YUSUF ERDOĞAN

140304010

EL YAZISI RAKAM TANIYAN YAPAY SİNİR AĞI

**Proje amacı:**

Mnist dosyaları\* kullanılarak 0-9 arası el yazısı rakamları tanıyan, test verileri üzerinde tanımlama yapabilen yapay sinir ağının matlab ortamında oluşturulması, eğitilmesi,test edilip belirli bir yüzdede başarı elde etmek amaçlanmıştır.

*Mnist dosyaları: 60000 farklı örnek resim ve etiketten oluşan ubyte tipinde dosya*

**Giriş:**

Yapay sinir ağları dersi kapsamında basit bir yapay sinir ağının nasıl oluşturulacağını, hangi algoritmaların kullanılabileceğini, her iterasyonda neler olduğunu teorik olarak öğrendik. Öğrendiklerimizi pekiştirmek ve mesleğimiz gereği bu teorik bilgilerin hayata nasıl geçirileceğini öğrenmek amacıyla daha önce "Yann LeCun","Corinna Cortes","Christopher J.C.Burges" tarafından 1998 yılında çözülmüş olan el yazısı rakamların ayırt edilmesi problemi üzerine bu projeyi ödev olarak aldık.

Yapay zekayı eğitmek için 60000 resim örneği ve bu resimlerin rakamsal değerlerini ifade eden etiket(label) dosyalarına yapay zekanın eğitilmesinde kullanmak sahibim.

Projenin çözmesi gereken sorun; 60000 resim dosyası ve 60000 etiketi öğrenip kendisine verilen 10000 farklı resmi sınıflandırmak. Sınıflandırma sonucunu bildirip yüzde kaç başarı ile sınıflandırma yaptığını kullanıcıya bildirmek.

**Yöntem:**

Öncelikle, MNIST dosyalarını matlab 2017 programına aktardım. Bu dosyaları okuma işlemini iki scripti kaynakta belirteceğim "Yann LeCun"'un sitesinden indirdim ve dosyaları okumak için bu scriptleri kullandım. Yapay sinir ağımın giriş parametrelerini resimler, çıkış parametresini ise etiketlerden oluşturduğum birim matrisler olarak ayarladım. Ağın hedef, gizli katman sayısı,döngü sayısı,eğitim fonksiyonları,aktivasyon fonksiyonlarını belirttim. Performans fonksiyonunu belirtip ağımı eğittim.

Test aşamasında test resimlerinin her elemanını matlab 2017 programının sim() fonksiyonu yardımıyla ağ içerisinde teste tabi tuttum. Çıktıların tüm değerini bir matrise aktarıp aynı değerleri her sütunun satırlarını tek tek alıp bire yakın olan değeri 1 gerisini sıfır kabul edip çok virgüllü sayıları satır sayısına göre 0-9 arası rakamlara çeviren matrise doldurdum. Bu matris ile elimde bulunan test etiketlerini karşılaştırıp başarı oranı,doğru ve yanlış tahmin sayılarını elde ettim.

**Verilerin Toplanması:**

MNIST dosyalarına ulaştığım adres:

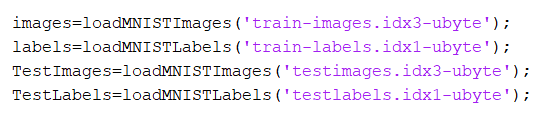
<http://yann.lecun.com/exdb/mnist/>

MNIST dosyalarını okumak için gerekli olan kütüphaneler:

<http://ufldl.stanford.edu/wiki/index.php/Using_the_MNIST_Dataset>

**Proje verileri:**

eğitim ve test verilerinin dosyalardan alınması işlemi;



loadMNISTImages() metodu dosyadan resimleri okuyor.

loadMNISTLabels() metodu dosyadan etiketleri okuyor.

**Ağın oluşturulması ;**



newff(): fonksiyonu yeni bir ileri beslemeli ağ oluşturuyor.

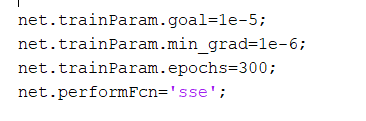
minmax(Giris): giriş matrisinin alt ve üst sınırlarını belirliyor.

[37,10] parametresi gizli katmanın sayısını ve içerdiği nöron sayısını belirtiyor.

{'logsig' ' logsig' }: aktivasyon fonksiyonları. Birinci parametre giriş ile gizli katman arasındaki aktivasyon fonksiyonu, ikinci parametre ise gizli katman ile çıkış katmanı arasındaki aktivasyon fonksiyonu.

trainscg : eğitim fonksiyonu.

Ağ Parametrelerinin verilmesi;



#trainParam# : parametre vermek için kullanılır.

.goal: ağın eğitim amacını belirtiyor. Ağ eğitimdeyken iterasyon sayısı bitmemişse ağ bu hedefe ulaşana kadar öğrenmeye devam eder.

.min\_grad: Eğitim eğrisinin hedef fonksiyona olan eğimini belirtir. Bu şart gerçekleştiğinde hedefe ulaşmış sayılır.

.epochs : ağın iterasyon sayısı.

.performFcn='sse': ağın performans fonksiyonunu belirtir.bir çok performans fonksiyonu vardır bu örnekte sse(sum square errors)'u kullandım.

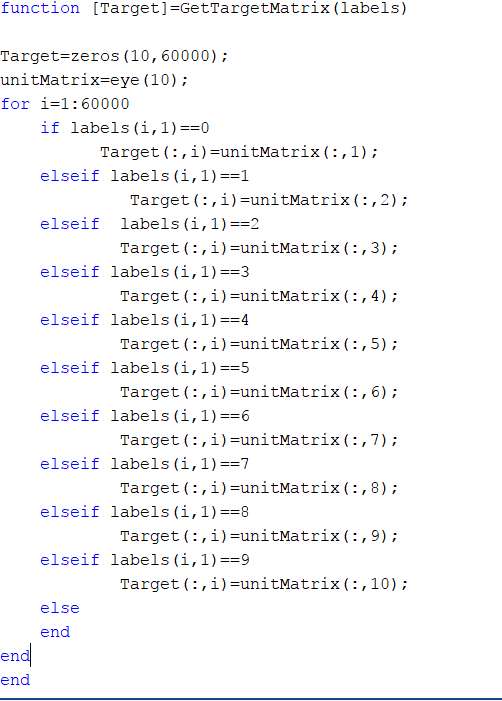
**Eğitim;**



net:Sinir ağının adını temsil eder.

Train: Matlab 2017 programının neural network tool eklentisinin ağ eğitim fonksiyonudur. Üç parametre ile eğittim. Ağın kendisi, giris matrisi(eğitim resimleri),hedef matris. Burada hedef matris decimal tabanda verildiği için bu matrisi kendi yazdığım script ile birim matrise çevirip parametre olarak yolladım.

Hedef matrisi birim matrise çevirme fonksiyonu;

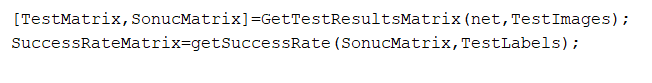


Bu fonksiyon daha önce dosyadan çektiğimiz etiket matrisini alıyor, boş bir hedef matrisi oluşturuyor,0-10 arası 10 rakam olduğu için 10x10 luk bir birim matris oluşturuyor ve etiket matrisinin o anki elemanına göre boş hedef matrisinin etiket matrisi ile aynı indexe sahip satırına birim matrisin o indexteki satırını yerleştiriyor.

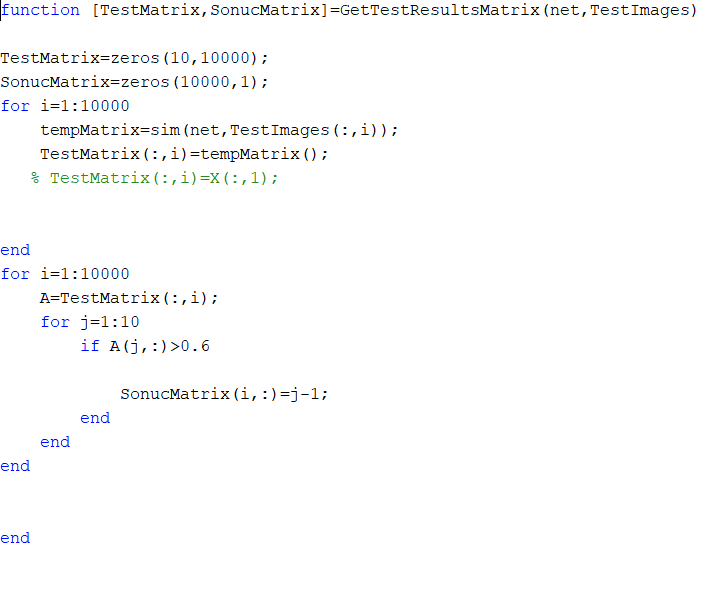
|  |  |
| --- | --- |
| 0 | [1 0 0 0 0 0 0 0 0 0] |
| 1 | [0 1 0 0 0 0 0 0 0 0] |
| 2 | [0 0 1 0 0 0 0 0 0 0] |
| 3 | [0 0 0 1 0 0 0 0 0 0] |
| 4 | [0 0 0 0 1 0 0 0 0 0] |
| 5 | [0 0 0 0 0 1 0 0 0 0] |
| 6 | [0 0 0 0 0 0 1 0 0 0] |
| 7 | [0 0 0 0 0 0 0 1 0 0] |
| 8 | [0 0 0 0 0 0 0 0 1 0] |
| 9 | [0 0 0 0 0 0 0 0 0 1] |

Bu çevirmeleri yukarıdaki tabloya göre yaptırdım.

Test aşaması;



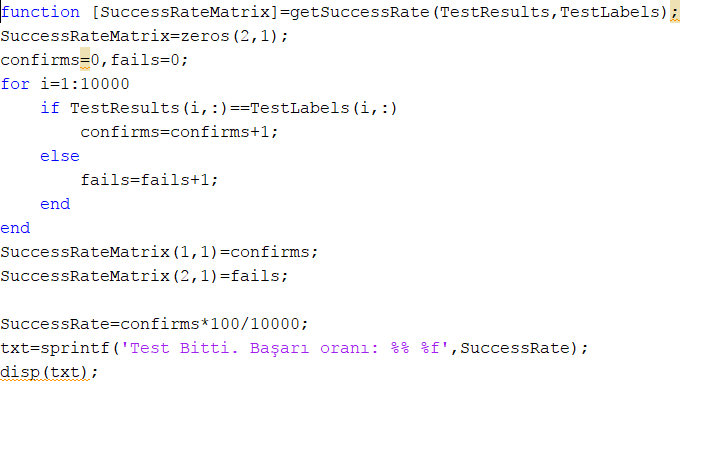
Yukarıdaki kod parçasını otomatik çalıştıran bir script yazdım. GetTestResultsMatrix() metodu ağ ve test resimleri olmak üzere iki parametre alıyor ve ağın verdiği tüm sonuçları TestMatrix ine yazıyor. Kendi yorumladığı değerleri ise SonucMatrix e yazıyor.



Test verisi boyutunda bir matris ve test etiketleri boyutunda bir matris oluşturdum. Geçici bir matrise sim() metodu ile ağda teste soktuğum test resimlerinin her bir elemanının değerlerini atıp bu değerleri TestMatrix te biriktirdim. İkinci for da ise TestMatrix te biriktirdiğim bu değerlerin her satırını 0.6 ile karşılaştırdım . Bu değerden büyük olan döngü değerinin bir eksiğini sayı olarak kabul ettim. Böylece elimde hem ham sonuçların olduğu hem de yorumlanmış sonuçların olduğu iki matris oldu.

Başarı oranının hesaplanması;

Elimde bulunan yorumlanmış değerleri test verilerinin etiketleriyle karşılaştırdım.



Yukardaki fonksiyonun döndüğü SuccessRateMatrix doğru ve yanlış sayısını tutan 2,1 boyutunda bir matris. Tüm yorumlanmış verilerle elimde bulunan etiket verilerini karşılaştırdım. Doğru ve yanlış sayılarını buldum. Bu sonuçları SuccessRateMatrix e attım. Daha sonra ise yüzdelik başarıyı hesaplayıp kullanıcıya bildirdim.

Sonuçlar:

mse( mean square error) ve sse(sum square error) performans fonksiyonlarının karşılaştırılması

goal: 1e-5

Epoch:300

Train function: trainscg

Layer: [66 10]

|  |  |  |
| --- | --- | --- |
|  | sse | mse |
| Süre: | 03:25 | 03:06 |
| Başarı: | %94.8800 | %58 |

Öğrenme fonksiyonlarının karşılaştırılması:

Transfer function: logsig,logsig

goal: 1e-5

epoch:300

layer: [66 10]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | trainscg | trainrp | traincgb | traincgp |
| Süre: | 03:20 | 01:55 | 06:45 | 07:37 |
| Başarı: | %95.0400 | %93.320 | %94.88 | %39.4100 |

Gizli katman nöron sayılarının karşılaştırılması:

Performans function: sse

Activation function: logsig,logsig

Train function: trainscg

Epochs:300

goal:1e-5

|  |  |  |  |
| --- | --- | --- | --- |
|  | [37 10] | [66 10] | [100 10] |
| Süre: | 02:27 | 03:26 | 04:42 |
| Başarı: | %93.650 | %94.890 | %95.49 |

Aktivasyon fonksiyonlarının karşılaştırılması:

Goal: 1e-5

epochs:300

Train function: trainscg

labels: [37 10]

performance function: sse

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | logsig-logsig | tansig-tansig | purelin-purelin | tansig-logsig | purelin-logsig | purelin-tansig |
| Süre: | 02:29 | 02:21 | 02:07 | 02:20 | 02:14 | 02:09 |
| Başarı: | %93.67 | %87.730 | %40.950 | %93.270 | %37.620 | %10.860 |

Sonuç:

Proje boyunca bir sinir ağının nasıl eğitileceğini, nasıl test edileceğini ve nasıl programlanabileceğini öğrendim. %90 üzerinde bir doğrulukla çalışan yapay sinir ağı programladım ve istenenin üzerinde bir performans veriyor. Gözlemlediğim kadarıyla Epoch sayısını arttırmak bir yerden sonra işe yaramıyor. Goal ve min\_grad doğru orantılı ve her zaman hedefe ulaşamıyoruz. Aktivasyon fonksiyonlarından logsig-logsig i kullanmak bu projede en mantıklısı. Hedefi 1e-5 te tutmak %95 gibi bir doğruluk oranı için yeterli.Öğrenme fonksiyonlarının hepsini kullanamıyoruz bu projede. Örneğin trainlr hata veriyor çalışmıyor. En verimli öğrenme fonksiyonu bu projede trainscg ve en hızlı-verimlisi trainrp. Gizli katman nöron sayısını optimum belirlemek için örnek sayısı / (giriş katmanı+çıkış katmanı)\*alpha fonksiyonunda alphaya 1-10 arası deneme değerler vererek hesapladığımda internetten edindiğim bilgiler doğrultusunda alphaya 2 verip hesaplamanın katmandaki nöron sayısını azaltıp karmaşıklığı önlediğini gözlemledim.

Kaynak:

<https://www.mathworks.com/help/nnet/ug/multilayer-neural-network-architecture.html>

<https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw/1097>

<https://www.mathworks.com/help/nnet/ug/train-and-apply-multilayer-neural-networks.html>

<http://ufldl.stanford.edu/wiki/index.php/Using_the_MNIST_Dataset>

<http://yann.lecun.com/exdb/mnist/>