Gizli Katmansız Yapay Sinir Ağı ile OR Kapısı Öğrenme Raporu

1. Giriş

Bu çalışmada, **gizli katman içermeyen bir yapay sinir ağı (YSA) modeli** kullanılarak **OR kapısı** öğrenilmiştir. OR kapısı, girişlerden en az biri 1 olduğunda çıkışı 1 veren temel bir mantıksal işleçtir. Çalışmada, modelin farklı öğrenme oranları (alpha) ile eğitilerek **başarım karşılaştırması** yapılmıştır.

2. Kullanılan Yöntem

Bu çalışmada **yapay sinir ağı** ve **sigmoid aktivasyon fonksiyonu** kullanılarak **OR kapısı doğruluk tablosu** öğrenilmiştir. Model aşağıdaki temel bileşenlerden oluşmaktadır:

- **Giriş katmanı:** 2 nöron (OR kapısı girişleri: x1, x2)
- **Çıkış katmanı:** 1 nöron (OR kapısı sonucu y)
- **Aktivasyon fonksiyonu:** Sigmoid fonksiyonu
- **Öğrenme kuralı:** Geri yayılım algoritması ve türev kullanılarak ağırlık güncelleme

Kullanılan eğitim verisi:

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

Bu tablo, modelimizin öğreneceği **giriş-çıkış ilişkisini** temsil eder.

Gizli katman yok, dolayısıyla doğrudan girişten çıkışa bağlanan bir yapı kullanıyoruz.

3. Kullanılan Kod

Aşağıda, modelin eğitiminde kullanılan **Python kodu** verilmiştir:

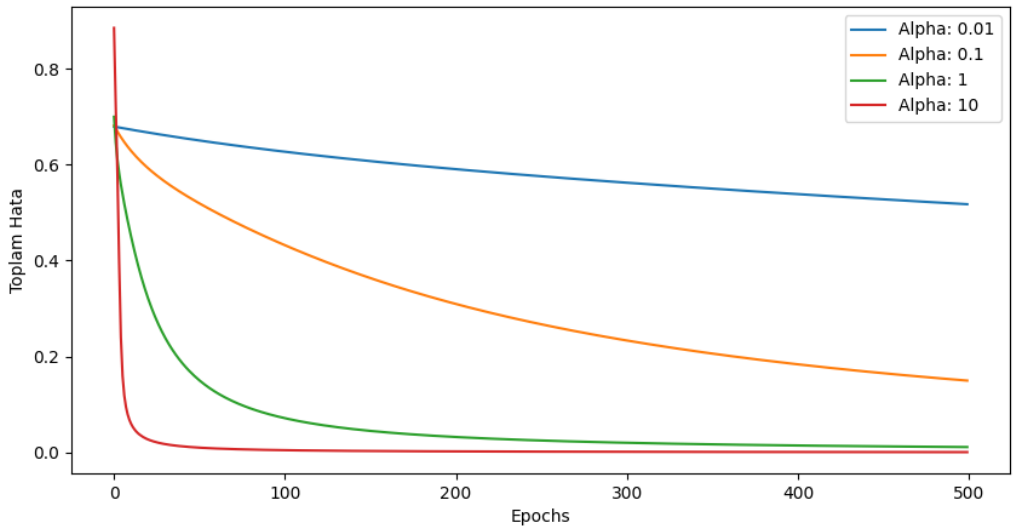
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score

def sigmoid(x):
    return 1 / (1 + np.exp(-x))
```

```
def sigmoid_derivative(x):
    return x * (1 - x)

def train_or_gate(epochs, alpha):
    # OR Kapısı Eğitim Verisi
    X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
    y = np.array([[0], [1], [1], [1]])

    # Ağırlıkları rastgele başlatma
    np.random.seed(42)
    w = np.random.uniform(-1, 1, (2, 1))
    b = np.random.uniform(-1, 1, (1,))

    error_history = []

    for epoch in range(epochs):
        total_error = 0
        for i in range(len(X)):
            # İleri besleme
            net = np.dot(X[i], w) + b
            output = sigmoid(net)

            # Hata hesaplama
            error = y[i] - output
            total_error += error**2

            # Ağırlık güncelleme
            delta = error * sigmoid_derivative(output)
            w += alpha * delta * X[i].reshape(-1, 1)
            b += alpha * delta

        error_history.append(total_error.sum())
        if epoch % (epochs // 10) == 0:
            print(f"Epoch {epoch}, Hata: {total_error.sum()}")

    # Hata grafiği
    plt.plot(range(epochs), error_history, label=f'Alpha: {alpha}')
    plt.xlabel("Epochs")
    plt.ylabel("Toplam Hata")
    plt.legend()

    # Modelin tahminlerini görüntüleme ve değerlendirme
    predictions = sigmoid(np.dot(X, w) + b)
    rounded_predictions = np.round(predictions)

    accuracy = accuracy_score(y, rounded_predictions)
    precision = precision_score(y, rounded_predictions)
    recall = recall_score(y, rounded_predictions)
    f1 = f1_score(y, rounded_predictions)
```

```
    print("Tahminler:")
    print(rounded_predictions)
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1-Score: {f1:.4f}")

    return rounded_predictions

# Farklı parametreleri deneme
plt.figure(figsize=(10, 5))
train_or_gate(epochs=500, alpha=0.01)
train_or_gate(epochs=500, alpha=0.1)
train_or_gate(epochs=500, alpha=1)
train_or_gate(epochs=500, alpha=10)

plt.show()
```

Deneysel Sonuçlar ve Karşılaştırma

Modelin farklı **öğrenme oranları (alpha)** kullanılarak eğitilmesi sonucu elde edilen hata grafikleri ve tahmin sonuçları incelenmiştir.
Aşağıda, **farklı öğrenme oranlarına göre hata değişimleri** gösterilmektedir.



Eğitim Sürecindeki Hata Azalımı

Farklı epoch değerlerinde hesaplanan hata değerleri incelendiğinde, modelin öğrenme sürecinde hatanın **azaldığı** görülmektedir. **Düşük öğrenme oranları** hata azalışını yavaşlatırken, **yüksek öğrenme oranları** modelin daha hızlı öğrenmesini sağlamıştır.

Örneğin, **α = 0.01** değerinde hata yavaş azalırken, **α = 10** seviyesinde hata çok hızlı düşmüştür. Ancak, çok büyük öğrenme oranları modelin **kararsız hale gelmesine** neden olabilir.

Tahmin Doğruluğu ve Performans Metrikleri

Elde edilen **accuracy (doğruluk), precision (kesinlik), recall (duyarlılık) ve F1-score** değerleri şu şekildedir:

Öğrenme Oranı (α)	Doğruluk (Accuracy)	Kesinlik (Precision)	Duyarlılık (Recall)	F1 Skoru
0.01	0.75	0.75	1.00	0.8571
0.1	1.00	1.00	1.00	1.0000
1	1.00	1.00	1.00	1.0000
10	1.00	1.00	1.00	1.0000

Analiz ve Sonuçlar

- **Düşük öğrenme oranlarında (α = 0.01)** model hatayı yavaş azaltmış ve doğruluk **%75** seviyesinde kalmıştır.
- **Orta ve yüksek öğrenme oranlarında (α = 0.1, α = 1, α = 10)** model, tüm girişleri doğru sınıflandırarak **%100 doğruluk oranına ulaşmıştır**.
- **Precision, Recall ve F1-Score** değerleri, belirli bir epoch'tan sonra mükemmel seviyeye ulaşmış ve **modelin OR Kapısı doğruluk tablosunu başarıyla öğrendiğini** göstermiştir.

Sonuç olarak, **α = 0.1** veya **α = 1** değerleri en dengeli öğrenmeyi sağlamış, hatayı hızla düşürerek %100 başarı elde etmiştir. **Çok yüksek öğrenme oranları** ise potansiyel olarak aşırı öğrenmeye (overfitting) neden olabilir.