

# BOĞAZIÇI ÜNİVERSİTESİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜNDE YAPAY SINIR AĞLARI ÜSTÜNE YAPILAN ARAŞTIRMALAR

H.Levent Akın, Ethem Alpaydın ve Fikret Gürgen  
Boğaziçi Üniversitesi Bilgisayar Mühendisliği Bölümü, 80815, Bebek-İstanbul

## 1.GİRİŞ

Bölümümüzde Yapay Sinir Ağları (YSA) üstüne çalışmalar 1989 yılında başlamıştır. Bu çalışmalar,

- Kuramsal temeller ve
  - Uygulamalar
- üstünde yoğunlaşmıştır.

Kuramsal temeller üstündeki araştırmalar

- Ağ yapıları,
  - Öğrenme yöntemleri ve
  - Klasik yöntemlerle bellek gereksinimi, öğrenme hızı ve başarısı kıstaslarında karşılaştırma
- yönlerindedir.

Uygulamalar ise

- Örüntü tanıma
  - Denetim
  - Beceri öğrenme
- konularındadır.

YSA üstüne çalışan grupta halen üç tam zamanlı öğretim üyesi ve dört araştırma görevlisi çalışmaktadır. Bu çalışmalar TÜBİTAK EEEAG ve Boğaziçi Üniversitesi Araştırma Fonu tarafından desteklenmektedir. Ayrıca YSA üzerine tamamlanmış ve devam eden yüksek lisans tezleri ve bitirme projeleri de vardır.

## 2. YSA'NIN KURAMSAL TEMELLERİ ÜSTÜNE ARAŞTIRMALAR

Bir YSA'nın kalitesini gösteren üç kıstas vardır:

1. *Deneme kümesi üzerindeki başarısı*, ağın öğrenme sırasında karşılaşmamış olduğu örüntülere nasıl bir genelleştirme gösterdiğini belirtir.
2. *Ağın karmaşıklığı*, ağ büyüklüğüne ve işlemcilerin karmaşıklığına bağlıdır. Ağın büyüklüğü bağlantı sayısı ile her bir bağlantı ağırlığı için gereken bit sayısı çarpılarak bulunur ve ağ için gereken toplam belleğin büyüklüğünü gösterir. İşlemcilerin karmaşıklığı kullanılan doğrusal olmayan fonksiyona, giren ve çıkan bağlantı sayısına, işlem ve bellek duyarlılığına bağlıdır.
3. *Öğrenme zamanı*, öğrenme kümesi üzerindeki başarının belli bir değerden aşağıya düşmesi için gereken öğrenme işleminin aldığı zamanı gösterir.

Her ne kadar herhangi bir uygulamada tüm bu üç kıstasın birden göz önüne alınması gerekirse de, yaygın olarak yapay sinir ağı öğrenme algoritmaları bunlardan sadece birincisini, yani başarıyı kıstas alır ve olabildiğince yükseltmeye çalışırlar. Halbuki, ağ karmaşıklığının ve öğrenme zamanının da olabildiğince küçük tutulabilmesi gerekir ve

bu iki kıstas, YSA, veya genelde bir öğrenen sistemin eldeki uygulama için olabildiğini gösterir. Örneğin, bir robotik uygulamasında sistemin gerçek zamanda ortama uyabilmesi istenir, ve böylesi bir durumda on binlerce döngü gerektiren bir algoritma uygun olmaz.

Herhangi bir uygulama için bir YSA yapısının veya öğrenme metodunun seçilmesi deneme-yanılma ile yapılır. Ağ tasarımcısı, uygulama ve öğrenme algoritması ile ilgili bilgisini kullanarak kaba seçimler yapar ve bu seçimler ile bir YSA oluşturulur ve bir öğrenme kümesi üzerinde bağlantı değerleri öğretilir. YSA daha sonra bir deneme kümesi üzerinde denenerek bu seçimlerin uygunluğu görülür ve gerekli görülürse, ağ yapısı veya öğrenme metodunda değişiklikler yapılarak bu tekrarlanır. Örneğin, ileri besleyen bir ağda geri hata yayma metodu kullanılıyorsa, ağın kaç saklı katmanı olacağı ve her bir saklı katmanda kaç ünite olması gerektiği ancak deneme yanılma ile belirlenebilir; kullanılacak tek bilgi en azın en iyi olduğudur. Occam'ın dediği gibi "Açıklamalar olabildiğince basit tutulmalıdır." Öğrenme sonrasında oluşturulmuş ağı da, girdiler ile çıktılar arasındaki fonksiyonun bir ifadesi olarak görünce, bunun da en basitinin, yani en küçük ağın en inandırıcı ifade olduğunu görebiliriz.

## 2.1. AĞ YAPILARI

### 2.1.1. Bellek tabanlı YSA

Bellek tabanlı YSA'larda hem öğrenme çok hızlıdır, hem de gereken YSA öğrenme sırasında oluşturulur; ağ yapısı ile ilgili her hangi bir ön kabule gerek yoktur. Bu yöntemlerde öğrenme kümesinin bir bellekte saklanan bir alt kümesinden bir yaklaşık değer hesaplanır. Örneğin, Büyü ve Öğren (Grow and Learn - GAL) (Alpaydın, 1990) yönteminde öğrenme şu şekilde olur:

0. Öğrenme kümesinden gelişigüzel bir örnek alınır.
1. Eldeki YSA'nın bu girdiye verdiği çıktı hesaplanır.
2. Eğer verilen çıktı gereken çıktıya yeterince yakınsa her hangi bir şey yapılmaz. Eğer arada tanımlanmış bir tolerans değerinden daha büyük bir fark var ise, bu girdi ve çıktı YSA'nın belleğine eklenir. Bu değişiklik, YSA'nın bir sonraki seferde benzer bir girdi ile karşılaştığında doğru çıktı verebilmesini sağlar.
3. Öğrenme kümesinde görülmemiş örnek kaldı ise, aşama (0)'e atlanır.

YSA'nın belleğini P diye adlandıralım ve  $Px_i$  örnek i nin girdisi,  $Py_i$  'de çıktısı olsun. x girdi olarak verilince bellek içinden en yakın örnek, diyelim c, bulunur. YSA'nın çıktısı, O, örnek c nin çıktısıdır.

$$\|Px_c - x\| = \min_i \|Px_i - x\|$$

$$O = Pyc$$

Bu yaklaşıma göre  $Px_c$  nin en yakın olduğu bütün girdi değerleri için sistemin çıktısı Pyc olur. Sınıflandırma uygulamalarında Py değerleri sınıf kodlarıdır.  $\|Px_i - x\|$  ve  $\|Px_j - x\|$  değerlerinin eşit olduğu noktada, yani x, i ve j'inci örneklere eşit uzaklıkta bulunduğu, çıktı ani olarak  $Py_i$ 'den  $Py_j$ 'ye değişir; bu yumuşak bir yaklaşım değildir.

Bunun için fonksiyonun Taylor açılımındaki ikinci terimi de ekleyerek daha iyi bir yaklaşım yapılabilir. Bunun için  $x$   $k$  boyutlu ise, en yakın  $k+1$  örnek bulunur ve bunlardan geçen  $k$  boyutlu hiperdüzlem hesaplanır. YSA'nın çıktısı bu hiperdüzlemde  $x$ 'e karşı gelen nokta olur.

### **2.1.2. Neocognitron**

Bu ağ yapısı Fukushima tarafından geliştirilmiştir(Fukushima,1988). Özellikle harf ve rakam tanımada kullanılır. Hiyerarşik ve yerel olarak bağlı aynı ağırlıkların tekrar kullanıldığı bir yapısı vardır. Bölümümüzde A. İrfan Oyman tarafından bitirme projesi olarak bu ağ yapısının PC üzerinde uyarlaması yapılmış ve katmanlardaki örüntülerin seçilmesinde gözönüne alınacak kıstaslar üzerinde çalışılmıştır(Oyman, 1992).

### **2.1.3. Hopfield Modeli**

Burç Uzalp, yüksek lisans tezi olarak Hopfield Modelini(Hopfield 1982) incelemiş ve bu modelin bir elektronik devreyle uyarlamasını yaparak bir optimizasyon problemini çözmüştür (Uzalp, 1991).

### **2.1.4. İki yönlü ilişkili bellek (BAM)**

İki yönlü ilişkili bellek ve bozulmuş verinin onarılması üzerine (Güvensoy, 1991, Kasımoğlu, 1992) algoritmalar kişisel bilgisayar için yazılmıştır.

### **2.1.5. Zaman gecikmeli sinir ağı (TDNN)**

Konuşma işaretinin tanınması için yaygın olarak kullanılan bu yapı (Waibell, 1989) uygulamalar da kullanılmaktadır. Konuşma işaretinin tanınması için Japonca veriler üzerinde denenip başarılı sonuçlar alınmıştır.

### **2.1.6. 3-pencere taramalı sinir ağı (NN3sw)**

Konuşma işaretinin tanınması için geliştirilmiş bu yapı (Gürgen, 1991) birbirinden bağımsız üç ayrı pencerenin veri üzerine uygulanıp önemli özelliklerin kullanılması amacıyla geliştirilmiştir.

### **2.1.7. İleri beslemeli model ve çeşitli performans ölçülerinin konuşma işareti üzerinde değerlendirilmesi**

Hatanın geriye yayılması eğitim algoritması ile çeşitli performans ölçüleri konuşma işaretinin tanınması problemi için değerlendirilmiştir. Uğur Ünlüakın bu konu da lisansüstü çalışmasına devam etmektedir.

## **2.2. ÖĞRENME YÖNTEMLERİ**

Çeşitli öğrenme yöntemleri konusunda bitirme projesi ve yüksek lisans tezi düzeyinde çalışmalar yapılmış ve yapılmaktadır.

### **2.2.1 Hata Geri Yayma**

Bu yöntemin bir türevinin(Filippi ve Reyneri, 1990) PCye uyarlanması ve standard hata geri yayma yöntemiyle karşılaştırılması konusunda Mehmet Yağcı bir bitirme projesi yapmıştır (Yağcı,1991). Bu algoritmanın bazı problemler için sonuç vermediği ve bazı durumlarda karasız davranış gösterdiği anlaşılmıştır.

Fusun Ertemalp ise şu anda bitirme projesi olarak hata geri yayma yönteminin dört transputerdan oluşan bir donanım üstünde koşut gerçekleşmesini yapmaktadır(Ertemalp, 1993).

### **2.2.2 Levenberg-Marquardt Yöntemi**

Serdar Günizi, bitirme projesi olarak Levenberg-Marquardt (Press, Flannery, Teukolsky ve Vetterling, 1986) yönteminin uyarlamasını ve standart hata geri yayma yöntemiyle karşılaştırılmasını yapmıştır (Günizi, 1991).

### **2.2.3. Öğrenme Yöntemlerinin Karşılaştırılması**

Haluk Topçuoğlu halen yüksek lisans tez çalışması olarak öğrenme işlevini bir optimizasyon problemi haline dönüştürüp bilinen optimizasyon yöntemlerinin etkinliklerini karşılaştırmaktadır (Topçuoğlu, 1993).

### **2.2.4 Bulanık eğitime algoritması**

Çok katmanlı devrelerde kullanılmak üzere bulanık eğitime algoritması önerilmiştir (Gürgen, 1991). Sınır ağının genelleştirilmesini arttıran fazla eğitilmesine (overlearning) engel olan bu yöntem konuşma işaretinin tanınmasına uygulanmıştır. Japonca dağarcıktan oluşan veri tabanı için yapılan konuşma işareti tanıma deneyleri bunu kanıtlamıştır.

## **3. YSA UYGULAMALARI**

### **3.1. ÖRÜNTÜ TANIMA**

#### **3.1.1. Optik Harf Tanıma**

Bu konuda bölümümüzde ilk çalışma, Orhan Kalaycıoğlu tarafından bitirme projesi olarak yapılmıştır. Bu çalışmada harfler bir önışlemciden geçirilerek featureları çıkarılmakta ve çok katmanlı bir YSA bu harfleri tanımayı öğrenmektedir(Kalaycıoğlu, 1991). Halihazırda ise, TÜBİTAK EEEAG tarafından Ethem Alpaydın'a verilen destek ile YSA'nın optik harf tanımadaki uygunluğu araştırılmaktadır. Bu konuda yurt dışında çalışan çok sayıda araştırma gurubu vardır. Bölümüzde yüksek lisans tezi çalışmalarını sürdüren Selami Aratma bellek tabanlı YSA'nın elle yazılmış sayıların tanınması konusunda çalışmaktadır. Bu çalışmada bellek tabanlı YSA, özellikle hızlı öğrenebilmesi sayesinde, her bir kullanıcının kendi el yazısına çabuk ayak uydurabileceği için ümit vermektedir.

Bir başka yüksek lisans tezi Mehmet Yağcı tarafından, bellek tabanlı YSA'nın seçici dikkate bağlı veya etkin görmede kullanılması ile ilgilidir. Bu çalışmada tanıma sırasında girdinin tamamı yerine, sadece gerekli olduğu kadarının kullanılması ve tanıma süresi içinde 'dikkat'in görüntünün bazen bir bölümüne, bazen de değişik bir bölümüne yoğunlaştırılması ana düşüncedir. Bu girdinin boyut sayısını düşüreceği için sistemin karmaşıklığını azaltacaktır.

#### **3.1.2. Konuşma İşaretinin Tanınması**

Bu konuda Fikret Gürgen tarafından yürütülmekte olan TÜBİTAK EEEAG projesi mevcuttur. Projenin kapsamı içinde Uğur Ünlüakın çok sayıda sözcükten oluşan dağarcık için konuşmacıya bağımlı tanıma işlemini yüksek lisans tezi olarak yapmaktadır. Hatanın

geri yayma algoritmasının ileri beslemeli devrelere uygulanması ile oluşturulan deney düzeneği üzerinde çeşitli kriterlerin denenmesi ile % 95 leri geçen tanıma oranlarına erişilmiş bulunmaktadır.

Bu projenin bir sonraki uygulama alanı Türkçe dağarcık olarak düşünülmektedir. Bu nedenle gereken altyapının oluşturulması için çalışmalara devam edilmektedir. Konuşma işaretinin kalitesinin geliştirilmesi (Gürgen, 1989) ve tanınması işlemlerinin birleştirilmesi sonucu gerçek uygulama alanlarına yönelmesi planlanmaktadır.

### **3.2. DENETİM**

YSAların herhangi bir doğrusal olmayan fonksiyonu istenen doğrulukla temsil edebilmeleri özelliğinden yararlanarak göreceli olarak daha kolay şekilde doğrusal olmayan denetleyiciler yapmak olasıdır. Bu konuda bölümümüzde yapılan ilk çalışma, Turap Taşoğlu ve Levent Akın'ın yaptığı bir nükleer reaktör denetleyicisi tasarımıdır (Akın ve Taşoğlu, 1991). Şu anda aynı reaktör için değişik ağ yapılarıyla çalışmalar devam etmektedir.

Bulanık mantık denetleyicilerin tasarımı her örnek için ayrı emek gerektiren bir çabadır. Son zamanlarda tasarımda bulanık kümelerin en uygun parametrelerini bulmada YSAlar kullanılmaktadır. Halen Levent Akın tarafından yapılan bir aralıttırma da ise tüm bulanık kümelerin ve kuralların öğrenilebileceği bir YSA yapısı geliştirilmeye çalışılmaktadır.

### **3.3. BECERİ ÖĞRENME**

Beceri öğrenme konusundaki ilk çalışma Selami Aratma'nın bitirme projesi olarak yaptığı "Ping Pong Oynamasını Öğrenen Bir YSA"dır (Aratma, 1991). Bu çalışmada, iki boyutlu bir pingpong masası kabulüyle mükemmel bir rakibe karşı oynayan bir YSA geliştirilmiştir. YSA çok katmanlı perseptron yapısındadır ve topu karşılama konumunu ve rakibini şaşırtmak için gerekli vuruş açısını öğrenmektedir.

Ruhan Alpaydın ise yüksek lisans tezi olarak gene çok katmanlı bir YSAya bir blues caz gitaristinin doğaçlama çalma biçimini öğretmeye çalışmaktadır (Alpaydın, 1992). Caz doğaçlaması gruptaki diğer müzistenlerin kurdukları armonik zemin ve bu zeminin üzerine doğaçlamacının o an yarattığı melodidir. Bu doğaçlama, başka bir açıdan da bir parmaklamalar dizisidir. Telli çalgılarda bir ses birden fazla değişik parmaklama ile gerçekleştirilebilir. Doğaçlamacının fiziksel özellikleri ve çalgısının fiziksel yapısı zaman içinde doğaçlamacının müzik stilinde belirleyici rol oynar. Parmaklama dizilerini öğrenmek, doğaçlamacının müzik stiliş öğrenmektir. Bu çalışmada da zaman gecikmeli YSAya parmaklama dizisi eşlemesi öğretildi. Sonuç olarak caz müziğine uygun dört ölçülük özgün melodiler elde edildi.

#### 4. REFERANSLAR

- Akın, H. L. ve Taşoğlu, T., (1991) "Nuclear Reactor Control Using Backpropagation Neural Networks," ISCIS VI Proceedings, Vol. II, pp.889-898, Elsevier.
- Alpaydın, E. (1990) Neural models of incremental supervised and unsupervised learning, no 869, Doktora tezi, Ecole Polytechnique Federale de Lausanne, İsviçre.
- Alpaydın, E. (1990) Vision as a temporal process, Teknik Rapor, Laboratoire de microinformatique, EPFL, İsviçre.
- Alpaydın, E. (1991) GAL: Networks that grow when they learn and shrink when they forget, Teknik Rapor 91-032, International Computer Science Institute, Berkeley, Calif, ABD
- Alpaydın, R. ve Akın, H. L., (1992) "Connectionist Approach to Improvisation on the Guitar," ISCIS VII Proceedings, pp.573-576, EHEI.
- Aratma, S ve Akın, H.L, (1992), "Ping-Pong Oynamasını Öğrenen Yapay Sinir Ağı," I. Türk Yapay Zeka ve Yapay Sinir Ağları Sempozyumu, Bildiriler, pp.85-92.
- Aratma, S. (hazırlanıyor) Memory-based neural networks for function approximation, Yüksek Lisans Tezi, Bilgisayar Mühendisliği Bölümü, Boğaziçi Üniversitesi.
- Ertemalp, F. (hazırlanıyor), "Implementation of Backpropagation Algorithm on a Transputer Networks" Bitirme Projesi, Boğaziçi Üniversitesi.
- Filippi, E ve Reyneri. L., (1990) "Modified Backpropagation Algorithm for Fast Learning in Neural Networks," Electronics Letters.
- Fukushima, K. (1988), "A Neural Network for Visual Pattern Recognition" IEEE Computer, pp.65-75.
- Gürgen F. S., Sagayama S., Furui S.,(1992) "A Study of Line Spectrum Pair Representation for Speech Recognition," IEICE (Japan) Trans. Fundamentals. Vol. E75-A, No. 1.
- Gürgen F. S., Aikawa K., Shikano K., (1991) "Phoneme Recognition with Neural Networks using a novel fuzzy training algorithm," IEEE IJCNN'91 Singapore, pp. 572-577.
- Gürgen F. S., Aikawa K., Shikano K., (1991) " The improvement of phoneme recognition performance of a neural network using fuzzy training," SYNAPSE'91, in Senri Int. Ins.,
- Gürgen F. S. (1989) "Speech enhancement with Fourier-Bessel Coefficients of signal and noise" Ph D Dissertation University of Akron.

Gürgen F. S. and C. S. Chen (1990) " Speech Enhancement with Fourier-Bessel Coefficients," IEE Proc. Computer Speech and Vision.

Günizi, S., (1991) "Implementation of Levenberg-Marquardt Method in Neural Network," Bitirme Projesi, Boğaziçi Üniversitesi.

Güvensoy C. (1991) "Associative Memories" Bitirme Projesi Boğaziçi Üniversitesi.

Hopfield, (1982), "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," Proc. N.A.S., Vol.5, pp.2554-2558.

Kasımoğlu M. (1992) "Bidirectional Associative Memories" Bitirme Projesi Boğaziçi Üniversitesi .

Oyman, A.İ, (1992) "Implementation of the Neocognitron Artificial Neural Network Algorithm," Bitirme Projesi, Boğaziçi Üniversitesi.

Press, W. H., Flannery, B., Teukolsky, S. A. ve Vetterling, W. T., (1986), Numerical Recipes, Cambridge University Press.

Uzalp, B., (1991) "Neural Networks and a Hopfield Model Realization," Yüksek Lisans Tezi Bilgisayar Mühendisliği Bölümü, Boğaziçi Üniversitesi.

Ünlüakın U. (hazırlanıyor), Speech recognition with multilayer perceptron, Yüksek Lisans Tezi Bilgisayar Mühendisliği Bölümü, Boğaziçi Üniversitesi.

Waibel A., Hanazawa Shikano,(1989) "Phoneme recognition using Time-Delay Neural Networks," IEEE Trans. on ASSP, Vol. 37, no. 3.

Yağcı, M., (1991) "Simulation Testing of a Proposed Backpropagation Algorithm on Neural Networks," Bitirme Projesi, Boğaziçi Üniversitesi.

Yağcı, M. (hazırlanıyor) Selective attention strategies for vision, Yüksek Lisans Tezi, Bilgisayar Mühendisliği Bölümü, Boğaziçi Üniversitesi.

## EK

```
/*
    Hata geri yayma yontemi ile fonksiyon ogrenme
    Bir girdi -> NO_UNI tane uniteli bir sakli katman -> Bir cikti
    E. Alpaydın Ocak 1993
*/
#include <stdio.h>
#include <math.h>
#include <graphics.h> /*Turbo (Borland) C */
#include <stdlib.h>

/* Ogrenilecek fonksiyon */
#define fonk(x) (4.26*(exp(-(x))-
                4.0*exp(-2.0*(x))+3.0*exp(-3.0*(x))))
#define X_ILK 0.0 /* Fonksiyonun en kucuk ve en
buyuk */
#define X_SON 5.0 /* x ve y degerleri */
#define Y_ILK (-1.5)
#define Y_SON 1.5
#define KERE 600 /* Ogrenme dongu sayisi */
#define NO_UNI 6 /* Sakli katmandaki unite sayisi
*/
#define OGR_SAYI 100 /* Ogrenme kumesi eleman
sayisi */
#define kare(a) ((a)*(a))
struct nokta { /* Ogrenme kumesi */
    double x, y;
    char mark;
} nok [OGR_SAYI];
#define rnd (rand()%1000)/1000.0

/* Grafik ara yuz komut ve sabitleri */
#define WS 150
#define WIN_BOR 10
#define WINX_SIZE (2*WIN_BOR+(int)(X_SON-
X_ILK)*WS)
#define WINY_SIZE (2*WIN_BOR+(int)(Y_SON-
Y_ILK)*WS)
#define WF 1
#define GXI WIN_BOR
#define GXF (WIN_BOR+(X_SON-X_ILK)*WS)
#define GYI WIN_BOR
#define GYF (WIN_BOR+(Y_SON-Y_ILK)*WS)

#define nokta(xp,yp) line(xp,yp,xp,yp)
#define dogru(sx,sy,ex,ey) line(sx,sy,ex,ey)
#define box(sx,sy) dogru(sx-1,sy-1,sx+1,sy+1); \
dogru(sx+1,sy-1,sx+1,sy+1); \
dogru(sx-1,sy+1,sx+1,sy+1); \
dogru(sx-1,sy+1,sx-1,sy-1)

#define gmapx(x) (int)(GXI+(((x)-X_ILK)/(X_SON-
X_ILK))*(GXF-GXI))
#define gmapy(y) (int)(GYF-(((y)-Y_ILK)/(Y_SON-
Y_ILK))*(GYF-GYI))

GrafikAc()
{
    int graphdriver,
    graphmode,
    grapherror;
    detectgraph(&graphdriver,&graphmode);
    if (graphdriver < 0) {
        printf("Grafik ekraninda hata !\n");
        exit(1);
    }
    initgraph(&graphdriver,&graphmode,"");
    grapherror = graphresult();
    if (grapherror < 0)
    {
        printf("GrafikAc hata:%s.\n",
            grapherrormsg(grapherror));
        exit(1);
    }
}

fonk_goster()
{
    double x, y; int xp,yp,xo,yo,xf,yf;
    xo=gmapx(0.0); yo=gmapy(0.0);
    xf=gmapx(X_SON); yf=yo; dogru(xo,yo,xf,yf);
    xf=gmapx(X_ILK); dogru(xo,yo,xf,yf);
    xo=xf=gmapx(0.0); yf=gmapy(Y_ILK);
    dogru(xo,yo,xf,yf);
    yf=gmapy(Y_SON); dogru(xo,yo,xf,yf);
    for (x=X_ILK;x<=X_SON;x+=1.0/WS) {
        y=fonk(x); xp=gmapx(x);
        yp=gmapy(y); nokta(xp,yp);
    }
}

ogr_kumesi()
{
    int i,xp,yp;
    for (i=0;i<OGR_SAYI;i++) {
        nok [i].x=X_ILK+rnd*(X_SON-X_ILK);
        nok [i].y=fonk(nok [i].x);
        xp=gmapx(nok [i].x); yp=gmapy(nok [i].y);
        box(xp,yp);
    }
}

main()
{
    GrafikAc();
    bp_ilk();
    fonk_goster();
    ogr_kumesi();
    bp_ogren();
    bp_goster();
    for (;;) ;
    closegraph();
}

/* Hata geri besleme kismi */
#define ETA 0.1 /* Ogrenme katsayisi */
double w1 [NO_UNI], b1 [NO_UNI]; /* Sakli katmana
baglantilar */
double w2 [NO_UNI], b2; /* Ciktiya baglantilar */
double ha [NO_UNI], byo;

#define sig(a) (1.0/(1.0+exp(-(a))))
#define ilk_wt (rnd*0.2-0.1)

bp_ilk()
{
    int h;
    for (h=0;h<NO_UNI;h++) {
        w1 [h]=ilk_wt; b1 [h]=ilk_wt;
        w2 [h]=ilk_wt;
    }
    b2=ilk_wt;
}

double bp(xi)
double xi;
{
    int h;

```



```

    for (h=0;h<NO_UNI;h++)
        ha [h]=sig(xi*w1 [h]+b1 [h]);
    for (byo=0.0,h=0;h<NO_UNI;h++)
        byo+=ha [h]*w2 [h];
    byo+=b2;
    return (byo);
}

bp_goster()
{
    int xp,yp,h; double x,y,bp();
    for (x=X_ILK;x<=X_SON;x+=1.0/WS) {
        y=bp(x); yp=gmapy(y); xp=gmapx(x);
        nokta(xp,yp);nokta(xp,yp-1);
    }
}

bp_degistir(xi,yi)
double xi,yi;
{
    int h; double bp();
    byo=bp(xi);
    for (h=0;h<NO_UNI;h++)
        w2 [h]+=ETA*ha [h]*(yi-byo);
    b2+=ETA*(yi-byo);
    for (h=0;h<NO_UNI;h++) {
        w1 [h]+=ETA*(yi-byo)*w2 [h]*
            ha [h]*(1.0-ha [h])*xi;
        b1 [h]+=ETA*(yi-byo)*w2 [h]*
            ha [h]*(1.0-ha [h]);
    }
}

bp_ogren()
{
    int s,i,j;
    for (s=0;s<KERE;s++) {
        for (i=0;i<OGR_SAYI;i++) nok [i].mark='.';
        for (i=0;i<OGR_SAYI;i++) {
            do j=(int)(rnd*OGR_SAYI);
            while (nok [j].mark=='X');
            nok [j].mark='X';
            bp_degistir(nok [j].x,nok [j].y);
        }
    }
}

bp_hata_hesapla()
{
    int i; double hata, yo;
    for (hata=0.0,i=0;i<OGR_SAYI;i++) {
        yo=bp(nok [i].x);
        hata+=kare(nok [i].y-yo);
    }
    hata/=OGR_SAYI;
    printf("Hata geri besleme: %d
        üniteli bir saklı katman \n",NO_UNI);
    printf("Ogrenme kumesi üzerindeki hata :
        %8.4f\n",hata);
}
}

```