KARADENİZ TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Yapay Sinir Ağları Dersi Ödev Raporu

Adı SOYADI
Sefa Subaşı 401058

2023-2024 GÜZ DÖNEMİ

KARADENİZ TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

PROJENİN KONUSU

Tek ve çok katmanlı yapay sinir ağının modellenmesi.

2023-2024 GÜZ DÖNEMİ

ÖNSÖZ

Bu çalışmayı Karadeniz Teknik Üniversitesi Mühendislik Fakültesi bölümü olan Bilgisayar Mühendisliği, Yapay Sinir ağları dersinin ödevi için yapılmaktadır. Bu çalışmanın hayata geçirilmesi sürecinde bilgi ve tecrübelerinden faydalandığım değerli Prof. Dr. MURAT EKİNCİ hocamıza teşekkür ederim.

Adı SOYADI
Sefa Subaşı 401058
Trabzon 2023

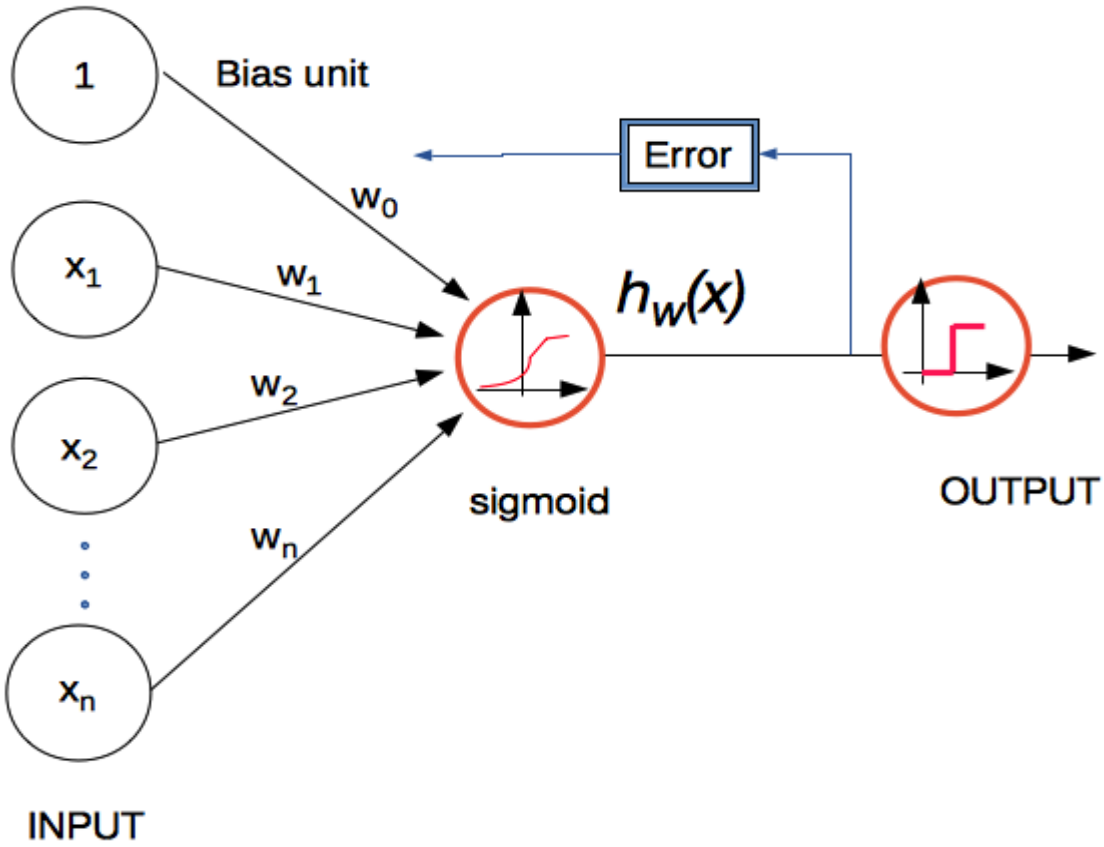
ÖZET

Bu raporda tek katmalı (perceptron) gözetimli (supervisor) yapay sinir ağının uygulamada nasıl modellendiği ve modelin koda nasıl uygulandığı anlatılmaktadır.

Tek Katmanlı Yapay Sinir Ağı (Perceptron)

GENEL MANTIK

Öncelikle 2 sınıfa ait girdiler (x1, x2, …., xn) kullanıcıdan istenir. Bu girdiler bizim giriş değerlerimizdir. Her girdi için bir ağırlık değerini programımız rasgele oluşturur. Daha sonra bu ağrılık değerlerini güncelleyerek adım adım eğitimi tamamlamış olacağız. İlk işlem olarak giriş değerleri ağırlıklarıyla çarpılır ve bias değeri eklenir. Tüm giriş değerleri için bu uygulanır ve çıkan sonuçlar toplanır. Daha sonra aktivasyon fonksiyonundan çıkan değer verilir. Bizim uygulamamız Supervisor(danışmanlı) eğitim olduğu için kullanıcıdan beklenen değer alınır. Bu beklenen değerden aktivasyon fonksiyonundan çıkan değer birbirinden çıkartılır. Eğer sıfır ise girdi doğru yerindedir. Fakat yanlış yerde ise buna göre ağırlıklar güncellenir. Bu şekilde girdi değerleri döngüye sokulur ve istenen sonuç elde edilene kadar ağırlık değerleri güncellenir. Bu güncellemeler sayesinde rasgele ağırlıklarla oluşan eğitim doğrumuz doğru yerine ulaşır ve eğitim tamamlanmış olur.



PROGRAMIN AKIŞI

0) Burada eğitim kodu başlamadan önce başlangıç değerleri ve sabitler verilmiştir. **epochs** değeri en fazla kaç adımda programın sonlandırılması için konulmuş bir limittir. **learningRate** öğrenme sabitidir. **error** değişkeni hatayı tutak için üretilmiştir. **totalerror** değişkeni toplam hatayı üzerinde tutar ve buna göre döngü sonladırılır.

1) Burada kullanıcıdan girilen örnek sayısı kontrol edilmektedir. 1 veya hiç örnek girmemesi durumunda uyarı verecektir.

2) Burada tüm örnekler **forward pass'e** sokulur. Her giriş bu döngüde işlenir ve bir sonuca götürülür.

3) Bu döngüde örnekler sırayla ağırlıklarıyla çarpılır ve **output** net toplam değerine eklenir. **Samples** tek boyutlu pointer mantığıyla oluşturulmuş bir dizidir. Burada tüm örnekler bulunmaktadır. Weights dizisinin içinde iste rasgele atanmış ağırlık değerleri bulunmaktadır.

4) Burada toplam **output** değerine **bias** değeri eklenir.

5) Çıkan **output** değeri aktivasyon fonksiyonuna sokulur ve **output** değeri güncellenir. Burada aktivasyon fonksiyonu olarak **Sigmoid** fonksiyonu kullanılmıştır.

6) Burada **error** değeri hesaplanır. Beklenen değerden çıktımız olana **output** değeri çıkartılır. Eğer **error** 0 ise eğitim tamamlanmıştır. Eğer 1 ise tekrardan ağrılıklar güncellenir.

7) **error** değeri **totalerror** değerine eklenir. Hata değerlerini toplamamızın sebebi diğer örnekleride eğitime dahil ederek tüm örneklere bakıldıktan sonra toplam hata değerine bakılır. Eğer toplam 0 dan büyükse ağırlıkları güncellemeye gideriz.

```
// Backward pass: update weights and bias
if(error!=0)
{
    for (int j = 0; j < inputDim; ++j) {
        Weights[j] += learningRate * error * Samples[i * inputDim + j];
    }
    bias[0] += learningRate * error;
}
chart1->Series["Series1"]->Points->AddXY(epoch, totalerror);
//Error Check
if (totalerror == 0)
{
    MessageBox::Show("Eğitim Tamamlandı...");
    break;
}
```

8

9

10

11

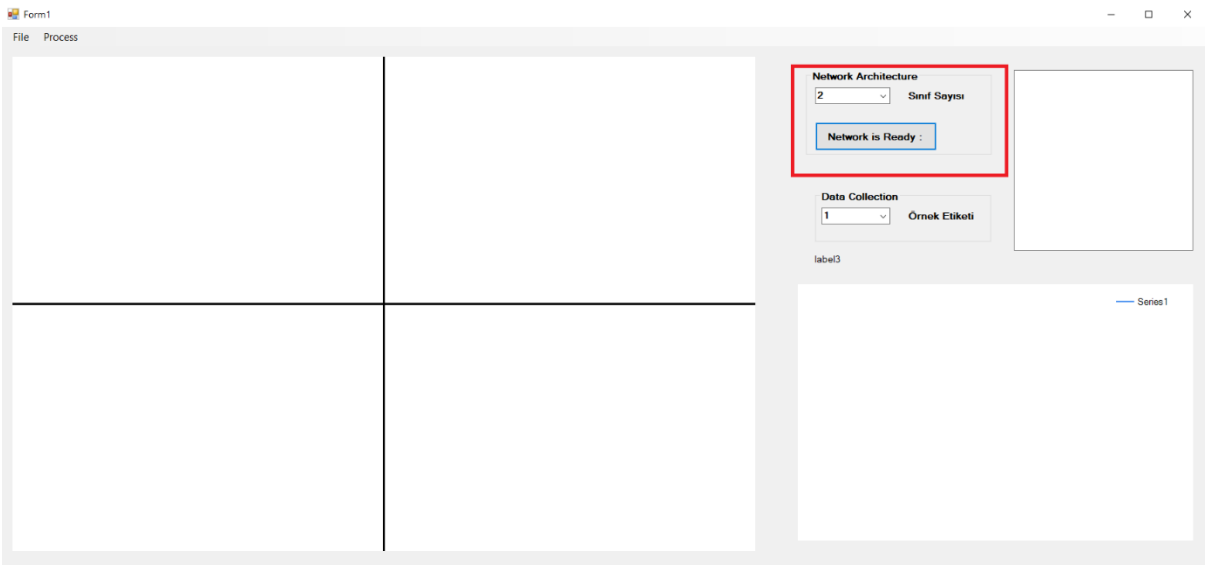
8) Eğer **error** değeri 0'dan farklı geldiyse **Backward pass** yapılır. Burada tüm ağırlıklar ve **bias** güncellenir. Ağırlık güncellemesi şu şekilde olur. Mevcut ağırlık değerinin üzerine öğrenme kat sayıyı, hata değeri ve örnek çarpılarak her ağırlık değeri güncellenir.

9) En sonunda **bias** da güncellenir. Bias'a öğrenme kat sayı ve hata çarpılarak bias'a eklenir.

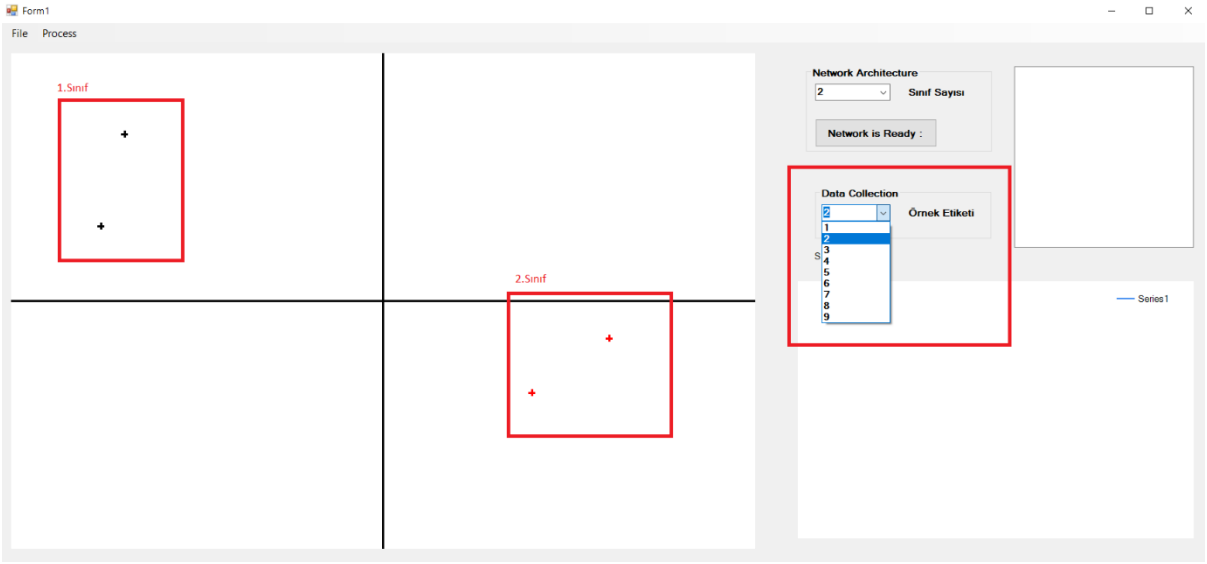
10) Grafiğimize her **epoch**'daki total **error** değeri eklenir.

11) Toplam hata değeri sıfıra ulaştığı zaman eğitim tamamlanmış olur ve beklenen değere ulaşılır. Daha sonra döngü kesilerek eğitim tamamlanır.

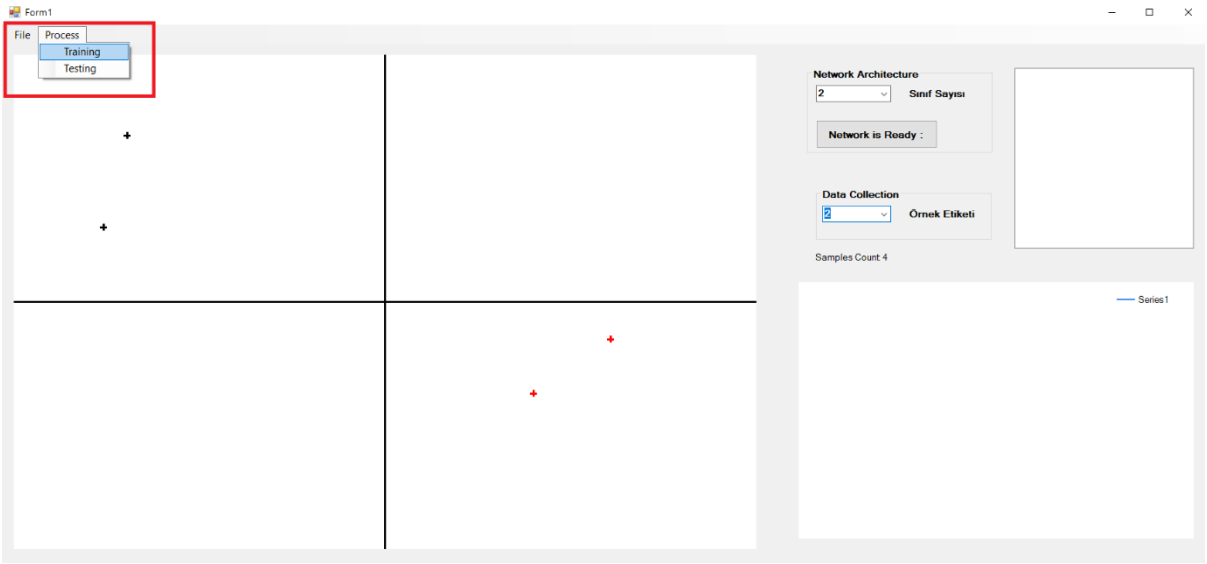
UYGULAMA



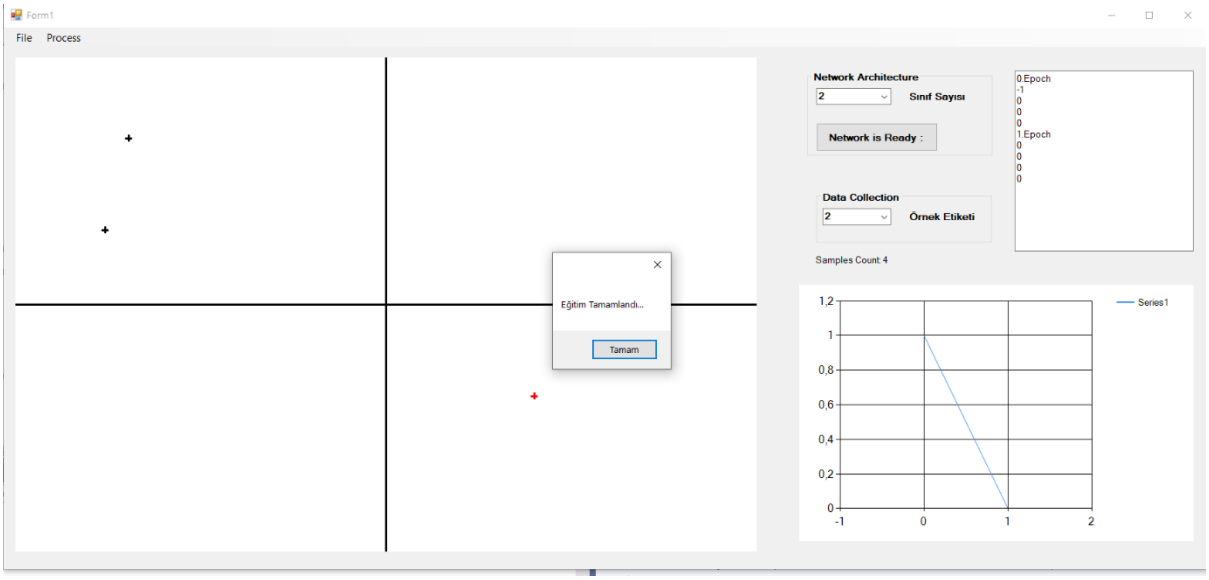
İlk olarak uygulamamız kaç sınıftan oluşacak o seçilir. Şu an bu program tek katmanlı perceptron olduğu için en fazla 2 sınıf seçilir. Sınıf sayısını kaydetmek için butona basılır.



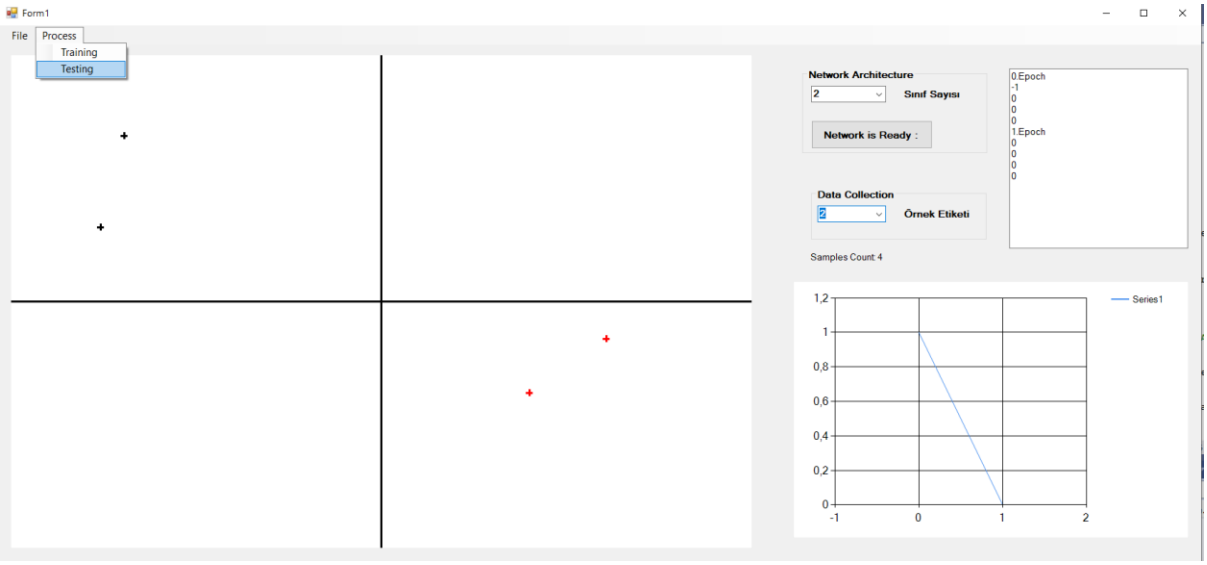
Her bir sınıf için sırayla örnekler eklenir.



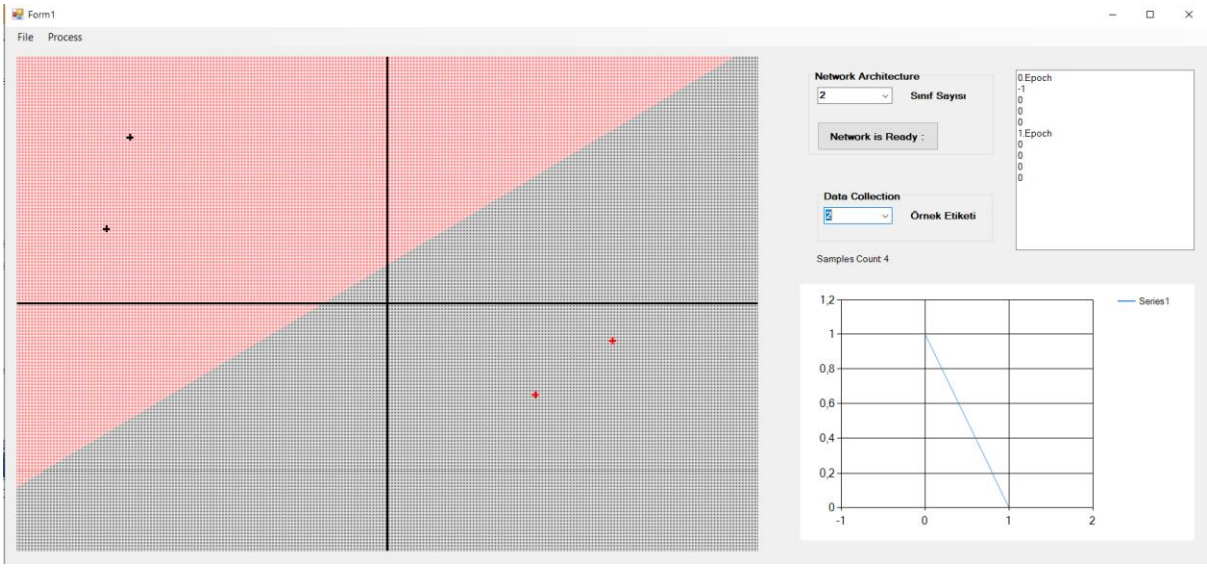
Buradan Training seçilerek uygulama eğitilir.



Eğitim Tamamlanmış olur. Burada kaç epoch da uygulamanın eğitildiği her örnek için error değeri sağ üst köşede listelenmiştir. Sağ alta ise hatanın nasıl bir ivmen ile azaldığı gözlemlenmektedir.



Eğitim tamamlandıktan sonra Testing'e basılır ve dağılım gözlemlenir.



Sonuç.

Kaynakça

Artifical Neural Systems - Jacek M. Zurada
https://mashimo.wordpress.com/tag/perceptron/
https://tr.newworldai.com/yapay-sinir-aglari-prof-dr-sadi-evren-seker/
https://www.youtube.com/watch?v=h1Y48mPXaEw
https://towardsdatascience.com/perceptron-learning-algorithm-d5db0deab975