Yapay Sinir Ağları Temelleri I: Temel Yapay Sinir Hücresi Modeli ve Parametrelerinin Tanıtımı

Biyolojik sinir sisteminin, dağıtık ve bağlantısal bilgi işleme süreçlerini taklit eden bağlantı yoğun dağınık bir bilgi işleme ağı olarak tarif edilebilir. Bu derste çok katmanlı ileri beslemeli (feedforward) yapay sinir ağlarını ve bunların eğitim algoritmaları incelenecektir.

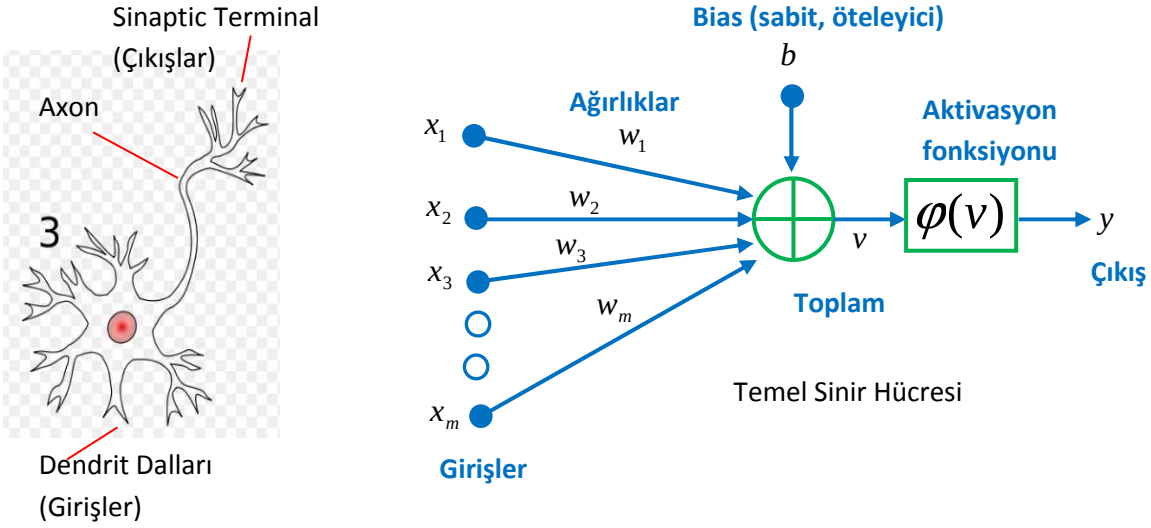
Kısa tarihsel gelişimi:

-1943 yılında fizyolojist Waracen McCulloch ve matematikçi Walter Pitts tarafından biyolojik beynin çalışmasını elektrik devreleri yardımı ile taklit etmek için önerilen basit sinir hücresi modellenmiştir.

-1961 yılında Rosen Blatt çok Kamanlı ağların eğitimi konusunda çalışmalar sunmuştur.

2000'lerden sonra bilgisayarların hesaplama hızı ve gücünün artması ile yapay sinir ağlarının uygulamaları yaygınlaşmaya başlamış ve son yıllarda bu artan hesaplama gücünün sağladığı imkanlar ile temelini yapay sinir ağlarının oluşturduğu deep learning (derin öğrenme) konusu gündeme gelmiştir. Yapay sinir ağları yapay zekâ uygulamalarının temel bileşeni haline dönüşmüş ve hesapsal zekânın en önemli konusu olmuştur.

Solda resmi görülen biyolojik sinir hücresinin çok basitleştirilmiş bir matematiksel modeli olarak sağdaki yapay sinir hücresi modeli önerilmiştir. Bu basitleştirilmiş sinir hücresi modelinin çok sayıda birbirine bağlanması ile yapay sinir ağı elde edilir.



Temel yapay sinir hücresi, girişlerine gelen değerlerini ($x_1, x_2, x_3, .., x_n$), her bir girişin ağırlık katsayıları ($w_1, w_2, w_3, .., w_n$) ile çarpar ve toplam biriminde toplar. Bu toplama, eğer kullanılmışsa girişten bağımsız bir değer olan bias katsayısı değeri b eklenir. Bias değeri çoğu ağda girişi sabit 1 olan ağırlık olarak gerçekleştirilir. İhtiyaç duyulmadığı durumlar için bias katsayısı değeri sıfır değerini alınabilir.

$$v = x_1 w_1 + x_2 w_2 + x_3 w_3 + \ldots + x_m w_m + b \Rightarrow v = \sum_{i=1}^{m} x_i w_i + b$$

Bu ifadeye ağırlıklı toplam denir ve bir x_i girişinden gelen değerin, bir w_i ağırlığı ile ağırlıklanarak toplama etki edeceğini ifade eder. Buna göre w_i değeri ne kadar büyük olursa x_i girişinin v toplamındaki ağırlığı veya etkisi o oranda artar. Ağırlık değeri sıfır olursa ($w_i = 0$) girişin (x_i) bir etkisi kalmaz. (Çünkü, $w_i x_i = 0$)

Ağırlıklı toplamı matris formunda ifade edebiliriz. Matris formu sinir hücresi modelinin daha derli toplu ifade edilmesini sağlar.

Giriş vektörü: $X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ : \\ x_m \end{bmatrix}$ bir sütün vektörü alınırsa

Ağırlık vektörü: $W = [w_1 \; w_2 \; w_3 \ldots w_m \;]$

olarak alınırsa, girişlerin ağırlıklı toplamı toplam

$$v = WX + b$$

ile ifade edilebilir.

Örnek: Yapay sinir hücresi ağırlıklı toplam modeli matris formunda ifadesinin $v = WX + b$'nin açık formda $v = \sum_{i=1}^{m} x_i w_i + b$ formülüne karşılık geldiğini gösteriniz.

$$v = WX + b \Rightarrow v = [w_1 \; w_2 \; w_3 \ldots w_m] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ : \\ x_m \end{bmatrix} + b = x_1 w_1 + x_2 w_2 + x_3 w_3 + \ldots + x_m w_m + b = \sum_{i=1}^{m} x_i w_i + b$$

Yapay sinir hücresi modelinde $\varphi(v)$ aktivasyon fonksiyonu olarak adlandırılır. Girişlerin ağılıklı toplamı $v = WX + b$, aktivasyon fonksiyonunda kullanılır ve yapay sinir hücresinin çıkışı,

$$y = \varphi(v)$$

olarak elde edilir. Burada ağırlıklandırılmış toplam $v = WX + b$ kullanılırsa, yapay sinir ağının çıkışı,

$$y = \varphi(v) = \varphi(WX + b)$$

olarak yazılabilir. Burada aktivasyon fonksiyonları genelde,

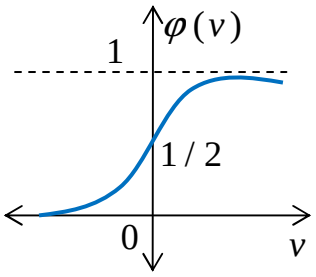
(i) türevlenebilir,

(ii) yumuşak değişen

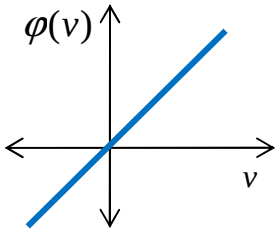
(iii) sınırlı değere sahip

fonksiyonlardan seçilir. Örneğin, sigmoid aktivasyon fonksiyonu $\varphi(v) = \frac{1}{1+e^{-v}}$, [0,1] aralığında sınırlıdır, yumuşak değişir ve türevlenebilir. Türevlenebilir olması ve çıkışın yumuşak değişim göstermesi gradyan iniş yönteminin uygulama başarısını artırır. Sigmoid yaygın kullanım bulan aktivasyon fonksiyonudur.

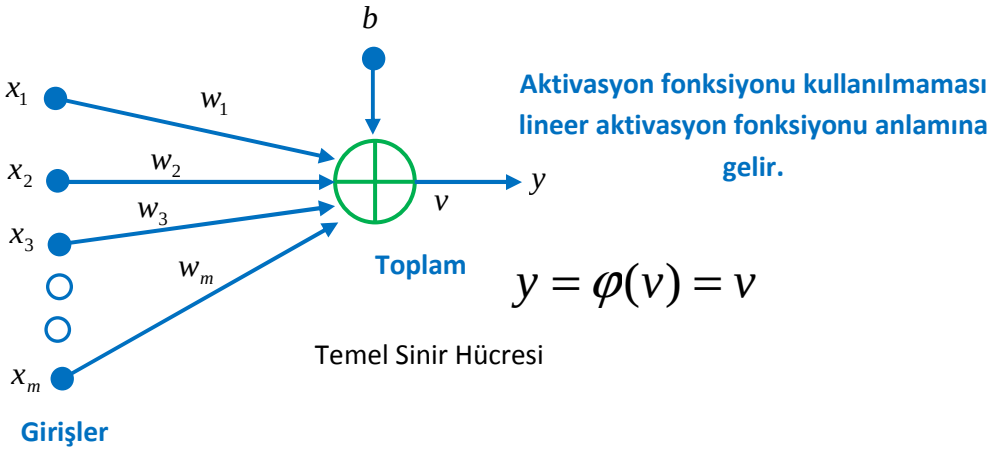
Sigmoid aktivasyon fonksiyonu



Lineer aktivasyon fonksiyonu

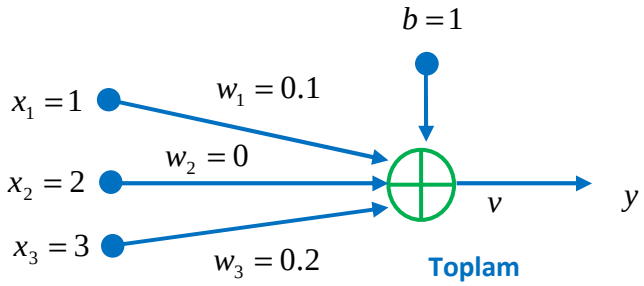


Lineer aktivasyon fonksiyonu $\varphi(v) = v$ olarak tanımlanır. Türevlenebilir, yumuşak değişir ancak bir aralıkta sınırlı değildir. Lineer aktivasyon fonksiyonu aslında ağırlıklı toplam toplam (v) üzerinde hiçbir etkiye sahip değildir. Ağırlıklı toplamı doğrudan çıkışa iletir. Dolayısı ile $y = \varphi(v) = v$. Lineer aktivasyon fonksiyonuna sahip bir sinir hücresi şöyle çizilebilir.



Temel Sinir Hücresi

Örnek: Aşağıda verilen sinir hücresi modelinde lineer aktivasyon fonksiyonu ($\varphi(v)=v$) kullanılmıştır. Sinir hücresinin çıkış değerini hesaplayınız? Sizce $x_2=2$ girişinin sinir hücresi çıkışında bir etkisi olur mu? Nedeni ile açıklayınız?



Öncelikle sinir hücresi modelinde verilenleri yazalım.

Girişler: $x_1=1,\ x_2=2,\ x_3=3$

Ağırlıklar: $w_1=0.1,\ w_2=0,\ w_3=0.2$

Bias: $b=1$

1. yol: Matris formunda yazılabilir ve hesaplanabilir.

$$v=WX+b \Rightarrow X=\begin{bmatrix}1\\2\\3\end{bmatrix},\ W=\begin{bmatrix}0.1 & 0 & 0.2\end{bmatrix},\ b=1$$

$$v=\begin{bmatrix}0.1 & 0 & 0.2\end{bmatrix}\begin{bmatrix}1\\2\\3\end{bmatrix}+1=0.1*1+0*2+0.2*3+1=0.1+0+0.6+1=1.7$$

lineer aktivasyon fonksiyonu kullanıldığı için $y=\varphi(v)=v$. Dolayısı ile sinir hücresi çıkışı

$y=v=1.7$

```
clear all
% Temel Sinir hücresi modeli için matris formunda hesaplamalar
% Temel sinir hücresi giriş vektörü
X=[1 2 3];
% Temel sinir hücresi ağırlık vektörü
W=[0.1 0 0.2];
% Temel sinir hücresi biad değeri
b=1;
% Lineer aktivasyon fonksiyonu durumunda çıkış
y=W*X'+b
```

2. yol: Açık formda yazılabilir ve hesaplanabilir

$$v=\sum_{i=1}^{m} x_i w_i + b \;\Rightarrow\; v=\sum_{i=1}^{m} x_i w_i + b = x_1 w_1 + x_2 w_2 + x_3 w_3 + b = 0.1*1+0*2+0.2*3+1=1.7$$

Lineer aktivasyon fonksiyonu kullanıldığı için çıkış $y=v=1.7$

Not: Burada $x_2=2$ girişinin toplamda bir etkisi olmamıştır. Çünkü bu girişe ait ağırlık katsayısı $w_2=0$ dır. Burada $x_2 w_2 = 2*0=0$ ve ağırlık toplam v'ye bir katkısı olmaz.

```
clear all
% Temel Sinir hücresi modeli için hesaplamalar
% Temel sinir hücresi giriş vektörü
X=[1 2 3];
% Temel sinir hücresi ağırlık vektörü
W=[0.1 0 0.2];
% Temel sinir hücresi biad değeri
b=1;
% Lineer aktivasyon fonksiyonu durumunda çıkış
top=0; % Seri toplamı için tutucu değişişken
% Seri toplamı hesaplanır ve top değişkeninde tutulur.
for i=1:length(X)
%Her döngüde bir girişle ile ilgili ağırlığı çarpar
top=top+X(i)*W(i);
end
% top değşkeninde tutulan seri toplamına bias eklenir
% çıkış hesaplanır
y=top+b
```

Örnek: Girişleri $x_1=3$, $x_2=1$, $x_3=4$ olan 3 girişli ve lineer aktivasyon fonksiyonuna sahip bir sinir hücresinde $w_1=1$, $w_2=2$ dir ve w_3 ağırlığı belirlenmemiştir. Bu hücrede bias kullanılmamıştır. Bu durumda sinir hücresinin, bu girişler için $y=9$ çıkışı verebilmesi için w_3 değerinin ne olması gerektiğini hesaplayınız.

Öncelikle sinir hücresi modelinde verilenleri yazalım.

Girişler: $x_1=3$, $x_2=1$, $x_3=4$

Ağırlıklar: $w_1=1$, $w_2=2$, $w_3=?$ (Burada $y=9$ istenen çıkışı verebilmesi için w_3 belirlenmesi isteniyor.)

Bias: $b=0$ (Bias kullanılmadığı için 0 alınır)

Çıkış: $y = 9$

Sinir hücresi modelinin açık formunu kullanalım:

$$v = \sum_{i=1}^{m} x_i w_i + b = x_1 w_1 + x_2 w_2 + x_3 w_3 + b$$

Lineer aktivasyon fonksiyonu kullanıldığı için

$$y = v = \sum_{i=1}^{m} x_i w_i + b = x_1 w_1 + x_2 w_2 + x_3 w_3 + b$$

$$y = x_1 w_1 + x_2 w_2 + x_3 w_3 + b$$

Verilenler burada kullanılırsa,

$9 = 3*1 + 1*2 + 4*w_3 + 0 \Rightarrow 9 = 5 + 4*w_3 \Rightarrow 4 = 4*w_3 \Rightarrow w_3 = 1$ elde edilir.

Not: Bu örnekte aslında basitleştirilmiş bir eğitim örneği görülebilir. Bir giriş için istenen çıkışı üretmesini sağlayan ağırlık belirme işlemine yapay sinir ağının eğitimi adı verilir. Bu örnekte sadece bir elemana sahip eğitim kümesi $\{(x_1 = 3,\ x_2 = 1,\ x_3 = 4, y_d = 9)\}$ için bir ağırlığın belirlenmesi durumu incelenmiştir. Optimizasyon işlemine gerek kalmadan bir denklem ve bir bilinmeyen durumu için çözülebildi, Çünkü tam belirlenmiş sistem ifade ediyor. Gerçek uygulamalarda 1000'lerce elemanlı eğitim kümeleri için onlarca ağırlık katsayısı aşırı belirlenmiş bir sistem ifade eder ve bu durumda aşırı belirlenmiş sistemin optimal çözümü aranabilir. Bu noktada, uygulamalarda gradyan iniş optimizasyon yönteminin çözümlerine başvurulur. Dolayısı ile ilerleyen derslerde göreceğimiz üzere yapay sinir ağlarının eğitiminde gradyan iniş yöntemi ve varyantları kullanılır ve ağın eğitimi gradyan iniş yönteminin performansına bağımlı olacaktır.

Yapay Sinir Hücresinin Ağırlıklı Toplam Modelinin Makine Öğrenmesi Açısından Geometrik Yorumu:

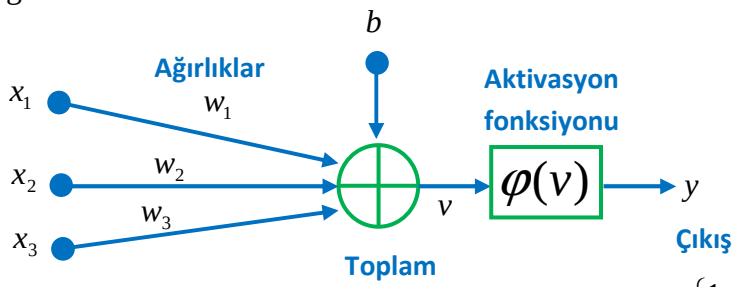
$v = x_1 w_1 + x_2 w_2 + x_3 w_3 + ... + x_m w_m + b$ ifadesi bir aslında m boyutlu uzayda bir düzlem ifade eder. Burada ağırlıklar w_i düzlemin eğimini ve b bias öteleme miktarını belirler. Örneğin,

İki girişli, basit bir nöron modeli $v = x_1 w_1 + x_2 w_2 + b$ düşünelim. $v = 0$ için $0 = x_1 w_1 + x_2 w_2 + b$, iki boyutlu Kartezyen koordinat sisteminde bir doğru ifade eder. Bu doğruyu sınıflama problemi için ayırıcı doğru olarak kabul edelim. Dolayısı ile $v = x_1 w_1 + x_2 w_2 + b$ için sinir hücresi çıkışı ikili eşikleyici aktivasyon fonksiyonu yardımı ile

$$y = \varphi(v) = \begin{cases} 1 & v \geq 0 \\ 0 & v < 0 \end{cases}$$

çıkışın $\{0,1\}$ değerleri üretmesi sağlanabilir. Eşikleyici aktivasyon fonksiyonu ile $0 = x_1 w_1 + x_2 w_2 + b$ ayırıcı eğrinin altında kalan bölge ($v < 0$) 0 olarak sınıflanır ve üstünde

kalan bölge ($v \geq 0$) ise 1 olarak sınıflanır. Aşağıda sınıflama için kullanılabilecek yapay sinir hücresinin modeli görülmektedir.



Şimdi bias'ın sınıfla probleminin çözümüne katkısını görmek için,

1) $b = 0$ için ayırıcı eğri $0 = x_1 w_1 + x_2 w_2$,

2) $b \neq 0$ için ayırıcı eğri $0 = x_1 w_1 + x_2 w_2 + b$ durumlarını grafik üzerinde inceleyelim.

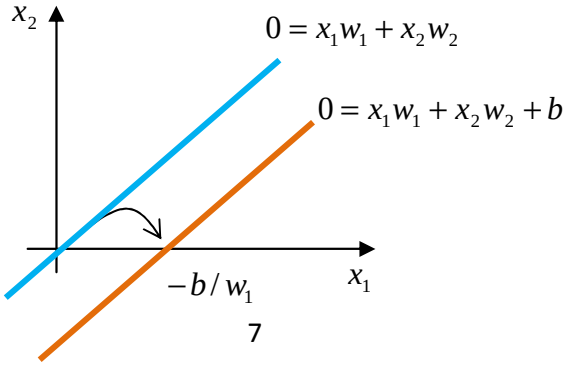
1) Aşağıdaki grafikte mavi çizgi ile $b = 0$ için ayırıcı eğri

$0 = x_1 w_1 + x_2 w_2$

mavi ile gösterilmiştir. x_1 eksenini kestiği noktayı bulmak için $x_2 = 0$ verilirse $x_1 = -\frac{w_2}{w_1} x_2 = -\frac{w_2}{w_1} 0 = 0$. x_1 eksenini 0 noktasında keser. Dolayısı ile ayırıcı doğru sadece orjin (0,0) geçer. Bu gözlem bias olmadığı durumda ($b = 0$)ayırıcı doğru orijinden noktasından (0,0) geçmek zorunda bırakır. Dolayısı ile ayırıcı doğru orijin noktasına (0,0) noktasına takılı kalır. Bu durum sinir hücresinin sınıflama performansını oldukça sınırlar. Çünkü ayırıcının doğrusunun orjinden geçmediği sınıflama problemleri olacaktır. Özetle bias'ın kullanılmaması, getirdiği sınırlama nedeni ile sınıflama performansına olumsuz etkiyebilir.

2) Oysaki, $b \neq 0$ için ayırıcı eğri $0 = x_1 w_1 + x_2 w_2 + b$ ve x_1 eksenini kestiği noktayı bulmak için $x_2 = 0$ uygulanırsa $x_1 = -\frac{w_2}{w_1} x_2 - \frac{b}{w_1} = -\frac{b}{w_1}$ elde edilir. Dolayısı ile bias değeri kullanılması durumda ($b \neq 0$), ayırıcı doğrunun eksenini kestiği nokta $x_1 = -\frac{b}{w_1}$ ile ayarlanabilir. Bu lineer ayrışabilir verilerin sınıflama probleminin çözümünde sinir hücresine önemli bir esneklik kazandırır ve sınıflama başarımını artırır.

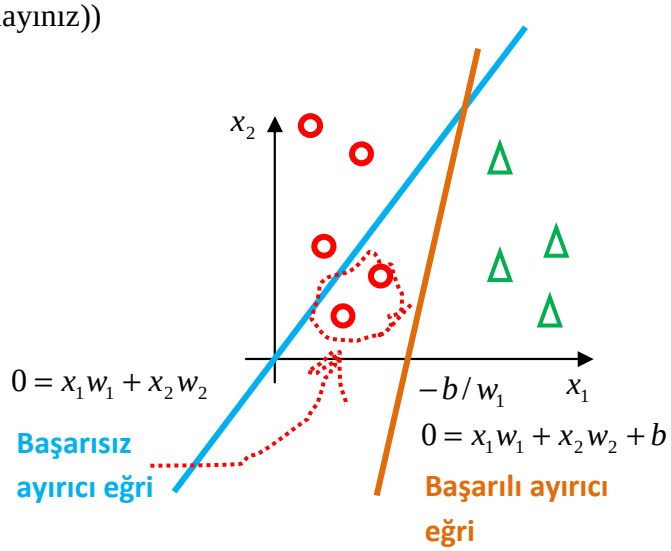
.



Bias'ın sınıflama başarımına katkısını görebilmek için aşağıdaki sınıflama probleminin çözümlerini inceleyelim.

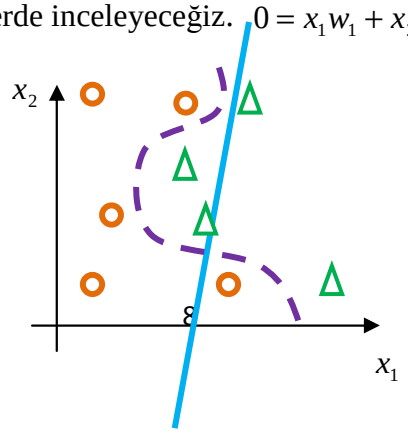
* Bias kullanılmaz ($b = 0$) ise yapay sinir ağının ayırıcı eğrisi orjinden geçmek ((0,0) notasından) zorunda kalır ve bu sınırlama nedeni ile üçgenlerin doğru sınıflandığı durumda iki adet kırmızı yuvarlak veri eğri altında kalır ve yeşil üçgen sınıfına dahil olarak yanlış sınıflanır. Dolayısı ile yeşil üçgenler lineer doğru ile doğru sınıflandığında iki kırmızı yuvarlak veri hatalı sınıflanır. Bu sınıflama başarısını düşürür.

* Bias kullanılır ($b \neq 0$) ise yapay sinir ağının ayırıcı eğrisi x_1 ekseni üzerinde b'nin ve w_1 ağırlık değerine göre kayabilir ($x_1 = -\frac{b}{w_1}$) ve daha başarılı bir sınıflama sağlayabilir. (Not: Bu sınıflama başarımı lineer ayrıştırılabilir veriler üstünde iki(binary) sınıflama yapıldığı durum için geçerlidir. (Önceki derslerimizden sınıflama problemleri ve lineer ayrıştırılabilir veriler konusunu hatırlayınız))

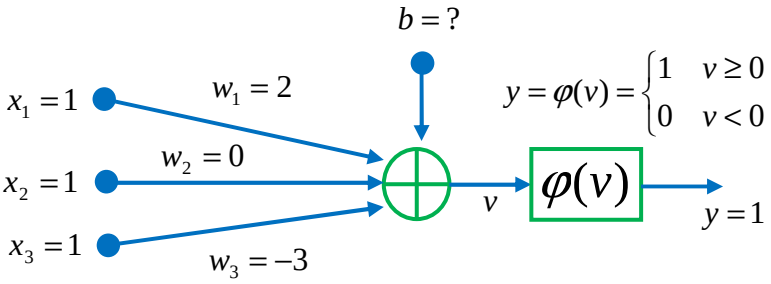


Sonuç: Bias kullanılırsa, ayrıştırma doğrusunun eksen üzerinde ötelenmesi kabiliyeti kazandırarak lineer ayrıştırma performansı artırılabilir. Eğer bias kullanılmaz ise ($b = 0$), ayrıştırma doğrusu (0.0) noktasından geçmek zorunda kalır ve bu durum lineer ayrıştırıcılık performansını düşürür. Sınıflama başarımı düşer.

Peki, veriler lineer ayrıştırılamaz veri olduğu durumda ne olur? Bias ile daha iyi sınıflama mümkün olsa da bu durumda tam başarım sağlanamaz. Aşağıda bu durumu gösterir grafik görülmektedir. Başarılı sınıflama için doğrusal olmayan ayırıcı eğrilerine ihtiyaç duyulur. Bu noktada yapay sinir ağlarında lineer olmayan aktivasyon fonksiyonu (Lineerliğin bozulması) ve çok katmanlı daha büyük sinir ağları kullanılması ile (model karmaşıklığının artırılması) bu sorun aşılır. Bunu gelecek derslerde inceleyeceğiz. $0 = x_1 w_1 + x_2 w_2 + b$



Örnek: Aşağıdaki tek sinir hücresinden oluşan ikili (binary) sınıflayıcının ($x_1 = 1$, $x_2 = 1, x_3 = 1$) girişini 1 olarak sınıflayabilmesi için bias'ın (b) sağlaması gereken koşulu bulunuz. Bu sinir hücresi için aktivasyon fonksiyonu binary eşikleyicidir. ($y = \varphi(v) = \begin{cases} 1 & v \geq 0 \\ 0 & v < 0 \end{cases}$)



Çözüme çıkıştan bakarak başlayalım. Bu sinir hücresinin

$$y = \varphi(v) = \begin{cases} 1 & v \geq 0 \\ 0 & v < 0 \end{cases}$$

Aktivasyon fonksiyonuna sahip olması durumunda 1 çıkışı verebilmesi için ağırlıklı toplamın $v \geq 0$ koşulunu sağlaması beklenir. Şimdi ağırlıklı toplamı yazalım.

$$v = \sum_{i=1}^{m} x_i w_i + b = x_1 w_1 + x_2 w_2 + x_3 w_3 + b$$

Verilenleri kullanalım:

Girişler: $x_1 = 1$, $x_2 = 1$, $x_3 = 1$

Ağırlıklar: $w_1 = 2$, $w_2 = 0$, $w_3 = -3$

Bias: $b = ?$ (Burada çıkışın 1 olabilmesi için $v \geq 0$ koşulunun sağlanmalıdır. Bunun için bias sağlaması gereken koşulu bulunmalı.)

$y = 1$ => $v \geq 0$ => $v = 1*2 + 1*0 + 1*(-3) + b \geq 0$ olmalıdır.

=> $2 + 0 - 3 + b \geq 0$ => $2 + 0 - 3 + b \geq 0$ => $b \geq 1$ elde edilir.

Not: Yukarıdaki soruda bu sınıflamayı sağlayan en küçük bias sorulsa idi bunun cevabı $b = 1$ olacaktı. Ancak bu durumda $v = 1*2 + 1*0 + 1*(-3) + 1 = 0$ olacak yani ($x_1 = 1$, $x_2 = 1, x_3 = 1$) noktası ayırma doğrusunun tam üzerinde olacaktır. Çünkü , $v = 0$ için elde edilecek olan ayırma doğrusu denklemini ($x_1 w_1 + x_2 w_2 + x_3 w_3 + b = 0$) sağlayacaktır.

Örnek: Bir önceki örneği lineer aktivasyon fonksiyonu ($\varphi(v) = v$) için çözülmüş olsaydı bias ne olurdu?

Lineer aktivasyon fonksiyonu durumda sinir hücresinin 1 çıkışı verebilmesi için $v = 1$ olmalıdır. Çünkü,

$y = \varphi(v) = v \Rightarrow y = v = 1$

Şimdi ağırlıklı toplamın 1 olabilmesi için bias hesaplayalım:

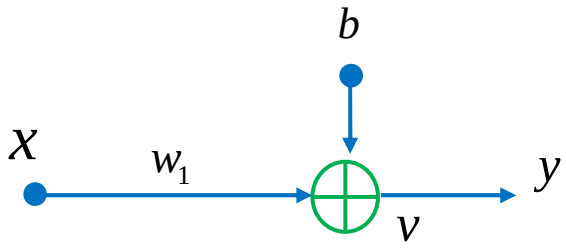
$$v = \sum_{i=1}^{m} x_i w_i + b = x_1 w_1 + x_2 w_2 + x_3 w_3 + b$$

$v = 1 \Rightarrow v = 1*2 + 1*0 + 1*(-3) + b = 1 \Rightarrow -1 + b = 1 \Rightarrow b = 2$ olmalıdır.

Not: Bu örnekte görüldüğü üzere aktivasyon fonksiyonun değişmesi katsayıların yeniden ayarlanmasını gerektirecektir. Yapay sinir ağının performansı, problemin doğasına uygun aktivasyon fonksiyonun seçimine bağımlıdır.

Gerçek uygulamalarda eğitim setinde çok fazla veri bulunur ve bu verilerin nispeten az sayıda ağılığa sahip bir yapay sinir ağı modeli tarafından öğrenilmesi istenir. Bu durumda eğitim verileri için karesel hatayı minimim yapan ağılıkların belirlenmesi gerekir. Eğitim problemi bir optimizasyon problemine dönüşür. Aşağıda gradyan iniş yöntemi ile bir sinir hücresinin eğitimi için ağırlık güncelleme ifadelerini elde edelim.

Örnek: Aşağıda lineer aktivasyon fonksiyonuna sahip sinir hücresi için $T = \{(x_1 = 1, y_{d1} = 3), (x_2 = 2, y_{d2} = 5), (x_3 = 3, y_{d3} = 7)\}$ eğitim kümesi için karesel hatayı minimum yapan (min $E = \frac{1}{2}\sum_{i=1}^{3}(y_{di} - y(x_i))^2$) gradyan iniş güncelleme çözümlerini elde ediniz.



Sinir hücresinin giriş-çıkış ilişkisini veren model:

$v = x_1 w_1 + b$

$y = v$

Dolayısı ile öğrenme modeli $y = x_1 w_1 + b$ elde edilir. (Bu modelin aslında lineer regresyon modeline karşılık geldiğini görünüz. O halde yapay sinir ağları lineer regresyon modellerini ifade edebilirler yada öğrenebilirler. Sinir hücresi ve katman sayısı arttıkça daha karmaşık

modelleri de ifade edebilirler. Bu durum yapay sinir ağlarının modelleme kabiliyetinin oldukça yüksek olduğuna işaret eder.)

Her bir veri için öğrenme hatası:

$(x_1 = 1, y_{d1} = 3)$ verisi için hata $e_1 = y_{d1} - y(x_1) = y_{d1} - (w_1 x_1 + b) = 3 - (w_1 1 + b)$

$(x_2 = 2, y_{d2} = 5)$ verisi için hata $e_2 = y_{d2} - y(x_2) = y_{d2} - (w_1 x_2 + b) = 5 - (w_1 2 + b)$

$(x_3 = 3, y_{d3} = 7)$ verisi için hata $e_3 = y_{d3} - y(x_3) = y_{d3} - (w_1 x_3 + b) = 7 - (w_1 3 + b)$

Bütün veri hatalarını dikkate alan karesel hata

$$E = \frac{1}{2}e_1^2 + \frac{1}{2}e_2^2 + + \frac{1}{2}e_3^2$$
$$= \frac{1}{2}(3 - (w_1 1 + b))^2 + \frac{1}{2}(5 - (w_1 2 + b))^2 + \frac{1}{2}(7 - (w_1 3 + b))^2$$
$$= \frac{1}{2}(3 - w_1 - b)^2 + \frac{1}{2}(5 - 2w_1 - b)^2 + \frac{1}{2}(7 - 3w_1 - b)^2$$

Belirlenmesi gereken elastik parametreler w_1 ve b için gradyan iniş güncelleme ifadeleri şöyle yazılır.

$$w_1 \leftarrow w_1 - \eta \frac{dE}{dw_1}$$

(Burada $w_1[n+1] \leftarrow w_1[n] - \eta \frac{dE}{dw}|_{w=w_1[n]}$ ifadesinin daha kolay yazılması için zaman indeksi n silinip eşitlik yerine $\leftarrow$ ile güncelleme işlemi ifade edilmiştir.)

$$b \leftarrow b - \eta \frac{dE}{db}$$

Burada türev ifadelerinin hesaplaması ve güncelleme ifadelerinde yerine yazılması gerekiyor.

$$\frac{dE}{dw_1} = \frac{2}{2}(-1)(3 - (w_1 1 + b)) + \frac{2}{2}(-2)(5 - (w_1 2 + b)) + \frac{2}{2}(-3)(7 - (w_1 3 + b))$$
$$= -3 + w_1 + b - 10 + 4w_1 + 2b - 21 + 9w_1 + 3b = -34 + 14w_1 + 6b$$

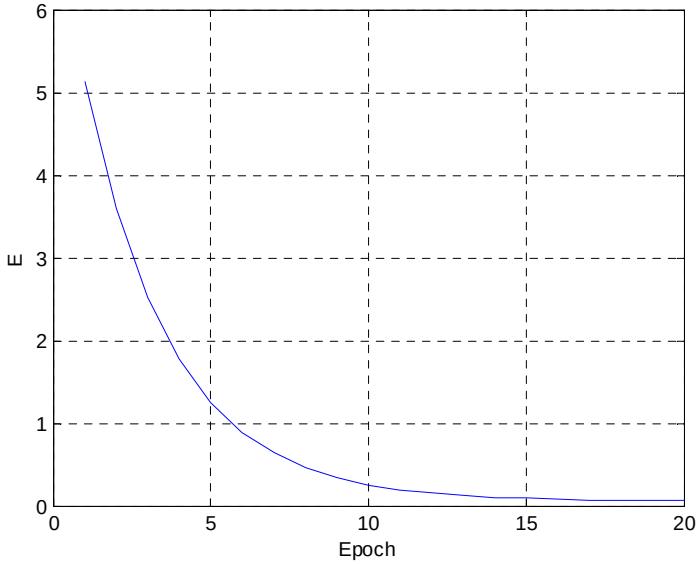
$$w_1 \leftarrow w_1 - \eta \frac{dE}{dw_1} \Rightarrow \boxed{w_1 \leftarrow w_1 - \eta(-34 + 14w_1 + 6b)}$$

$$\frac{dE}{db} = \frac{2}{2}(-1)(3 - (w_1 1 + b)) + \frac{2}{2}(-1)(5 - (w_1 2 + b)) + \frac{2}{2}(-1)(7 - (w_1 3 + b))$$
$$= (-3 + (w_1 1 + b)) + (-5 + (w_1 2 + b)) + (-7 + (w_1 3 + b)) = -15 + 6w_1 + 3b$$

$$b \leftarrow b - \eta \frac{dE}{db} \Rightarrow \boxed{b \leftarrow b - \eta(-15 + 6w_1 + 3b)}$$

olarak elde edilir. Bu güncelle denklemleri aşağıdaki Matlab kodunda her veri için bir kere güncelleme yapılan stokastik gradyan inişe göre 20 defa bütün eğtim kümesi verileri için güncelleme sağlayan 20 Epoch (Dönem) için çözülmüştür.

```
clear all;
% Eğitim seti
X=[1 2 3]
D=[3 5 7];
% Ağırlık katsayıları başlangıç değeri
w1=1;
b=1;
% Öğrenme katsayısı
eta=0.005;
% 10 epoch (Tüm verilerin eğitimde kullanma sayısı) işlem
for epoch=1:20
    % Her bir veri için bir kere ağırlık güncellesi
uygulanır
    % Bu durum stokastik gradyan iniş optimizasyonu adını
alır
    for j=1:length(X)
      w1=w1-eta*(-34+14*w1+6*b);
      b=b-eta*(-15+6*w1+3*b);
      % mevcut ağırlıklar için yapay sinir ağının çıkışı
hesaplanır
      v=w1*X(j)+b;
      y=v;
      e(j)=D(j)-y; % X(i) giriş için yapay sinir ağü
hatası
      fprintf('w1=%d; b=%d \n', w1,b);
    end
    % Bu epoch için karesel hata
    KareselHata=(1/2)*sum(e.^2);
    fprintf('*** Epoch = %d \n', epoch);
    fprintf('E=%d \n', KareselHata);
    E(epoch)=KareselHata;
end
figure(1)
plot(1:epoch,E)
xlabel('Epoch')
ylabel('E')
grid
% Ağırlık ve bias değerlerleri
fprintf('Optimizasyon Sonucunda Ağırlık,Bias ve Karesel
Hata Değerleri \n');
fprintf('w1=%d; b=%d \n', w1,b);
fprintf('E=%d \n', KareselHata);
```



Optimizasyon Sonucunda Ağırlık, Bias ve Karesel Hata Değerleri

w1=2.188385e+00; b=4.791947e-01

E=6.734910e-02

Görüldüğü gibi stokastik gradyan iniş yöntemi optimizasyonu, her bir veri için bir kere ağırlık güncellemesi yapmış ve karesel hatayı bütün veri kümesi için 0.0673 gibi düzeye çekmiştir.