T.C.

SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BULANIK MANTIK VE YAPAY SİNİR AĞLARINA GİRİŞ ÖDEV RAPORU

2024 GÜZ DÖNEMİ 2. ÖDEV

G211210029 - Aybüke Berfin SÜER

SAKARYA

Aralık, 2024

Yapay Sinir Ağları Performans Analizi ve Grafiksel Değerlendirme

Aybüke Berfin SÜER

G211210029
Dr. Öğr. Üyesi Muhammed Fatih ADAK

Giriş

Yapay sinir ağları (Artificial Neural Networks - ANN), makine öğrenimi ve veri bilimi alanında oldukça önemli bir yere sahiptir. Özellikle sınıflandırma, regresyon ve kümeleme gibi problemler üzerinde etkili çözümler sunmaktadır. Sinir ağlarının başarısı, kullanılan eğitim algoritmalarına ve ağ topolojisine bağlı olarak değişiklik gösterebilmektedir. Bu çalışmada, farklı yapay sinir ağı topolojilerinin performansları, momentumlu ve momentumsuz eğitim yöntemleri kullanılarak karşılaştırılmıştır.

Momentumlu eğitim, öğrenme sürecini hızlandırmak ve hata yüzeyinde daha stabil bir şekilde ilerlemek için eklenen bir yöntemdir. Bu yöntemde, önceki adımlardan elde edilen ağırlık değişiklikleri de hesaba katılarak daha kararlı bir güncelleme sağlanır. Momentumsuz eğitim ise klasik geri yayılım algoritmasını temel alır ve yalnızca mevcut iterasyondaki hataya odaklanır. Bu iki yöntemin performanslarının farklı ağ topolojilerinde nasıl değiştiği bu çalışmanın ana araştırma konusudur.

Özellikle farklı topolojilere sahip sinir ağlarının hatalarının grafiksel olarak karşılaştırılması, ağ tasarımı sırasında hangi topolojilerin daha verimli olduğunu görselleştirmek için önemlidir. Çalışmada, momentumlu ve momentumsuz eğitim modları için her biri 10 farklı topoloji denenmiş ve bu topolojilerin performans sonuçları analiz edilmiştir. Elde edilen hata oranları birer grafikle sunulmuş ve hangi topolojilerin daha başarılı olduğu belirlenmiştir.

Çalışmanın sonunda, momentumlu eğitimin genellikle daha iyi sonuç verdiği gözlemlenmiş ve bu bulgular detaylı bir şekilde raporlanmıştır. Yapılan analizler, yapay sinir ağları tasarımında topoloji seçiminin önemini vurgularken, momentum faktörünün performans üzerindeki etkisini de açıkça ortaya koymaktadır.

Bu raporda, öncelikle uygulamada kullanılan yöntemler ve algoritmalar açıklanacak, ardından deney sonuçları ve elde edilen grafikler detaylandırılacaktır. Son olarak, bulgular ışığında sonuçlar tartışılacak ve gelecekteki çalışmalar için öneriler sunulacaktır.

1. GELİŞTİRİLEN YAZILIM

Bu çalışmada, farklı yapay sinir ağı topolojilerinin performanslarını karşılaştırmak amacıyla momentumlu ve momentumsuz eğitim yaklaşımları kullanılmıştır. Bu yöntemlerin detayları ve uygulama adımları aşağıda açıklanmıştır.

1.1) Veri Seti

Veri seti üzerinde gerçekleştirilen işlemler:

1. **Veri Ayrımı**: Veri seti, eğitim ve test veri setleri olmak üzere ikiye bölünmüştür:
   Eğitim seti: %70
   Test seti: %30
2. **Veri Normalizasyonu**: Giriş değerleri [0,1][0, 1][0,1] aralığına ölçeklendirilmiştir. Bu işlem, öğrenme algoritmasının daha hızlı yakınsamasını sağlamıştır.

Kullanılan Yapılar:

- **Neuroph DataSet Sınıfı**: Neuroph kütüphanesinde yer alan DataSet sınıfı, veri setinin düzenlenmesi ve bölünmesi için kullanılmıştır.
- DataSetRow: Veri setindeki her bir veri örneğini temsil eden yapı.

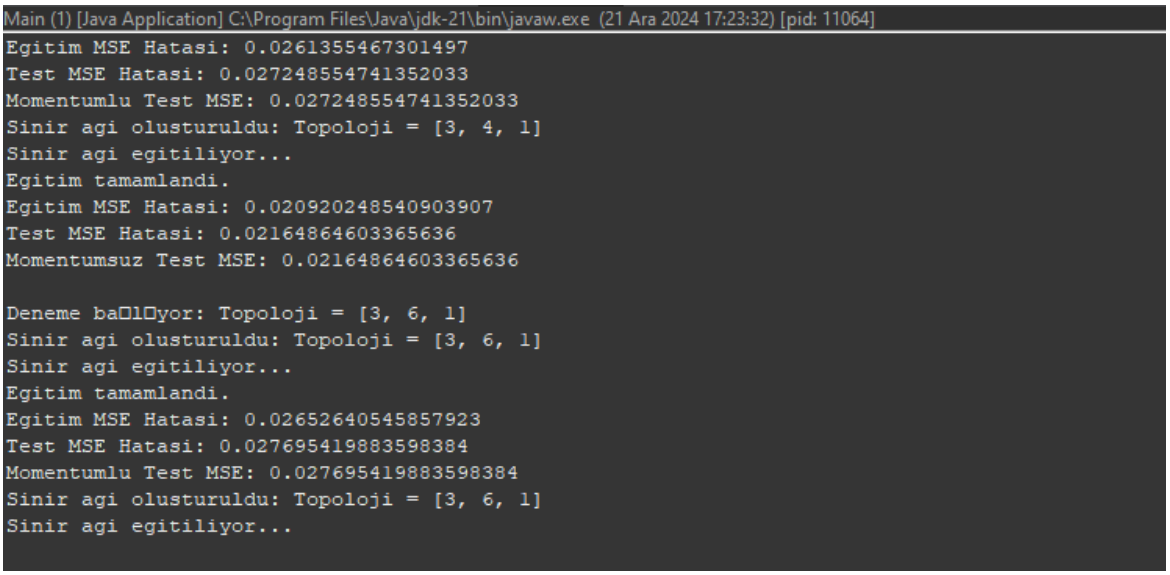
1.2) Sinir Ağı Topolojileri

10 farklı sinir ağı topolojisi tasarlanmış ve her biri test edilmiştir. Topolojiler şu parametrelerle tanımlanmıştır:

1. **Giriş Nöron Sayısı**: Veri setindeki özellik sayısı kadar giriş nöronu belirlenmiştir.
2. **Gizli Katmanlar**: Her topoloji farklı sayıda gizli katman ve nöron içermektedir:
   - Örneğin: 1 gizli katmanlı (5 nöronlu) ile 3 gizli katmanlı (10-15-5 nöronlu) topolojiler.
3. **Çıkış Katmanı**: Çıkış nöron sayısı, problem türüne göre belirlenmiştir (örneğin, sınıflandırma için 1 çıkış nöronu).

Kullanılan Yapılar:

- **Neuroph MultiLayerPerceptron Sınıfı**: Çok katmanlı ileri beslemeli yapay sinir ağlarını temsil etmek için kullanılmıştır.
- TransferFunctionType.SIGMOID: Gizli ve çıkış katmanlarındaki aktivasyon fonksiyonu olarak sigmoid kullanılmıştır.
- Sinir ağı ağırlıkları rastgele başlatılmış ve eğitim süreci boyunca güncellenmiştir



1.3)Momentumlu ve Momentumsuz Eğitim

Momentumlu Eğitim:
- Neuroph kütüphanesindeki MomentumBackpropagation sınıfı kullanılmıştır.
- Momentum, eğitim sürecinde öğrenme oranını iyileştirip daha hızlı yakınsama sağlamıştır.
- Momentum katsayısı (α\alphaα) varsayılan olarak 0.7 alınmıştır.

Momentumsuz Eğitim:
- Klasik geri yayılım algoritması uygulanmıştır. Bu yöntemde, yalnızca mevcut iterasyonun hatası dikkate alınarak ağırlıklar güncellenmiştir.
- Neuroph'un varsayılan geri yayılım algoritması olan BackPropagation sınıfı kullanılmıştır.

Kullanılan Yapılar:
- **MomentumBackpropagation ve BackPropagation**: Momentumlu ve momentumsuz öğrenme algoritmaları için Neuroph kütüphanesinden iki ayrı sınıf kullanılmıştır.
- Eğitim sırasında her epoch sonunda Mean Squared Error (MSE) değeri kaydedilmiştir.

1.4) Sonuçların Saklanması

Elde edilen sonuçlar şu şekilde saklanmıştır:
1. **Momentumlu Eğitim Sonuçları**: Her bir topoloji için ortalama hata oranı (Map<Integer, Double> kullanılarak) saklanmıştır.
2. **Momentumsuz Eğitim Sonuçları**: Benzer şekilde, momentumsuz eğitim için her topolojinin hata oranları ayrı bir yapıda tutulmuştur.

Kullanılan Yapılar:
- **LinkedHashMap**: Topolojilere karşılık gelen hata oranlarını düzenli bir sırada saklamak için kullanılmıştır.
    - **Anahtar**: Topoloji numarası.
    - **Değer**: İlgili topolojinin ortalama hata oranı.

1.5) Grafik Çizimi

Elde edilen sonuçlar görselleştirilerek momentumlu ve momentumsuz eğitim yaklaşımlarının karşılaştırılması yapılmıştır.(Topoloji-Hata grafiğini sonraki sayfalarda inceleyebilirsiniz.)

Grafik Türü: Çubuk grafik (Bar Chart) tercih edilmiştir.

- X ekseni: Topoloji numaraları.
- Y ekseni: Ortalama hata oranları.

Kütüphane: JFreeChart kütüphanesi kullanılmıştır.

- DefaultCategoryDataset: Veri setinin grafik için hazırlanmasında kullanılmıştır.
- ChartFactory: Çubuk grafik oluşturmak için kullanılmıştır.
- ChartPanel: Oluşturulan grafiğin kullanıcı arayüzünde gösterimi için kullanılmıştır.

1.6) Kod Yapısı

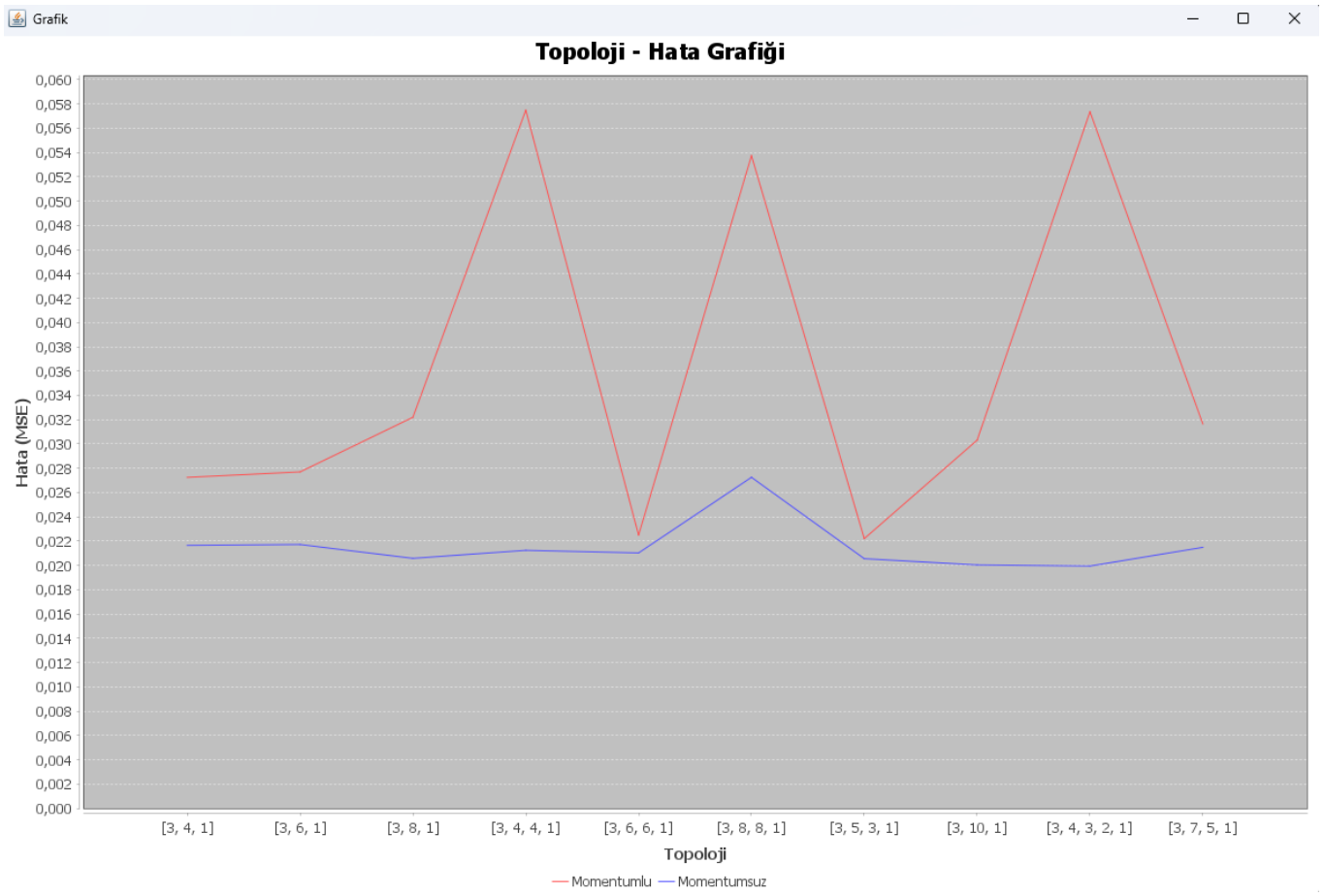
Programın temel işleyişi; veri işleme, momentumlu ve momentumsuz eğitim süreçlerini ayrı ayrı gerçekleştirme, farklı topolojilerle deneyler yapma ve sonuçların grafiksel analizini yapma üzerine kuruludur. NeuralNetworkProcess sınıfı, tüm bu süreçleri yöneten ana sınıftır. Bu sınıf, eğitim ve test veri setlerini saklamak için DataSet nesnelerini kullanır ve şu anda kullanılan sinir ağı modelini temsil eden bir MultiLayerPerceptron nesnesine sahiptir. Ayrıca, momentumlu ve momentumsuz eğitim sonuçlarını saklamak için Map<Integer, Double> veri yapıları kullanılmıştır; burada anahtar topolojiyi, değer ise bu topolojiyle elde edilen hata oranını temsil eder.

```
import org.jfree.chart.ChartFactory;

public class NeuralNetworkProcess {
    private DataSet trainingSet;
    private DataSet testSet;
    private MultiLayerPerceptron neuralNet;

    private Map<int[], Double> momentumResults = new LinkedHashMap<>();
    private Map<int[], Double> nonMomentumResults = new LinkedHashMap<>();

    private int[] bestMomentumTopology;
    private int[] bestNonMomentumTopology;

    public NeuralNetworkProcess(DataSet trainingSet, DataSet testSet) {
        this.trainingSet = trainingSet;
        this.testSet = testSet;
    }

    public void createNetwork(int... topology) {
        neuralNet = new MultiLayerPerceptron(TransferFunctionType.SIGMOID, topology);
        System.out.println("Sinir agi olusturuldu: Topoloji = " + Arrays.toString(topology));
    }

    public void trainNetwork(boolean useMomentum, double learningRate, double momentum, int maxIterations) {
        MomentumBackpropagation backPropagation = (MomentumBackpropagation) neuralNet.getLearningRule();
        backPropagation.setLearningRate(learningRate);
        backPropagation.setMomentum(useMomentum ? momentum : 0.0);
        backPropagation.setMaxIterations(maxIterations);

        try {
            System.out.println("Sinir agi egitiliyor...");
            neuralNet.learn(trainingSet);
            System.out.println("Egitim tamamlandi.");

            double trainingMSE = calculateMSE(trainingSet);
            System.out.println("Egitim MSE Hatasi: " + trainingMSE);

            double testMSE = calculateMSE(testSet);
            System.out.println("Test MSE Hatasi: " + testMSE);
        } catch (Exception e) {
            System.err.println("Egitim sirasinda hata olustu: " + e.getMessage());
        }
    }
```

Sinir ağlarını eğitmek için **trainWithMomentum** ve **trainWithoutMomentum** metodları bulunmaktadır. Bu metodlar, sırasıyla momentumlu ve momentumsuz geri yayılım algoritmalarıyla farklı topolojiler üzerinde eğitim yapar ve her bir topoloji için hata oranlarını ilgili veri yapılarına kaydeder. En iyi topolojiyi bulmak için kullanılan **findBestTopologies** metodu, momentumlu ve momentumsuz eğitim süreçlerinin sonuçlarını analiz eder ve hata oranı en düşük olan topolojileri belirler. Grafiksel analiz için **plotResults** metodu, JFreeChart kütüphanesini kullanarak momentumlu ve momentumsuz yöntemlerin sonuçlarını karşılaştıran çubuk grafikleri oluşturur. Grafik, her topolojinin hata oranlarını görsel olarak sunar ve kullanıcıya yöntemlerin performansını değerlendirme imkanı sağlar.

Farklı sinir ağı topolojilerinin oluşturulması, **TopologyGenerator** isimli yardımcı sınıf tarafından gerçekleştirilir. Bu sınıf, giriş ve çıkış nöron sayısına göre dinamik olarak çeşitli gizli katman kombinasyonları üretir.

Programda kullanılan Neuroph kütüphanesi, sinir ağlarının oluşturulması, eğitilmesi ve test edilmesi gibi temel işlemleri kolaylaştırırken; JFreeChart kütüphanesi, sonuçların grafiksel olarak analiz edilmesini sağlamıştır. Bu modüler kod yapısı, esnekliği sayesinde yeni yöntemlerin veya topolojilerin eklenmesine olanak tanır ve analiz süreçlerini daha etkili hale getirir.



Kaynakça

http://neuroph.sourceforge.net
https://neuroph.sourceforge.io/documentation.html
http://www.jfree.org/jfreechart/
https://www.jfree.org/jfreechart/guide.html
https://stackoverflow.com
https://sourceforge.net/p/neuroph/discussion/
https://github.com/jfree/jfreechart
https://github.com/neuroph