Örüntü Tanıma

Uygulama 4

K En Yakın Komşu (KNN) Algoritması

- KNN, temel olarak veri noktalarının benzerliklerine dayanarak tahmin yapan bir gözetimli makine öğrenmesi algoritmasıdır.
- Bir “model” eğitimi gerekmeksizin tahmin yapabilir.
- Sınıflandırma ve regresyon amaçlı kullanılabilir.
- **Sınıflandırma:**Veri noktası, kendisine en yakın K komşusunun çoğunlukta olan sınıfına atanır.
- **Regresyon**: Veri noktası, K komşusunun ortalaması veya ağırlıklı ortalaması kullanılarak tahmin edilir.

KNN Nasıl Çalışır

- Yeni bir veri noktası için tahmin yapılmak istendiğinde, eğitim kümesindeki tüm veri noktalarına olan mesafe hesaplanır.
- En küçük mesafeye sahip K veri noktası belirlenir.
- Eğer sınıflandırma yapılıyorsa, bu komşuların çoğunlukta olan etiketi yeni veri noktasına atanır. Eğer regresyon yapılıyorsa, komşuların hedef değerlerinin ortalaması alınarak tahmin yapılır.

İki Komşu Arası Mesafenin Hesaplanması

Minkowski Mesafesi

$$d(A,B) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$

İki Komşu Arası Mesafenin Hesaplanması

Manhattan Mesafesi (p=1)

$$d(A, B) = |x_2 - x_1| + |y_2 - y_1|$$

$$d(A, B) = \sum_{i=1}^{n} |x_i - y_i|$$

İki Komşu Arası Mesafenin Hesaplanması

Öklid Mesafesi(p=2)

$$d(A,B) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

$$d(A,B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 + \ldots}$$

Öklid Mesafesi Örnek

A(3,4) ve B(6,8) arası mesafe:

$$d(A,B) = \sqrt{(6-3)^2 + (8-4)^2} = \sqrt{3^2 + 4^2} = \sqrt{9+16} = \sqrt{25} = 5$$

Kosinüs Benzerliği

$$\cos(\theta) = \frac{A \cdot B}{||A|| \cdot ||B||}$$

1 → Vektörler benzer
0 → Vektörler birbirine dik (ilişkisiz)
-1 →Vektörler birbirine zıt

$$A = (1, 2, 3), \quad B = (4, 5, 6)$$

1. **Skaler çarpımı hesaplayalım:**

$$A \cdot B = (1 \times 4) + (2 \times 5) + (3 \times 6) = 4 + 10 + 18 = 32$$

2. **Vektör uzunluklarını hesaplayalım:**

$$||A|| = \sqrt{1^2 + 2^2 + 3^2} = \sqrt{1 + 4 + 9} = \sqrt{14}$$

$$||B|| = \sqrt{4^2 + 5^2 + 6^2} = \sqrt{16 + 25 + 36} = \sqrt{77}$$

3. **Kosinüs benzerliğini hesaplayalım:**

$$\cos(\theta) = \frac{32}{\sqrt{14} \times \sqrt{77}} \approx 0.97$$

KNN

Nokta	X1	X2	Sınıf
A	1	2	Kırmızı
B	2	3	Kırmızı
C	3	3	Mavi
D	5	5	Mavi

Yeni verilen bir X1=3, X2=2 noktasının sınıfını tahmin etmek istiyoruz.

Nokta	X1	X2	Mesafe Hesabı	Sonuç
A (Kırmızı)	1	2	$\sqrt{(3-1)^2+(2-2)^2}$	2.0
B (Kırmızı)	2	3	$\sqrt{(3-2)^2+(2-3)^2}$	1.41
C (Mavi)	3	3	$\sqrt{(3-3)^2+(2-3)^2}$	1.0
D (Mavi)	5	5	$\sqrt{(3-5)^2+(2-5)^2}$	3.61

- Bu noktalar arasından en yakın k kadarını seçiyoruz.

K=3 olsun:

C (Mavi) → 1.0

B (Kırmızı) → 1.41

A (Kırmızı) → 2.0

3 noktadan en fazla kırmızı olana yakın olduğuna göre tahminimizi kırmızı olarak yapıyoruz.

K kadar komşunun hepsi farklı sınıfta olursa ne yapabiliriz?

- **K değerini arttırma:**
  Böyle durumlarda seçilen k değerini geçici arttırmak bir çözüm olabilir.
- **Komşuları ağırlıklarına göre sınıflandırma**

  w=1/d gibi bir formül kullanılarak komşuları ağırlıklandırıp ağırlığı yüksek olan sınıf seçilebilir.
- **Daha önce tahmin edilen sonuçlara yakınlık**

  Daha önce tahmin edilen yakın sonuç varsa veya belli bölgede belirli sınıflara yoğunluk varsa onlar da hesaba katılabilir.

Veri Kümeleri ve Makine Öğrenmesi ile Kavramlar

- Bir makine öğrenimi modelini eğitirken veriyi genellikle üç bölüme ayırırız:
- **Train (Eğitim)**
- **Validation (Doğrulama)**
- **Test (Test)**

Train (Eğitim) Kümesi

- Modelin öğrenme sürecinde kullanılan, ağırlık ve parametreleri güncellemek için kullanılan asıl veridir.
- Genellikle verinin büyük bölümünü oluşturur (örneğin %70-80 civarı).
- Model, bu veriler üzerinde iteratif olarak çalışır ve hatayı (loss) minimize etmek için optimizasyon yapar.
- Derin öğrenmede eğitim seti üzerinde model sürekli geri yayılım (backpropagation) ile ağırlıkları günceller.

Eğitimde Neden Metrik Ölçüyoruz?

- Eğitim sürecinde modelin öğrenme eğilimini görmek ve eğitim sürecinin düzgün ilerleyip ilerlemediğini anlamak için.
- Modelin yeterli öğrenip öğrenmediğini veya daha fazla eğitime ihtiyaç duyup duymadığını anlamak için kullanılır.
- Eğitim setindeki çok yüksek doğruluk, modelin iyi öğrendiğini değil, bazen aşırı uyum (overfitting) olduğunu gösterebilir.

Validation (Doğrulama) Kümesi

- Modelin eğitim sırasında, hiperparametre ayarlamaları (örneğin learning rate (öğrenme katsayısı), epoch (döngü) sayısı, katman sayısı gibi) ve genel performans değerlendirmeleri için kullanılan veridir.
- Genellikle verinin daha küçük bir bölümüyle oluşturulur (%10-20 civarı).
- Model eğitim sırasında, her epoch sonrasında doğrulama verisi üzerinde değerlendirilir, ancak doğrulama verisi ile ağırlıklar güncellenmez.
- Modelin eğitim sırasında hiperparametre optimizasyonunda yol gösterici rol oynar.

Doğrulamada Neden Metrik Ölçüyoruz?

- Validation loss ve accuracy, modelin daha önce görmediği veriler üzerinde genelleme yeteneğini ölçer.
- Eğitim (train) loss'u düşerken doğrulama (validation) loss'u yükselirse, modelin aşırı uyuma (overfitting) yöneldiği anlaşılır. Bu da eğitim sürecinin durdurulması gerektiğini (early stopping) gösterir.
- Hyperparameter tuning sürecinde en iyi performansa sahip modeli seçebilmek için kritik öneme sahiptir.

Test Kümesi

- Modelin nihai olarak genelleme yeteneğinin, yani hiç görmediği yeni veriler üzerindeki performansının bağımsız olarak değerlendirilmesi için kullanılır.
- Verinin %10-20 civarıdır ve eğitim sürecinde hiç kullanılmamalıdır.
- Test kümesi üzerinde elde edilen sonuçlar, modelin gerçek dünya performansının en objektif göstergeleridir.
- Test sonuçları, modelin gerçek performansını temsil eder ve literatürde raporlanan asıl değerler bunlardır.
- Bu değerler, modelin tamamen dışarıdan gelen verilere ne kadar iyi genelleştirdiğini, başarısını ve kullanılabilirliğini yansıtır.

Kayıp (Loss) ve Doğruluk (Accuracy)

Kayıp fonksiyonu nedir?

Modelin tahmin ettiği değerler ile gerçek değerler arasındaki farkın matematiksel olarak hesaplanmasını sağlar.

Düşük loss değeri, modelin daha doğru tahminler yaptığı anlamına gelir.

Model eğitimi sırasında sürekli ölçüldüğünden hesaplaması kolay olan fonksiyonlar kullanılması tercih edilir.

- **Accuracy (Doğruluk) Nedir?**

Özellikle sınıflandırma problemleri için modelin doğru tahmin ettiği örneklerin toplam örneklere oranıdır.

Sıklıkla Kullanılan Kayıp Fonksiyonları

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

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

Binary Cross Entropy Loss

$$Loss = -[y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})]$$

Örneğin gerçek sınıf etiketi 1 olan bir örneğin model çıktısı 0.8.

$$Loss = -[1 \cdot \log(0.8) + (1 - 1) \cdot \log(1 - 0.8)]$$

=~ 0.2231

Makine Öğrenmesinde Sıklıkla Kullanılan Kavramlar

- Epoch (Döngü):
  Bir epoch, tüm eğitim verisinin model tarafından baştan sona tek sefer geçilmesine denir. Birden fazla epoch yapılması, modelin veriyi daha iyi öğrenmesine yardımcı olur. Fazla epoch, aşırı öğrenmeye (overfitting) neden olabilir, az epoch ise yetersiz öğrenmeye (underfitting) neden olabilir.
- Tensör:

Basit bir tanımla çok boyutlu matrislerdir.Örneğin bir görüntü (yükseklik × genişlik × renk kanalı) şeklinde 3B bir matris ile ifade edilebilir.

Yığın Boyutu (Mini Batch Size)

- Bir epoch içinde eğitim verisi küçük gruplara bölünür ve model bu küçük grupları kullanarak ağırlıkları günceller. Bu grupların her birine batch denir. Batch size ise tek seferde modelin göreceği örnek sayısını ifade eder.
- 1000 örnek var, batch size 100 ise her epoch için 10 batch olacak şekilde veriler ayrılır.
- Küçük batch size gürültülü güncellemelere (stochastic), büyük batch size daha stabil ama hesaplama yükü fazla güncellemelere neden olur. Genellikle batch size 32, 64, 128 gibi değerler kullanılır.

İterasyon (Iteration)

- Bir iterasyon, modelin tek bir batch verisi ile güncelleme yaptığı her adımı ifade eder. Başka bir deyişle, modelinizin her bir faklı sürümü bir iterasyondur.
- İterasyon Sayısı = Toplam Veri Sayısı/Yığın Sayısı

Öğrenme Oranı (Learning Rate)

- Modelin parametrelerini güncellerken, kayıp fonksiyonunun türevini kullanarak her adımda ne kadar ilerleyeceğini belirleyen katsayıdır.

Yüksek learning rate hızlı öğrenir ancak global minimumu kaçırabilir.

Düşük learning rate daha yavaş ama kararlı bir öğrenme sağlar.