodlar:

- Confusion Matrix (karışıklık matrisi)
- F-score
- Cross validation (k-fold)
- Bootstrap
- Confidence intervals } iki modelin başarısını karşılaştırma/ölçmede
- ROC eğrisi }
- Holdout
- Random subsampling

⇒ Bias → isabet, variance → farklılık

↓

high bias ve low variance ⇒ underfitting
low bias ve high variance ⇒ overfitting

⇒ Yüksek varyans → eğitim verilerinde düşük hata, doğrulama verilerinde yüksek hata durumu
- modele eğitim verisi eklemek } çözüm
- regularization (düzenlileştirme) }

⇒ Yüksek bias → Hem eğitim hem de doğrulama verilerinde yüksek hata durumu
- model eğitiminde iterasyonlara devam etmek } çözüm
- öğrenme algoritmasını farklılaştırmak }

⇒ Regularization yöntemleri:

- Lasso regularization } ağırlık azaltımı
- Ridge regularization }
- Early stopping regularization
- Dropout regularization
- Data augmentation
- Batch Normalization
- Ensemble Learnings

⇒ Curse of dimensionality (boyut laneti): Çok yüksek boyutlu veri kümelerinde karşılaşılan problem
- veri seyrekliği
- mesafe konsantrasyonu

⇒ Ensemble: Tek bir hipotez yerine hipotez uzayından bir hipotez topluluğu seçmek ve tahminleri birleştirmektir. Tek bir güçlü sınıf yerine çok sayıda zayıf sınıflayıcı kullanılır. Hipotezler birbirinden farklılaştıkça hataları arasındaki korelasyon azalır ve topluluk öğrenmesi daha kullanışlı olur.

⇒ Ensemble sınıflayıcıları oluşturma:

- Eğitim setini manipüle ederek
  - bagging → ortalama, majority voting
  - boosting → ağırlık (önceliği) ile, weighted voting
- Giriş özelliklerini (input features) manipüle ederek
  - random forests
- Sınıf etiketlerini (class label) manipüle ederek
  - hata düzeltici çıkış kodlaması
- Öğrenme algoritmasını manipüle ederek:
  - YSA veya karar ağacında rastgelelik eklemek

⇒ Random forest → Kaşula göre hangi algoritmanın çalışacağını, hangi şartta hangi algoritmanın daha başarılı olacağını belirleyen algoritmadır.

⇒ Adaboost → Algoritmalar başarılarına göre bir ağırlığa sahip olur ve ağırlık zaman zaman değişir. Toplam hatanın kontrol altına alınması amaçlanır.

16) CLUSTERING

⇒ Clustering algoritmaları verileri gruplayarak kümeleme işlemi yapar.

⇒ Unsupervised bir tekniktir. Sınıf etiketi (class label) kullanmaz, feature similarity yapar.

⇒ Similarity matrix hesaplayarak benzer özelliğe sahip verileri kümeler. Bu hesaplamada çeşitli metriklerden yararlanılır.

↳ Öklid, minkowski, manhattan

⇒ Şekil, yoğunluk, konum bazlı kümeleme yapılabilir.

⇒ Kümeleme (clustering) algoritmaları:

- K-means ve varyantları
- Hierarchical clustering
- Density-based clustering

⇒ Hierarchical clustering'te kümeler arası uzaklığı belirleme:
- MIN (tek bağ)
- MAX (tam bağ)
- Ortalama
- Merkezler arası uzaklık

17) K-MEANS

⇒ Algoritma aşamaları:
1) Veri kümesi k alt kümeye ayrılır.
2) Her kümenin ortalaması hesaplanır: merkez nokta
3) Her nesne en yakın merkez noktanın olduğu kümeye dahil edilir.
4) Nesnelerin kümelenmesinde değişiklik olmayana kadar 2. adıma geri dönülür.

⇒ K-means değerlendirme: Hataların kareleri toplamını (SSE) azaltmak için; } en az SSE değerine sahip olan kümeleme seçilir.
- k küme sayısı artırılabilir
- Başlangıç için farklı merkez noktaları seçilebilir.

⇒ Karmaşıklığa etkili olan faktörler:
- Nokta sayısı
- Küme sayısı
- İterasyon sayısı
- Nitelik sayısı

⇒ Soru çözümü:
- k kadar rastgele nokta seç.
- Seçilen noktalar ile diğer noktaların mesafelerini ayrı ayrı hesapla
- En küçük mesafeye göre I'ları kümele.
- k kadar noktanın, new c değerleri, kümelerindeki I'ların aritmetik ortalamasıyla bulunur.
- Grafik çizilir. I'lar kümelenir, new c değerleri belirtilir.
- İterasyon sayısı kadar devam edilir. (c → new c)

18) ASSOCIATION RULE VE APRIORI ALGORİTMASI

$A \rightarrow B$ Support = $\frac{A,B \text{ sayısı}}{\text{satır sayısı}}$! Confidence = $\frac{A,B \text{ sayısı}}{A \text{ sayısı}}$

⇒ support count = support x satır sayısı