# K-En Yakın Komşuluk Algoritması ile MNIST Rakam Sınıflandırması

Yiğit Buğra KÜÇÜK
Yapay Zeka Mühendisliği • 230212048
230212048@ostimteknik.edu.tr • October 16, 2025

***Abstract*—Bu çalışma, K-En Yakın Komşuluk (KNN) algoritmasının temel prensiplerini uygulamalı olarak incelemektedir. MNIST el yazısı rakamları veri seti üzerinde L1 (Manhattan) ve L2 (Öklid) mesafe metrikleri kullanılarak kapsamlı bir performans analizi gerçekleştirilmiştir. Geliştirilen KNN sınıflandırıcı, farklı k değerleri ve mesafe metrikleri altında test edilmiş, en iyi performans k=7 değeri ve L2 mesafe metriği ile %98.89 doğruluk oranında elde edilmiştir. Ayrıca, implementasyon scikit-learn kütüphanesi ile karşılaştırılmış ve benzer doğruluk değerleri gözlemlenmiştir.**

## I. Giriş

K-En Yakın Komşuluk (KNN) algoritması, makine öğrenmesinde en temel ve sezgisel sınıflandırma algoritmalarından biridir. Bu çalışmada, KNN algoritmasının sıfırdan implementasyonu gerçekleştirilmiş ve MNIST veri seti üzerinde performans değerlendirmesi yapılmıştır.

Algoritmanın temel amacı, bir test örneğinin en yakın k komşusunu bularak, bu komşuların çoğunluk sınıfına dayalı olarak sınıflandırma yapmaktır. Çalışmanın ana odak noktaları; farklı k değerlerinin, mesafe metriklerinin model performansı üzerindeki etkisini analiz etmek ve kendi implementasyonumuzun endüstri standardı kütüphanelerle karşılaştırmalı değerlendirmesini yapmaktır.

## II. Kod Açıklaması

### A. Implementasyon Yaklaşımı

KNN algoritması dört temel fonksiyon üzerine inşa edilmiştir:

- **fit():** Eğitim verilerini ve etiketlerini saklayarak modeli hazırlar
- **compute_distances():** Test ve eğitim örnekleri arasındaki mesafeleri hesaplar
- **predict():** k-en yakın komşuyu bularak çoğunluk oylaması ile tahmin yapar
- **score():** Modelin doğruluk skorunu hesaplar

Mesafe hesaplama fonksiyonunda hem L1 (Manhattan) hem de L2 (Öklid) metrikleri implemente edilmiştir:

$$\mathrm{L1}(x, y) = \sum_{i=1}^{n} |x_i - y_i| \tag{1}$$

$$\mathrm{L2}(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2}$$

### B. Karşılaşılan Zorluklar

Implementasyon sürecinde şu zorluklarla karşılaşılmıştır:

- Büyük ölçekli mesafe matrislerinin hesaplama karmaşıklığı
- Bellek yönetimi ve verimli dizin işlemleri
- Farklı k değerlerinde model kararlılığının sağlanması

### C. Çözüm Stratejileri

Yukarıdaki zorlukları aşmak için aşağıdaki stratejiler geliştirilmiştir:

- Vektörleştirilmiş işlemler ile hesaplama performansı optimize edilmiştir
- NumPy kütüphanesinin etkin kullanımı ile bellek verimliliği artırılmıştır
- k değeri analizi ile en uygun parametre seçimi yapılmıştır

## III. Deneysel Sonuçlar

### A. Test Accuracy Sonuçları

Temel KNN modeli k=3 ve L2 mesafe metriği ile eğitilmiş ve test edilmiştir. Elde edilen sonuçlar:

| Metric | Value |
|---|---|
| Test Accuracy | 0.9833 |
| Hesaplama Süresi | 0.08 saniye |

### B. K Değeri Analizi

k değerinin model performansı üzerindeki etkisini analiz etmek için [1, 3, 5, 7, 9, 11, 15, 21] değerleri test edilmiştir:

| k Değeri | Accuracy |
|---|---|
| 1 | 0.9778 |
| 3 | 0.9833 |
| 5 | 0.9861 |
| 7 | **0.9889** |
| 9 | 0.9806 |
| 11 | 0.9833 |
| 15 | 0.9806 |
| 21 | 0.9722 |

### C. Karışıklık Matrisi ve Örnek Görselleştirmeler

Modelin sınıflandırma performansı karışıklık matrisi ile detaylı olarak analiz edilmiştir. Ayrıca, rastgele seçilen 10 örnek üzerinde model tahminleri görselleştirilmiştir.

### D. Mesafe Metrikleri Karşılaştırması

L1 ve L2 mesafe metriklerinin performans karşılaştırması aşağıdaki tabloda sunulmuştur:

| k Değeri | L1 Accuracy | L2 Accuracy | Fark |
|---|---|---|---|
| 1 | 0.9750 | 0.9778 | 0.0028 |
| 3 | 0.9778 | 0.9833 | 0.0056 |
| 5 | 0.9778 | 0.9861 | 0.0083 |
| 7 | 0.9750 | **0.9889** | 0.0139 |
| 9 | 0.9778 | 0.9806 | 0.0028 |
| 11 | 0.9778 | 0.9833 | 0.0056 |
| 15 | 0.9750 | 0.9806 | 0.0056 |
| 21 | 0.9667 | 0.9722 | 0.0056 |

### E. Scikit-learn Karşılaştırması

Kendi implementasyonumuz scikit-learn kütüphanesinin KNeighborsClassifier sınıfı ile karşılaştırılmıştır:

| Metric | Kendi KNN | Scikit-learn |
|---|---|---|
| Accuracy | 0.9833 | 0.9833 |
| Hesaplama Süresi | 0.22s | 3.87s |

## IV. Analiz ve Yorumlar

### A. En İyi k Değeri Analizi

Deneysel sonuçlara göre, k=7 değeri %98.89 doğruluk oranı ile en iyi performansı göstermiştir. k değeri 7'ye kadar arttıkça modelin doğruluğu artmış, bu değerden sonra ise azalma eğilimi göstermiştir. Bu durum, k=1 değerinde overfitting, k=21 değerinde ise underfitting gözlemlendiğini işaret etmektedir.

### B. L1 vs L2 Mesafe Metrikleri Analizi

L2 (Öklid) mesafe metriği, tüm k değerlerinde L1 (Manhattan) metriğine göre daha iyi performans göstermiştir. Özellikle k=7 değerinde iki metrik arasındaki fark en yüksek seviyeye (0.0139) ulaşmıştır. Bu sonuç, MNIST görüntülerinin piksel uzayında Öklid mesafesinin daha anlamlı bir benzerlik ölçütü sağladığını göstermektedir.

### C. Scikit-learn Karşılaştırması Analizi

Kendi implementasyonumuz ile scikit-learn kütüphanesinin doğruluk değerleri tamamen aynı çıkmıştır. Ancak, hesaplama süreleri karşılaştırıldığında scikit-learn'in daha yavaş çalıştığı gözlemlenmiştir. Bu durum, scikit-learn'in daha genel amaçlı ve optimize edilmiş veri yapıları kullanmasından kaynaklanıyor olabilir.

## V. Sonuç

Bu çalışmada, KNN algoritmasının sıfırdan başarılı bir implementasyonu gerçekleştirilmiş ve MNIST veri seti üzerinde kapsamlı bir performans analizi yapılmıştır.

### A. Öğrenilenler

- KNN algoritmasının temel prensipleri ve implementasyon detayları
- k parametresinin model karmaşıklığı ve genelleme performansı üzerindeki kritik etkisi
- Farklı mesafe metriklerinin sınıflandırma performansı üzerindeki etkileri
- Endüstri standardı kütüphanelerle karşılaştırmalı analiz metodolojisi

### B. Gelecek Çalışma Önerileri

- Çok çözünürlüklü arama teknikleri ile hesaplama karmaşıklığının azaltılması
- Farklı veri setleri üzerinde algoritma performansının değerlendirilmesi
- Ağırlıklı KNN ve diğer KNN varyasyonlarının implementasyonu
- GPU hızlandırma ile büyük ölçekli veri setlerinde performans iyileştirmesi

## Kaynakça


1) Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. IEEE Transactions on Information Theory.
2) Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research.
3) LeCun, Y., Cortes, C., & Burges, C. J. C. (1998). The MNIST database of handwritten digits.