# #Matlab ile Yapay Sinir Ağları #

- Bir excel var. Excelde yaş-gelir-cinsiyet-telefon fatura bilgileri var. Veriler tamamen rastsal. Veriyi önce matlaba aktaracağız ve ardından işleyeceğiz

- Veri = xlsread('veri.xlsx'); // excel dosyasındaki verileri çeker.
  ||
  xlsread('veri.xlsx', 'Sayfa1');
  ⇓
  cinsiyet verileri NaN olarak görünür.
  Kategorik olan cinsiyet verisini rakam haline dönüştürelim.

⇒ EĞER(C2="F"; 1; 0) # F ise 1 değilse 0 yapacak.
EĞER(C2="M"; 1; 0) # M ise 1 " 0 yapacak.
} Exceli değiştiriyoruz matlabe sayısal veri vermeliyiz.

Kategorik değişkenlerde, N tane seçenek varsa N-1 sütun çevirmek yeterlidir.

- input = veri(:, 1:3); // tüm satırları al, 1,2,3. sütunu al.
  output = veri(:, 4); // Sadece 4. sütun ve tüm satırlar.
  ||  └→ (Tahmin edilmesi istenen sütun burası)
  ouput = veri(:, end);

→ 2 şekilde NN çalıştıracağız = ① NN toolbox

Start → Toolboxes → Neural Network → Neural Fitting Tool (nftool)

Next → Adjust Network size → Number of hidden Neurons = 25 → Next → Train → Next → Plot Regression, Plot Error Hist, Plot Fit → Retrain → Next → Next → Save Results → Finish
(Nelerl kayıt edecekseniz onları seçiniz)

Finish → (output'a gerçek değerler, output1 e tahmin edilen değerler)

Next → input = input, target = output (workspace değişkenleri)

Samples are = Matrix rows (1 verim satırlarda)

Next → Training = %70, Validation = %20, Testing = %10 } Değiştirebilirsiniz

Next → Number of hidden Neurons = 15 → Neural Network Start → Next → Next → (input-output curve fitting)

(input 3 yaş gelir cinsiyet) → Sigmoid (15 hidden layer) → ... → (output 1) (1 output layer)

Levenberg-Marquardt (Back propagation türü)

(R = Gerçek ile tahmin arasındaki korelasyon) [-1, 1] (1'e ne kadar yakınsa o kadar iyi) (R²'nin 1'e yakın olması önemli.)
(MSE = Tahmin ile gerçek arası fark) (e^-1 = 10^-1, e^-3 = 10^-3)
eğri sola yatmış, eğri sağa yatmış

YSA-1

- output1 = output1'; // değişkenin Transpozunu aldık.
- çıktılar = [output output1]; // 1. sütunda gerçek değerler, 2. sütunda tahmin değerleri.

Training, Validation, Testing hepsi var.

- r = corr(output, output1); // output ile output1 arasındaki korelasyonu hesaplar.

workspace'i save ettik ve "network1.mat" dosyasını oluşturduk.

## ② NN Kodlama;

- NN toolbox'da sadece 1 tane hidden layer vardı, ayrıca back propagation tek seçenekti, learning rate görmedik... Bunları kodlama ile yapacağız.

- input = veri(:, 1:3); // NN girdisi.
  target = veri(:, end); // Gerçekleşen değerler NN çıktısını karşılaştıracağız.
  (Veriyi Ayırdık)

→ current folder → New File → Function → neuralnetwork.m → ok
(2 hidden layer olacak şekilde fonksiyonu yazalım.)

- Her function, end ile biter. (Sol taraf = çıktı değişkenleri; Sağ taraf = girdi)
- = 'den sonra fonk. adı yazılır. Fonk. adı ile dosya adı aynı olmalıdır.
- Dosya isminin yanındaki * işareti değişiklik yapıldığını ve kayıt edilmediğini gösterir.
- % işareti yorum satırı anlamına gelir.
- [ ] varsa virgüle gerek yok, ( ) varsa virgüle gerek var.
- Girdi — Training, Validation
  Çıktı — Training, Validation
  → NN oluştur. → Training girdi ve çıktılarını kullanarak ağı eğit. → Eğitilmiş NN.

Validation girdi → Ağa ver → Eğitilmiş Ağdan çıkan çıktıları al. → Gerçek çıktılar ile karşılaştır.

- %70 training, %30 validation olarak ayıracağız.

- noofdata = size(input, 1); // inputun satır sayısını alır. 2 dersek sütun sayısını alır.
  ntd = noofdata * training_rate; // number Of Training Data // (training_rate = [0,1])
  ntd = round(noofdata * training_rate); // büyüklü değerler gelmesin diye. (yuvarla) → 38.5 → 39
  0,65 0,75 0,85 ... (55 * 0.7 = 38.5)

  xt = input(1:ntd, :); // training girdisi.
  xv = input(ntd+1:end, :); // validation girdisi

- Gerçeklesen satisida training ve testing diye ayırmalıyız.

- `yt = target(1:ntd);` // training için gerçeklesen data.
  `yv = target(ntd+1:end);` // validation için gerçeklesen data. } Sonuç verileri.

  - Target sadece 1 adet sutündan olusuyor.
  - NN çıktılarını yt ve yv ile karsılastıracak

10. Ders: → Veriler satır bazında yazılmış. Bundan dolayı transpozunu alarak verileri sutün bazlı hale getiriyoruz.

- `Xt = Xt';`
  `Xv = Xv';`
  `yt = yt';`
  `yv = yv';`
  } Transpozlarını Aldık. NN'ke sokmadan önce.

→ 3 adet veri (yas, maas, cinsiyet)'den, faturayı tahmin etmek istiyoruz
Verilerin hepsi rakam ama sok farklı. Bundan dolayı normalize ediyoruz. (0-1 arasına sekiyoruz.)

- `Xtn = mapminmax(Xt)`
  `Xvn = mapminmax(Xv)`
  ~~`ytn = mapminmax(yt)`~~
  ~~`yvn = mapminmax(yv)`~~
  } [-1,+1] arası normalize işlemi yaptık.

$\frac{1}{1+e^{-x}}$ = sigmoid fonk. [0,1]

→ `ytn = mapminmax(yt);` // Normalize edilmis training output (target) verisi.

→ `[ytn, PS] = mapminmax(yt);` // PS: normalizasyonun nasıl yapıldığı ile alakalı bilgiyi tutar.

→ Ağ bilgilerinin saklanacağı değişkeni olusturalım: (n1 = 1. katman nöron sayısı. n2 = 2. katma " ".)

Ağı Olusturduk

```
net = newff(Xtn, ytn, [n1, n2], {'tansig','purelin'}, 'trainlm');
net.trainParam.lr = Lrate;
net.trainParam.epochs = 10000;
net.trainParam.goal = 1e-20; // Belli bir düzeye ulasınca epoch sayısını durdur demek (10^-20)
net.trainParam.show = NaN; //Sonuçları bitince göster.
```

(n1: 2 hidden layer) (tansig: sigmoid algılar) (trainlm: levenberg back probagation)

`net = train(net, Xtn, ytn)` // Ağı Eğitiyoruz, eğitilen ağı net ağının üzerine yazdık.
(net: eğitilen ağ adı, Xtn: eğitim girdisi, ytn: eğitim çıktısı)

~~`yen = sim(net, Xvn);`~~ // eğitilmis net ağını kullanır girdi olarak Xvn kullan yen çıktısı al.

↳ Denormalization yaparak, yv ile karsılastırmak gerekir.

`ye = mapminmax('reverse', yen, PS);` //yen'i denormalize ediyoruz. PS'yi normalize ederken kaydettik.

14/3 - Veriler çok rastsal ve az veri olduğu için sonuçlar çok iç açıcı gelmedi. Performansın en iyi bunu olduğunu da bilmiyoruz. Performansını ölçmeniz gerekiyor

- ye = ye'; // NN çıktısı
  yv = yv'; // Gerçekleşen çıktı
  ) Transpozlarını Aldık.

## #Performans Ölçütleri#

① MPE - Mean Percentage Error:

(Gerçekleşenin kaç katı tahmin edilmiş)

$$\sum \frac{(\text{Tahmin} - \text{Gerçekleşen})}{\text{Gerçekleşen}} = \sum \frac{ye - yv}{yv}$$

ortalama da olabilir.

② MAPE - Mean Absolute Percentage Error:

(Hataların mutlak değerlerini alıyoruz. Yüzde veriyor.)

$$\text{ortalama}\left(\frac{|\text{Tahmin} - \text{Gerçekleşen}|}{\text{Gerçekleşen}}\right) = \text{ortalama}\left(\frac{|ye - yv|}{yv}\right)$$

③ RMSE - Root Mean Square Error:

(Karelerini alıyoruz. Kaçık hataları daha iyi görüyoruz) Birim Verir.

$$\sqrt{\text{ortalama}(\text{Tahmin} - \text{Gerçekleşen})^2} = \sqrt{\text{ortalama}(ye - yv)^2}$$

④ $R^2$ :

(Korelasyona yakın bir değer üretir.)

↓

[0,1] arası bir değer gelmesini bekleriz.

- SSerror = Hatanın Sapması
- SStotal = Gerçekleşenin Sapması

$$R2 = 1 - SSerror/SStotal$$

$\sum (\text{Gerçekleşen} - (\text{Gerçekleşen})_{ortalama})^2 \rightarrow SStotal$

$\sum (\text{Tahmin} - \text{Gerçek})^2 \rightarrow SSerror$

- MAPE = mean((abs(ye - yv)./yv)); // Nokta koymamızın sebebi matris elemanlarını 1'e bölmeye çalışmamız.
  ↳ Bunun 0'a yaklaşması önemli.
- // SStotal: gerçekleşen sapma toplamı
  // SSerror: hatanın sapma toplamı.

```
SStotal = sum((yv-mean(yv)).^2); // Eleman eleman kare aldırmak için . koyduk
SSerror = sum((ye-yv).^2);
R2 = 1 - SSerror/SStotal;
```

↳ Genelde ölçümler için R2 sonucunu kullanırız.

↳ 1'e yaklaşması önemli.

YSA-4