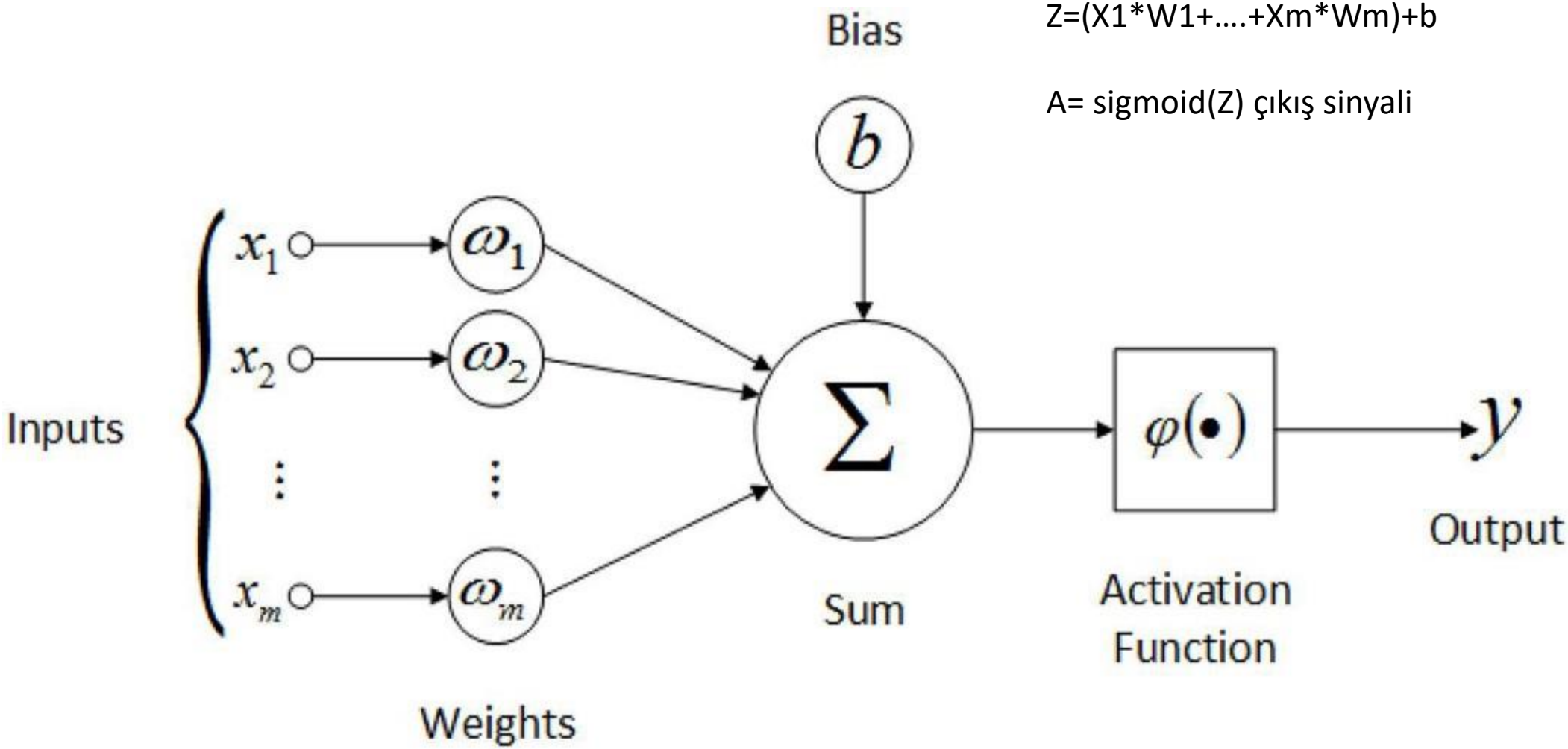
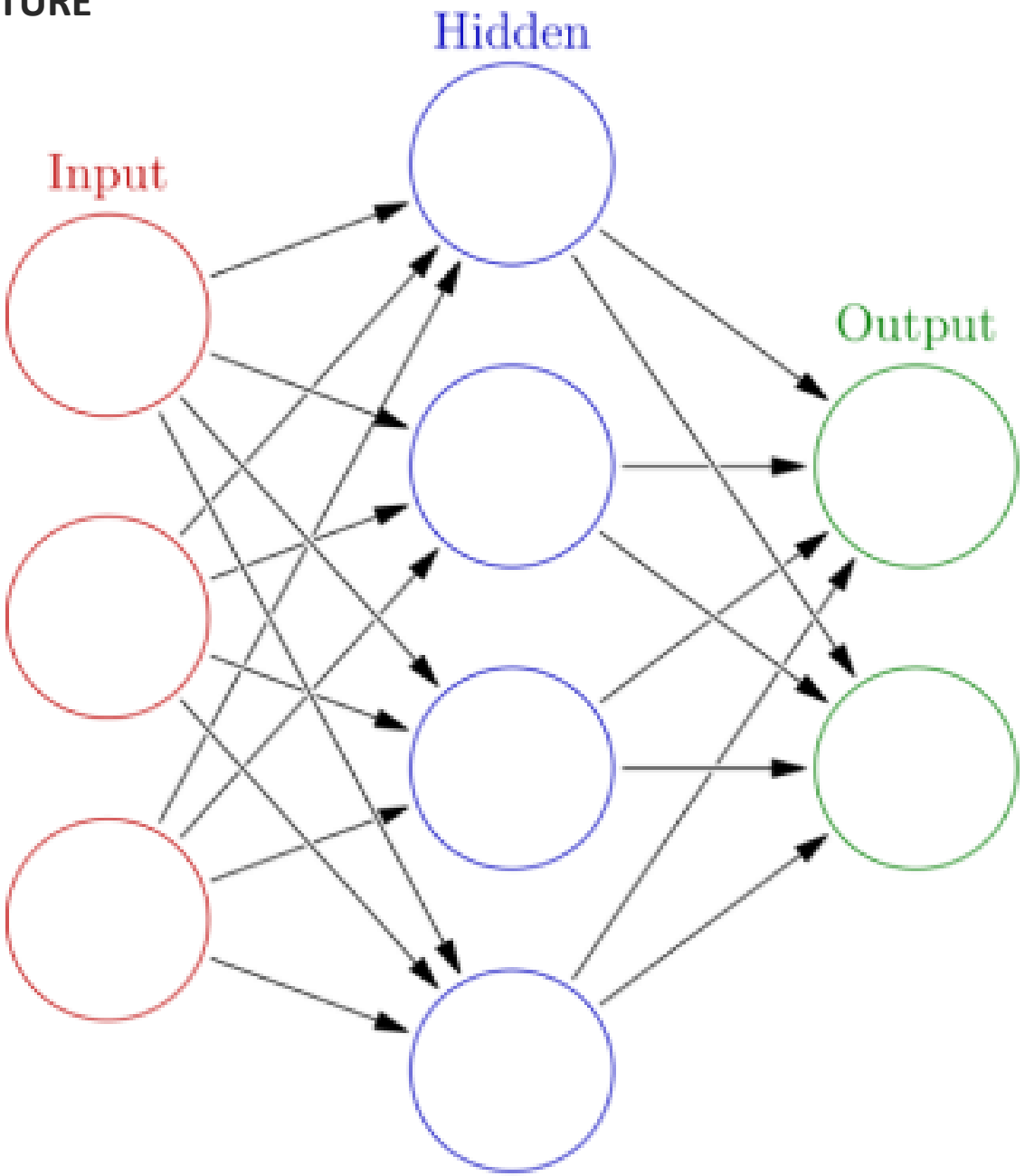
ARTIFICIAL NEURONS

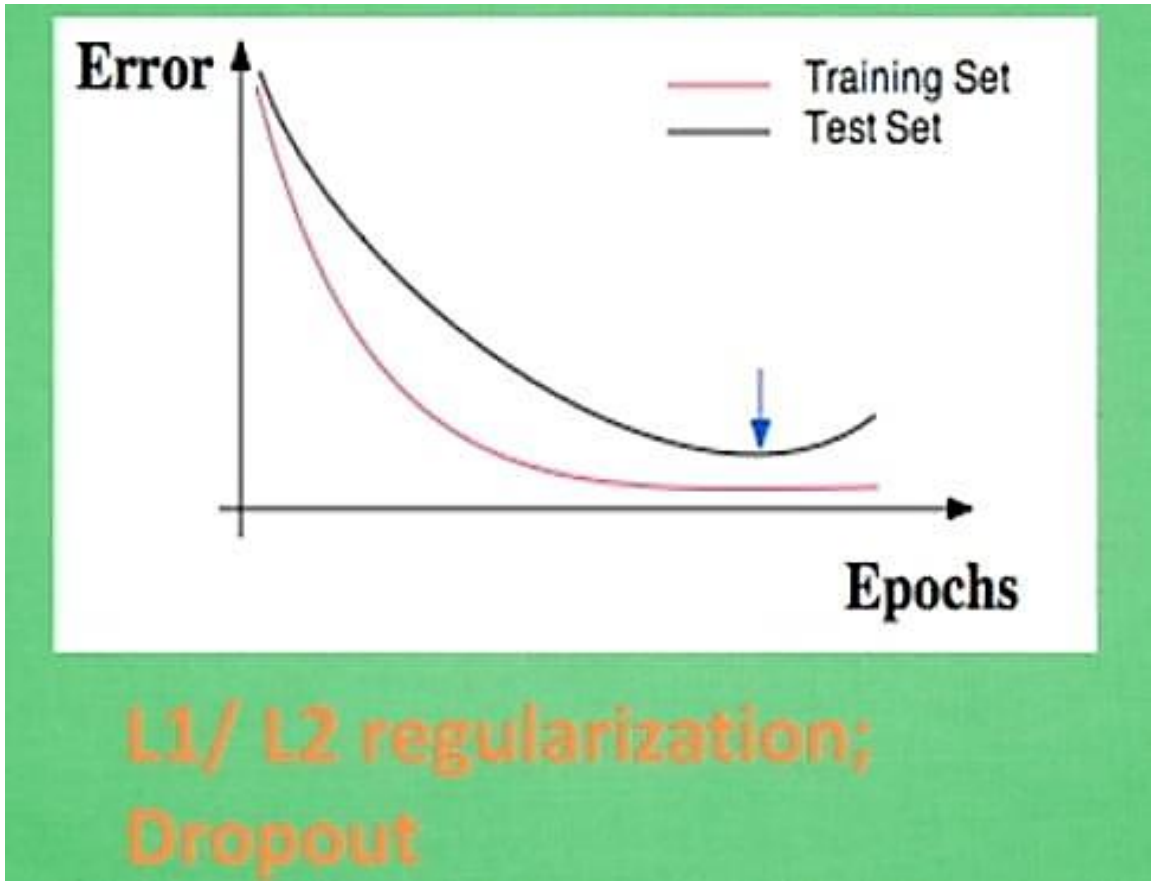
Z=(X1*W1+….+Xm*Wm)+b

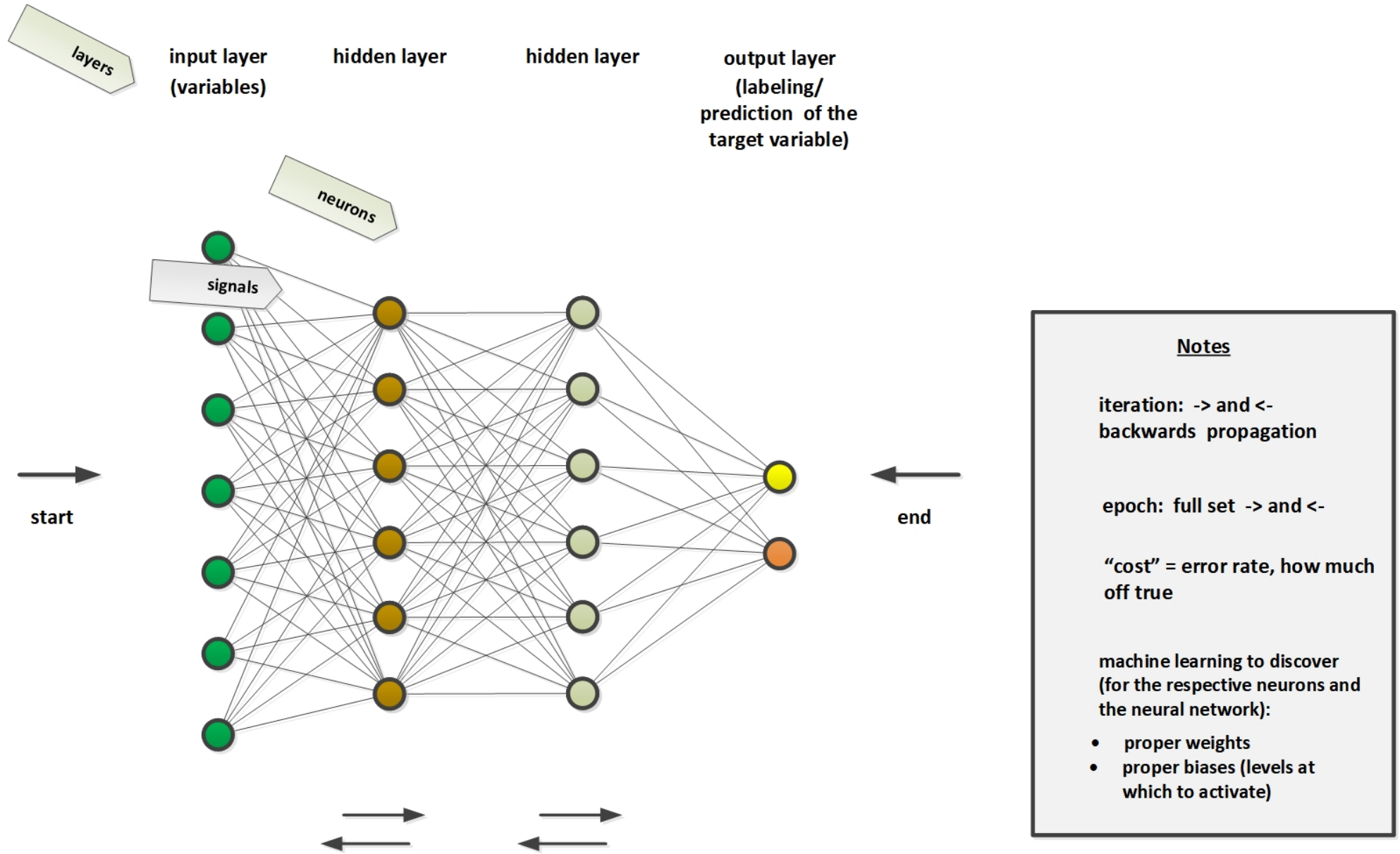
A= sigmoid(Z) çıkış sinyali



NEURAL NETWORK STRUCTURE




Error
Training Set
Test Set
Epochs
L1/ L2 regularization;
Dropout



Notes

iteration: -> and <-
backwards propagation

epoch: full set -> and <-

"cost" = error rate, how much off true

machine learning to discover (for the respective neurons and the neural network):

- proper weights
- proper biases (levels at which to activate)

An Artificial Neural Network (Simple Visual)

LOSS (COST) FONKSİYONU

Loss fonksiyonu, tahmin edilen değerin, gerçek değerinden ne kadar uzak olduğunu hesaplar. Amacımız bu değeri sıfıra yaklaştırmaktır. Ardından optimizasyon ile model kendini günceller.

OPTIMIZASYON

Loss değerinin en düşük olduğu noktayı bulmak. Bu durum modelin tüm inputları işleyip, doğru outputları ürettiği anlamına gelir. Loss ne kadar düşükse, output değerlerimizin büyük çoğunluğu da doğru olur.

Optimizasyon ile loss değerini olabildiğince düşürmeye çalışacağız. Burada en önemli parametre öğrenme oranı (learning rate)'dir. Learning rate'e çok küçük bir sayı atanırsa adım adım her değer kontrol edileceği için loss değeri düşürülebilse de çok zaman alacaktır. Aksine learning rate'e çok yüksek bir değer atanırsa da loss'un minimum olduğu değere erişilemeyebilir. Kısacası learning rate'in optimum değerde olması çok önemlidir.

Çok çeşitli optimizasyon algoritması bulunmaktadır ancak ADAM algoritması özellikle tavsiye edilir. Bu algoritmaların matematiksel derinliğine girmeyeceğim. Ancak tensorflow kütüphanesinde mevcut olduklarını biliniz.

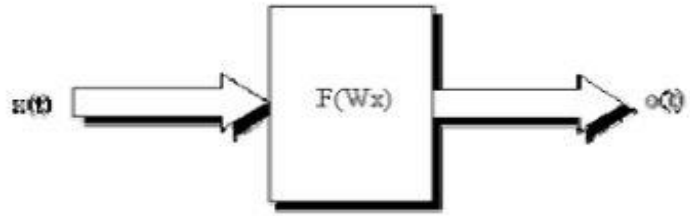
Yapay Sinir Ağları 2’e ayrılır:

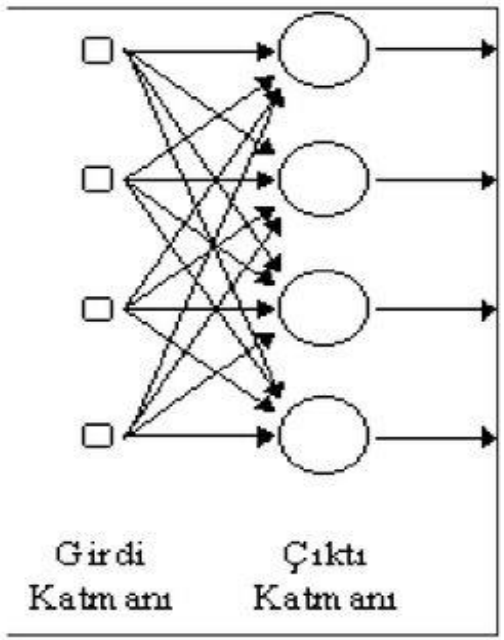
1.İleri Beslemeli Ağlar: İleri beslemeli yapay sinir ağlarında aktarılan bilgiler sadece ileri doğru hareket eder yani girişten çıkışa doğru bir hareket söz konusudur. Bir katmandaki nöronlar sadece kendinen sonra gelen katmana gider, ardından sonuç(çıkış) katmanına geçer. İleri beslemeli yapay sinir ağları, giriş katmanındayken dışarıdan gelen bilgilere herhangi bir değişiklik yapmadan bir sonraki katmana aktarır. Bu süreç içinde bilgi, orta ve çıkış katmanında fonksiyonlarla belli bir ağırlığa sahip olur ve çıkış nöronuna yönlendirir.

2.Geri Beslemeli Yapay Sinir Ağları: En az bir hücrenin çıkışı, diğer herhangi bir hücreye giriş olarak verilir bundan dolayı genellikle geri beslemeli yapay sinir ağlarında bir geciktirme eleman üzerinden yapılır. Besleme işlemi bir katmandaki hücreler arasında olmayabilir, bu sebeple doğrusal ilişkinin varlığından söz edilemez. Bu sebeple yapay sinir ağlarının geri beslemenin yapısı veri setine göre değişkenlik gösterebilir.

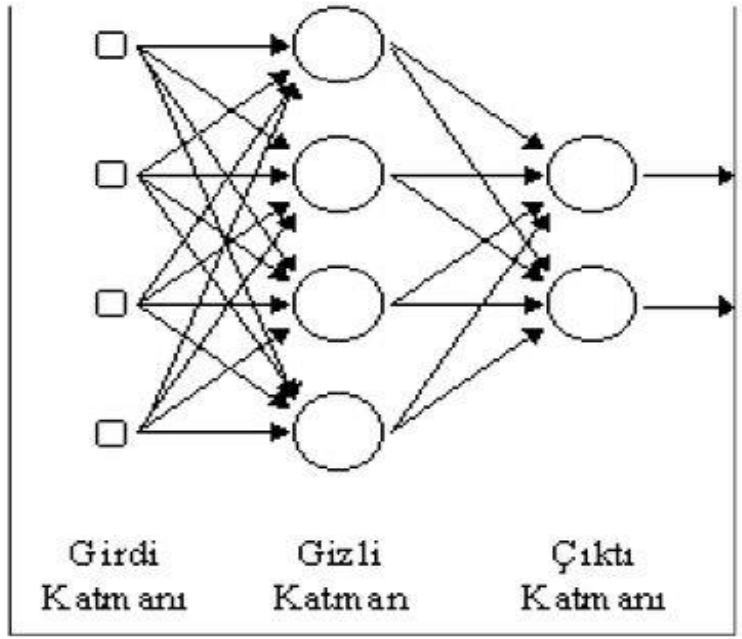
YSA'ların Yapılarına Göre Sınıflandırılması

- **İleri Beslemeli Ağlar(Feedforward neural nets) :**





Tek katmanlı ileri beslemeli ağ modeli

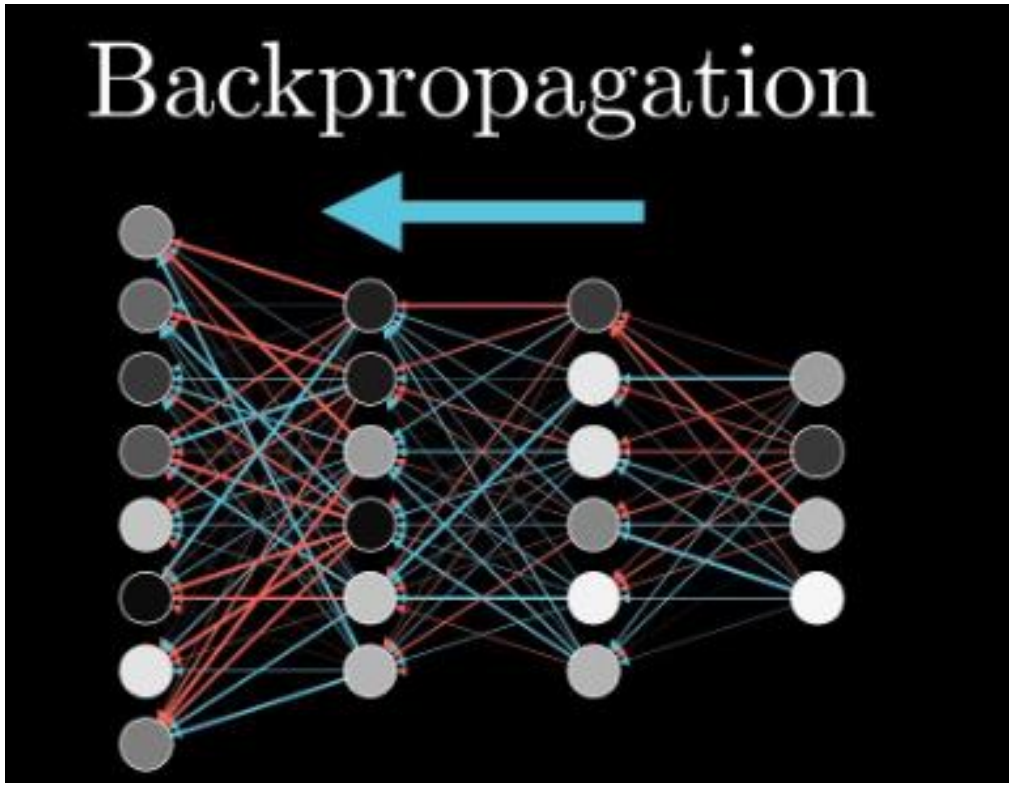


Çok katmanlı ileri beslemeli ağ modeli

BACKPROPAGATION

İlk önce weights ve bias değerlerine rastgele değerler atanır. Backpropagation metodu ile modelimiz weights ve bias değerlerini daha doğru hesaplayacaktır. Aynı şekilde bu metot da tensorflow kütüphanesinde mevcuttur.

Geri yayılım ile amacımız ağdaki ağırlıkların her birini güncellemek ve bu sayede gerçek çıktının hedef çıktıya daha yakın olmasına neden olmasını sağlamaktır. Bu amaç icin Gradient Descent (azalma) optimizasyon algoritması kullanılır. Bu algoritma denklemin local minimuna yakınsamak için birinci türev kullanır. Hata fonksiyonunun lokal minimumunu bu algoritmayı kullanarak hesaplayacağız.



2 gizli katmanlı birinci modelimiz oluşturalım.

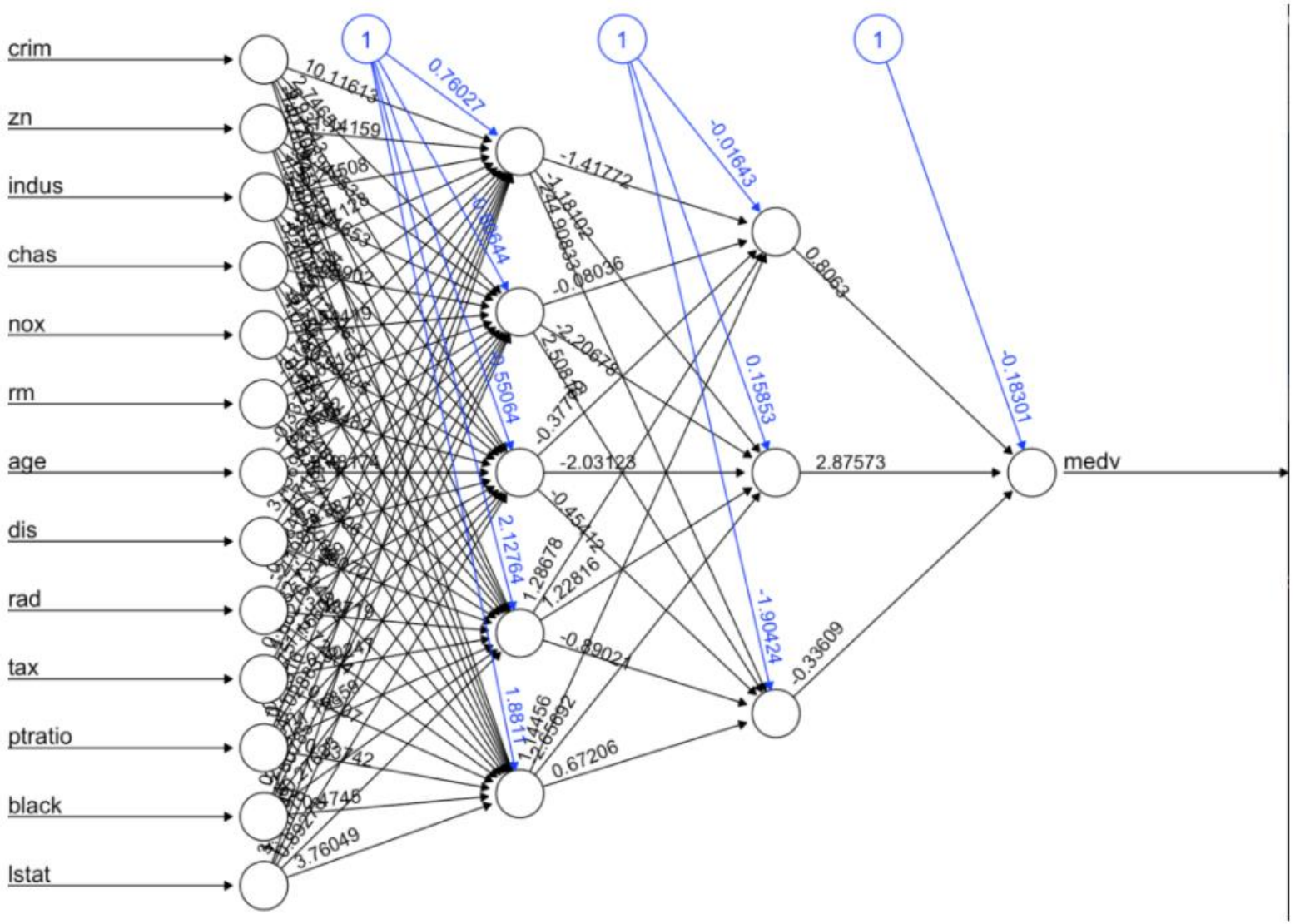
```
model_1_nn_5_3 <- neuralnet(duzeltilmis_veri
                , data=egitim_verisi
                , hidden=c(5,3)
                , linear.output=TRUE)
```

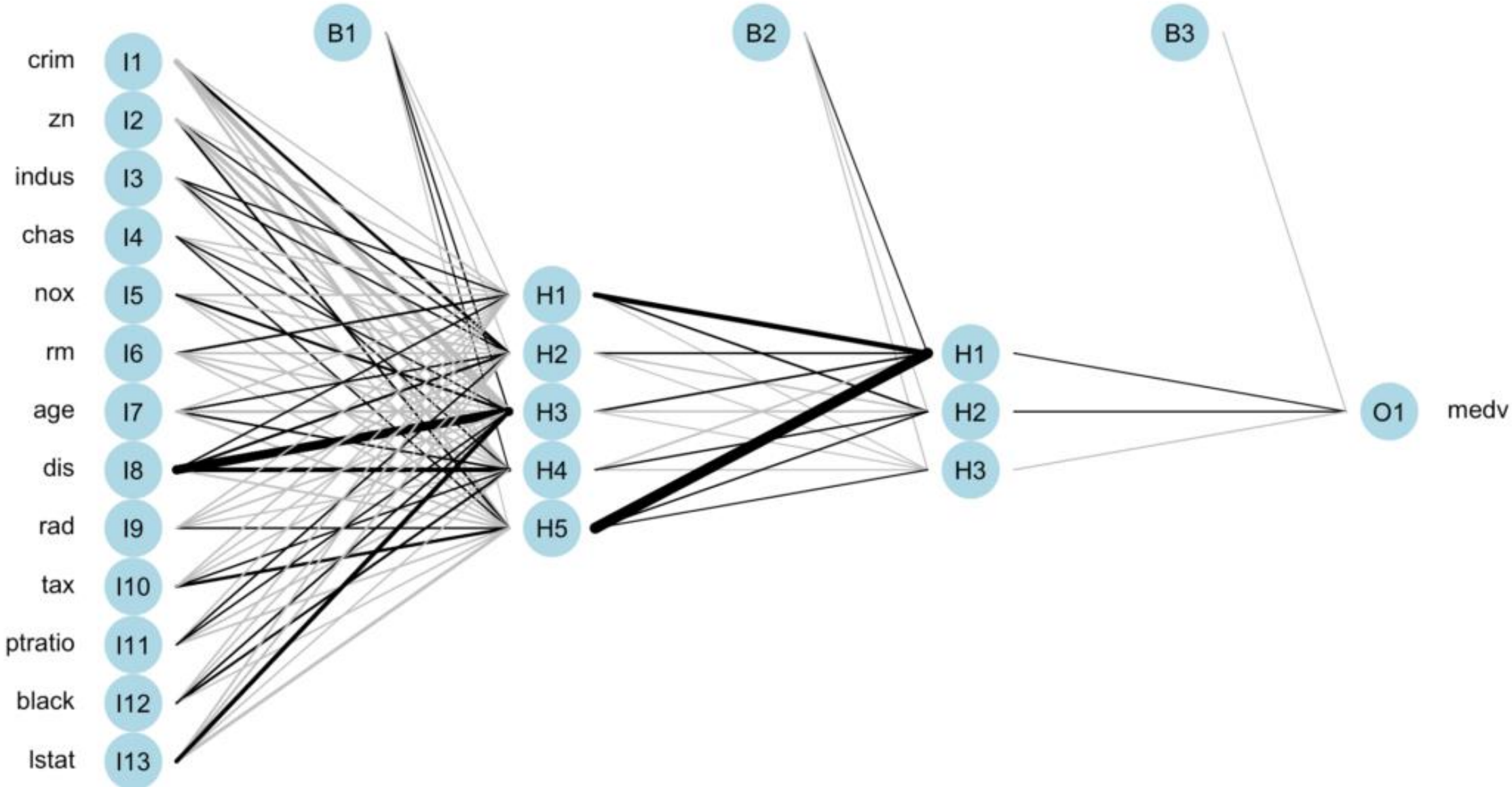
2 katmandan oluşan modelimiz;

```
plot(model_1_nn_5_3)
```

Z=(X1*W1+…….+X13W13)+b(1)

A=Sigmoid(Z) Çıkış sinyali



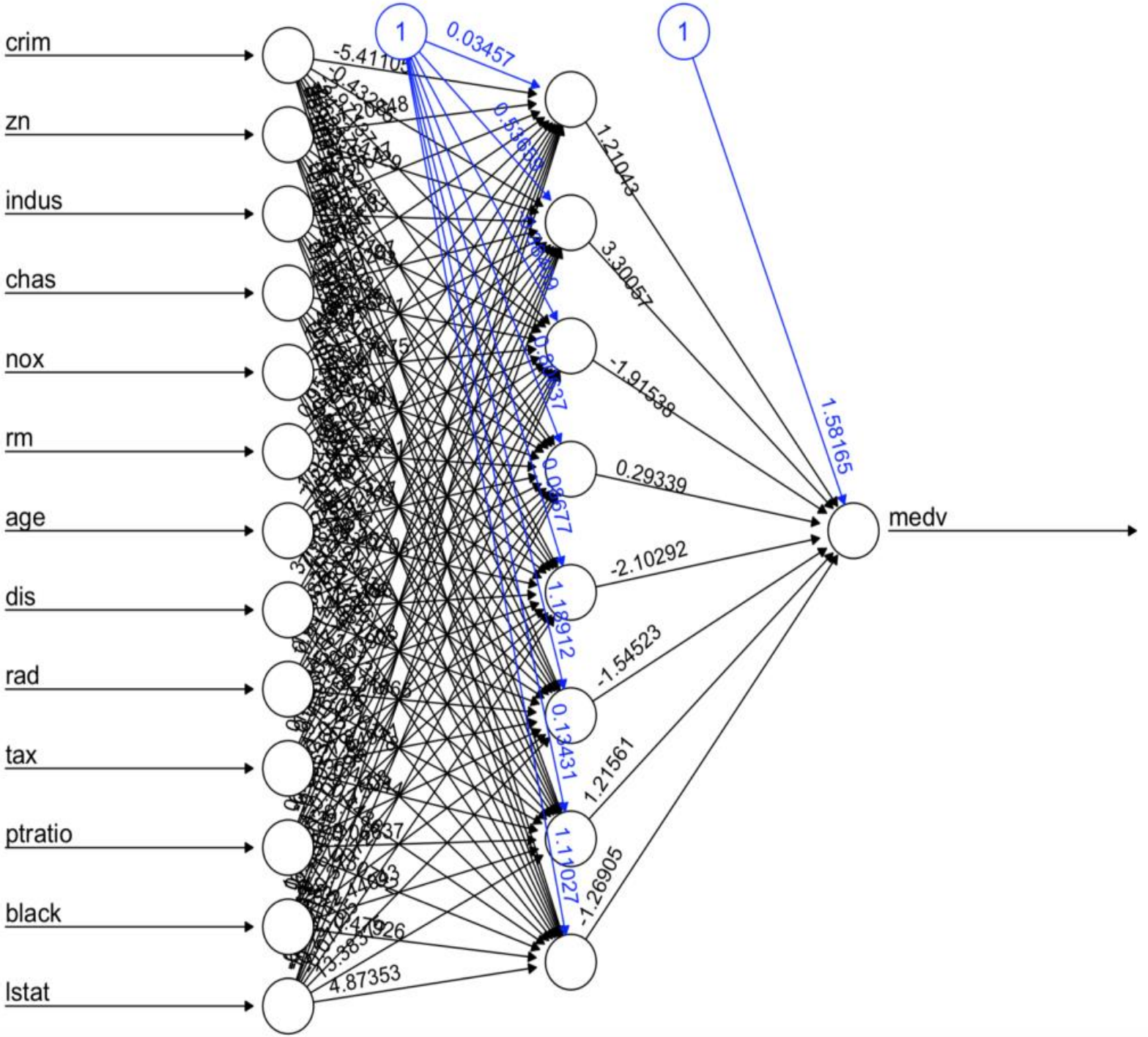

B1
B2
B3
crim
zn
indus
chas
nox
rm
age
dis
rad
tax
ptratio
black
lstat
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
I12
I13
H1
H2
H3
H4
H5
H1
H2
H3
O1
medv

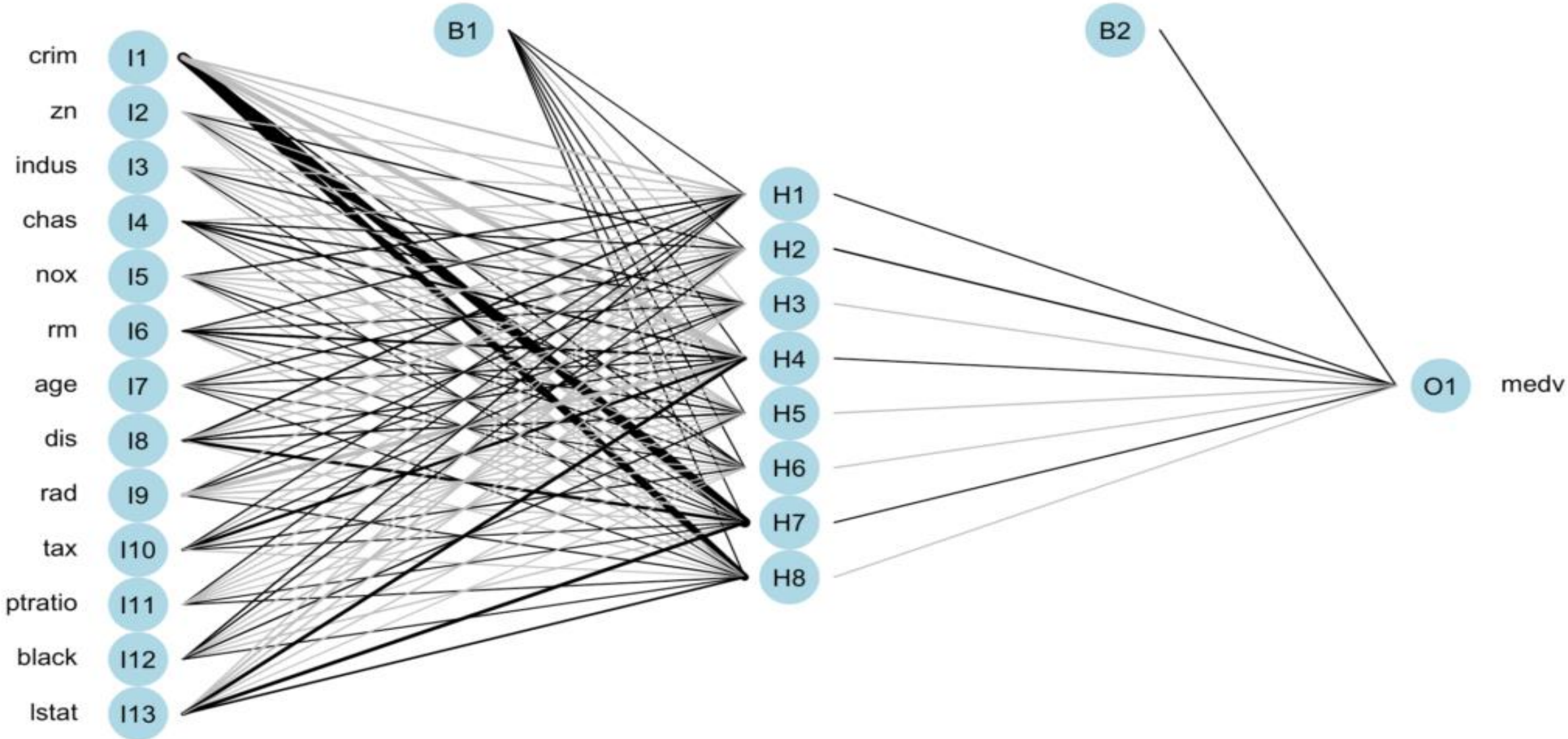
Tek gizli katmanlı ikinci modelimiz.

```
model_2_nn_8 <- neuralnet(duzeltilmis_veri
                , data= egitim_verisi
                , hidden=8,
                , linear.output = TRUE)
```

Tek katmandan oluşan modelimiz;

```
plot(model_2_nn_8)
```




crim
zn
indus
chas
nox
rm
age
dis
rad
tax
ptratio
black
lstat
I1
I2
I3
I4
I5
I6
I7
I8
I9
I10
I11
I12
I13
B1
B2
H1
H2
H3
H4
H5
H6
H7
H8
O1
medv

OVERFITTING/UNDERFITTING

Eğitilen modelin test setinde iyi sonuçlar vermesine rağmen, hiç görmediği yeni/farklı datalarda yanlış sonuçlar vermesine “overfitting” denir. Tam tersi duruma ise “underfitting” denir.

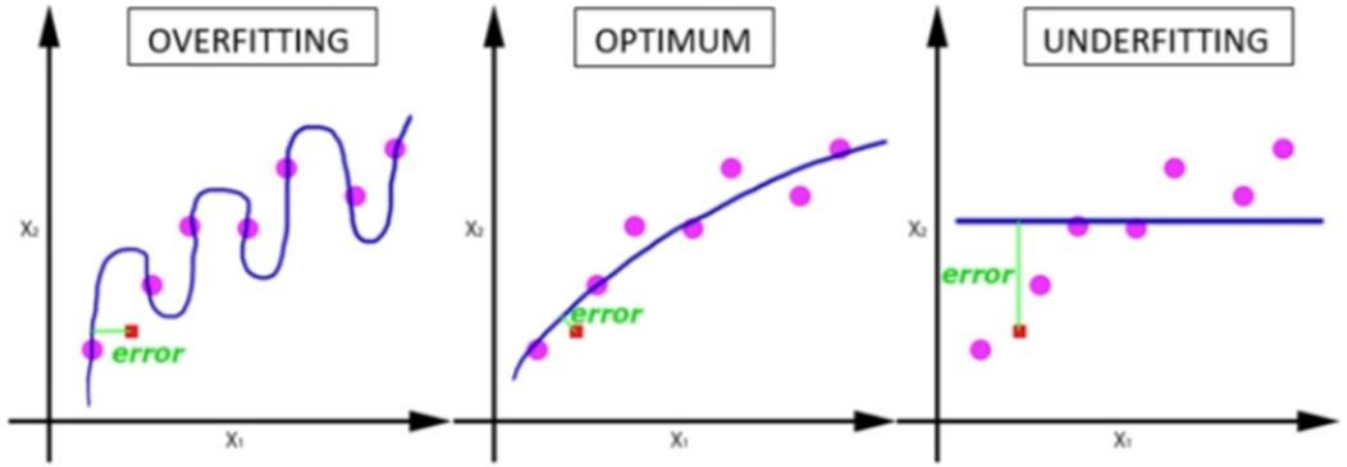
Overfitting’in oluşmasının sebebi modelin verileri genelleştirememesidir ve bizim için büyük bir problem oluşturmaktadır.

Bu problemi aşağıdaki yöntemlerle çözebiliriz:

- Data setine daha fazla veri eklemek
- Data Augmentation yöntemi ile elimizde bulunan veriyi eğitim esnasında modifiye ederek daha fazla veri yaratarak, modeli eğitmektir.
- Dropout fonksiyonu ile sinir ağlarında düzenleme yapılır. Her ileri gidişte layerdaki bazı nöronlar rasgele sıfırlanır ve böylece modelin verinin sadece tek bir noktaya odaklanması engellenir.

Overfitting

- Eğitilen modelin eğitim setini çok iyi öğrenip genelleştirme yapamamasına overfitting denir.
- Underfitting ise tam tersi, modelin veriyi hiç öğrenememesi.



Yapay Sinir Ağlarının Eğitilmesi

Yapay sinir ağı öğrendikten sonra daha önce verilmeyen girişler uygulanarak ağ çıkışları gözlemlenir.

Genelde eldeki örneklerin %80'i ağa verilip ağ eğitilir. Daha sonra geri kalan %20'lik kısım verilip ağın davranışları incelenir ve bu işleme "ağın test edilmesi" denir.

Eğitimde kullanılan örnekler setine "eğitim seti", test için kullanılan sete ise "test seti" denir.

ANN-Hata Fonksiyonu (Loss Function)

İyi bir ANN minimum hataya sahiptir.

İyi bir öğrenme fonksiyonu ANN'yi yüksek hatalı bir konfigürasyondan düşük hatalıya taşır.

ANN'deki hata, hata ölçüm fonksiyonları ile ölçülür.

Bu hata fonksiyonları eğitim kümesindeki örneklerin hatalarının tümünü temsil eder. Problemin tipine ve tasarımcının seçimine göre farklı hata fonksiyonları kullanılabilir.

En sık kullanılanları verilmiştir:

- Mean absolute error
- Mean squared error
- Sum squared error

ANN-Epoch

ANN eğitimi batch processing(çevrim içi) modda yapılır.

Tüm eğitim verileri sisteme verilir. Sistem tüm bu girişleri işlemeyi bitirdikten sonra, başka bir batch proses olan ağırlıkların ve bias' ın güncellenmesi için başlar.

Epoch: tüm giriş verilerinin işlenmesindeki tek bir iterasyon.

- Her epoch'da ANN öğrenir.
- Aynı girişleri çok kereler uygulamanın sonucu olarak, sistem kendini az bir hata ile eğitebilir hale gelir.
- Çok sayıda epoch ile sistem tam olarak eğitilmiş kabul edilir.
- Epoch sistemin kendini eğitim verisine göre eğitebilmesi için gereklidir.

ANN-Learning Rate

Büyük öğrenme oranı: sistemin veriyi çok hızlı öğrenir, toplam hata artar.

Düşük öğrenme oranı: sistem çok yavaş öğrenir, eğitim zamanı artar ancak hata azalır.

η ile temsil edilir.

ANN-Weights and Bias (Ağırlıklar)

ANN eğitimi ağırlıklardaki değişim demektir.

Ağırlıkların farklı kombinasyonları ANN' nin farklı performanslarını temsil edecektir.

Ağırlıklar ANN zekasını oluşturur. ANN' nin öğrenme yeteneğinden bahsediyorsak ağrılıklarından bahsediyoruzdur.

Bias:

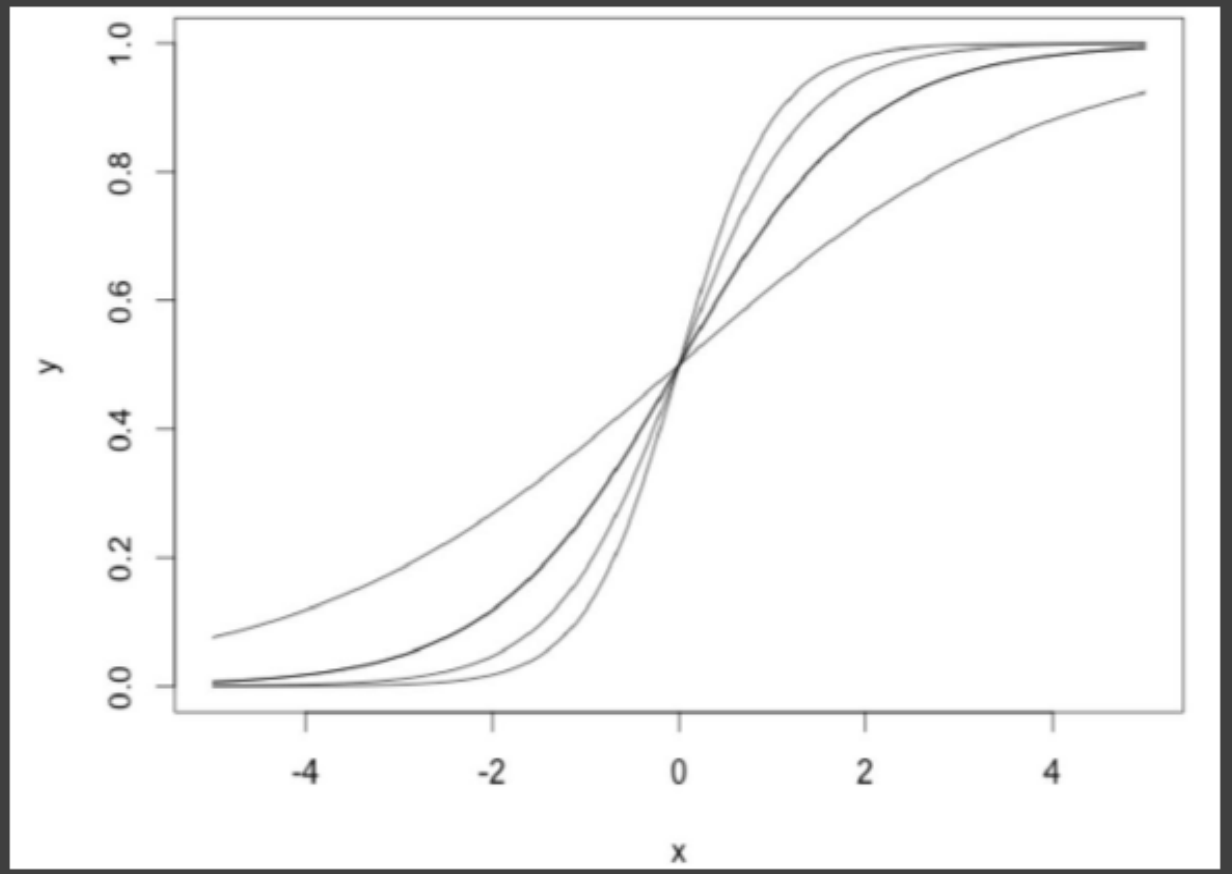
- Sistem performansını etkiler.
- Bias da öğrenmede bir ağırlık gibi alınır.
- Giriş sinyallerinin toplamı 0 olduğunda öğrenme gerçekleşmez. Çıkış değerleri hep 1 olan bias nöronları, nöronların giriş sinyallerinin sürekli sıfırdan farklı olmasını sağlarlar.
- Öğrenmeyi hızlandırırken yerel optimum değerlere takılmayı güçleştirirler.

Neden bias nöronlarına ihtiyaç var?

- X tek bir giriş, w ve b ise sinir ağının ağırlıkları ve biası olsun:

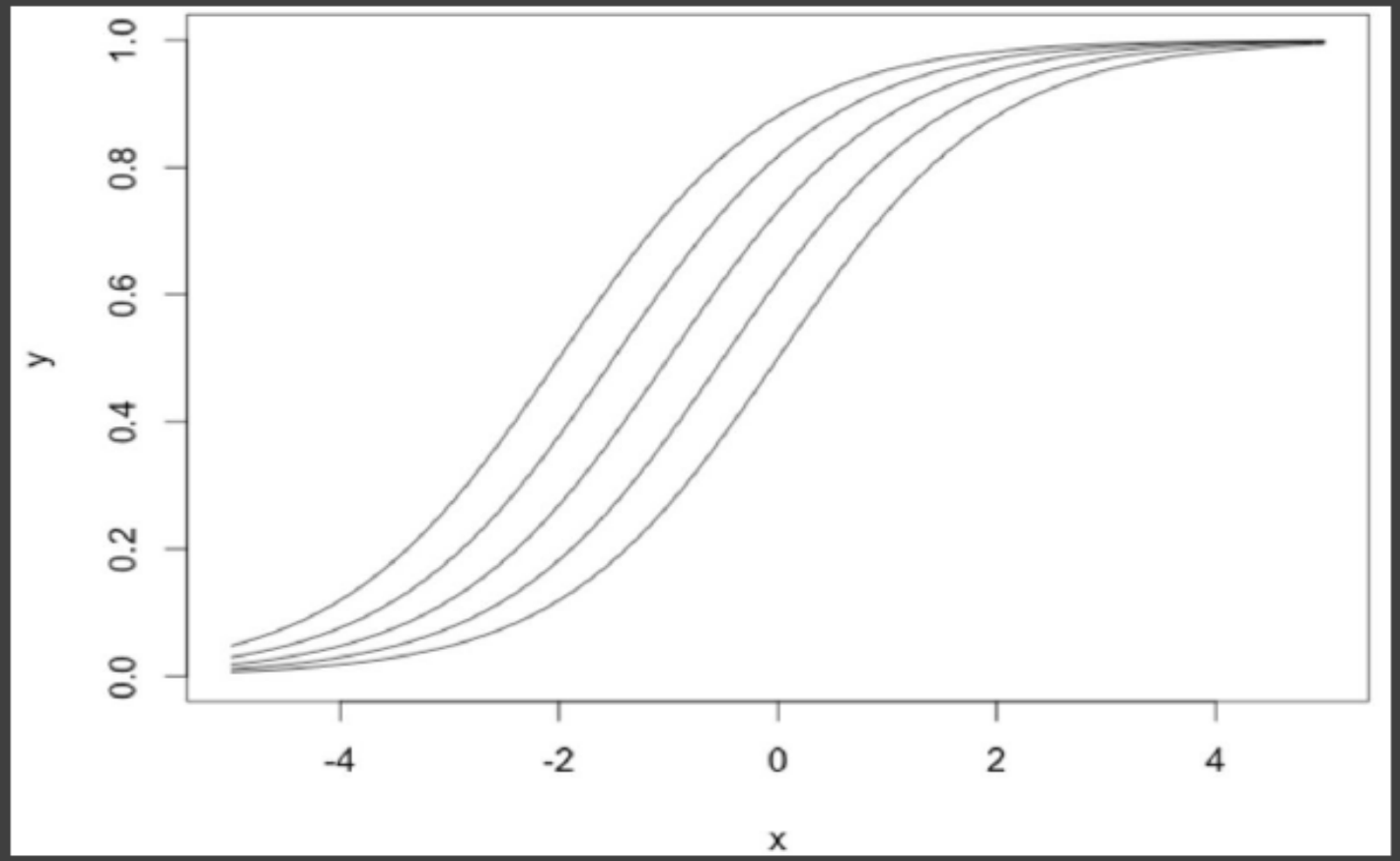
$$f(x, w, b) = \frac{1}{1 + e^{-(wx + b)}}$$

- Nöronun ağırlıkları aktivasyon fonksiyonunun eğimini veya şeklini değiştirmenizi sağlar.

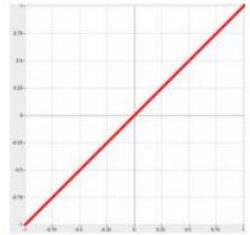
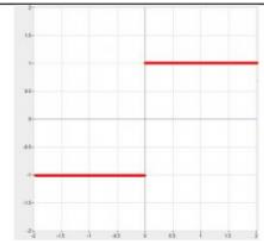
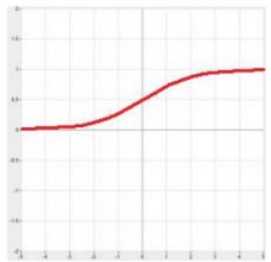
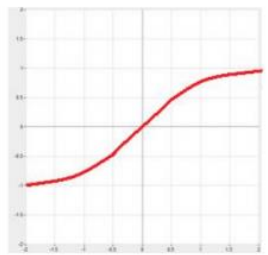


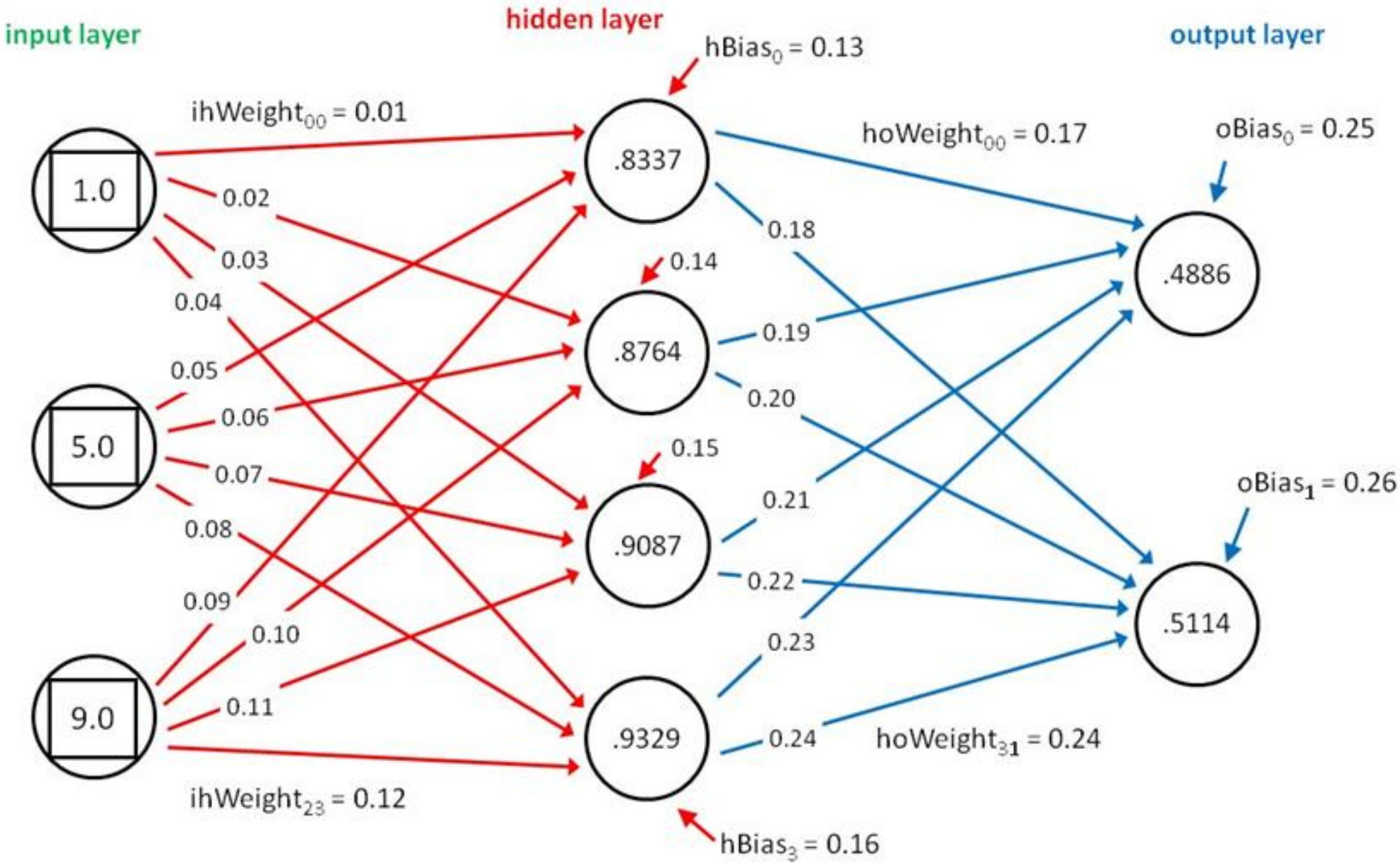
$$f(x, 0.5, 0.0) f(x, 1.0, 0.0) f(x, 1.5, 0.0) f(x, 2.0, 0.0)$$

Neden bias nöronlarına ihtiyaç var?

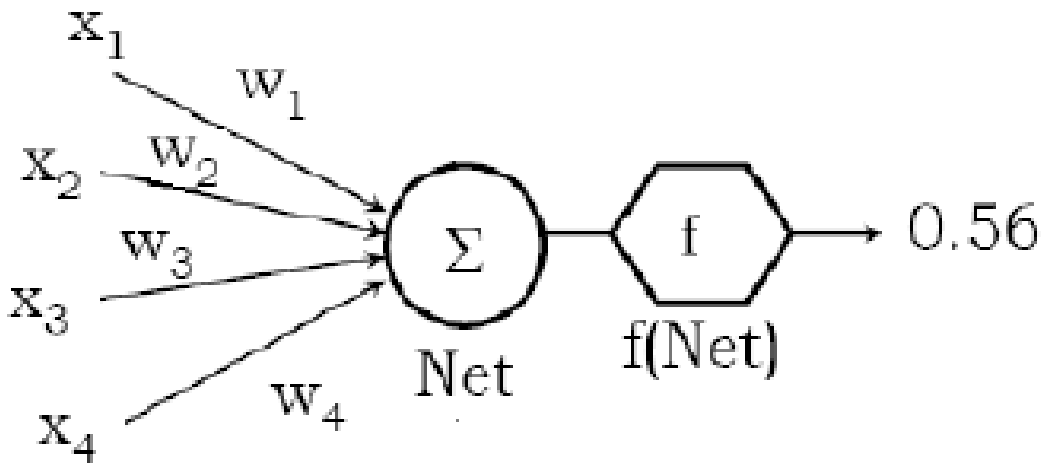


$$f(x, 1.0, 1.0) f(x, 1.0, 0.5) f(x, 1.0, 1.5) f(x, 1.0, 2.0)$$

Doğrusal (Lineer) Aktivasyon Fonksiyonu		F(NET)=A* NET (A sabit bir sayı)	Doğrusal problemler çözmek amacıyla aktivasyon fonksiyonu doğrusal bir fonksiyon olarak seçilebilir. Toplama fonksiyonundan çıkan sonuç, belli bir katsayı ile çarpılarak hücrenin çıktısı olarak hesaplanır.
Adım (Step) Aktivasyon Fonksiyonu		$F(Net)=\begin{cases} 1 \text{ if Net>Eşik Değer} \\ 0 \text{ if Net<=Eşik Değer} \end{cases}$	Gelen Net girdinin belirlenen bir eşik değerin altında veya üstünde olmasına göre hücrenin çıktısı 1 veya 0 değerini alır.
Sigmoid Aktivasyon Fonksiyonu		$F(Net)=\frac{1}{1+e^{-Net}}$	Sigmoid aktivasyon fonksiyonu sürekli ve türevi alınabilir bir fonksiyondur. Doğrusal olmayışı dolayısıyla yapay sinir ağı uygulamalarında en sık kullanılan fonksiyondur. Bu fonksiyon girdi değerlerinin her biri için 0 ile 1 arasında bir değer üretir.
Tanjant Hiperbolik Aktivasyon Fonksiyonu		$F(Net)=\frac{e^{Net}+e^{-Net}}{e^{Net}-e^{-Net}}$	Tanjant hiperbolik fonksiyonu, sigmoid fonksiyonuna benzer bir fonksiyondur. Sigmoid fonksiyonunda çıkış değerleri 0 ile 1 arasında değişirken hiperbolik tanjant fonksiyonunun çıkış değerleri -1 ile 1 arasında değişmektedir.
Eşik Değer Fonksiyonu		$F(Net)=\begin{cases} 0 \text{ if Net<=0} \\ \text{Net if } 0<\text{Net}<1 \\ 1 \text{ if Net>= 1} \end{cases}$	Gelen bilgilerin 0 dan küçük-eşit olduğunda 0 çıktısı, 1 den büyük-eşit olduğunda 1 çıktısı, 0 ile 1 arasında olduğunda ise yine kendisini veren çıktılar üretilebilir.
Sinüs Aktivasyon Fonksiyonu		F(Net) = Sin(Net)	Öğrenilmesi düşünülen olayların sinüs fonksiyonuna uygun dağılım gösterdiği durumlarda kullanılır.


input layer
hidden layer
output layer
hBias0 = 0.13
ihWeight00 = 0.01
hoWeight00 = 0.17
oBias0 = 0.25
1.0
0.02
0.03
0.04
0.14
.8337
0.18
.4886
0.05
0.06
0.07
0.19
.8764
0.20
5.0
0.15
0.21
oBias1 = 0.26
0.08
.9087
0.22
.5114
0.09
0.10
0.23
9.0
0.11
.9329
0.24
hoWeight31 = 0.24
ihWeight23 = 0.12
hBias3 = 0.16

Girişler	Ağırlıklar
$x_1 = 0.5$	$w_1 = -0.2$
$x_2 = 0.6$	$w_2 = 0.6$
$x_3 = 0.2$	$w_3 = 0.2$
$x_4 = 0.7$	$w_4 = -0.1$



Hücrenin net girdisi;

$Net = \Sigma x_i w_i \qquad i=1...4$

$Net = 0.5*(-0.2)+0.6*0.6+0.2*0.2+0.7*(-0.1)$

$Net = 0.23$

- **Sigmoid aktivasyon fonksiyonuna göre hücrenin çıkışı;**

$f(Net) = 1 / (1+e^{-0.23})$

$f(Net) = 0.56$