Makine Öğrenmesi Ders Notları
Hazırlayan: Barış Baykant Alagöz
İnönü Üniversitesi, Bilgisayar Mühendisliği, 2023-2024 Bahar Dönemi Lisans Dersi
Sorularınız ve düzeltme önerileriniz için e-mail: baykant.alagoz@inonu.edu.tr

Yapay Sinir Ağları Temelleri IV

Yapay Sinir Ağlarının Eğitim Algoritmalarının Geri Planındaki Matematik:

Yapay sinir ağlarının matematiksel temelini oluşturan temel konular aşağıda incelenmektedir.

a) *Zincir Kuralı*: Bileşke fonksiyonların (kompozit: iç içe yazılabilen fonksiyonların) türevleri zincir kuralı ile kolaylıkla yazılabilir.

Örneğin $y = f(x), x = g(v), v = h(z)$ bileşke fonksiyonları verilsin,

$y = f(x), x = g(v), v = h(z)$ => $y = f(g(h(z)))$ yazılabilir ve iç içe geçen fonksiyonların birlikte yazımı bileşke fonksiyon ifade eder. $y = f(g(h(z)))$ nun z göre türevini alınmak isteyelim.

Bu fonksiyonların türevleri zincir kuralına göre

$$\frac{\partial y}{\partial z} = \frac{\partial y}{\partial x}\frac{\partial x}{\partial v}\frac{\partial v}{\partial z}$$

yazılabilir.

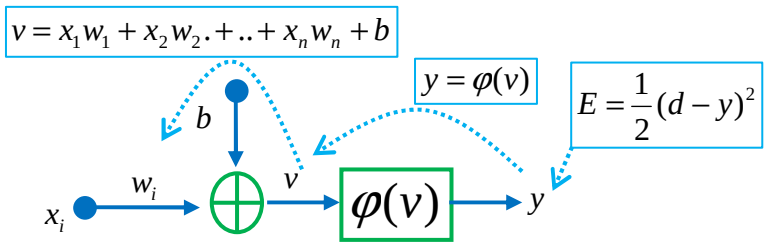
Zincir kuralına yapay sinir ağlarında karesel hatanın fonksiyonun ağırlık katsayısına göre türevini alırken ihtiyaç duyulur. Hatanın, nöron çıkışına göre, nöron çıkışının, ağılıklı toplama göre, ağırlıklı toplamın ağırlığa göre türevi alınır. ($\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial v}\frac{\partial v}{\partial w_i}$) Bunu aşağıda inceleyelim. Ağırlık optimizasyonu için gradyan iniş çözümünü yazalım.

$$w_i \leftarrow w_i - \alpha \frac{\partial E}{\partial w_i}.$$

Burada hatanın ağırlığa göre kısmı türevi $\frac{\partial E}{\partial w_i}$ bulunmalıdır. Hatanın karesel hata olduğunu varsayalım. Aşağıdaki tek sinir hücresi için çıkıştan ağılığa kadar bileşke fonksiyon olduğunu görelim.

$E = \frac{1}{2}(d-y)^2$,

$y = \varphi(v)$,



$v = x_1w_1 + x_2w_2 + x_3w_3 + ... + x_nw_n + b$ bileşke fonksiyonları için zincir kuralı ile

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial v}\frac{\partial v}{\partial w_i}$$

yazılır. Böylece gradyan iniş yöntemi çözümü bileşke fonksiyon ile ifade edilen yapay sinir ağı modellerinde uygulanabilir. Eğitim algoritmaları bu çözümler ile oluşturulur. Bu çözümler ilerleyen bölümlerde delta kuralı olarak adlandırılacaklar.Bu türev ifadesinin bir kısmı delta $\delta_i = \varphi'(v_i)e_i$ olarak ifade edilecektir.

b) ***Gradyan İniş Yöntemi***: Ağırlıkların optimal olarak belirlenmesi için yapay sinir ağlarının eğitiminde gradyan iniş yöntemi yaygın kullanım bulur. Sinir ağları ağırlık güncellemesi için

$$w_i \leftarrow w_i - \alpha \frac{\partial E}{\partial w_i}$$

olarak ifade edilebilir.

Makine öğrenmesinde, eğitim aşamasında gradyan iniş optimizasyonu işlemi 3 farklı şekilde uygulanır:

$T = \{(x_1, d_1), (x_2, d_2), (x_3, d_3), .., (x_p, d_p)\}$ bir eğitim kümesi olsun. Bu eğitim kümesi için yazılan karesel öğrenme hatası gradyan iniş yöntemi ile minimize edilir.

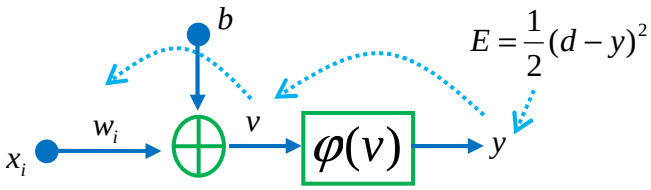
$\min E(w)$

Yukarıda verilen optimizasyon işlemi makine öğrenmesi eğitim algoritmalarında 3 farklı şekilde uygulanır.

(i) *Stokastik Gradyan İniş Çözümü*: Burada eğitim kümesinin her bir verisi için bir kere parametre güncellemesi yapılır. Dolayısı ile p veriden oluşan $T = \{(x_1, d_1), (x_2, d_2), (x_3, d_3), .., (x_p, d_p)\}$ kümesinin tamamı için eğitim süreci p defa parametre güncellemesi gerçekleştirir. Buna stokastik gradyan iniş denir. Bu kümeden seçilen bir veri (x_h, d_h) olsun. Bu tek veri için karesel hata,

$E = \frac{1}{2}(d_h - y)^2$ yazılır. Burada gradyan iniş çözümü $w_i \leftarrow w_i - \alpha \frac{\partial E}{\partial w_i}$ olur.

Varsayalım tek bir sinir hücresi için yazıyoruz.



$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial v}\frac{\partial v}{\partial w_i}$ zincir kuralı yazılırsa,

$\frac{\partial E}{\partial y} = -(d_h - y) = -e_h$, (Çünkü $\frac{\partial E}{\partial y} = \frac{2}{2}(-1)(d_h - y) = -(d_h - y) = -e_h$. Veri için öğrenme hatası $e_h = d_h - y$ alındı.)

$\frac{\partial y}{\partial v} = \varphi'(v)$, (Çünkü $y = \varphi(v)$ => $\frac{\partial y}{\partial v} = \varphi'(v)$)

$\frac{\partial v}{\partial w_i} = x_i$ (Çünkü, $v = x_1 w_1 + x_2 w_2 + x_3 w_3 + ... + x_n w_n + b$ => $\frac{\partial v}{\partial w_i} = x_i$, $i = 1,2,3,,, p$)

$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial v}\frac{\partial v}{\partial w_i} = -e_h.\varphi'(v).x_i$ bu ifade gradyan inişte kullanılırsa,

$w_i \leftarrow w_i - \alpha \frac{\partial E}{\partial w_i}$ => $w_i \leftarrow w_i + \alpha.e_h.\varphi'(v).x_i$

elde edilir.

Formülü sadeleştirmek için $\varphi'(v).e_h$ ifadesine delta δ adı verilir. Böylece delta eğtim kuralı için güncelleme ifadesi elde edilir.

$w_i \leftarrow w_i + \alpha.\delta.x_i$

$\delta = \varphi'(v).e_h$

Bu formülasyona delta öğrenme kuralı denmiştir. Yapay sinir ağlarının eğitiminde yaygın olarak stokastik gradyan iniş yöntemi yani delta kuralı uygulanır.

(ii) *Yığın mod (Batch-mode) Gradyan İniş Çözümü*: Burada eğitim kümesinin tamamı için karesel hata yazılır ve tamamı için bir defa parametre güncellemesi yapılır. Dolayısı ile p veriden oluşan $T = \{(x_1, d_1), (x_2, d_2), (x_3, d_3), .., (x_p, d_p)\}$ kümesi tamamı için sadece bir defa parametre güncellenir.

Tek sinir hücresi modeli üzerinde uygulayalım. Bütün eğitim kümesi için karesel hatayı yazalım:

$$E = \frac{1}{2}\sum_{j=1}^{p} e_i^2 = \frac{1}{2}\sum_{j=1}^{p}(d_j - y)^2 = \frac{1}{2}(d_1 - y)^2 + \frac{1}{2}(d_2 - y)^2 + ... + \frac{1}{2}(d_p - y)^2$$

için gradyan iniş yazılırsa $w_i \leftarrow w_i - \alpha \frac{\partial E}{\partial w_i}$. Burada zincir kuralı $\frac{\partial E}{\partial w_i}$ için yazılabilir.

$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial v}\frac{\partial v}{\partial w_i}$ zincir kuralı yazılırsa,

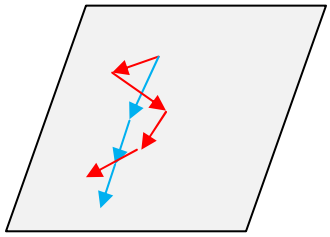
$\frac{\partial E}{\partial y} = \frac{1}{2}\sum_{j=1}^{p} 2.(-1)(d_j - y) = -\sum_{j=1}^{p}(d_j - y) = -\sum_{j=1}^{p} e_j$,

$\frac{\partial y}{\partial v} = \varphi'(v)$, $\frac{\partial v}{\partial w_i} = x_i$

$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial v}\frac{\partial v}{\partial w_i} = -(\sum_{j=1}^{p} e_j).\varphi'(v).x_i$ bu gradyan inişte kullanılırsa,

$$w_i \leftarrow w_i + \alpha.(\sum_{j=1}^{p} e_j).\varphi'(v).x_i$$

Not: Yığın-mod gradyan inişte eğitim kümesindeki bütün girişler tek tek sinir hücresine uygulanır ve hepsi için ağın çıkış değerleri bulunur. Eğitim kümesinin tamamı için toplam eğitim hatası ($\sum_{j=1}^{p} e_j$) hesaplandıktan sonra bir defa ağırlık güncellemesi yapılır. Dolayısı ile yığın-mod eğitimin işlem yükü, p adet öğrenme hatası (e_i) hesaplaması neden ile bir adet hata hesaplanan (e_h) stokastik gradyan iniş yönteme göre daha fazladır. Ancak, yığın-mod eğitimde bütün veriler dikkate aldığı için optimal çözüme yakınsama performansı stokatik gradyan iniş yöntemine göre daha iyi olması beklenir. Sonuçta, yığın-mod gradyan iniş yöntemi ağırlık güncelleme başına işlem maliyetini artırır ancak yakınsama hızı iyileştirir çünkü her parametre güncellenmesinde bütün verileri dikkate aldığı için optimal noktaya daha kolay(az güncellemede) ulaşabilir. Stokastik gradyan iniş her parametre güncellemesini sadece bir veriye göre yapar ve optimal noktaya daha fazla güncelleme ile ulaşmasına yol açabilir. Aşağıda lokal minimuma zigzag şeklinde inen stokastik gradyan iniş adımları olabilir. Çünkü, bir veri başına gradyan (türevi) dikkate alır. Mavi adımlar ve daha düz inen yığın mod gradyan iniş olabilir. Çünkü, bütün verileri dikkate alır ve daha isabetli adım atar.



(iii) *Mini-yığın mod (Mini Batch-mode) Gradyan İniş Çözümü*: Burada eğitim kümesinin bir parçası(kısmı) için karesel hata yazılır ve parametre güncellemesi yapılır. Dolayısı ile p veriden oluşan $T = \{(x_1, d_1), (x_2, d_2), (x_3, d_3), .., (x_p, d_p)\}$ kümesinde m adet veriden oluşan bir altkümesi için bir defa parametre güncellenir. Burada $1 < m < p$'dir.

Tek sinir hücresi modeli üzerinde uygulanırsa,

$E = \frac{1}{2}\sum_{j=1}^{m} e_j^2 = \frac{1}{2}\sum_{j=1}^{m} (d_j - y)^2 = \frac{1}{2}(d_1 - y)^2 + \frac{1}{2}(d_2 - y)^2 + ... + \frac{1}{2}(d_m - y)^2$ için gradyan iniş yazılırsa $w_i \leftarrow w_i - \alpha \frac{\partial E}{\partial w_i}$,

$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial v}\frac{\partial v}{\partial w_i}$ zincir kuralı yazılırsa,

$$\frac{\partial E}{\partial y} = \frac{1}{2}\sum_{j=1}^{p} 2.(-1)(d_j - y) = -\sum_{j=1}^{m}(d_j - y) = -\sum_{j=1}^{m} e_j \ , \ \frac{\partial y}{\partial v} = \varphi'(v) \ \ , \frac{\partial v}{\partial w_i} = x_i$$

$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial v}\frac{\partial v}{\partial w_i} = -\sum_{j=1}^{m} e_j.\varphi'(v).x_i$ bu gradyan inişte kullanılırsa,

$$w_i \leftarrow w_i + \alpha.(\sum_{j=1}^{m} e_j).\varphi'(v).x_i$$

Not: Mini yığın-mod gradyan inişde, yığın-mod gradyan iniş ile stokastik gradyan iniş arasında bir işlem yükü ve yakınsama performansı elde edilebilir. Bunun için mini-yığın mod gradyan inişin için m $(1 < m < p)$ ile ayarlanabilir. Aşağıda tabloda p adet veriye sahip eğitim kümesi için kıyaslama:

Gradyan İniş Yöntemleri	*Açıklama*	*Ağırlık Güncelleme*	*Güncelleme başına performans*
Stokastik GD	Bir veri için bir kere ağırlık güncellenir.	$w_i \leftarrow w_i + \alpha.e_h.\varphi'(v).x_i$	Az işlem maliyetli fakat yakınsama performansı düşebilir.
Yığın-mod GD	Bütün eğitim kümesi için bir kere ağırlık güncellenir.	$w_i \leftarrow w_i + \alpha.(\sum_{j=1}^{p} e_j).\varphi'(v).x_i$	Yüksek işlem maliyetli fakat yakınsama performansı iyi olabilir.
Mini yığın-mod GD	$1 < m < p$ aralığında m adet veri için bir kere ağırlık güncellenir.	$w_i \leftarrow w_i + \alpha.(\sum_{j=1}^{m} e_j).\varphi'(v).x_i$	m veri sayısına bağlı olarak diğer iki yöntem arasında bir performans sağlayabilir.

c) ***Sigmoid fonksiyonun türevi*** : $\varphi(v) = \frac{1}{1+e^{-v}}$ sigmoid fonksiyonun türevi,

$\frac{d\varphi(v)}{dv} = \varphi(v) - \varphi(v)^2$ dolayısı ile

$\varphi'(v) = \varphi(v)(1 - \varphi(v))$ yazılabilir.

İspat: [Aşağıda türevin $\left(\frac{f}{g}\right)' = \frac{f'g - g'f}{g^2}$ özelliği kullanılır.]

$\varphi(v) = \frac{1}{1+e^{-v}}$ =>

$$\frac{d\varphi(v)}{dv} = \frac{0.(1+e^{-v}) - (-1)e^{-v}}{(1+e^{-v})^2} = \frac{e^{-v}}{(1+e^{-v})^2} = \frac{-1+1+e^{-v}}{(1+e^{-v})^2}$$

$$= \frac{1+e^{-v}}{(1+e^{-v})^2} - \frac{1}{(1+e^{-v})^2} = \frac{1}{1+e^{-v}} - \frac{1}{(1+e^{-v})^2}$$

$$= \frac{1}{1+e^{-v}} - \frac{1}{(1+e^{-v})^2} = \varphi(v) - \varphi(v)^2 = \varphi(v)(1-\varphi(v))$$

$$\varphi'(v) = \varphi(v)(1-\varphi(v))$$

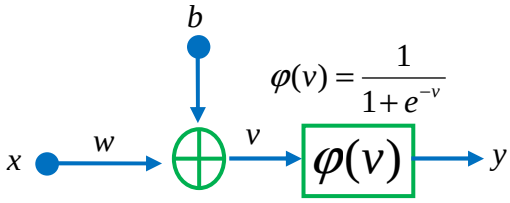
Not: Böylece bilgisayarda türev hesaplamasında işlem yükü çok düşer. Çünkü sadece $\varphi(v)$ değeri bir kez hesaplanırsa, türev gereken her yerde $\varphi'(v) = \varphi(v)(1-\varphi(v))$ şeklinde kullanabilir. Bu her defasından $\frac{d\varphi(v)}{dv} = \frac{1}{1+e^{-v}} - \frac{1}{(1+e^{-v})^2}$ hesaplamasına göre daha düşük işlem maliyetlidir.

Örnek: Aşağıda tek girişli bir nöron için yapay sinir ağının eğitimini sağlayan ağırlık (w) ve bias (b) katsayılarının stokastik gradyan iniş çözünü elde ediniz. Aktivasyon fonksiyonu olarak sigmoid seçilmiştir. ($\varphi(v) = \frac{1}{1+e^{-v}}$) Eğitim setinde p adet veri olduğunu varsayalım.

$T = \{(x_1, d_1), (x_2, d_2), (x_3, d_3), .., (x_p, d_p)\}$ eğitim kümesinden her bir veri için ayrı ayrı ağırlık güncellemesi için stokastik gradyan iniş (tek verili güncelleme) uygulayalım. Bu kümenin bir (x_h, d_h) verisi için hata

$$E = \frac{1}{2}(d_h - y)^2$$

Tek girişli nöron için ağırlıklı toplam



$$v = xw + b$$

$$y = \varphi(v)$$

Ağırlıkların belirlenmesi için minimize edilecek eğitim hatası

$$\min E = \frac{1}{2}(d_h - y)^2$$

Optimizasyon problemini bir (x_h, d_h) verisi için stokastik gradyan iniş yöntemi ile w ve b çözümlerini yazalım.

$w \leftarrow w - \alpha \frac{dE}{dw}$

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

* Önce $w \leftarrow w - \alpha \frac{dE}{dw}$ çözümleyelim. Bunun için $\frac{dE}{dw}$ çözmek için zincir kuralını yazalım.

$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial v}\frac{\partial v}{\partial w}$ hesaplanmalıdır.

$\frac{\partial E}{\partial y} = -(d_h - y) = -e_h$ (çünkü karesel hata ifadesinde bir (x_h, d_h) verisi için ağırlık güncelliyoruz. Dolayısı ile $E = \frac{1}{2}(d_h - y)^2$ idi.)

$\frac{\partial y}{\partial v} = \varphi'(v) = \varphi(v)(1-\varphi(v))$ (çünkü burada $y = \varphi(v) = \frac{1}{1+e^{-v}}$ sigmoid fonksiyonu, türevi bir önceki sayfada çıkarılmıştı.)

$\frac{\partial v}{\partial w} = x$ (çünkü burada $v = xw + b$)

Türevler zincir kuralında değerlendirilirse,

$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial v}\frac{\partial v}{\partial w} = -e_h.\varphi(v)(1-\varphi(v)).x$ elde edilir. Bu gradyan iniş çözümünde kullanılarak ağırlık güncelleme ifadesi,

$w \leftarrow w - \alpha \frac{dE}{dw}$ => $\boxed{w \leftarrow w + \alpha e_h.\varphi(v)(1-\varphi(v)x}$ elde edilir.

Buradan ileride detaylı göreceğimiz delta kural şöyle geliştirilir.

$\delta = \varphi'(v)e$ ifadesine delta δ denir. Bu durumda ağırlık güncellemesi delta kuralına göre

$\boxed{w \leftarrow w + \alpha\delta.x}$, burada $\boxed{\delta = \varphi'(v)e = e_h.\varphi(v)(1-\varphi(v))}$ alınmıştır.

Görüldüğü üzere aslında delta kuralı temelde gradyan iniş çözünün δ parametresi tanımı ile daha sade ifadesi sonucunda ortaya konmuştur.

Şimdi aynı işlem adımlarını bias güncellemesi $b \leftarrow b - \alpha \frac{dE}{db}$ için çözelim.

Bunun için $\frac{dE}{db}$ çözmek için zincir kuralını yazalım.

$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial v}\frac{\partial v}{\partial b}$ hesaplanmalıdır.

$\frac{\partial E}{\partial y} = -(d_h - y) = -e_h$ (yukarıda açıklandı)

$\frac{\partial y}{\partial v} = \varphi'(v) = \varphi(v)(1-\varphi(v))$ (yukarıda açıklandı)

$\frac{\partial v}{\partial b} = 1$ (çünkü burada $v = xw + b$)

Türevler zincir kuralında değerlendirilirse,

$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial y}\frac{\partial y}{\partial v}\frac{\partial v}{\partial w} = -e_h.\varphi(v)(1-\varphi(v).1 = -e_h.\varphi(v)(1-\varphi(v))$ elde dilir.

Bu gradyan iniş çözümünde kullanılarak bias güncelleme ifadesi,

$b \leftarrow b - \alpha\frac{dE}{db}$ => $b \leftarrow b + \alpha e_h.\varphi(v)(1-\varphi(v))$ elde edilir.

Yukarıda $\delta = \varphi'(v)e = e_h.\varphi(v)(1-\varphi(v))$ ifadesine delta δ denmişti. Bu durumda bias güncellemesi delta kuralına göre

$b \leftarrow b + \alpha\delta$ elde edilir.

Bu ağın tek bir (x_h, d_h) verisi için güncelleme ifadeleri,

$$w \leftarrow w + \alpha\delta.x$$
$$b \leftarrow b + \alpha\delta$$
$$\delta = e_h.\varphi(v)(1-\varphi(v)$$

elde edilmiştir.

Not: Bu tek sinir hücresinin matematiksel modelinin aslında bir logistik regresyon modeline karşılık geldiğini görünüz. Logistik regresyon modeli $C(x) = \frac{1}{1+e^{-mx-b}}$ ile ifade edilmişti. Dolayısı ile bu çözüm aynı zamanda stokastik gradyan iniş ile lojistik regresyen problemi çözümünü ifade eder. Bu tek bir sinir hücresinin lojistk regresyon modelini üretebileceğini gösterir.

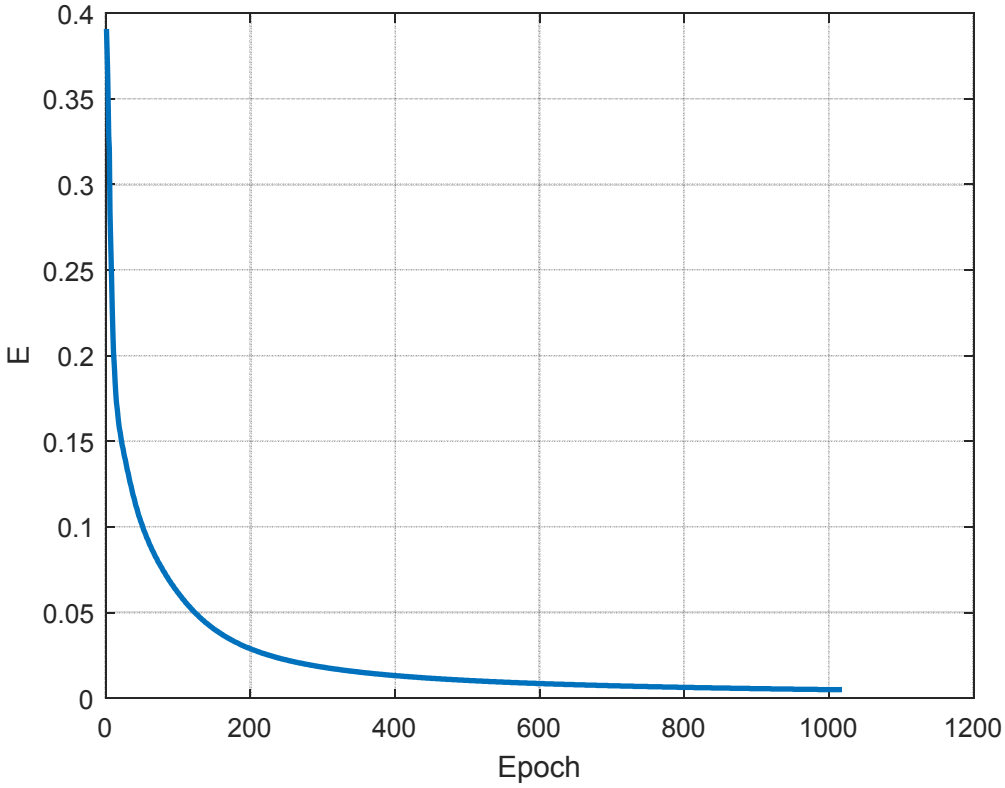
Daha büyük sinir ağları için benzer şekilde zincir kuralı ile çözümleme yapılabilir. Ancak problemin çözüm formülasyonunun karmaşıklığı, ağın katman sayısı artışı ile aşırı yükselir. Her yeni ağ modeli durumda zincir kuralı yardımı ile elle çözmek yerine, bu çözümler geriye yayılım (backpropagation) yöntemi adı verilen hatanın geriye yayılımı gibi düşünerek bilgisayarda hesaplamaya uygun bir genel formülasyon geliştirilmiştir. Bu genelleştirilmiş çözüm şekli geriye yayılım algoritması (backpropagation) olarak isimlendirilmiştir. Programcılar geriye yayılım algoritmasını doğrudan kullanırlar. Ancak, bu algoritmanın

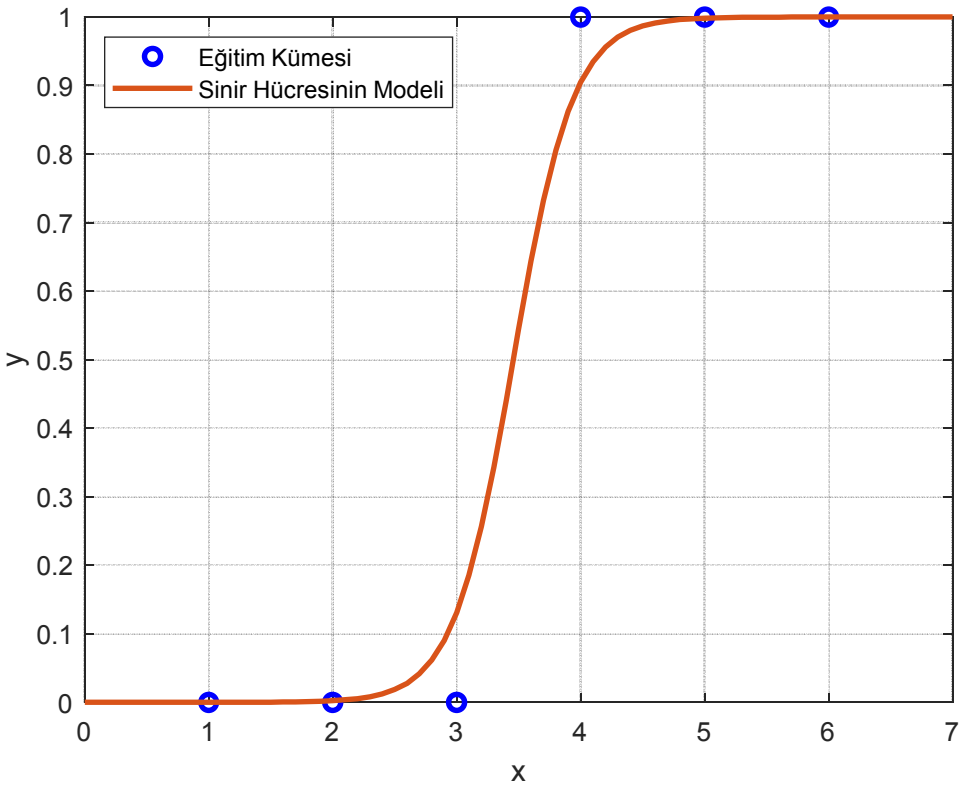
temeli bu basitleştirilmiş örnekte görüldüğü üzere gradyan iniş çözümleri zincir kuralı yardımı ile elde edilir.

Yukarıdaki yapay sinir ağı örneğinin uygulamasını Matlab'da yapalım:

```
clear all;close all
% Tek girişli bir sinir hücresinin ikili sınıflama problemi için
eğitimi
x=[1 2 3 4 5 6];
d=[0 0 0 1 1 1];
% Öğrenme katsayısı
etaw=1;
etab=1;
% Ağırlıkların başlangıç değerleri rastgele
w=rand;
b=rand;
MaxEpoch=1200;
% Eğitime başlanıyor
h=0;
for i=1:MaxEpoch % Epoch (dönem sayısı)
    % Tokastik gradyan iniş  uygulanır.
    % Her veri için bir kez ağırlık güncellenir.
    for h=1:length(x)% Veri indisi
    v=w*x(h)+b;
    AktivasyonFonk=1/(1+exp(-v));
    y=AktivasyonFonk;
    e(h)=d(h)-y;
    % her bir veri için ağırlık güncelleme  (stochastic gradyan iniş)
    delta=e(h)*AktivasyonFonk*(1-AktivasyonFonk);
    w=w+etaw*delta*x(h);
    b=b+etab*delta;
    end
    % Ortalama karesel hata durdurma kriteri
         E(i)=(1/length(e))*sum((e).^2);
    % Durdurma kriteri
    if E(i)<0.05
        fprintf('Durdurma krite sağlandı \n');
        break;
    end
end
% Eğitilmiş sinir hücresi modelinin cevabını çizmek
% için x giriş etrafında  çıkışları
x1=min(x)-1:0.1:max(x)+1;
v=w*x1+b;
y=1./(1+exp(-v));

figure(1)
plot(x,d,'ob',x1,y,'LineWidth',2)
xlabel('x')
ylabel('y')
legend('Eğitim Kümesi','Sinir Hücresinin Modeli')
grid
figure(4)
plot(1:i,E,'LineWidth',2)
xlabel('Epoch')
```


0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
E
0
200
400
600
800
1000
1200
Epoch


Eğitim Kümesi
Sinir Hücresinin Modeli
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
y
0
1
2
3
4
5
6
7
x