Çok katmanlı ileri sürümlü YSA'da standart geri yayıyım ve momentum geri yayılım algoritmalarının karşılaştırılması

(Eğitim/Hata geri yayılım)

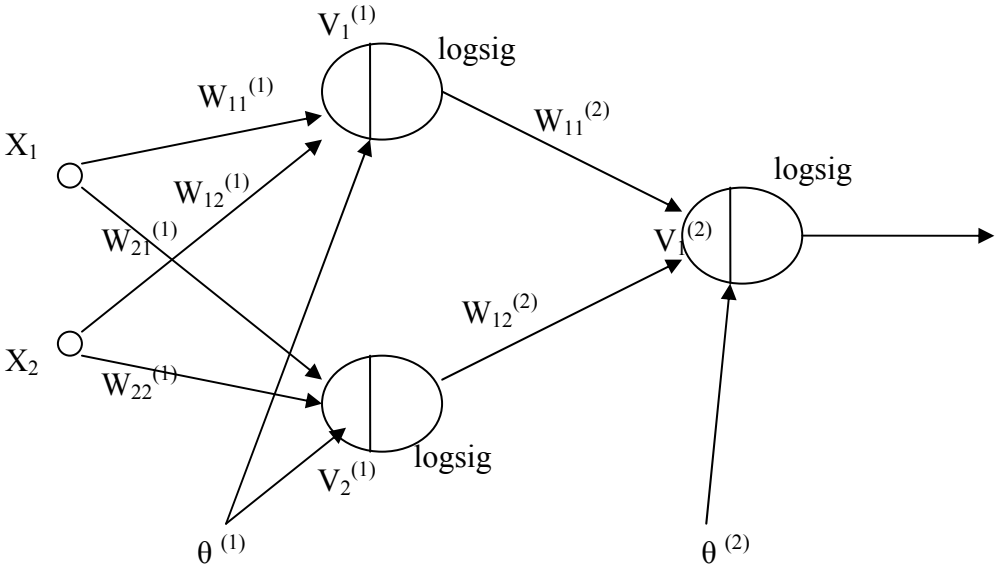
Halim Cem KEFELI
www.cemkefeli.com

Özetçe

Bu çalışmada çok katmanlı ve ileri sürümlü bir YSA' da geri hesaplama için iki algoritma yapısı incelenmiş ve aşama aşama bu algoritmaların özellikleri verilmiştir. Çalışma boyunca kullanımının kolaylığından dolayı MATLAB™ programı kullanılmış ve algoritmanın kodları MATLAB™ program dilinde yazılmıştır.

1. Giriş

Bu çalışmada özel-veya (EX-OR) işlevini gerçekleyen {2,1} yapısında sigmaoidal aktivasyonlu bir YSA' nın geri yayılım ile eğitilmesi konusunda iki yöntem incelenmiştir. Yöntemler standart geri yayılım ve momentum geri yayılım algoritmalarıdır. YSA' nın başlangıç parametreleri ağa rasgele olarak gösterilmiş ve giriş-çıkış örnekleri ağa rasgele olarak verilmiştir. Tüm çalışma boyunca maliyet fonksiyonu(Cost Function) olarak 4 iterasyon öncesine ait hataların karesel ortalaması kullanılmıştır. Maksimum iterasyon sayısı olarak 1000 değeri seçilmiştir. Şekil 1 'de incelenen YSA için ağ yapısı gösterilmektedir.



Şekil 1 :İncelenen YSA için ağ yapısı

YSA giriş ve çıkış katmanlarından oluşmaktadır ve YSA için gizli katman bulunmamaktadır. Giriş katmanı iki hücreden ve çıkış katmanı da bir hücreden oluşmaktadır. Tüm YSA için kullanılacak giriş parametreleri için eşitlikler denklem 1.1 - 1.5 arasında verilmiştir. Bu eşitliklerde W'lar kollardaki ağırlık parametrelerini, θ lar her bir katman için sıfırıncı dereceden bağımsız değişkenin katsayısını, X ise hali hazırda ağ girişinde bulunan giriş sinyalini göstermektedir.

$$W^{(1)} = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix}_{2X2} \quad \text{(denklem.1.1)}$$

$$W^{(2)} = \begin{bmatrix} W_{11} & W_{21} \end{bmatrix}_{1X2} \quad \text{(denklem.1.2)}$$

$$\theta^{(1)} = \begin{bmatrix} \theta_1 \\ \theta 2 \end{bmatrix} \quad \text{(denklem1.3)}$$

$$\theta^{(2)} = \begin{bmatrix} \theta_1 \end{bmatrix} \quad \text{(denklem.1.4)}$$

$$X = \begin{bmatrix} X_1 & X_2 \end{bmatrix} \quad \text{(denklem.1.5)}$$

1.1 Çok Katmanlı YSA İçin İleri Hesaplama

Çok katmanlı YSA' da bir hücre için çıkış değerinin bulunması temel olarak üç aşamadan oluşmaktadır. İlk aşama olarak hücrelere giriş yapan kollardaki sinyaller ile bu sinyallerin yol aldığı kollardaki ağırlık katsayıları çarpılarak bir değer elde edilmektedir. İkinci aşama olarak elde edilen bu değer bir sabit sayı ile toplanarak sıfırıncı dereceden bağımsız değişkenin katsayısı denkleme dahil edilmektedir. Bu aşamada elde edilen değer 'V' sembolü ile gösterilmektedir. Son aşama olarak ise elde edilen bu değerler toplamı aktivasyon fonksiyonu adı verilen belirli bir fonksiyondan geçirilerek hücre çıkışı elde edilmektedir. Son aşama ile elde edilen değer 'Y' sembolü ile gösterilmektedir. Genel olarak sinüs ve kosinüs fonksiyonları elde etme kolaylığından dolayı sıkça tercih edilmektedir bununla beraber bu çalışma boyunca sigmaoidal bir aktivasyon fonksiyonunun kullanılması uygun görülmüştür. Çalışma boyunca kullanılan ve ileri hesaplama için gerekli olan eşitlikler denklem 1.6 – 1.11 arasında verilmektedir. Denklemlerde kullanılan Ψ o hücre için kullanılan aktivasyon fonksiyonunu göstermektedir.

$$V^{(1)} = \left[W^{(1)^T} \times X^T + \theta^{(1)} \right] \quad \text{(denklem.1.6)}$$

$$V^{(1)} = \left[\begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix}_{2X2}^T \times \begin{bmatrix} X_1 & X_2 \end{bmatrix}^T + \begin{bmatrix} \theta_1 \\ \theta 2 \end{bmatrix} \right] \quad \text{(denklem.1.7)}$$

$$V^{(2)} = \left[W^{(2)^T} \times Y^{(1)} + \theta^{(2)} \right] \quad \text{(denklem.1.8)}$$

$$V^{(2)} = \left[\begin{bmatrix} W_{11} & W_{21} \end{bmatrix}_{1X2}^T \times Y^{(1)} + \theta^{(2)} \right] \quad \text{(denklem.1.9)}$$

$$Y^{(1)} = \Psi\left(V^{(1)}\right) \quad \text{(denklem.1.10)}$$

$$Y^{(2)} = \Psi\left(V^{(2)}\right) \quad \text{(denklem.1.11)}$$

1.2. Çok Katmanlı YSA İçin Standart Geri Yayılım Algoritması

Çok katmanlı YSA' da ileri hesaplama yöntemi ile bir tek iterasyon için bir YSA çıkışı elde edilmektedir. İleri hesaplama boyunca sistem parametreleri ile ilgili herhangi bir güncelleme ya da değiştirme yapılmamaktadır. Fakat sabit ,hiç değişmeyen, başlangıç değerleri ile eğitilen bir ağın tüm olası giriş değerleri için makul sonuçlar verememesi gerçeği bulunmaktadır. Dolayısı ile yolların ağırlık katsayılarının güncellenmesi ile bu tekdüze ağ yapısından uzaklaşılmaktadır. YSA için geri hesaplama aşamasında bahsi geçen bu nedenlerden dolayı belirli kurallara bağlı kalınarak parametre güncellemesi yapılmaktadır. Geri besleme için önerilen akış şeması şekil 2'de verilmiştir. Çalışma boyunca kullanılan ve geri hesaplama için gerekli olan eşitlikler denklem 1.12 – 1.18 arasında verilmektedir. Bu denklemlerde Y_a sistem çıkışını, Y_d istenen sitem çıkışını göstermektedir. $e(n)$ Y_d ile Y_a arasındaki mutlak olmayan fark değerini belirtmektedir. $\varepsilon(n)$ son dört iterasyon için karesel ortalama hatayı vermektedir. $\varepsilon(n)$ değeri sayesinde ağ yapısı çıkışındaki hata, belirli bir eşik değerindeki hata seviyesinden aşağıya düştüğü zaman sistemin eğitilmiş olduğu sonucuna varılmaktadır. δ lar ilgili katman için gradyent sinyallerini göstermektedir. ∇W ağırlık parametrelerinin güncellenme miktarını saklamaktadır. Öğrenme katsayısı olarak isimlendirilen η ise ∇W parametresinin oluşturulması sırasında kullanılmaktadır ve ∇W parametresinin güncelleme değeri için bir sınırlayıcı görevi görmektedir.

$$Y_a = Y_2 \qquad \text{(denklem.1.12)}$$

$$e(n) = Y_d - Y_a \qquad \text{(denklem.1.13)}$$

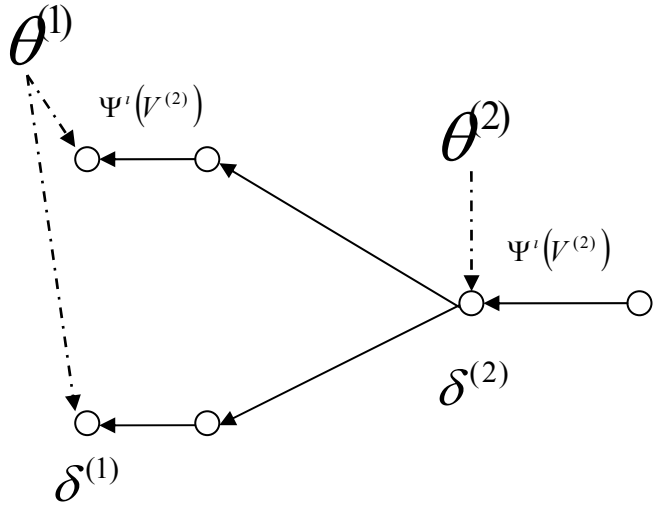
$$\varepsilon(n) = \frac{\left(e(n)^2 \quad + \quad e(n-1)^2 \quad + e(n-2)^2 \quad + \quad e(n-3)^2\right)}{4} \qquad \text{(denklem.1.14)}$$

$$\delta^{(2)} = e(n).\Psi^1\left(V^{(2)}\right) \qquad \text{(denklem.1.15)}$$

$$\nabla W_{ji} = \eta.\delta_j(n).yi(n) \qquad \text{(denklem.1.16)}$$

$$W^{(2)} = W^{(2)} + \nabla W_2 \qquad \text{(denklem.1.17)}$$

$$\delta^{(1)} = \Psi^1\left(V^{(2)}\right)\left[w^{(2)^T} * \delta^{(2)}\right] \qquad \text{(denklem.1.18)}$$



Şekil 2 :İncelenen YSA için geri besleme

1.2.1 Standart Geriye Yayılım Algoritması İçin Yazılan Program Kodları

```
clc;
clear all;
close all;
DUYARLILIK=0.00001;
%W1=[-0.88 0.05 ; -0.02 0.08]
%W2=[0.07 ; -0.06]
%TETA1=[0 ; 0]
%TETA2=[0]
W1=rand(2)/10;                  % Giriş katmanı için rastgele ağırlık parametreleri
W2=rand(2,1)/100;               % Çıkış katmanı için rastgele ağırlık parametreleri
TETA1=rand(2,1)/100;            % Giriş katmanı için rastgele sabit sayılar
TETA2=rand(1)/100;              % Çıkış  katmanı için rastgele sabit sayılar
GIRISLER=[1 1 ; 0 0 ; 0 1 ; 1 0];  % İkili sayı fırmatında giriş değerleri
YDLER=[0 0 1 1];                % İkili sayı fırmatındaki bu giriş değerli için arzu edilen çıkış değerleri
NU=1;                           % Öğrenme katsayısı
i=1;
while( i<1000 )                 % Toplamda 250 tur (epoch) için
   for sayi=1:1:4
      V1=W1'*GIRISLER(sayi,:)'+TETA1;        % Formüllere uygun olarak gerekli denklemler
      % işleniyor ve sonuçlar elde ediliyor
      Y1=logsig(V1);
      V2=W2'*Y1+TETA2;
      Y2=logsig(V2);
      Ya=Y2;
      e(i)=YDLER(sayi)-Ya;
      if(i>=4)
         EPSILON(i)=(e(i)^2+e(i-1)^2+e(i-2)^2+e(i-3)^2)/4;
         % Son dört iterasyon için hataların karesel ortalama değeri elde ediliyor.
         plot(EPSILON);
         %  Her aşama için bulunan hata değerleri ve karesel ortalama hata değerleri ekrana bastırılıyor.
      end
      delta2=e(i)*logsig(1-logsig(V2));      % Geri hesaplama için delta değerleri bulunuyor
      deltaw2=NU*delta2*Y1;
      yeni_W2=W2+deltaw2;                    % ve bulunan bu delta değerleri yardımıyla yeni ağırlık
      % katsayıları belirleniyor.
      delta1=logsig(1-logsig(V1)).*(delta2*W2);
      deltaw1=NU.*(delta1*GIRISLER(sayi,:));
      yeni_W1=W1+deltaw1;
      W1=yeni_W1;                            %  Bulunan yeni ağırlık katsayıları güncelleniyor.
      W2=yeni_W2;
      i=i+1;
   end
end
```

1.3. Çok Katmanlı YSA İçin Momentum Geri Yayılım Algoritması

Çok katmanlı YSA' da bir başka geri yayılım yöntemi olan Momentum geri yayılım algoritması aslında işleyiş bakımından standart geriye yayılım algoritmasından çok büyük farklılıklar barındırmamaktadır. Standart geri yayılım algoritmasına ek olarak ağırlık katsayılarının güncellenebilmesi için alfa (α) ve beta (β) gibi iki değere gereksinim duyulmaktadır. Algoritmayı standart geri yayılımdan ayıran en önemli özellik ise ağırlık katsayısı güncellemesi yapılırken iki iterasyon öncesine ait ağırlık katsayısı değerlerinin de kullanılmasıdır. Momentum geri yayılım algoritması için güncelleme eşitlikleri denklem 1.19 – 1.20 arasında verilmiştir.

$$\nabla W_{ji} = \eta.\delta_j(n).yi(n) + \alpha\Delta W_{ji}(n-1) + b\Delta W_{ji}(n-2) \qquad \text{(denklem.1.19)}$$

alfa (α) ve beta (β) : Sabit reel sayılar (denklem.1.20)

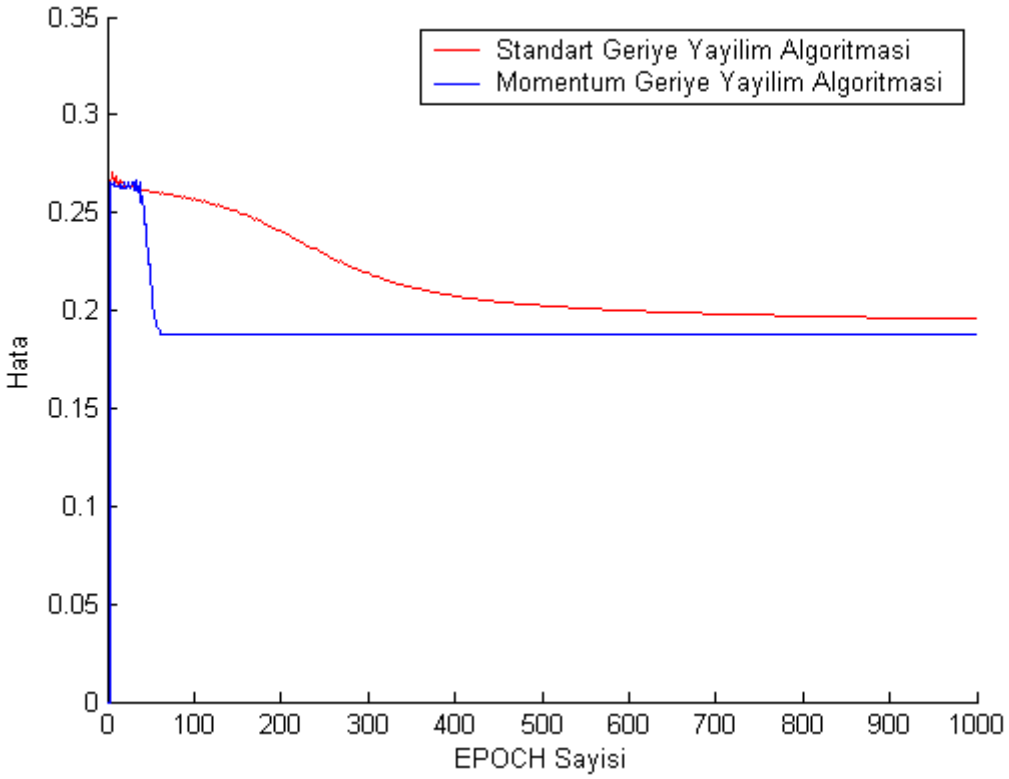
1.3.1 Momentum Geriye Yayılım Algoritması İçin Yazılan Program Kodları

```
clc;
clear all;
close all;
DUYARLILIK=0.00001;
W1=[-0.88 0.05 ; -0.02 0.08]
W2=[0.07 ; -0.06]
TETA1=[0 ; 0]
TETA2=[0]
% W1=rand(2)/10;                 % Giriş katmanı için rastgele ağırlık parametreleri
% W2=rand(2,1)/100;               % Çıkış katmanı için rastgele ağırlık parametreleri
% TETA1=rand(2,1)/100;            % Giriş katmanı için rastgele sabit sayılar
% TETA2=rand(1)/100;              % Çıkış  katmanı için rastgele sabit sayılar
GIRISLER=[1 1 ; 0 0 ; 0 1 ; 1 0];  % İkili sayı fırmatında giriş değerleri
YDLER=[0 0 1 1];                % İkili sayı fırmatındaki bu giriş değerli için arzu edilen çıkış değerleri
NU=1;                           % Öğrenme katsayısı
i=1;
beta=0.1;                  % Momentum geri yayılımın tipik özelliği olan alfa ve beta katsayıları
alfa=1;

while( i<1000 )
   for sayi=1:1:4
      V1=W1'*GIRISLER(sayi,:)'+TETA1;
      Y1=logsig(V1);
      V2=W2'*Y1+TETA2;
      Y2=logsig(V2);
      Ya=Y2;
      e(i)=YDLER(sayi)-Ya;
      if(i>=4)
         EPSILON(i)=(e(i)^2+e(i-1)^2+e(i-2)^2+e(i-3)^2)/4;
         % Son dört iterasyon için hataların karesel ortalama değeri elde ediliyor.
         plot(EPSILON);
         %  Her aşama için bulunan hata değerleri ve karesel ortalama hata değerleri ekrana bastırılıyor.
      end
      delta2=e(i)*logsig(1-logsig(V2));
      deltaw2=NU*delta2*Y1;
      yeni_W2=W2+deltaw2;
      delta1=logsig(1-logsig(V1)).*(delta2*W2);
      deltaw1=NU.*(delta1*GIRISLER(sayi,:));
      yeni_W1=W1+deltaw1;
      if( i>2 )
         % Momentum geri yayılım kullanılarak ağırlık parametreleri güncelleniyor
         W1=deltaw1 + alfa*W1_SAKLA(:,:,i-1) + beta*W1_SAKLA(:,:,i-2);
         W2=deltaw2 + alfa*W2_SAKLA(:,:,i-1) + beta*W2_SAKLA(:,:,i-2);
         W1_SAKLA(:,:,i)=W1;
         W2_SAKLA(:,:,i)=W2;
      else
         W1=yeni_W1;
         W2=yeni_W2;
         W1_SAKLA(:,:,i)=W1;
         W2_SAKLA(:,:,i)=W2;
      end
      i=i+1;
   end
end
```

2. Sonuçlar

Çok katmanlı YSA' da geri yayılım kullanılarak ağın makul parametreler ile çalışması sağlanmaktadır. Böylece başlangıçta uygun olmayan değişken değerleri geri yayılım algoritması sayesinde kolayca makul değerlere çekilebilmektedir. Standart Geri Yayılım ve Momentum Geri Yayılım Algoritmaları ile elde edilen hata grafikleri Şekil.3'te verilmektedir. Grafikten Momentum geri yayılım algoritmasının başarımının standart geri yayılım algoritmasına göre daha iyi olduğu söylenebilmektedir. Fakat bununla beraber momentum geri yayılımda herhangi bir iterasyon aşamasında hatalı bir güncelleme yapılırsa, sistem boyunca bu hatanın etkisi hatalı geri beslemeden dolayı görülmektedir.



Şekil 3 :İncelenen YSA için farklı algoritmaların karesel ortalama hata değerleri

Çok katmanlı ileri sürümlü YSA'da standart geri yayıyım ve momentum geri yayılım algoritmalarının karşılaştırılması

(Eğitim/Hata geri yayılım)

Halim Cem KEFELI
www.cemkefeli.com