YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YAPAY NÖRON AĞLARINDAN ÇİFT-YÖNLÜ İLİŞKİLİ BELLEK

YÜKSEK LİSANS TEZİ
ELEKTRONİK VE HABERLEŞME MÜHENDİSİ
SİBEL ÖGE

İSTANBUL 1993

İÇİNDEKİLER

ÖNSÖZ

Bu tezin verilmesinde ve hazırlanmasında bana büyük katkı ve yardımları olan Sayın Hocam Doç.Dr.Fikret GÜRGEN'e aynı zamanda tezin yazılma aşamasındaki yardımlarından ötürü Sayın Ercan MEREY'e teşekkürlerimi sunarım.

Elektronik ve Haberleşme Mühendisi

Sibel ÖGE

ÖZET

Hopfield tarafından tanıtılan tek katmanlı, tek-yönlü öz-ilişkilendirici daha sonra Kosko tarafından genişletilerek, çift katmanlı ve çift-yönlü hale getirilmiştir. Bu yapıya da Çift-yönlü İlişkili bellek adı verilmiştir. Bu yapı küçük bir ilişki matrisiyle karışık ilişkiyi başarabilir. Fakat, Kosko tarafından kullanılan bu Hebbian kuralına dayalı ilişki matrisi,çok küçük data eğitim seti içerisinde bile kesin hatırlamayı başaramayabilir. Uygulamalara bağlı olarak özel bir eğitim çiftinin veya tüm çiftlerin kesin hatırlanması önemli olabilir. Bu durumda Wang'ın ileri sürdüğü Çoklu Eğitim metodu ile ilişki matrisi oluşturulur ve istenilen çiftler üzerinde çoklu eğitim uygulanır. Böylece istenilen çiftlerin kesin hatırlanması başarılır. Hatırlama işlemi ve enerji hesaplama işlemi her iki yöntemde de aynıdır. Üzerinde durduğumuz bir diğer model ise Yüksek Dereceden ÇYİB dir. Bu modelin hatırlama işleminde tüm eğitim çiftleri ve girilen derece önemlidir. Eğitim çiftlerinin fazlalığı ve hatırlama başarısı bakımından bu yöntem genel olarak daha iyi bir performans gösterir.

SUMMARY

Hopfield introduced a one layer unidirectional autoassociator and it has been extended to be two layer and bidirectional by Kosko. This structure is called a bidirectional associative memory. It can achieve heteroassociation with a smaller correlation matrix. But the form of the Hebbian rule-based correlation matrix used by Kosko cannot guarantee recall of an important patern even in a very small set of traning data. Depending on the application, it may be important to guarantee that a traning pair will be recalled. In this care, Multiple Training Encoding Strategy which introduced by Wang is used and Multiple Training is applied to the desired pairs. So that the desired pairs can guarantee recall. The recall algorithm and energy calculations for the vectors are same in the two method. The other coding strategy is High-order Bidirectional Associative Memory. In this method training pairs and initial order is important. This method significantly improves the storege capacity and error-correcting capability of the bidirectional associative memory.

BÖLÜM 1 YAPAY NÖRON AĞLARI

1-1 YAPAY NÖRON AĞLARINA GİRİŞ

1940'lı yıllardan sonra temelini Von Neumann'ın attığı işlemci yapısı ortaya çıkmış ve günümüze kadar olan sürece damgasını vurmuştur. Bu yapının temel özelliği tek bir işlemciden oluşması ve işlemleri belli bir algoritmaya göre yapmasıdır. Günümüze kadar bir çok uygulamada başarıyla kullanılan bu yapının son yıllarda bazı problemlerin çözümünde yetersiz kaldığı ortaya çıkmıştır. Bu tip problemlerin iki temel özelliği vardır:

1) Kötü veya eksik tanımlanmış olmaları

2) Çok fazla işlem yoğunluklu olmaları

Patern algılama , ses tanıma ve robot kontrol uygulamaları bu tip problemlere örnek olarak gösterilebilir. Bu problemlerin ortak özellikleri bunların geleneksel yöntemlerle çözülememesine karşın canlı organizmalar tarafından kolaylıkla çözülebilmeleridir. Canlı organizmaların bu özellikleri farkedilince araştırmacılar insan beyninin yapısını ve işleyişini incelemeye başlamışlardır. Buradan öğrenilenlerden yola çıkarak yeni bir mimari yapının geliştirilmesine yönelmişlerdir. Yapay nöron ağları (YNA) bu çalışmaların sonucunda ortaya çıkmışlardır.

YNA üzerindeki çalışmaların temeli 40-45 yıl öncesine dayanır. Bu tarihlerde konuyla ilk olarak McCullough ve Pitts isimli bilim adamları ilgilenmişler ve YNA'nın

temelini atmışlardır. Bu ilk çalışmalar sonucunda fazla bir başarı sağlanamamış , konu ile uzunca bir süre ilgilenilmemiştir. Yukarıda belirtilen ihtiyaçların ortaya çıkmasından sonra bu konudaki araştırmalara yeniden başlanmış ve özellikle son 10 yılda büyük ilerlemeler kaydedilmiştir [10].

1-2 YAPAY NÖRON AĞLARININ YAPISI

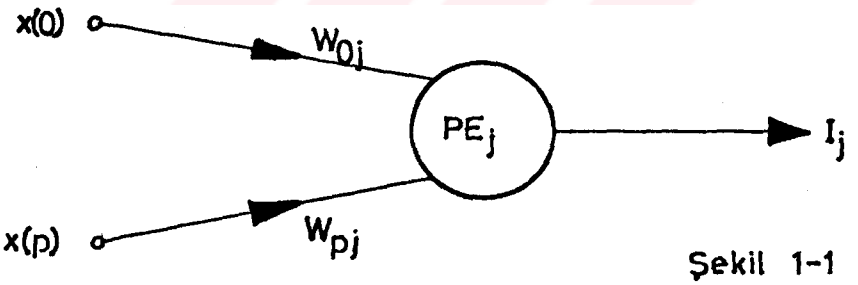
YNA , insan beynindeki nöronların çok basite indirgenmiş bir karşılığı olan elektronik işlemcilerden ve bunların birbirleriyle bağlantılarından oluşur. Canlı organizmaların sinir sistemlerinin , özelliklede beynin temel taşı nöron denilen sinir hücreleridir. Bir insan beyninde yaklaşık 15 milyar nöron ve her nöronun da 10 bin girişi vardır. Bu 10 bin girişe diğer nöronlardan 10 bin tanesinin çıkışları bağlıdır. Nöronlar girişlerine gelen sinyalleri işleyip sonuçta elde ettikleri çıkış sinyalini bağlantılı oldukları diğer nöronlara gönderirler.

Araştırmacılar bu bilgilerden yola çıkarak insan beyninin yapısına göre modellenmiş yeni bir mimari yapı ortaya çıkarmışlardır. Nasıl ki biyolojik organizmalardaki nöronlar ve bunların birbirleriyle bağlantıları bir sinir ağı meydana getirir ise aynı şekilde YNA'ında kullanılan işlemciler ve bunların birbirleriyle bağlantıları da bir elektronik ağ oluştururlar. Bu işlemcilerin birden fazla girişi , bir merkezi işlem birimi ve tek bir çıkışı vardır. Bu tek çıkış diğer işlemcilerin girişlerine bağlantı yapabilmek için birçok dala ayrılır [10].

Bir nöronun girişiyle buna bağlantılı olan başka bir nöronun çıkışının birbirlerine bağlandıkları noktaya sinaps denir. Sinapsın işlevi gelen sinyalin sadece bir kısmını geçirmektir. Canlı organizmalarda bu olay kimyasal bir yapıda gerçekleşir. Sinapslar sinir sisteminin hafıza birimleridir. Girilen bilgi bütün sinapslara dağılarak

saklanır. YNA'ında sinapsların karşılığı 'bağlantı ağırlıkları' adı verilen birimlerle gösterilir. Bu sinapslar gelen sinyalin değerini kendi ağırlık değerleriyle çarparak sinyalin ağırlıklı bir değerini oluştururlar. Bu ağırlık değerleri kullanılan devrenin yapısına göre uygulama sırasında ya sabit kalırlar ya da belli bir kurala göre değişimler gösterirler. Sabit ağırlıklar genelde belli bir 'mapping' yapan yani bir girişe karşılık olarak çıkışını belli bir duruma götüren devrelerde kullanılır. Değişken ağırlıklar ise adaptif kontrol sistemleri ve eğitilebilir patern algılayıcılar gibi adaptif devrelerde kullanılırlar. Ağırlık değişimleri ordinary türevsel (fark) eşitlikleri ile tanımlanır; bu nedenle değişken ağırlıklı devreler doğrusal olmayan dinamik sistemlerdir.

Şimdi YNA'ında kullanılan bir işlemcinin yapısını şekil yardımıyla görelim.



Şekil 1-1

Giriş vektörü [x(0),.....,x(p)] olsun. İlk olarak giriş vektörünün her elemanı kendi bağlantısının ağırlık değeriyle çarpılır ve herbir giriş için ağırlıklı bir giriş değeri bulunur. Daha sonra her girişdeki ağırlıklı değerler toplanarak bu işlemci için tek bir ağırlıklı toplam giriş

değeri bulunur. Bu değeri I(j) ile gösterirsek:

$$I(j) = \Sigma W(i,j)*X(i)$$

Bundan sonraki aşamada bu ağırlıklı toplam doğrusal olmayan bir transfer fonksiyonundan geçer ve bu işlemcinin çıkışı (Y(j)) bulunur.

$$Y(j) = f (I(j))$$

Birçok uygulamada transfer fonksiyonu olarak sigmoidal fonksiyon kullanılır:

$$f (I(j)) = 1 / (1 + \exp(-I(j)))$$

YNA işlemcileri genelde analog yapıdadırlar. Fakat yukarıdaki örnekte de görüldüğü gibi sayısal olarak gerçeklenip kullanıldıkları uygulamalarda vardır. YNA'daki işlemciler geleneksel yöntemlerde kullanılan sayısal işlemcilere göre hem daha basit yapıdadırlar hem de daha yavaş çalışırlar. Fakat bunların birçoğu aynı anda paralel olarak çalıştıklarından sonuçta YNA'nın hız ve kapasiteleri daha fazla olmaktadır. YNA 'katman' denilen işlemci gruplarının ard arda konması ve bunlar arasında çeşitli bağlantılar yaratılmasıyla oluşur. Bir katman aynı transfer fonksiyonuna sahip , aynı yapıdaki işlemcilerin yan yana gelmeleri ile oluşur. Bir katmanda en az bir işlemcinin bulunması gerekirken bu sayı için bir üst sınır yoktur. Aynı katmandaki işlemcilerin birbirleriyle bağlantıları yoktur, fakat diğer katmanlardaki işlemcilere çeşitli fiziksel bağlantılar yapabilirler [10].

1-3 YAPAY NÖRON AĞLARININ ÖZELLIKLERI

Bir paralel dağılmış bilgi işlem yapısı (yönlü graf yapısı) şeklindeki YNA'nın aşağıdaki alt tanımları ve kısıtlamaları vardır.

1- Düğümler işlem elemanları olarak tanımlanır.

2- Düğümler arasında bağlantılar (connections) vardır. Her bağlantı gecikmesiz tek yönlü iletim yolu olarak görev yapar.

3- Her işlem elemanı istenildiği sayıda giriş bağlantısı alabilir.

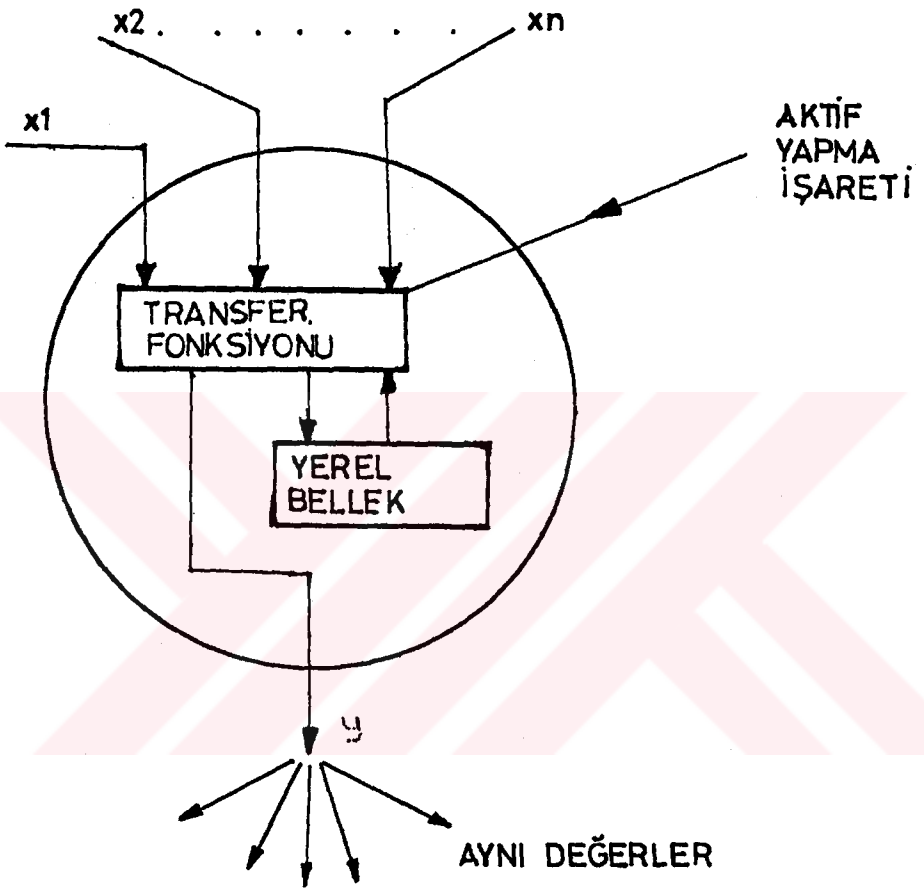
4- Her işlem elemanında tek bir çıkış bağlantısı olabilir. Fakat bu bağlantı kopya edilebilir. Yani aynısı olmak koşuluyla istediğimiz kadar çıkış bağlantısı kabul edebiliriz.

5- Işlem elemanları yerel bellek (local memory) taşıyabilirler.

6- Her işlem elemanının, yerel belleği ve giriş sinyallerini kullanabildiği bir transfer fonksiyonu vardır. Transfer fonksiyonu işlem elemanının çıkış sinyalini üretir. Diğer bir deyişle kabul edilebilir giriş işaretleri yerel bellekte bulunan değerlerden biri olabilir. Transfer fonksiyonları, sürekli veya kısmen sürekli olabilirler. Kısmen sürekli çalışma konumunda bir aktif yapma giriş işareti vardır. Bu işaret aktif halde iken eleman bir çıkış işareti üretebilir.

7- Giriş işaretleri yapay nöron ağına bilgi taşır ve sonuç çıkış işaretlerinden alınabilir.[13]

Asağıda tek bir islem elemanının (nöronun) genel sekli gözükmektedir:



Şekil 1-2

BÖLÜM 2 İLİŞKİLİ BELLEK

2-1 İLİŞKİLİ BELLEĞİN TANIMI

Tarihsel olarak ilişkili bellek (associative memory) modeli nöron ağları araştırmaları için esas odaklardan biri olmuştur. Nöron ağında (NA) giriş değerleri ikili (binary) ve sürekli olmak üzere ikiye ayrılabilir. İkili girişi de süpervizörlü eğitimli ve süpervizörsüz eğitimli giriş olmak üzere ikiye ayırabiliriz. Süpervizörlü eğitime Hopfield ağını ve perceptronları örnek verebiliriz. Bunlar sınıflayıcılar veya ilişkili bellekler olarak kullanılabilirler [8].

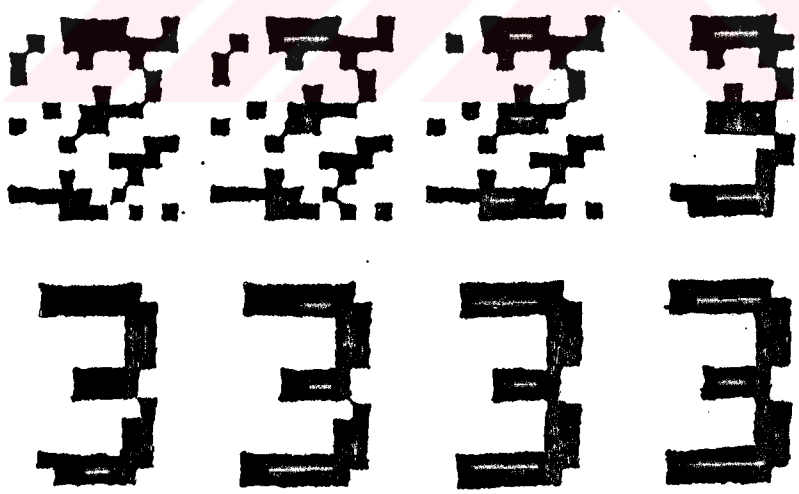
İlişkili bellek, tam depolanmış paternin, gürültülü veya eksik giriş paterninden geri alınmasına olanak verir. Yani ilişkili nellek modelleri, ikili (binary) paternlerin depolanması ve geri alınmasına dayanır. Bu modeller tipik olarak Hamming ölçüsü kullanımı yaparlar ve bunların bağlantıları sadece tam sayı değerlerini üzerlerine alırlar.

1950'lerden sonra ortaya çıkan ilişkili bellek modelleri mevcut teknolojinin RAM ve ROM'un kullanılması taraftarı olduğundan fazla ilgi toplayamadı.

Hopfield 1982 yılında dışçarpım depolama kuralına dayanan enerji minumizasyonu modelini ileri sürünce ilişkili belleğe ilgi arttı. Sınırlı kapasite ve donanım (hardware) verime sahip Hopfield ağına dayanarak, 1982 den beri pek çok yeni bellekler ileri sürüldü. Bunlardan

önemlileri, Unary (Hamming) ağı ve Sparsely-distributed ileri besleme modelidir [6].

a.



b.

Şekil 2-1

Arkadaki şekilde ilişkili bellek hatırlamasına örnek verilmiştir. Şekil a'da eğitilecek (depolanacak) çiftler gösteriliyor. Şekil b'deki ilk resim parçası ağa uygulanıyor. Bu, 3 rakamının her bitinin birbirinden bağımsız olarak %25 olasılıkla değiştirilmiş halidir. Bu ağa uygulanınca, ağ son şekil parçasında görüldüğü gibi 3'e yakınsar.

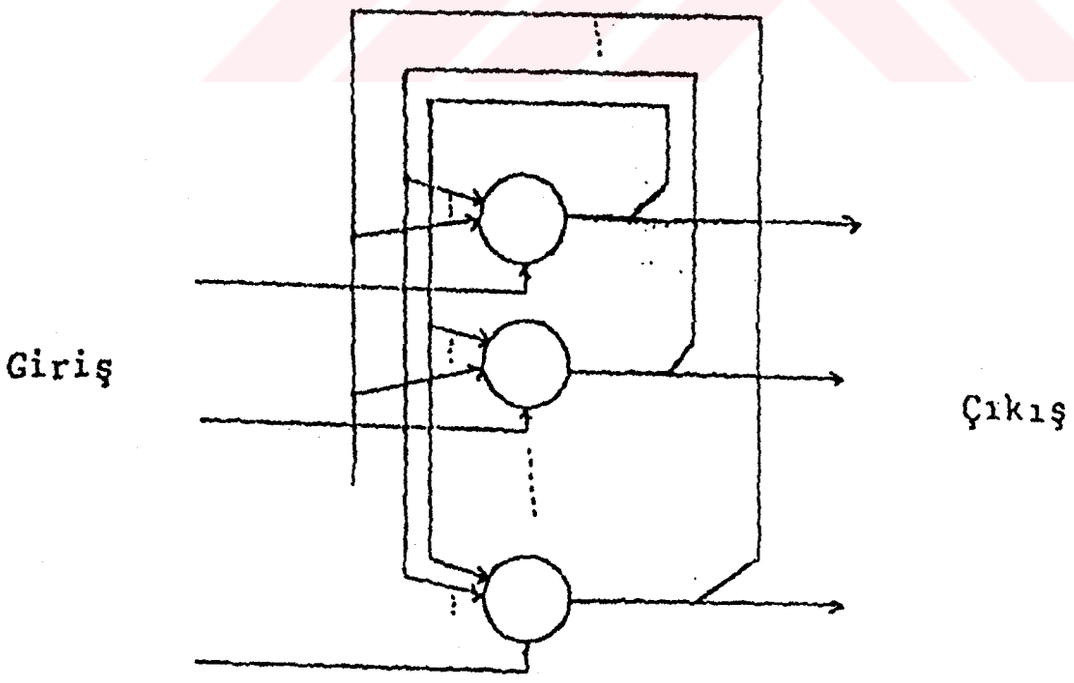
İnsanlarda da bu bellek hatırlaması sıkça olur. Örneğin uzaktan beyaz Mazda 323 model fakat plakasını tam okuyamadığımız bir araba görmüşsek ve bizimde bu model arabaya sahip bir arkadaşımız varsa, o kişinin imajı hafızamızda oluştuğu gibi araba plakasını da hatırlarız.

İlişkili belleğin bir diğer önemli kavramıda enerji fonksiyonudur. Enerji fonksiyonun en önemli özelliği, onun her zaman azalması veya sabit kalmasıdır. Bu kavram üzerinde daha sonra durulacaktır.

Burada ilişkili bellek modeli olarak kısaca Hopfield ve Hamming ağından bahsedeceğiz.

2-2 HOPFIELD VE HAMMING AĞI

Hopfield ağı normalde ikili (binary) girişler kullanır ve bu giriş elemanları siyah beyaz bir resmin pixel elemanlarını veya her karakterin 8 bit ASCII gösterimindeki bitlerini temsil edebilirler. Bu ağda ağırlıklar önceden belirlenir. Aşağıdaki şekilde Hopfield ağı modeli görülmektedir [8].



Şekil 2-2

Görüldüğü gibi ağ tek katmandan oluşur. Burada katı sınırlamalı fonksiyona sahip N tane hesapsal birim bulunur. $t+\tau$ anında bu ağın i. nöronunun çıkışı şu şekilde ifade edilir:

$$s_i(t+\tau) = f(\Sigma C_{ij} s_j(t)) = f(v_i(t)) \qquad j=1...N$$

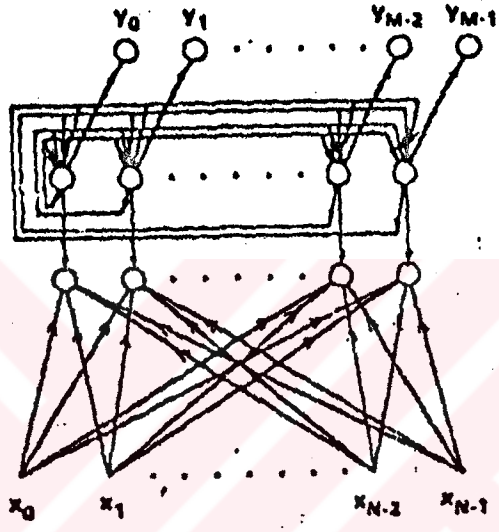
Burada C_{ij}, j. nörondan bilgi alan i. nöronun sinaptik ağırlığıdır. $s_i(t)$ i. nöronun t anındaki durumu, $v_i(t)$ ise o andaki potansiyelidir. f'de nöronun aktivasyon fonksiyonudur [7]. Hopfield ağı ilişkili bellek olarak kullanıldığında depolanabilecek örnek sayısı M, bu ağdaki hesapsal birim sayısı N ile sınırlıdır. Bu sınırlama $M<0.15N$ şeklindedir [8].

Hopfield ağının ilişkili bellek özellikleri durum uzayında çekici sabit noktaların (attractor fixed points) var olmasına dayanır. (σ^k k=1...p) p tane ilk örnek paterni varsayalım. Problem σ^k paternlerinin çekici sabit noktalar olduğu ve başka çekicilerin mevcut olmadığı C matrisini bulmaktır [7].

Arkada Hamming ağının şekli gözükmektedir. Hamming ağı işlemsel birimlere sahip 3 katmandan oluşur. Giriş katmanı N birime sahiptir. (+1,0) değerlerini alabilir. İç katman G iç birime sahiptir. Çıkış katmanı N' birime sahiptir. (+1) ayrık aktivasyon değerlerini alır. Bu ağda yapılan bir match sonunda sadece bir iç birim aktif olur. Birden fazla iç birimin aktif olduğu modeller sparsely-distributed modelleridir [6].

Hamming modelinde $M<=G$ dir. Yani depolanabilecek örnek

sayısı iç birimdeki eleman sayısına bağlı ve giriş eleman sayısı N'den bağımsızdır. Bu yüzden Hamming ağı Hopfield ağından daha avantajlıdır. Bağlantı sayısı olarakda Hamming ağının daha az bağlantıya sahip olma üstünlüğü vardır [6].



Şekil 2-3

2-3 ÖZ-ILIŞKILI VE KARIŞIK-ILIŞKILI BELLEK

Tek yönlü ilişkili bellek ----> öz-ilişkili

$A \rightarrow M \rightarrow A' \rightarrow M \rightarrow A'' \rightarrow M \rightarrow \ldots\ldots A_r \rightarrow M \rightarrow A_r$

Eğer nöron ağına girdiğimiz işaret x veya gürültülü bir x ise çıkışta doğru bir y=x hatırlamak istersek kullanılacak model öz-ilişkili tek yönlü bellektir. Görüldüğü gibi bu tek yönlü bellekte geçiş sadece M ilişki matrisi vasıtasıyladır.

Çift yönlü ilişkili bellek ----> karışık-ilişkili

$A \rightarrow M \rightarrow B$

$A' \leftarrow M^T \leftarrow B$

$A' \rightarrow M \rightarrow B'$

$A'' \leftarrow M^T \leftarrow B'$

.

.

.

$A_r \rightarrow M \rightarrow B_r$

$A_r \leftarrow M^T \leftarrow B_r$

Eğer giriste x varken çıkışta $y \neq x$ olmasını istersek karışık-ilişkili bellek kullanırız. Çift-yönlü ilişkili bellek bir karışık-ilişkili bellektir. Çünkü (A_i, B_i) şeklindeki eğitim çiftlerini hatırlarken bir yönde geçiş M vasıtasıyla, diğer yönde geçiş M^T vasıtasıyla yapılarak çift yönlülük ve karışık ilişkilik sağlanır.

BÖLÜM 3 ÇİFT-YÖNLÜ İLİŞKİLİ BELLEK

3-1 ÇİFT-YÖNLÜ İLİŞKİLİ BELLEK NEDİR ?

(A_i,B_i) şeklindeki data çiftlerini depolayıp hatırlayabilecek çift katmanlı, lineer olmayan geri beslemeli en küçük nöron ağı çift-yönlü ilişkili bellektir (ÇYİB). ÇYİB karışık-ilişkili bir bellektir. Bir yöndeki geçiş M ilişki matrisi ile diğer yönden geçiş M^T ile sağlanır [1].

Çift-yönlü ilişkili bellek içinde $(A_1,B_1)....(A_m,B_m)$ çiftleri depolandığında ve M, M^T elde edildğinde (A,B) çiftini ağa uygulayalım. Hatırlama işlemini ilk önce A yada ilk önce B'yi kullanarak yada ikisini eşzamanlı kullanarak başlatabiliriz. Kolaylık için çift-yönlü ilişkili belleğe A_i depolanmış paterni uygulanırsa aşağıdaki sinyal-gürültü ifadesi elde edilir:

$$A_iM = (A_iX_i^T)Y_i + \Sigma\ (A_iX_j^T)Y_j \qquad j = 1....m \quad i<>j$$

veya A_i'nin çift kutuplu (bipolar) versiyonu X_i'yi kullanırsak:

$$X_iM = (X_iX_i^T)Y_i + \Sigma\ (X_iX_j^T)Y_j \qquad j = 1....m \quad i<>j$$
$$= nY_i + \Sigma\ (X_iX_j^T)Y_j$$
$$= nY_i + \Sigma\ (n-2H(A_i,A_j))Y_j$$

Yukarıdaki eşitlikte $x_{ij}=X_iX_j^T$ gürültü kuvvetlendirme katsayılarıdır. $H(A_i,A_j)$ ise A_i ile A_j arsındaki Hamming uzaklığıdır. Eşitliğin sağ tarafındaki ilk terim doğru çıkışın hatırlanmasıyla yükümlü olan sinyal terimidir. Kalan terim ise doğru çıkışı geri almak için minimize

edilecek gürültü terimini ifade eder. Eğer gürültü terimi doğru çıkışı elde etmek için yeterince küçük ise mükemmel hatırlama başarılır. Mükemmel hatırlama özelliği sadece erişilen denge noktalarının depolanan data çiftlerine tekabül edenlerle aynı olmasıyla sağlanır. Diğer durumlarda çift-yönlü ilişkili belleğin hatırlama işlemi doğru data çiftlerinin geri alınmasında başarısız olur. Bu olaya iki kutuplu kodlama yönteminde yanlış öğrenme (mislearning) adı verilir. Gerçekte ÇYİB'de yanlış öğrenme doğrudan doğruya , depolanan data kodlarıyla ve depolanan datalar arasındaki Hamming uzaklıklarıyla ilişkilidir. Bu, çift-yönlü ilişkili belleğin kod-bağımsız ilişkili bellek olmadığını ifade eder. Buradaki sorun, mükemmmel hatırlama özelliğini sağlayabilmekte hangi şartlar altında , yanlış öğrenmeden kaçınabileceğimizi bulmaktır. Kosko aşağıdaki süreklilik şartını bu durum için önermiştir [9].

$$1/n\ H(A_i,A_j) \approx 1/p\ H(B_i,B_j)$$

İyi hatırlama için sağlanması gereken bir diğer şart da depolanacak eğitim çiftlerinin sayısının bu çiftlerdeki vektörlerin boyutlarından ufak olmasıdır. Yani ;

$$m < \min(n,p)$$

3-2 KOSKO'NUN ÇIFT-YÖNLÜ ILIŞKILI BELLEK MODELI

Çift-yönlü ilişkili bellek ilk olarak Kosko tarafından tanıtılmıştır. Bu model şu işlemleri takip etmektedir :

$\{ (A_1,B_1),(A_2,B_2),.....,(A_i,B_i),.....,(A_N,B_N) \}$

şeklinde N tane eğitim çiftimiz olsun. Burada N depolamak istediğimiz örnek sayısını ifade etmektedir. Her bir A_i ve B_i vektörü şu şekilde tanımlı olsun :

$A_i=(a_{i1},a_{i2},...,a_{ij},...,a_{in})$

$B_i=(b_{i1},b_{i2},...,b_{ij},...,b_{ip})$

Her bir a_{ij} ve b_{ij} açık (ON) ya da kapalı (OFF) olabilir. Ikili (binary) modda 1 ve 0 değerlerini alırlarken, çift kutuplu modda 1 ve -1 değerlerini alırlar. A_i ve B_i vektörlerinin eleman sayıları (boyutları,uzunlukları) aynı olmak zorunda değildir. A_i vektörlerinin boyutları n, B_i vektörlerinin boyutları p olarak kabul edilebilir [2-4].

Bu eğitim çiftlerini depolamak ve istediğimiz zaman yeniden elde edebilmek için bir matris oluşturulur.

$$M = \Sigma X_i^T Y_i \qquad i=1...N$$

Bu şekilde oluşturulan n*p 'lik matrise ilişki matrisi (correlation matrix) adı verilir. Bu tanımlanan matrisdeki X_i ve Y_i, ikili modda olan A_i ve B_i'nin, çift kutuplu moddaki gösterimleridir. Çift kutuplu kodlama modu daha avantajlı olduğu için ilişki matrisi bu şekilde oluşturulmuştur [1].

Nöron ağına (α,β) çiftini veya sadece α vektörünü ilk koşul (giriş) olarak uygularsak, ağdan buna en yakın bir (A_i,B_i) çiftini geri almak isteriz. (α,β) ile başlama,

$$(\alpha,\beta),(\alpha',\beta'),(\alpha'',\beta''),.....,(\alpha_F,\beta_F)$$

şeklindeki sonlu ard arda gelişlere neden olur. (α_F,β_F) sonuçta nöron ağından alacağımız vektör çiftidir. Yani ağın ulaşabildiği denge noktasıdır. Bu sonlu ard arda gelişler şu şekilde belirlenir [2-4].

$\beta'=\emptyset(\alpha*M)$ $\quad\quad$ $\alpha'=\emptyset(\beta'*M^T)$

$\beta''=\emptyset(\alpha'*M)$ $\quad\quad$ $\alpha''=\emptyset(\beta''*M^T)$

. $\quad\quad$.

. $\quad\quad$.

. $\quad\quad$.

Burada $\emptyset$ fonksiyonu şu şekilde tanımlıdır.

$\emptyset F= G = (g_1,g_2,....,g_n)$ $\quad\quad$ $F=(f_1,f_2,....,f_n)$

$f_i<0$ ise $g_i=0$ (ikili modda), $g_i=-1$ (çift kutuplu modda)

$f_i>0$ ise $g_i=1$ (her iki modda)

$f_i=0$ ise önceki g_i

Yani, yukarıdaki formüllerde parantez içi çarpımlarından bir vektör elde edilir ve bu vektörün elemanları 0'dan büyükse 1, küçükse 0 değerini alır.

Her (α,β) vektör çifti için $E=-\alpha M\beta^T$ şeklinde bir enerji fonksiyonu tanımlıdır ve her bir kod çözme iterasyonunda bu E enerji değeri azalır. (α_F,β_F) denge noktasına ulaşıldığında, $E=-\alpha_F M\beta_F^T$ enerjisiide lokal enerji minimumuna ulaşmış olur [2-5]. Artık (α_F,β_F) den bir Hamming uzaklıktaki noktaların enerjileri, onun enerjisinden daha küçük olamaz. Burada (A_i,B_i) eğitilmiş çiftinin geri alınmasına ilişkin önemli bir sonuç vardır. Eğer (A_i,B_i) çiftinin koordinatlarını kullanarak değerlendirilen E enerjisi lokal minimum teşkil etmezse, başlangıçta $\alpha=A_i$ bile olsa bu çift hatırlanamaz [2].

3-3 ÇOKLU EĞITIMLI ÇIFT-YÖNLÜ ILIŞKILI BELLEK MODELI

Bir önceki bölümde söylenilenler dikkate alınırsa, Kosko tarafından kullanılan, Hebbian kuralına dayalı ilişki matrisi formu, çok küçük bir eğitim setinde dahi önemli bir paternin hatırlanmasını garanti edemez. En azından tek bir çiftin hatırlanması garanti edilmek isteniyorsa, bu çoklu eğitim metodu kullanılmak suretiyle sağlanılabilir. Aynı zamanda tüm eğitim çiftlerine çoklu eğitim uygulanarak tüm çiftlerin hatırlanması sağlanabilir [2-4].

Şayet (A_j,B_j) çiftinin kesinlikle hatırlanlmasını istiyorsak, Kosko'nun M_o ilişki matrisine şöyle bir P matrisi ilave edilmelidir.

$$P = (q-1)\ X_j^T\ Y_j$$

Böylece yeni ilişki matrisimiz şu hali alır :

$$M = M_o + P = \Sigma\ X_i^T\ Y_i + (q-1)\ X_j^T\ Y_j \qquad (i=1...N)$$

Bu durumda bu çift için tanımlı E enerjisi şöyledir :

$$E(A_j,B_j) = -\ A_i\ (\ \Sigma\ X_i^T\ Y_i\ )\ B_j^T - (q-1)\ A_i\ X_j^T\ Y_j\ B_i^T$$

E enerjisinin q'nun uygun seçimi ile yeterince küçültülmesi, bu çiftin enerjisinin bir Hamming uzaklıktaki tüm çiftlerin enerjilerinden küçük olmasını sağlar ve böylece bu çift hatırlanır. Tüm çiftlerin hatırlanmasını sağlayan aşağıdaki M ilişki matrisindeki q_i'ler lineer programlama veya ardarda (sequantial) çoklu eğitimi gibi metodlarla bulunabilir [3].

$$M = \Sigma\ q_i\ X_i^T\ Y_i \qquad i=1...N$$

3-4 YUKSEK DERECELI ÇIFT-YÖNLÜ ILIŞKILI BELLEK MODELI

ÇYIB'in iyi hatırlaması için, depolama kapasitesi, kullanılan sinirlerin sayısı ile sınırlıdır. Ayrıca güvenilir hatırlama için patern setleri arasında süreklilik şartı aranır. Yüksek dereceli lineer olmayan (nonlineer) ağların (high order BAM, HOBAM), lineer bellek modelleri ile karşılaştırılmalarında, bellek kapasitesi ve hata düzeltme kapasitesi bakımından daha iyi sonuçlar verdiği belirlenmiştir [5].

k. dereceli ÇYIB'e X_κ gibi bir giriş uygulandığında, vektör çiftini hatırlama işleminin bir iterasyonu söyledir:

$$Y = \text{sign} \left(\Sigma\, Y_i \left[X_i^T X_\kappa \right]^\kappa \right) \qquad i=1...N$$

$$X = \text{sign} \left(\Sigma\, X_i \left[Y_i^T Y \right]^\kappa \right) \qquad i=1...N$$

Burada X ve Y vektörleri sütun vektörleri şeklinde tanımlıdır. Ayrıca sign(a) fonksiyonunda a>0 ise çıkış 1, diğer durumlarda -1'dir. Köşeli parantez içerisindeki vektörlerin çarpımı sonucu elde edilen tamsayı, seçilen k derecesine göre kuvvetlendirilerek çıkış vektörüne etki eder.

3-5 ENERJI FONKSIYONU VE ENERJININ KARARLILIGI

Bundan önceki konularda da belirtildiği gibi E enerji fonksiyonu şu şekilde tanımlıdır:

$$E = A\ M\ B^T$$

Burada herhangi bir M ilişki matrisinin tek-yönlü ve çift-yönlü olarak kararlılığını inceleyeceğiz. Daha açığı E enerjisinin, ÇYİB'de herhangi bir M matrisiyle kararlı noktaya varıncaya kadar azaldığını ($\Delta E < 0$) ispat edeceğiz.

Tek-yönlü kararlılık izlendiğinde, eğer k. nörondaki durum değişikliği $\Delta a_k = a_{k2} - a_{k1}$, $\Delta E = E_2 - E_1$ 'e neden olursa, E enerjisi şu şekilde yazılabilir:

$$E(A) = -\sum_{i,j=1\ldots n,\ i<>k,\ j<>k} a_i a_j m_{ij} - a_k \sum_{j=1\ldots n} a_j m_{kj} - a_k \sum_{i=1\ldots n} a_i m_{ik} \qquad (1)$$

Böylece E_2-E_1 farkını almak ve Δa_k ile bölmek şunu verir:

$$\Delta E / \Delta a_k = -\sum a_j m_{kj} - \sum a_i m_{ik} \qquad (2)$$

$$= -AM_k^T - AM^k$$

Burada M_k, M ilişki matrisinin k. satırı, M^k ise k. sütunudur. Eğer M simetrikse, (2) eşitliğinin sağ tarafı sadece $-2AM^k$ olur. AM^k, a_k nöronu için giriş etkinlik toplamıdır. Klasik McCulloch-Pitts iki değerli nöron modellerinde olduğu gibi, eğer $AM^k>0$ ise a_k'nın eşiği +1, $AM^k<0$ ise -1'dir. Bu nedenle Δa_k ve AM_k aynı işaretlidir ve birbirleriyle çarpımları pozitiftir (yada sıfırdır). Buradan $\Delta E = -2\,\Delta a_k\,(AM^k) < 0$, E enerjisi sınırlandırıldığından, tek-yönlü işlem $E(A_r)$ lokal enerji

minimumu olan bir A_r'e yakınsar.

Eğer M simetrik değilse, genellikle tek-yönlü öz-ilişkili bellek işlemi kararsızdır. (2) denklemindeki AM_k^T ifadesi a_k 'dan diğer nöronlara çıkış etkinleşme tolamıdır ve genellikle $AM_k^T <> AM^k$ 'dır. Eğer çıkış toplamı büyüklüğü giriş toplamı büyüklüğünü geçerse ve iki toplamın işaretleri farklıysa, $\Delta E>0$ oluşur. Dolayısıyla titreşim olur.

ÇYİB hatırlama işlemi bir lineer-olmayan geri besleme işlemidir. A alanı yada a nüfusundaki her a_i nöronu ve B 'deki her b_j nöronu bağımsız ve asenkron (yada senkron) olarak, diğer nüfuslardaki nöronların sebep olduğu kendilerindeki giriş toplamını incelerler. Sonra giriş toplamının eşik değerini geçmesi, eşit olması yada küçük olmasına göre durum değiştirir yada değiştirmezler. Yani her bir nöron, girişlerinin toplamının sayısal bir eşik değerini aşması yada aşmamasına göre açık (+1) yada kapalı (-1) olurlar. Eğer girişlerin toplamı eşik değerine eşitse, nöron o anki durumunda kalır. b_j için girişlerin toplamı:

$$AM^j = \Sigma\, a_i m_{ij} \qquad i=1...n \qquad (3)$$

Burada M^j, M ilişki matrisinin j. sütunudur. a_i için girişlerin toplamı benzer şekilde:

$$BM_i^T = \Sigma\, b_j m_{ij} \qquad j=1...p \qquad (4)$$

Burada M_i, M'in (M^T'nin) i. satırı (sütunu) dır. Tüm nöronlar için eşiği 0 alırız. Özet olarak, a_i ve b_j için eşik fonksiyonları şöyledir:

$$BM_i^T > 0 \quad \text{ise} \quad a_i = 1 \ , \qquad (4)$$

$$BM_i^T > 0 \quad \text{ise} \quad a_i = 0 \ ,$$

$$AM^j > 0 \quad \text{ise} \quad b_j = 1 \ , \qquad (5)$$

$$AM^j < 0 \quad \text{ise} \quad b_j = 0 \ ,$$

Şimdi, ÇYİB'in (A_r,B_r) gibi kararlı bir duruma herhangi bir M matrisi ile ulaşılacağını ve bunun enerji fonksiyonunun lokal minimumuna karşılık geleceğini ispat edeceğiz.

E enerjisi $\{0,1\}^{n}*\{0,1\}^{p}$ faz uzayında ayrık yörüngeler boyunca azalır. Bunun böyle olduğunu, durum değişkenlerindeki Δa_i ve Δb_j değişimlerinin $\Delta E < 0$ üreteceğini göstermekle ispatlayacağız. Dikkat etmek gerekir ki, Δa_i , $\Delta b_j \in \{-1,0,1\}$ ikili durum değişkenleri içindir ve Δa_i , $\Delta b_j \in \{-2,02\}$ çift kutuplu durum değişkenleri içindir. Sadece a_i ve b_j'deki sıfır olmayan değişimleri dikkate almamız gerekir. Enerji fonksiyonu ifadesini çift toplam olarak yeniden yazarsak:

$$E(A,B) = - \Sigma \Sigma \, a_i b_j m_{ij} \qquad i=1...n \ , \ j=1...p$$

$$= - \Sigma_{i<>k} \Sigma_j \, a_i b_j m_{ij} - a_k \Sigma_j \, b_j m_{kj}$$

$$= - \Sigma_i \Sigma_{j<>k} \, a_i b_j m_{ij} - b_k \Sigma_i \, a_i m_{ik} \qquad (6)$$

a_k durum değişikliği yüzünden, $\Delta E = E_2 - E_1$ söyledir:

$$\Delta E / \Delta a_k = - \Sigma_j \, b_j m_{kj} = - BM_k^T \qquad (7)$$

(4) denklemindeki eşik kuralı nedeniyle a_k için girişlerin toplamı olarak, (7) denkleminin sağ tarafıdır. Buradan, eğer $0 < \Delta a_k = 1 - 0 = 1$ ise (4) denklemi $BM_k^T > 0$ olmasını sağlar ve böylece $\Delta E = - \Delta a_k \, (BM_k^T) < 0$ olur. Benzer şekilde, eğer $\Delta a_k < 0$ ise (4) denklemi

tekrar a_k 'nın giriş toplamı işaretinin, $BM_k^T < 0$ ile aynı olmasını sağlar ve böylece $\Delta E = - \Delta a_k (BM_k^T) < 0$ olur. Aynı yolla, Δb_k durum değişikliği nedeniyle enerji değişimi söyledir:

$$\Delta E / \Delta b_k = - \Sigma a_i m_{ik} = - AM^k \qquad i=1...n \qquad (8)$$

Yine (5) denklemindeki eşik kuralından, (8) denkleminin sağ tarafını, b_k için giriş toplamının negatifi olarak kabul ederiz. Buradan $\Delta b_k > 0$ 'ın sadece $AM^k > 0$ ise oluşacağı ve $\Delta b_k < 0$ 'ın sadece $AM^k < 0$ ise oluşacağı açıktır. Her iki durumda da $\Delta E = - \Delta b_k (AM^k) < 0$ dır. $\Delta a_i = \Delta b_j = 0$ iken $\Delta E = 0$ dır. Buradan, gerektiği gibi $\{0,1\}^n * \{0,1\}^p$ de (yada $\{-1,1\}^n * \{-1,1\}^p$ de) ayrık yörüngeler boyunca $\Delta E < 0$ dır [1].

3-6 DOGRU ÖGRENME KOŞULU

Bu bölümde bir öz-ilişkili ÇYIB için her bir depolanmış bilginin mükemmel hatırlanmasını sağlayacak yumuşatılmış bir yeter koşul türetilecektir.

Farzedelim (A_1,A_1), ... (A_m,A_m) şeklindeki m tane data çiftini depolamak istiyoruz. Burada A_i m boyutlu ikili (binary) bir vektördür ve her bir data çifti her bir girişin kendisi ile ilişkilendirilmesi neticesinde oluşturulmuştur. öz-ilişkili bellek matrisi M, bu m çifti çift kutuplu (bipolar) kodlama metodu yoluyla öğrenir.

$$M = X_1^T X_1 + ... + X_m^T X_m$$

Burada $X_i = [x_1^{(i)}, ... , x_n^{(i)}] \in \{-1,1\}^n$ şeklinde A_i'ye karşı düşen çift kutuplu (bipolar) bir vektördür. Eşikten (thresholding) önce çift-yönlü ilişkili belleğin çıkışı şöyle ifade edilir:

$$X_{çıkış} = X_i M$$
$$= X_i X_1^T X_1 + \Sigma (X_i X_j^T) X_j \quad j=1...m \quad j<>i$$
$$= nX_i + \Sigma (n-2H_{ij}) X_j \quad " \quad "$$

Burada $H_{ij} = H(A_i,A_j)$ 'dır. Her elemanın $X_{çıkış}$ çıkış vektörü, Hamming uzaklıklarının toplamına, depolanan vektörlerin boyutuna ve data çiftlerinin sayısına bağlıdır. Örneğin, $X_{çıkış}$'ın k. elemanının ifadesi şöyledir:

$$x_k^{(çıkış)} = nx_k^{(i)} + \Sigma (n-2H_{ij}) x_k^{(j)} \quad j=1 ... m \quad j<>i$$

Burada $x_k^{(i)}$ ve $x_k^{(j)}$ sırasıyla X_i ve X_j'nin k. elemanlarıdır.

Mükemmel hatırlama için yeter koşulu türetmeye, özel bir durum olan, tüm $i<>j$ için $H_{ij} < n/2$ kabulü ile

başladık. Bu kabül yukarıdaki eşitlikteki $(n-2H_{ij})$ terimine pozitif olmaya zorlar. Bu durumda n'nin $\Sigma_{i<>j}(n-2H_{ij})$ 'dan büyük olması makuldür. Yeter koşul H_{ij} toplamının bulunmasıyla n boyutu ve m data çifti sayısı terimlerine bağlanır. Burada $x_k^{(cıkış)}$'ın işareti her zaman, hangi $x_k^{(j)}$ bileşimi kullanıldığına bağlı olmaksızın $x_k^{(i)}$'nin işaretine uygun gelir. En basit yol, en kötü $x_k^{(j)}$'nin bileşiminin düşünülmesidir. En kötü bileşimden kastedilen, bunun, k. elemanın mükemmel hatırlanmasını önlemek için $\Sigma_{i<>j}(n-2H_{ij})$'nin etkisini arttırır. $H_{ij} < n/2$ kabullü bu özel durumda en kötü bileşim tüm $x_k^{(j)}$'lerin, $x_k^{(i)}$'lerin ters işaretlisi olmasıdır. Tüm i<>j için eğer $x_k^{(i)} = 1$ ise $x_k^{(j)} = -1$ dir. Bu durumda

$$x_k^{(cıkış)} = n(1) + \Sigma\ (n-2H_{ij})(-1) \quad j=1 \ldots m \quad j<>i$$

$$= (2-m)n + 2\ \Sigma\ H_{ij} \qquad " \qquad "$$

Doğru çıkışı elde etmek için, yukarıdaki denklemin sağ tarafı $x_k^{(i)}$ ile aynı işaretli olmalıdır. Bu durumda yeter koşul şu şekilde yazılır:

Tüm i=1 ... m için $\Sigma\ H_{ij} > (m-2)n/2$ j=1 ... m j<>i

Buna bir öz-ilişkili ÇYİB için doğru öğrenme koşulu adı verilir ve tüm X_i girişleri için mükemmel hatırlamayı sağlar.

Yukarıdaki koşulun tüm H_{ij} 'lerin n/2'den küçük olduğu özel durumdan türetilmiş olması önemlidir. Doğru öğrenme koşulu ñ'nün tanıtımı ile genelleştirilir. ñ kaç H_{ij}'nin n/2'den büyük olduğunun sayısıdır.

$$x_k^{(cıkış)} = nx_k^{(i)} + \Sigma\ (n-2H_{ij})\ x_k^{(j)} \quad j=1 \ldots m\ j<>i$$

Yukarıdaki eşitliğin sağ tarafına pozitif ve negatif $(n-2H_{ij})$'lere ayrılmasıyla şu ifade yazılabilir:

$$x_k^{(cikis)} = nx_k^{(i)} + \Sigma\,(n-2H_{ij})x_k^{(j)} + \Sigma\,(n-2H_{i\mu l})x_k^{(\mu l)}$$

$$j = 1 \ldots m \qquad j \neq i,\mu 1,\ldots,\mu ñ \qquad l = 1 \ldots ñ$$

Burada ñ, $(n-2H_{ij})<0$ şeklindeki data çifti sayısıdır. Gerçekte, bir önceki irdelemede ñ=0 ayarlamasıyla yeter koşulu türettik. Bundan sonra istenilen, ñ'nün mümkün olan tüm değerlerini kapsayan koşulu incelemektir. Bunun için şu anlaşılmalıdır ki; her bir ñ için, $\Sigma(n-2H_{ij})$'nin etkisini arttıran en kötü bir $x_k^{(j)}$'nin bileşmi mevcuttur. Bu kötü bileşime dayandırılarak yukarıdaki ifade şu şekilde yeniden yazılabilir:

$$x_k^{(cikis)} = nx_k^{(i)} + \Sigma\,(n-2H_{ij})x_k^{(j)} + \Sigma\,|n-2H_{i\mu l}|x_k^{(\mu l)}$$

$$j = 1 \ldots m \qquad j \neq i,\mu 1 \ldots \mu ñ \qquad l = 1 \ldots ñ$$

Doğru öğrenme koşulunun formülleştirilmesiyle, en kötü bileşenden dolayı gürültü terimi artsa bile mükemmel hatırlama başarılır. Bu nedenle, genelleştirilmiş doğru hatırlama formülü şu hale gelir [9]:

$$n > \Sigma\,|\,n-2H_{ij}\,| \qquad j = 1 \ldots m \qquad j \neq i$$

Aşağıda doğru hatırlama koşuluna örnek verilmiştir:

ÖRNEK: Farzedelim (A_1,A_1), ... ,(A_4,A_4) ilişki kodlamasını kullanarak depo edilmiş olsun ve her bir A_i şu şekilde verilmiş olsun:

A_1 = [1011001111011011]

A_2 = [1111001101010011]

A_3 = [1110000111011101]

A_4 = [0110001101001111]

İlişkili bellek matrisi olarak hizmet veren ilişki matrisi şu şekilde oluşturulur:

$$M = X_1^T X_1 + \ldots + X_4^T X_4$$

Şimdi X_1'i çift-yönlü ilişkili belleğe giriş olarak uygulayalım. Eşikten (thresholding) önceki çıkış şöyledir:

$$X_{(çıkış)} = X_1 M$$

$$= X_1 X_1^T X_1 + \Sigma\ (X_1 X_j^T) X_j \qquad j = 1 \ldots 4 \qquad j<>1$$

$$= 16 X_1 + \Sigma\ (16 - 2H_{1j}) X_j \qquad " \qquad "$$

$X_{çıkış}$'ın katı sınırlamalı transfer (eşik) fonksiyonuna uygulanmasıyla ÇYİB çıkışı elde edilir:

$$A_{çıkış} = [1111001111011011]$$

Burada $A_{çıkış}$'ın ikinci bit'i doğru olarak hatırlanmamıştır. Verilen data çiftleri için doğru öğrenme koşulunu kolayca kontrol edebiliriz:

$$H(A_1, A_2) = 3$$

$$H(A_1, A_3) = 6$$

$$H(A_4, A_4) = 6$$

Tüm Hamming uzaklıkları n/2'den küçük olduğu için ñ'yü 0 olarak tayin ederiz. ñ = 0 durumundaki doğru öğrenme koşuluna dayanılarak yukarıdaki Hamming uzaklılarının toplamı 16 dan büyük olmalıdır. Bu nedenle verilen data çiftleri mükemmel hatırlama için yeter koşulu sağlayamaz [9].

3-7 ÇIFT KUTUPLU MODUN IKILI MODA GÖRE USTUNLUGU

Çift-yönlü ilişkili bellekde çift kutuplu modun ikili moda göre daha uygun olmasının temel nedeni şudur: İkili paternlerin içindeki sıfırların, ekleme yapıldığında görmemezlikten gelinmesi fakat çift kutuplu paternlerdeki -1 lerin böyle olmamasıdır. 1 + 0 = 1 fakat 1 + (-1) = 0 dır. Eğer sayılar sinaptik güçleri temsil eden matris elemanları ise, ikili (binary) sayıların çarpımı ve toplamı sadece uyarıcı bağlantılar yada sıfır ağırlık bağlantıları üretir. Oysa çift kutuplu (bipolar) sayıların çarpımı ve toplamı uyarıcı ve yasaklayıcı bağlantılar üretir.

Elimizde çift kutuplu modla oluşturulmuş M bellek matrisi ve A giriş vektörü varken, A ile M 'in vektör çarpımını mı, yoksa X (A'nın çift kutuplu moddaki gösterimi) ile M 'in vektör çarpımını mı yapmalıyız? Genel olarak, durum vektörlerinin ikili mi, çift kutuplu kodlamasını mı kullanmalıyız?

Ortalama olarak çift kutuplu kodlama daha iyidir.

$$AM = (AX_j^T)Y_j + \Sigma\ (AX_i^T)Y_i \qquad i<>j \qquad i=1...m$$

Burada $H(A,A_j)$, A ile A_j arasındaki Hamming uzaklığıdır. X_i , ikili moddaki A_i 'nin çift kutuplu moda dönüşümüdür. Yani X_i , 0 ların yerini -1 alan A_i dir. Burada A, A_j ye diğer tüm depolanmış A paternlerinden daha yakındır. Daha öncede belirtildiği gibi, yukarıdaki eşitliğin sağ tarafındaki ilk terim sinyal terimi ve ikinci terim gürültü terimidir. Parantezli terimler, $a_i = AX_i^T = X_iA^T$ noktasal çarpımlardır. Bu nedenle yukarıdaki eşitlik, depolanmış

çıkış paternlerinin lineer bileşimi olarak yazılabilir:

$$AM = a_j Y_j + \Sigma\, a_i Y_i \qquad i<>j \qquad i=1...m$$

Y_j yi kuvvetlendirmek için a_j , Y_i yi düzeltmek için a_i isteriz. Eğer $H(A,A_i) > n/2$ ise A, A_i^c 'ye (A_i 'nin komplementine) A_i den daha yakındır. Buradan $a_i < 0$ istediğimiz için, Y_i Y_i^c 'ye dönüştürülebilir. Eğer $H(A,A_i) < n/2$ ise A , A_i ye A_i^c den ,daha yakındır. Böylece $a_i > 0$ olmasını isteriz. Eğer $H(A,A_i) = n/2$ ise A , A_i ile A_i^c arsında eşit uzaklıkta bir yerdedir ve böylece $a_i = 0$ olmasını isteriz. Eğer $Y_i = X_i$ ve M öz-ilişki matrislerinin toplamı ise bu istekler (koşullar) kayıtsız şartsız geçerlidir.

Eğer M ile A 'nın çift kutuplu dönüşümü X 'in vektör çarpımını yaparsak, şunu elde ederiz:

$$XM = x_j Y_j + \Sigma\, x_i Y_i \qquad i<>j \qquad i=1...m$$

Burada $x_i = XX_i^T$ dir. Yine $H(A,A_i) > n/2$ için $x_i < 0$, $H(A,A_i) < n/2$ için $x_i > 0$ ve $H(A,A_i) = n/2$ için $x_i = 0$ olmasını isteriz.

Çift kutuplu kodlama, terimlerin güçlerinde ve düzeltme katsayılarının işaretlerinde, ikili kodlamadan daha iyidir. Şu avantajlarını sıralayabiliriz:

1) $H(A,A_i) > n/2$ iken $x_i < a_i$

2) $H(A,A_i) < n/2$ iken $x_i > a_i$

3) $H(A,A_i) = n/2$ iken her zaman $x_i = 0$ dır.

Sonuçta istenilen istekler (koşullar) ortalama olarak çift kutuplu modda daha iyi sağlanır [1].

3-8 ÇYIB'IN PERFORMANSINI ÖGRENMEME İLE GELIŞTIRME

Burada Kosko'nun ileri sürdüğü ÇYIB'in kapasitesini ve gürültü toleransını geliştirmeyi hedefleyen bir kodlama yönteminden bahsedeceğiz. Bu yöntemle, depolanacak patern çiftlerine tekabül eden enerji minimumları arttırılır ve eşzamanlı olarak istenmeyen yada süpriz durumlar yokedilir. Bu metod, istenilen bölgelerde lokal minimumlara teşvik etmesi nedeniyle, literatürlerde tanımlanmış olan çoklu eğitim işleminin genişlemesidir. Ek olarak enerji ifadesindeki öğrenmeme terimide, süpriz durumları yoketmek için mevcuttur. Nöron ağını kurmak için süpriz durumlar ve parametre değerleri deneysel olarak belirlenir. Bu konu üzerine yapılmış bilgisayar simülasyonları sonucunda, öğrenmeme , ağın depolama kapasitesinde ve tek başına çoklu eğitimle başarılamayacak seviyede gürültü toleransındaki artış elde edilmiştir.

Bilindiği gibi Kosko'nun kolama metodu her bir eğitilmiş çifte karşı düşen bir lokal minumum yaratmayı sağlamaz. Dolayısıyla özel bir çift için bile hatırlanma garantisi yoktur. Çoklu eğitim metodu depolanan özel vektör çiftlerine tekabül eden bir enerji minimumuna neden olmasına rağmen bu, ağın, o çiftten kısa bir Hamming uzaklıkta bulunan süpriz minimuma yakınsamasını önlemez. Burada ele alınan öğrenmeme veya iptal işlemi bu gibi süpriz minimumların yok edilmesine olanak verir. Youn ve Kak , Hopfield tek-yölü öz-ilişkilisinin performansını geliştirmek için öğrenmeme (unlearning) tekniğini

uygulamışlardır. İstenilen durumlar için ağırlık matrisi, önce Kosko'nun ÇYİB metodu ile aynı düzende, dışçarpımların toplamı yardımıyla belirlenir. Süpriz durumlar, her bir ideal istenilen durumların Hopfield modelinde giriş olarak kullanılmasıyla belirlenir. İstenilen durumların komplementi olmayan bunlar, dış çarpımlarının bir öğrenmeme oranı ile çarpılıp, ağırlık matrisinden çıkarılması sayesinde öğrenilmez. Eş zamanlı olarak istenilen durumları arttırmak için hiç bir çalışma yapılmazsa bir kaç öğrenmeme saykılından (iterasyonundan) sonra performans küçülecektir. öğrenmeme oranın seçimi çok kritiktir. Büyük değerlerinin, ağırlık matrisinin kararlı duruma yakınsamasını önleyen sonsuz döngülere neden olduğu kaydedilmiştir. İstenilen durumların çoklu eğitiminin, süpriz durumları öğrenmemeyle birleşimi sayesinde ve eğitim ile öğrenmeme faktörlerinin tamsayı değerlerinin seçimiyle, başarılı depolama ve hatılamanın olacağı belirlenmiştir.

Genel olarak S süpriz durumun var olduğu, N vektör çiftini depolamak için öğrenmeme kodlama metodu aşağıdaki gibi bir ağırlık matrisi gerektirir:

$$Mus = \sum_{k=1...N} q_k X_k^T Y_k - \sum_{s=1...S} q_{ss} X_{ss}^T Y_{ss}$$

Burada (X_k, Y_k) depolanacak paternler ve (X_{ss}, Y_{ss}) ler süpriz durumlardır. q_k ve q_{ss} sırasıyla çoklu eğitim ve öğrenmeme faktörleridir. Bunların değerleri deneysel olarak belirlenmelidir [11].

BÖLUM 4 BILGISAYAR SIMULASYONU

4-1 BILGISAYAR PROGRAMININ ÇALIŞTIRILMASI VE YAPTIĞI IŞ

Bu program, Kosko'nun modeli, çoklu eğitim modeli ve yüksek-dereceli ÇYIB modeli olarak çalışabilir. Girilen bir vektör ve seçilen bir ÇYIB modeli ile girdiğimiz vektöre en yakın bir vektör çiftini elde edebiliriz.

Programda, her birinin boyutu n (A için) ve p (B için) olan N tane vektör çifti giriş olarak alınır. Pointer depolama yapısı kullanıldığı için N, n ve p değiştirilebilir. İstenildiği zaman yeni vektör çiftleri girilebilir. Başlangıç olarak vektör çifti sayısı, A ve B vektörlerinin uzunlukları programa verilir. Sonra program bunları kullanarak, klavyeden vektörleri alır. İstenildiği takdirde yeni bir eğitim çiftini girmek istemediğimizi program başlarken sorulan soruya h (hayır) diyerek belli edebiliriz. Bu durumda giriş dosyasında, vektör listesi pointerında depolanmış olan vektörleri kullanırız. Sonra seçilen kodlama yöntemine göre bir ilişki matrisi oluşturulur. Bu matris yardımıyla çiftlerin enerjileri hesaplanabildiği gibi klavyeden girilen bir vektöre en yakın gelen vektör ve onun eşi yani vektör çifti bulunabilir. Üçüncü yani yüksek-dereceli metodda bir ilişki matrisi oluşturulmaz. Bunda direk depolanan çiftler ve girilen vektör yardımıyla, vektör çifti hatırlanır.

Programda tip hatasının önlenebilmesi için tüm girişler kontrol edilir. Bir hata söz konusu ise klavyeden

giriş yapan kişi uyarılır. Eğer ilk olarak Çoklu eğitim metodu seçildiyse, Kosko'nun metoduna dönüş olamaz. Fakat tersi söz konusudur. Yani her iki metodlada işlem yapılmak isteniyorsa önce Kosko metodunu kullanmak gerekir. Çoklu eğitim tüm vektör çiftleri minimum noktaya varıncaya yada 1000 adım atılıncaya kadar devam ettirilir. Yüksek-dereceli metod her zaman kullanılabilir. Bu metodda hatırlama ise ya iki iterasyon sonucunda aynı vektör elde edilinceye kadar yada 20 iterasyon tamamlanılıncaya kadar devam eder. Işlemlerden sonra çıkışlar hem ekrana hemde çıkış dosyasına yazılır.

4-2 BILGISAYAR PROGRAMININ YAPISI

Bu program modüler yapıda yazılmıştır. Toplam olarak 23 tane prosedürü içermektedir. Asagıda her bir prosedürün görevi açıklanmıştır.

1- Input-check prosedürü : Bu prosedür tip (yazılım) hatalarını önlemek için girişleri kontrol eder. Hatalı giriş varsa uyarır. Girişler karakter olarak okunur ve sonra tamsayıya dönüştürülür. Burada başka bir prosedüre dallanma yoktur.

2- Rread1 prosedürü : Bu prosedür klavyeden girilen o elemanlı (yani n yada p boyutlu) j tane (yani N tane) vektörü okur ve onları giriş vektör dosyasına yazar. Aynı zamanda input-check prosedürünü kullanır.

3- Rread prosedürü : Bu prosedür o elemanlı (n yada p boyutlu) bir vektörü giriş dosyasından okur ve bunları, listenin başındaki bir başlangıç pointeri ile pointer listesine depolar. Başka bir prosedür kullanmaz.

4- Listvector prosedürü : Bu j (yada N) tane vektörden oluşan bir liste oluşturur. Buradaki her bir vektör o (n yada p) elemana sahiptir ve listenin başlangıcı bir başlangıç pointeri ile gösterilir. Rread prosedürünü kullanır.

5- Wr_arr prosedürü : Bu, karakter dizisini (string) çıkış dosyasına yazar. Başka prosedür kullanmaz.

6- Wr_int prosedürü : Bu prosedür tamsayı şeklindeki sayıları karakter tipine dönüştürür ve çıkış dosyasına yazar. Başka prosedür kullanmaz.

7- Wnatrix prosedürü : Mat pointeri ile gösterilen ilişki matrisini bulur. Bu matris Y vektörlerinin, X vektörlerini transpozeler ile çarpımlarının tolamıdır. Başka prosedür kullanmaz.

8- Energy prosedürü : Bu prosedür, -X * Matrix * Y'nin transpozesi şeklindeki enerjiyi bulur. Başka prosedür kullanmaz.

9- Wwrite prosedürü : Bu prosedür, ilişki matrisini çıkı dosyasına ve ekrana yazar. Başka prosedür kullanmaz.

10- Energydif prosedürü : Bu, hatırlama işleminin iki adımında hesaplanmış olan, önceki ve yeni enerji farkını bulup, minimumda oluşacak kararlı bir enerjiye ulaşmaya çalışır. Bu arada bir önceki ve o anki enerji eşit olur. E olarak enerji prosedürünü kullanır.

11- Multi_train prosedürü : Burada, çoklu eğitimin uygulandığı çift adına, ilişki matrisine $X^T * Y$ eklenir. Başkşa prosedür kullanılmaz.

12- Test_1_hamming prosedürü : Bu prosedür, X ve Y den oluşan bir vektör çiftinin bir Hamming uzaklıktaki vektörlerini kontrol eder. Amaç bu çiftden daha düşük bir enerjiye sahip bir vektör çiftinin mevcut olup olmadığının kontroludur. Eğer bu şekilde bir vektör varsa asıl çiftimizde değiştirilmesi gereken bitlerin yeri belirlenir Enerji prosedürünü kullanır.

13- Multi_process prosedürü : BU prosedür, Kosko algoritmasına göre oluşturulmuş olan ilişki matrisi ile X ve Y vektör listesini giriş olarak alır. Eğer özel

çiftin minimum enerjisine ulaşılmadıysa, bu matris için, her bir adımda $X^T * Y$ 'yi ilave eder. Çoklu eğitim adımları bu prosedürde en çok 1000 adımdır. Bu prosedür ek olarak, Test_1_hamming ve Multi_train prosedürlerini kullanır.

14- Recall prosedürü : Bu prosedür bir başlama vektörü ve ilişki matrisi verildiğinde, bir vektör çiftini hatırlamak için kullanılır. Energydif prodesürünüde kullanır.

15- Write_vector prosedürü : Bir vektörü ekrana ve çıkış dosyasına yazar. Wr_int prosedürünü kullanır.

16- Rec_read1 prosedürü : Bu prosedür, o elemanlı bir vektörü okur ve hatırlama işlemi (Recall prosedürü) için bunun giriş kabul edileceği bir dosyaya yazar. Input_check prosedürünü kullanır.

17- Rec_read prosedürü : Bu, Kosko veya Çoklu Eğitim kodlama metodu ile oluşturulmuş ilişki matrisi için bir vektör çiftini hatırlar. Rread, Wr_arr, Write_vector ve Recall prosedürlerini kullanır.

18- High_Recall1 prosedürü : Bu prosedür, başlangıç olarak girilen bir vektör için, yüksek düzen metodu ile vektör çiftinin hatırlanmasını sağlar. Başka bir prosedür kullanmaz.

19- High_Recall prosedürü : Bu prosedür, yüksek düzen metodunu kullanmak üzere, girilen bir vektörü bir dosyaya, high_recall1 prosedüründe gerekli olan giriş olarak yazar ve hatırlama işlemi dönüşünde vektör çiftini yazdırır. High_Recall1, Rread1, Write_vector prosedürlerini kullanır.

20- Disp2 prosedürü : Bu prosedür pointerlerin bellek bölgesinde gösterdiği alanlara geri döner. Amaç daha sonra bunların kullanılabilmesidir. Başka prosedür kullanılmaz.

21- Disp1 prosedürü : Bu prosedür pointerlerin bellek bölgesinde gösterdiği alanlara geri döner. Amaç daha sonra bunların kullanılabilmesidir. Disp2 prosedürü kullanılır.

22- Disp prosedürü : Bu prosedür, programdan çıkarken pointerlerin bellek bölgesinde gösterdiği alanlara geri döner. Amaç daha sonra bunların kullanılabilmesidir. Disp1 ve Disp2 prosedürünü kullanır.

23- Menu prosedürü : Bu prosedür, kullanıcı ile çevre arasındaki haberleşmeyi sağlar. Ayrıca, Input_check, Rread, Listvector, Wr_arr, Wwrite, Wmatrix, Multi_process, Rec_read1, Rec_read, Energy, High_Recall1, High_Recall ve Disp prosedürlerini kullanır.[12]

4-3 BILGISAYAR SIMULASYONU SONUCLARI

Bilgisayar simülasyonunda üç örnek ele alınmıştır. Bunlardan ilk ikisinin sadece Kosko'nun metodu ve yüksek dereceli metod için karşılaştırılmaları yapılabilmiştir. Çünkü alınan eğitim setleri ile tüm eğitim çiftlerini hatırlayabilecek çoklu eğitimli ilişki matrisi, programımızda makul bir süre içerisinde elde edilememiştir. Fakat üçüncü örnekte bu sağlanmış ve her üç metodun karşılaştırılması yapılmıştır.

ÖRNEK 1 :

Bu örnekte n=20 ve p=9 olan 4 eğitim çiftimiz var. Bunlar :

A_1=[1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1]

A_2=[1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1]

A_3=[1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1]

A_4=[1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1]

B_1=[1 1 1 0 0 0 0 1 0]

B_2=[1 0 0 0 0 0 0 0 1]

B_3=[0 1 0 1 0 0 1 0 1]

B_4=[1 1 1 0 1 1 1 0 0]

Bu eğitim çiftleri ele alınarak, tüm A vektörleri ve bunlardan 1 Hamming, 2 Hamming, 3 Hamming uzaktaki vektörler giriş olarak alınıp ağa uygulanmıştır. Sonuçlar aşağıda tablo halinde verilmiştir. Burada aradaki 1 Hamming uzaklık için kasdedilen, herhangi bir A vektörü ve bu vektörden herhangi bir biti farklı olan bir A_x vektörüdür.

Kosko'nun ÇYIB Modeli için hatırlama yüzdeleri:

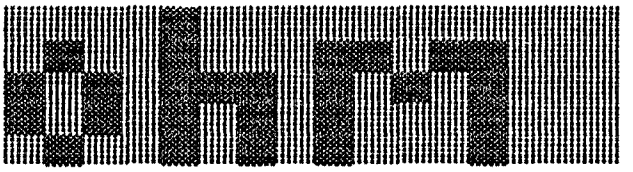
	Kendisini	1 Hamming uzaklık (20 durum)	2 Hamming uzaklık (190 durum)	3 Hamming uzaklık (1140 durum)
A_1	Hatırlamadı	% 0	% 0	% 0
A_2	Hatırladı	% 100	% 100	% 100
A_3	Hatırlamadı	% 0	% 0	% 0
A_4	Hatırladı	% 100	% 86	% 76

İkinci Dereceden ÇYIB Modeli için hatırlama yüzdeleri:

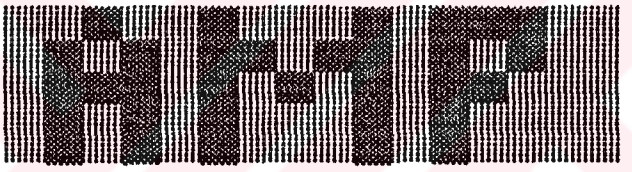
	Kendisini	1 Hamming uzaklık (20 durum)	2 Hamming uzaklık (190 durum)	3 Hamming uzaklık (1140 durum)
A_1	Hatırladı	% 75	% 74	% 68
A_2	Hatırladı	% 100	% 100	% 98
A_3	Hatırladı	% 75	% 28	% 0
A_4	Hatırladı	% 100	% 100	% 96

ÖRNEK 2 : Bu örnekte n=80 ve p=64 olan üç eğitim çifti seti ele alınmıştır. Bunlar; ohm-R, amp-A, volt-V . Aşağıda bu çiftler verilmiştir. Bunların sadece Kosko'nun modeli ve yüksek dereceli ÇYİB modeli yönünden karşılaştırılmaları yapılmıştır.

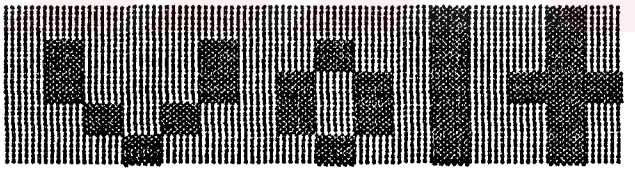
Burada ▒▒▒ 0'ı, ■ ise 1'i temsil etmektedir.



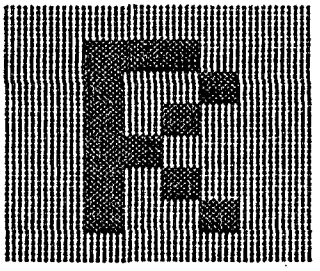
A_1



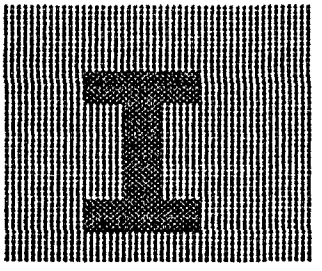
A_2



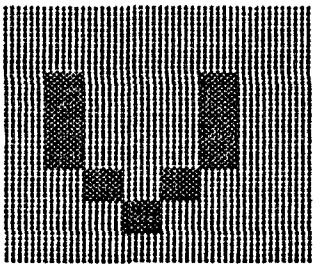
A_3



B_1



B_2



B_3

Yukarıdaki eğitim çiftlerinin her bir A vektörü sırasıyla Kosko'nun ÇYİB modeli ve ikinci dereceden ÇYİB modeline giriş olarak uygulanmıştır. Kosko'nun modeli için çıkışda hiç bir çift doğru olarak hatırlanmaz iken ikinci dereceden ÇYİB modeli ile tüm çiftler doğru olarak hatırlanmıştır. Yine bu A_1, A_2 ve A_3 vektörlerinin 1 Hamming uzaklığındaki (80*3=240 tane vektör) vektörler bu modellere giriş olarak uygulanmıştır. Sonuç olarak Kosko'nun modeli yine hiçbirini hatırlamayıp %0 başarı gösterirken, ikinci dereceden ÇYİB modeli tüm bu hatalı A giriş vektörlerine karşı doğru olan eğitim çiftlerini hatırlayarak %100 başarı sağlamıştır.

ÖRNEK 3 :

Bu son örnekde ise n=p=9 olan aşağıdaki üç çiftin, Kosko'nun ÇYİB modeli, Çoklu eğitimli ÇYİB modeli ve İkinci dereceden ÇYİB modeli için karşılaştırılmaları yapılmıştır. Sırasıyla A_1, A_2, A_3 ve bunlardan 1 Hamming, 2 Hamming, 3 Hamming uzaklıktaki vektörler giriş olarak her bir modele uygulanmış ve alınan sonuçlar aşağıda tablolar halinde verilmiştir. Ayrıca örnek olması amacıyla, A_1 ve bundan 1 Hamming uzaktaki vektörlerin bilgisayar programına giriş olarak uygulanması ve bundan elde edilen (her üç model için) çıktılar tablolardan sonra verilmiştir.

A_1=[1 0 0 1 1 1 0 0 0] B_1=[1 1 1 0 0 0 0 1 0]

A_2=[0 1 1 1 0 0 1 1 1] B_2=[1 0 0 0 0 0 0 0 1]

A_3=[1 0 1 0 1 1 0 1 1] B_3=[0 1 0 1 0 0 1 0 1]

Kosko'nun ÇYIB Modeli için hatırlama yüzdeleri:

	Kendisini	1 Hamming uzaklık (9 durum)	2 Hamming uzaklık (36 durum)	3 Hamming uzaklık (84 durum)
A_1	Hatırladı	% 56	% 25	% 36
A_2	Hatırlamadı	% 0	% 0	% 0
A_3	Hatırladı	% 67	% 84	% 36

Çoklu Eğitimli ÇYIB Modeli için hatırlama yüzdeleri:

	Kendisini	1 Hamming uzaklık (9 durum)	2 Hamming uzaklık (36 durum)	3 Hamming uzaklık (84 durum)
A_1	Hatırladı	% 89	% 70	% 65
A_2	Hatırladı	% 56	% 70	% 48
A_3	Hatırladı	% 56	% 14	% 0

Ikinci Dereceden ÇYIB Modeli için hatırlama yüzdeleri:

	Kendisini	1 Hamming uzaklık (9 durum)	2 Hamming uzaklık (36 durum)	3 Hamming uzaklık (84 durum)
A_1	Hatırladı	% 89	% 78	% 83
A_2	Hatırladı	% 56	% 0	% 0
A_3	Hatırladı	% 100	% 84	% 36

Sonuç olarak, bu örneklerin bilgisayar simülasyonu sonuçlarını daha kısa ifade edecek olursak ilk örnek için Kosko'nun metodu yanlış girilen 4050 tane giriş için (yani çeşitli Hamming uzaklıklardaki girişlerden) %59, Ikinci dereceden ÇYIB modeli ise %89 başarı sağlar. Burada başarıdan kasdedilen, çıkışta her iki vektöründe (A,B), doğru olarak hatırlanmasıdır. Ikinci örnekte girilen 240 tane hatalı giriş için Kosko'nun modeli %0, Ikinci dereceden ÇYIB modeli ise %100 başarı sağlamıştır. Son örnekte ise toplam 387 tane hatalı girişlere karşı Kosko'nun modeli %32, Çoklu Eğitimli ÇYIB modeli %42 ve Ikinci Dereceden ÇYIB modeli %45 başarı elde etmiştir.

Aşağıda örnek olması amacıyla, üçüncü örnekteki A_1 vektörünün birer Hamming uzaklığındaki vektörler (hatalı vektörler) her üç modele giriş olarak uygulanmış ve bilgisayardan alınan çıktılar verilmiştir.

EĞİTİLECEK "A" VEKTÖRLERİ (BIPOLAR FORM)

1	-1	-1	1	1	1	-1	-1	-1
-1	1	1	1	-1	-1	1	1	1
1	-1	1	-1	1	1	-1	1	1

EĞİTİLECEK "B" VEKTÖRLERİ (BIPOLAR FORM)

1	1	1	-1	-1	-1	-1	1	-1
1	-1	-1	-1	-1	-1	-1	-1	1
-1	1	-1	1	-1	-1	1	-1	1

KOSKO METODU SONUÇLARI

İLİŞKİ MATRİSİ (KOSKO ALGORİTMASI)

-1	3	1	1	-1	-1	1	1	-1
1	-3	-1	-1	1	1	-1	-1	1
-1	-1	-3	1	-1	-1	1	-3	3
3	-1	1	-3	-1	-1	-3	1	-1
-1	3	1	1	-1	-1	1	1	-1
-1	3	1	1	-1	-1	1	1	-1
1	-3	-1	-1	1	1	-1	-1	1
-1	-1	-3	1	-1	-1	1	-3	3
-1	-1	-3	1	-1	-1	1	-3	3

A BAŞLAMA VEKTÖRÜ :

1	0	0	1	1	1	0	0	0

İTERASYON SAYISI :

2

HATIRLANAN A VEKTÖRÜ:

1 0 0 1 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

1 1 1 0 0 0 0 1 0

A BAŞLAMA VEKTÖRÜ :

0 0 0 1 1 1 0 0 0

İTERASYON SAYISI :

2

HATIRLANAN A VEKTÖRÜ:

1 0 0 1 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

1 1 1 0 0 0 0 1 0

A BAŞLAMA VEKTÖRÜ :

1 1 0 1 1 1 0 0 0

İTERASYON SAYISI :

2

HATIRLANAN A VEKTÖRÜ:

1 0 0 1 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

1 1 1 0 0 0 0 1 0

A BAŞLAMA VEKTÖRÜ :

1 0 1 1 1 1 0 0 0

İTERASYON SAYISI :

2

HATIRLANAN A VEKTÖRÜ:

1 0 0 0 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

0 1 1 1 0 0 1 1 0

A BAŞLAMA VEKTÖRÜ :

1 0 0 0 1 1 0 0 0

İTERASYON SAYISI :

2

HATIRLANAN A VEKTÖRÜ:

1 0 0 0 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

0 1 1 1 0 0 1 1 0

A BAŞLAMA VEKTÖRÜ :

1 0 0 1 0 1 0 0 0

İTERASYON SAYISI :

2

HATIRLANAN A VEKTÖRÜ:

1 0 0 1 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

1 1 1 0 0 0 0 1 0

A BAŞLAMA VEKTÖRÜ :

1 0 0 1 1 0 0 0 0

İTERASYON SAYISI :

2

```
 HATIRLANAN A VEKTÖRÜ:
    1     0     0     1     1     1     0     0     0
 HATIRLANAN B VEKTÖRÜ:
    1     1     1     0     0     0     0     1     0

A BAŞLAMA VEKTÖRÜ :
    1     0     0     1     1     1     1     0     0
 İTERASYON SAYISI :
    2
 HATIRLANAN A VEKTÖRÜ:
    1     0     0     1     1     1     0     0     0
 HATIRLANAN B VEKTÖRÜ:
    1     1     1     0     0     0     0     1     0

 A BAŞLAMA VEKTÖRÜ :
    1     0     0     1     1     1     0     1     0
 İTERASYON SAYISI :
    2
 HATIRLANAN A VEKTÖRÜ:
    1     0     0     0     1     1     0     0     0
 HATIRLANAN B VEKTÖRÜ:
    0     1     1     1     0     0     1     1     0

 A BAŞLAMA VEKTÖRÜ :
    1     0     0     1     1     1     0     0     1
 ITERASYON SAYISI :
    2
```

HATIRLANAN A VEKTÖRÜ:

1 0 0 0 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

0 1 1 1 0 0 1 1 0

ÇOKLU EĞİTİM METODU

İLİŞKİ MATRİSİ (ÇOKLU EĞİTİM)

-1	5	3	1	-1	-1	1	3	-3
1	-5	-3	-1	1	1	-1	-3	3
-1	-3	-5	1	-1	-1	1	-5	5
5	-1	1	-5	-3	-3	-5	1	-1
-1	5	3	1	-1	-1	1	3	-3
-1	5	3	1	-1	-1	1	3	-3
1	-5	-3	-1	1	1	-1	-3	3
-1	-3	-5	1	-1	-1	1	-5	5
-1	-3	-5	1	-1	-1	1	-5	5

A BAŞLAMA VEKTÖRÜ :

1 0 0 1 1 1 0 0 0

İTERASYON SAYISI :

2

HATIRLANAN A VEKTÖRÜ:

1 0 0 1 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

1 1 1 0 0 0 0 1 0

A BAŞLAMA VEKTÖRÜ :

0 0 0 1 1 1 0 0 0

ITERASYON SAYISI :

2

HATIRLANAN A VEKTÖRÜ:

1 0 0 1 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

1 1 1 0 0 0 0 1 0

A BAŞLAMA VEKTÖRÜ :

1 1 0 1 1 1 0 0 0

ITERASYON SAYISI :

2

HATIRLANAN A VEKTÖRÜ:

1 0 0 1 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

1 1 1 0 0 0 0 1 0

A BAŞLAMA VEKTÖRÜ :

1 0 1 1 1 1 0 0 0

ITERASYON SAYISI :

2

HATIRLANAN A VEKTÖRÜ:

1 0 0 1 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

1 1 1 0 0 0 0 1 0

A BAŞLAMA VEKTÖRÜ :

1 0 0 0 1 1 0 0 0

İTERASYON SAYISI :

2

HATIRLANAN A VEKTÖRÜ:

1 0 0 0 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

0 1 1 1 1 1 1 1 0

A BAŞLAMA VEKTÖRÜ :

1 0 0 1 0 1 0 0 0

İTERASYON SAYISI :

2

HATIRLANAN A VEKTÖRÜ:

1 0 0 1 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

1 1 1 0 0 0 0 1 0

A BAŞLAMA VEKTÖRÜ :

1 0 0 1 1 0 0 0 0

İTERASYON SAYISI :

2

HATIRLANAN A VEKTÖRÜ:

1 0 0 1 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

1 1 1 0 0 0 0 1 0

A BAŞLAMA VEKTÖRÜ :

1 0 0 1 1 1 1 0 0

İTERASYON SAYISI :

2

HATIRLANAN A VEKTÖRÜ:

1 0 0 1 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

1 1 1 0 0 0 0 1 0

A BAŞLAMA VEKTÖRÜ :

1 0 0 1 1 1 0 1 0

İTERASYON SAYISI :

2

HATIRLANAN A VEKTÖRÜ:

1 0 0 1 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

1 1 1 0 0 0 0 1 0

A BAŞLAMA VEKTÖRÜ :

1 0 0 1 1 1 0 0 1

İTERASYON SAYISI :

2

HATIRLANAN A VEKTÖRÜ:

1 0 0 1 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

1 1 1 0 0 0 0 1 0

YÜKSEK DERECE METODU

A BAŞLAMA VEKTÖRÜ :

1 0 0 1 1 1 0 0 0

YÜKSEK DERECELI ÇYIB SONUÇLARI :

HATIRLANAN A VEKTÖRÜ:

1 0 0 1 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

1 1 1 0 0 0 0 1 0

A BAŞLAMA VEKTÖRÜ :

0 0 0 1 1 1 0 0 0

YÜKSEK DERECELI ÇYIB SONUÇLARI :

HATIRLANAN A VEKTÖRÜ:

1 0 0 1 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

1 1 1 0 0 0 0 1 0

A BAŞLAMA VEKTÖRÜ :

1 1 0 1 1 1 0 0 0

YÜKSEK DERECELI ÇYIB SONUÇLARI :

HATIRLANAN A VEKTÖRÜ:

1 0 0 1 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

1 1 1 0 0 0 0 1 0

A BAŞLAMA VEKTÖRÜ :

1 0 1 1 1 1 0 0 0

YUKSEK DERECELI ÇYIB SONUÇLARI :

HATIRLANAN A VEKTÖRÜ:

1 0 0 1 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

1 1 1 0 0 0 0 1 0

A BAŞLAMA VEKTÖRÜ :

1 0 0 0 1 1 0 0 0

YUKSEK DERECELI ÇYIB SONUÇLARI :

HATIRLANAN A VEKTÖRÜ:

0 1 1 1 0 0 1 1 1

HATIRLANAN B VEKTÖRÜ:

1 0 0 0 0 0 0 0 1

A BAŞLAMA VEKTÖRÜ :

1 0 0 1 0 1 0 0 0

YUKSEK DERECELI ÇYIB SONUÇLARI :

HATIRLANAN A VEKTÖRÜ:

1 0 0 1 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

1 1 1 0 0 0 0 1 0

A BAŞLAMA VEKTÖRÜ :

1 0 0 1 1 0 0 0 0

YUKSEK DERECELI ÇYIB SONUÇLARI :

HATIRLANAN A VEKTÖRÜ:

1 0 0 1 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

1 1 1 0 0 0 0 1 0

A BAŞLAMA VEKTÖRÜ :

1 0 0 1 1 1 1 0 0

YÜKSEK DERECELİ ÇYIB SONUÇLARI :

HATIRLANAN A VEKTÖRÜ:

1 0 0 1 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

1 1 1 0 0 0 0 1 0

A BAŞLAMA VEKTÖRÜ :

1 0 0 1 1 1 0 1 0

YÜKSEK DERECELİ ÇYIB SONUÇLARI :

HATIRLANAN A VEKTÖRÜ:

1 0 0 1 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

1 1 1 0 0 0 0 1 0

A BAŞLAMA VEKTÖRÜ :

1 0 0 1 1 1 0 0 1

YÜKSEK DERECELİ ÇYIB SONUÇLARI :

HATIRLANAN A VEKTÖRÜ:

1 0 0 1 1 1 0 0 0

HATIRLANAN B VEKTÖRÜ:

1 1 1 0 0 0 0 1 0

BÖLÜM 5 SONUÇ

Yapay nöron ağlarının temel konularından biri olan çift-yönlü ilişkili belleğin uygulama alanları genel olarak patern ve ses tanıma işlemleridir. Bu çalışmada ÇYİB için üç kodlama yöntemi üzerinde durulmuştur. İlk iki kodlama yönteminde yani Kosko'nun ÇYİB modeli ve Çoklu Eğitimli ÇYİB modelinde önce bir ilişki matrisi oluşturulmuştur. Daha sonra bu matrisden yararlanarak girilen bir vektöre karşı düşen vektör çiftinin hatırlanması sağlanmıştır. Üçüncü kodlama yöntemi olan yüksek dereceden ÇYİB'de ise istenilen dereceye ve eğitim çiftlerine bağlı olarak çıkışta yine girilen vektöre karşı düşen çift elde edilir. Yapılan bilgisayar simülasyonu örneklerinde bu derece 2 alınmış ve sonuçta en iyi performansın, bu yöntemle, daha sonra Çoklu Eğitim yöntemiyle ve en son olarak Kosko'nun yöntemi ile alındığı belirlenmiştir.

Burada tek yönlü ilişkili bellek yerine çift-yönlü ilişkili bellek kullanılmasının temel nedeni, ÇYİB'in daha küçük bir ilişki matrisi ile hatırlamayı sağlayabilmesidir. Bir diğer üstünlüğü ise tek yönlü ilişkili bellekte girilen vektöre (A yada hatalı A'ya) karşı tek bir vektör (doğru A) hatırlanabilirken, ÇYİB'de hem A hem de B olmak üzere iki vektör birden hatırlanabilir.

BILGISAYAR

PROGRAMI

```
program cyib(input,output);
uses crt,dos;
type
   vectorpointer=^vector;
   vector=record                    { vektör tipi,col=sütunun sayısı }
             col,j,v:longint; { v=j pozisyonundaki vektörün eleman degeri }
             next:vectorpointer
          end;
    veclistpointer=^veclist;
    veclist=record           { matris tipi,n=satırın sayısı,vv=matrisin }
               vv:vectorpointer; { satır sayısında=count daki }
               count,n:longint;    { ilk pointer elemanı }
               nnext:veclistpointer
             end;
    arr_type=packed array[1..41] of char;
    fil_type=file of longint;
var
   out:file of char;
   arr1,arr2,arr3,arr4,arr5,arr6,arr7,arr9,arr10,arr11,arr12,arr13:arr_type;
   vec_flag1,vec_flag2:file of boolean;
   vectors1,vectors2:fil_type;
   an8,bn8,an9,bn9,a,b,aa,bb,lp,lo: vectorpointer;
   ddii,c,d,e,ffn,ff,matrix,xxlist,yylist,pp,qq:veclistpointer;
   px,py,jj,oo,eee,coun,eeoi,iii,uij,ooo:longint;
   fla1,fla2:boolean;
   cchh:char;



procedure input_check(chra:char;
                      var intg:longint);
var
   l,lm:longint;
begin
   l:=0;
   lm:=100;
   while ord(chra)<>13 do
   begin
      if ((ord(chra)<48)or(ord(chra)>57)or(lm=0)) then
      begin          { giris geçerli tamsayı değil }
         readln;
         writeln(' GEÇERLI GIRIŞLER = 1000 DEN KUÇUK TAMSAYILAR ');
         writeln(' YENI GIRIŞ :');
         read(chra);
         l:=0;
         lm:=100;
      end;
      l:=l+(ord(chra)-48)*lm;      { karakteri tamsayıya dönüştürme }
      lm:=trunc(lm/10);
      read(chra);
   end;
   while lm<>0 do
   begin
      l:=trunc(l/10);
```

```
      lm:=trunc(lm/10);
   end;
   intg:=1;
   read(chra);
end;



procedure rread1 (j,o:longint);
var
   i,l,k:longint;
begin
   fla1:=true;
   for i:=1 to j do
   begin
      writeln('VEKTÖRÜ GIRIN (0 veya 1)');
      for l:=1 to o do
      begin
         read(cchh);                { vektor okuma }
         input_check(cchh,k);       { tip kontrolu }
         while ((k<>0)and(k<>1)) do
         begin
            writeln('0 veye 1 GIRIN');  { gecerli bir tamsayı degil }
            read(cchh);
            input_check(cchh,k);
         end;
         write(vectors1,k);          { giris vektör dosyasına yazma }
      end;
   end
end;



procedure rread (o:longint;
                 var pvec:vectorpointer;
                 var vect:fil_type);
var
   l,k:longint;
begin
   for l:=1 to o do
   begin
      read(vect,k);     { dosya okuma }
      if k=0 then
      k:=-1;
      pvec^.v:=k;
      pvec^.j:=1;
      pvec^.col:=o;
      if l<>o then
      begin
         new(a);
         pvec^.next:=a;     { yeni pointer yaratma }
         pvec:=a;
      end
   end;
```

```
   pvec^.next:=nil;
end;


procedure listvector(j,o:longint;
                     var lvec:veclistpointer);
var
   l:longint;
begin
   for l:=1 to j do
   begin
      new(b);
      bb:=b;
      rread(o,b,vectors1);     { vektör okuma }
      lvec^.vv:=bb;
      lvec^.count:=l;
      lvec^.n:=j;
      if l<>j then
      begin
         new(c);              { yeni vectorpointer yaratma }
         lvec^.nnext:=c;
         lvec:=c
      end;
   end;
   lvec^.nnext:=nil
end;



procedure wr_arr(arr:arr_type);
var
   i:longint;
begin
   for i:=1 to 2 do
   begin
      cchh:=chr(13);
      write(out,cchh);    { carriage return }
      cchh:=chr(10);      { new line }
      write(out,cchh);
   end;
   for i:=1 to 41 do
   write(out,arr[i]);     { çıkış dosyasına yazma }
   for i:=1 to 2 do
   begin
      cchh:=chr(13);
      write(out,cchh);
      cchh:=chr(10);
      write(out,cchh);
   end;
end;
```

```
procedure wr_int(numb:longint);
var
   ccou,ccvv,i,l,j1,j2:longint;
begin
   ccou:=4;
   ccvv:=1;
   if numb<0 then
   ccou:=ccou-1;
   l:=10;
   while trunc(abs(numb/l))<>0 do
   begin
      l:=l*10;
      ccou:=ccou-1;
      ccvv:=ccvv+1
   end;
   l:=trunc(l/10);
   for i:=1 to ccou do
   begin
      cchh:=' ';           { karakterlerin formatı }
      write(out,cchh)
   end;
   if numb<0 then
   begin
      cchh:='-';
      write(out,cchh);     { çıkış dosyasına bir '-' yazma }
   end;
   numb:=abs(numb);
   for i:=1 to ccvv do
   begin
      j1:=trunc(numb/l);
      numb:=numb-l*j1;
      j2:=abs(j1);
      j2:=j2+48;
      cchh:=chr(j2);
      write(out,cchh);     { çıkış dosyasına dijitleri yazma }
      l:=trunc(l/10)
   end;
end;


procedure wmatrix(xlist,ylist:veclistpointer;
                  var mat:veclistpointer);
var
   bp,k,l,t:longint;
begin
   d:=mat;
   for l:=1 to xlist^.vv^.col do
   begin
      new(a);
      d^.vv:=a;
      for k:=1 to ylist^.vv^.col do
      begin
         a^.v:=0;
```

```
            a^.col:=ylist^.vv^.col;
            a^.j:=k;
            if k<>ylist^.vv^.col then     { tüm elemanları sıfır }
            begin                         { olan matris yaratma }
               new(b);
               a^.next:=b;
               a:=b
            end
         end;
         a^.next:=nil;
         d^.count:=1;
         d^.n:=xlist^.vv^.col;
         if 1<>xlist^.vv^.col then
         begin
            new(e);
            d^.nnext:=e;
            d:=e
         end
      end;
      d^.nnext:=nil;
      d:=mat;
      for l:=1 to xlist^.n do
      begin
         a:=xlist^.vv:                    { iliski matrini bulma }
         b:=ylist^.vv;
         for t:=1 to xlist^.vv^.col do
         begin
            aa:=d^.vv;
            bp:=a^.v;
            a:=a^.next;
            for k:=1 to ylist^.vv^.col do
            begin
               aa^.v:=aa^.v+bp*b^.v;
               aa:=aa^.next;
               b:=b^.next
            end;
            b:=ylist^.vv;
            d:=d^.nnext
         end;
         xlist:=xlist^.nnext;
         ylist:=ylist^.nnext;
         d:=mat;
      end
   end;


procedure energy(x,y:vectorpointer; mat:veclistpointer;
                 var ee:longint);
var
   z,g:longint;
begin
   ee:=0;
   d:=mat;
```

```
   while x<>nil do
   begin
      z:=x^.v;
      x:=x^.next;
      aa:=y;
      a:=d^.vv;
      g:=0;
      while aa<>nil do
      begin
         g:=g+aa^.v*a^.v;
         aa:=aa^.next;
         a:=a^.next;
      end;
      ee:=ee+z*g;
      d:=d^.nnext;
   end;
   ee:=-ee;
end;


procedure wwrite(wvec:veclistpointer);
var
   i:longint;
begin
   while wvec<>nil do
   begin
      a:=wvec^.vv;
      while a<>nil do
      begin
         wr_int(a^.v);          { çıkış dosyasına yazma }
         if ((a^.v>=0)and(a^.v<10)) then
         write('   ',a^.v)
         else
         write('  ',a^.v);
         a:=a^.next
      end;
      writeln;
      for i:=1 to 2 do
      begin
         cchh:=chr(13);         { carriage return }
         write(out,cchh);
         cchh:=chr(10);         { new line }
         write(out,cchh);
      end;
      wvec:=wvec^.nnext
   end;
   readln
end;


procedure energydif(ee:longint; x,y:vectorpointer; mat:veclistpointer;
                    var flag:boolean;
                    var gh:longint);
```

```
var
   fh:longint;
begin
   energy(x,y,mat,fh);
   gh:=fh;
   if fh-ee=0 then    { minimum enerjiye ulasıldı }
   flag:=true
   else
   flag:=false;
end;


procedure multi_train(mulx,muly:vectorpointer;
                      var multm:veclistpointer);
var
   bp,k,l,t:longint;
   mula,mulb:vectorpointer;
begin
   d:=multm;
   mulb:=muly;
   for l:=1 to mulx^.col do
   begin
      mula:=d^.vv;
      for t:=1 to muly^.col do
      begin
         mula^.v:=mula^.v+mulx^.v*mulb^.v;
         mula:=mula^.next;
         mulb:=mulb^.next;
      end;
      d:=d^.nnext;
      mulx:=mulx^.next;
      mulb:=muly;
   end
end;


procedure test_1_hamming(vecx,vecy:vectorpointer;mat:veclistpointer;
                         var posx,posy,en,eoi:longint);
var
   l,minn,s:longint;
   veca:vectorpointer;
begin
   posx:=0;
   posy:=0;
   veca:=vecx;
   energy(vecx,vecy,mat,s);     { baslama enerjisini hesaplama }
   minn:=s; en:=s; eoi:=s;
   for l:=1 to vecx^.col do
   begin                  { X vektöründen bir hamming uzaktaki }
      veca^.v:=0-veca^.v;       { vektörlerin testi }
      energy(vecx,vecy,mat,s);
      if s<minn then
```

```
      begin
         en:=s;
         minn:=s;
         posx:=1
      end;
      veca^.v:=0-veca^.v;
      veca:=veca^.next
   end;
   veca:=vecy;
   for l:=1 to vecy^.col do
   begin                  { Y vektöründen bir hamming uzaktaki }
      veca^.v:=0-veca^.v;       { vektörlerin testi }
      energy(vecx,vecy,mat,s);
      if s<minn then
      begin
         posx:=0;
         en:=s;
         minn:=s;
         posy:=1
      end;
      veca^.v:=0-veca^.v;
      veca:=veca^.next
   end;
   eoi:=eoi-en;
end;



procedure multi_process(mtx,mty:veclistpointer;
                        var mtmat:veclistpointer);
var
   mta,mtb,mtd:veclistpointer;
   flag1:boolean;
   count,l,mtposx,mtposy,mten,mteoi:longint;
begin
   count:=0;
   mtd:=mtmat;
   flag1:=true;
   while ((flag1=true) and (count<1000)) do
   begin
      count:=count+1;
      flag1:=false;
      mta:=mtx; mtb:=mty;
      for l:=1 to mtx^.n do
      begin
         test_1_hamming(mta^.vv,mtb^.vv,mtd,mtposx,mtposy,mten,mteoi);
         if mteoi>0 then
         begin
            multi_train(mta^.vv,mtb^.vv,mtd);
            flag1:=true;
         end;
         mta:=mta^.nnext;
         mtb:=mtb^.nnext;
      end;
```

```
   end;
   writeln(count);
   if count=1000 then
   writeln('1000 ADIMLI COKLU EGITIMDEN SONRA GARANTILI HATIRLAMA YOK');
end;


procedure recall(matt:veclistpointer;
                 axx,bxx:vectorpointer;
                 var aq,bq:vectorpointer;
                 var rrdf:longint);


var
   eeee,fff,i,j,klm,s,it:longint;
   ddq:veclistpointer;
   aaq,bbq,fbbq,faaq,aaxq,bbxq,vvv:vectorpointer;
   con,flag:boolean;
begin
   it:=0;
   klm:=matt^.vv^.col;
   s:=matt^.n;
   aaxq:=axx;
   bbxq:=bxx;
   aaq:=aq;
   bbq:=bq;
   while aaxq<>nil do
   begin
      aaq^.v:=0;          { (a,b) iki bos vektor yaratma }
      bbq^.v:=0;
      aaq^.col:=s;
      bbq^.col:=klm;
      aaxq:=aaxq^.next;
      if aaxq<>nil then
      begin
         new(faaq);
         new(fbbq);
         aaq^.next:=faaq;
         bbq^.next:=fbbq;
         aaq:=faaq;
         bbq:=fbbq
      end
      else
      begin
         aaq^.next:=nil;
         bbq^.next:=nil;
      end
   end;
   aaxq:=axx;
   bbxq:=bxx;
   con:=false;
   eeee:=0;
   while con=false do
```

```
begin
   ddq:=matt;
   bbq:=bq;                              { bx=a*m }
   for i:=1 to axx^.col do
   begin
      vvv:=ddq^.vv;
      for j:=1 to bxx^.col do
      begin
         bbq^.v:=bbq^.v+aaxq^.v*vvv^.v;
         bbq:=bbq^.next;
         vvv:=vvv^.next;
      end;
      bbq:=bq;
      ddq:=ddq^.nnext;
      aaxq:=aaxq^.next;
   end;
   bbq:=bq;
   for i:=1 to bxx^.col do
   begin
      if bbq^.v>=0 then bbq^.v:=1
      else bbq^.v:=-1;
      bbq:=bbq^.next
   end;
   bbq:=bq;
   aaq:=aq;
   aaxq:=axx;
   ddq:=matt;
   for i:=1 to axx^.col do          { ax=bx * M_transpose }
   begin
      vvv:=ddq^.vv;
      for j:=1 to bxx^.col do
      begin
         aaq^.v:=aaq^.v+vvv^.v*bbq^.v;
         vvv:=vvv^.next;
         bbq:=bbq^.next;
      end;
      ddq:=ddq^.nnext;
      bbq:=bq;
      aaq:=aaq^.next
   end;
   aaq:=aq;
   for i:=1 to axx^.col do
   begin
      if aaq^.v>=0 then aaq^.v:=1
      else aaq^.v:=-1;
      aaq:=aaq^.next
   end;
   aaxq:=aq;
   bbxq:=bq;
   energydif(eeee,aaxq,bbxq,matt,con,fff);    { iki adım }
   eeee:=fff;              { arasındaki enerji farkını bulma }
   rrdf:=fff;
   it:=it+1;
end;
```

```
   writeln('ITERASYON SAYISI:',it);
   wr_arr(arr11);
   wr_int(it);
   writeln;
end;


procedure write_vector(aka:vectorpointer;ivq:longint);
var
   avq,bvq:longint;
begin
   bvq:=aka^.col;
   bvq:=bvq-ivq;
   for avq:=1 to bvq do      { vektör yazma }
   begin
      if aka^.v=1 then
      begin
         write('  1');
         wr_int(1)
      end
      else
      begin
         write('  0');;
         wr_int(0);
      end;
      aka:=aka^.next
   end;
   cchh:=chr(13);            { carriage return }
   write(out,cchh);
   cchh:=chr(10);            { new line }
   write(out,cchh);
   writeln;
   readln
end;


procedure rec_read1(o:longint);
var
   i,l,k:longint;
begin
   fla2:=true;
   writeln('VEKTÖRÜ GIRIN (0 veya 1)');
   for l:=1 to o do
   begin
      read(cchh);
      input_check(cchh,k);
      while ((k<>0)and(k<>1)) do      { yanlış giriş }
      begin
         writeln('0 veya 1 GIRIN');
         read(cchh);
         input_check(cchh,k);
      end;
```

```
      write(vectors2,k);          { dosyaya yazma }
   end
end;


procedure rec_read(o,si:longint;matt:veclistpointer;
                   var axx,bxx:vectorpointer);
var
   alp,alo,azz:vectorpointer;
   i:longint;
begin
   azz:=axx;
   reset(vectors2);
   read(vectors2,o);
   rread(o,azz,vectors2);    { A başlangıç vektörünü okuma }
   read(vectors2,si);
   close(vectors2);
   clrscr;
   writeln(arr4);
   wr_arr(arr4);
   azz:=axx;
   write_vector(azz,0);      { A başlangıç vektörünü yazma }
   new(azz);
   alp:=azz;
   new(azz);
   alo:=azz;
   bxx^.col:=si;
   recall(matt,axx,bxx,alp,alo,i);  { vektör çiftini hatırlama }
   writeln(arr5);
   wr_arr(arr5);
   write_vector(alp,0);      { hatırlanan A vektörünü yazma }
   writeln(arr6);
   wr_arr(arr6);
   write_vector(alo,0)       { hatırlanan B vektörünü yazma }
end;


procedure high_recall1(k:longint; xxlist,yylist:veclistpointer;
                  exx,fxx:vectorpointer;
                  var eq,fq:vectorpointer);
var
   i,j,l,u,uk,s,it:longint;
   eeq,ffq,feeq,fffq,eexq,ffxq,h,hh,x,y,fh:vectorpointer;
   pp,qq:veclistpointer;
   bay:boolean;
begin
   pp:=xxlist;
   qq:=yylist;
   it:=0;
   eexq:=exx;
   ffxq:=fxx;
   eeq:=eq;
```

```
ffq:=fq;
while eexq<>nil do
begin
   eeq^.v:=0;
   ffq^.v:=0;
   eeq^.col:=xxlist^.vv^.col;
   ffq^.col:=yylist^.vv^.col;
   eexq:=eexq^.next;
   if eexq<>nil then
   begin
      new(feeq);
      new(fffq);
      eeq^.next:=feeq;
      ffq^.next:=fffq;
      eeq:=feeq;
      ffq:=fffq;
   end
   else
   begin
      eeq^.next:=nil;
      ffq^.next:=nil;
   end
end;
eexq:=exx;
ffxq:=fxx;
eeq:=eq;
ffq:=fq;
new(h);
hh:=h;
for i:=1 to yylist^.vv^.col do
begin
   new(fh);
   h^.col:=yylist^.vv^.col;
   h^.next:=fh;
   h:=fh;
end;
h:=hh;
bay:=false;
while ((bay=false) and (it<20)) do
begin
   for i:=1 to yylist^.vv^.col do
   begin
      h^.v:=ffq^.v;
      h:=h^.next;
      ffq:=ffq^.next;
   end;
   h:=hh;
   ffq:=fq;
   for i:=1 to xxlist^.n do
   begin
      x:=xxlist^.vv;
      u:=0;
      for j:=1 to xxlist^.vv^.col do
      begin
```

```
      u:=u+x^.v*eexq^.v;
      x:=x^.next;
      eexq:=eexq^.next;
   end;
   uk:=1;
   for l:=1 to k do uk:=uk*u;
   y:=yylist^.vv;
   for j:=1 to yylist^.vv^.col do
   begin
      ffq^.v:=ffq^.v+uk*y^.v;
      ffq:=ffq^.next;
      y:=y^.next;
   end;
   ffq:=fq;
   eexq:=exx;
   xxlist:=xxlist^.nnext;
   yylist:=yylist^.nnext;
end;
xxlist:=pp;
yylist:=qq;
for i:=1 to yylist^.vv^.col do
begin
   if ffq^.v>0 then ffq^.v:=1
   else ffq^.v:=-1;
   ffq:=ffq^.next;
end;
ffq:=fq;
eeq:=eq;
for i:=1 to xxlist^.n do
begin
   y:=yylist^.vv;
   u:=0;
   for j:=1 to yylist^.vv^.col do
   begin
      u:=u+y^.v*ffq^.v;
      y:=y^.next;
      ffq:=ffq^.next;
   end;
   uk:=1;
   for l:=1 to k do uk:=uk*u;
   x:=xxlist^.vv;
   for j:=1 to xxlist^.vv^.col do
   begin
      eeq^.v:=eeq^.v+uk*x^.v;
      eeq:=eeq^.next:
      x:=x^.next;
   end;
   eeq:=eq;
   ffq:=fq;
   xxlist:=xxlist^.nnext;
   yylist:=yylist^.nnext;
end;
xxlist:=pp;
yylist:=qq;
```

```
      for i:=1 to xxlist^.vv^.col do
      begin
         if eeq^.v>0 then eeq^.v:=1
         else eeq^.v:=-1;
         eeq:=eeq^.next;
      end;
      eeq:=eq;
      ffq:=fq;
      eexq:=eq;
      ffxq:=fq;
      it:=it+1;
      s:=0;
      for i:=1 to yylist^.vv^.col do
      begin
         if ffq^.v<>h^.v then s:=s+1;
         ffq:=ffq^.next;
         h:=h^.next;
      end;
      ffq:=fq;
      h:=hh;
      if s=0 then bay:=true;
   end;
end;



procedure high_recall(o,oo:longint; xxlist,yylist:veclistpointer;
                  var exx,fxx:vectorpointer);
var
   elp,elo,ezz:vectorpointer;
   k:longint;
begin
   ezz:=exx;
   reset(vectors2);
   read(vectors2,o);
   rread(o,ezz,vectors2);
   read(vectors2,oo);
   close(vectors2);
   clrscr;
   writeln(arr4);
   wr_arr(arr4);
   ezz:=exx;
   write_vector(ezz,0);
   new(ezz);
   elp:=ezz;
   new(ezz);
   elo:=ezz;
   fxx^.col:=oo;
   writeln('KAÇINCI DERECEDEN ÇYIB ISTIYORSUNUZ ?');
   readln(k);
   writeln(arr13);
   wr_arr(arr13);
   writeln;
   high_recall1(k,xxlist,yylist,exx,fxx,elp,elo);
```

```
   writeln(arr5);
   wr_arr(arr5);
   write_vector(elp,0);
   writeln(arr6);
   wr_arr(arr6);
   write_vector(elo,0);
end;


procedure disp2(var pif:vectorpointer);
var
   pif1:vectorpointer;
   i,j:longint;
begin
   if pif<>nil then
   begin
      j:=pif^.col;
      for i:=1 to j do
      begin                { hatırlanılabilip hatırlanılamayacağının kontrolu
         pif1:=pif;
         pif:=pif^.next;
         if pif1<>nil then
         dispose(pif1)
      end
   end
end;


procedure disp1(var gif:veclistpointer);
var
   gif1:veclistpointer;
begin
   while gif<> nil do
   begin                    { dispose yapılıp yapılamayacağının kontrolu }
      gif1:=gif;
      gif:=gif^.nnext;
      disp2(gif1^.vv);
      dispose(gif1)
   end
end;


procedure disp(var qmatrix,qxxlist,qyylist:veclistpointer;
               var qan9,qbn9:vectorpointer);
begin
   if qmatrix<>nil then    { dispose yapılıp yağılamayacağının kontrolu }
   disp1(qmatrix);
   if qxxlist<>nil then
   disp1(qxxlist);
   if qyylist<>nil then
   disp1(qyylist);
```

```
   if qan9<>nil then
   disp2(qan9);
   if qbn9<>nil then
   disp2(qbn9);
end;



procedure menu;
var
   ss1,ss2,ss3,ss4,ss6,ss7,ss9,ss11,ss12,ss13:char;
begin
   clrscr;
   writeln(' BAŞLAMA GIRIŞ DOSYALARI AÇACAKMISINIZ (e/h) ?');
   readln(cchh);
   if ((cchh='e')or(cchh='E')) then
   begin
      fla1:=false;
      rewrite(vec_flag1);
      write(vec_flag1,fla1);
      close(vec_flag1);
      rewrite(vec_flag2);
      write(vec_flag2,fla1);
      close(vec_flag2);
   end;
   reset(vec_flag1);    { giriş dosyalarının boş olu olmadığının kontrolu }
   reset(vec_flag2);
   read(vec_flag1,fla1);
   read(vec_flag2,fla2);
   close(vec_flag1);
   close(vec_flag2);
   ss1:=' ';
   ss2:=' ';
   ss3:='.';
   ss4:='.';
   ss6:='.';
   ss7:='.';
   ss9:='.';
   ss11:='.';
   if fla1=true then
   ss11:=' ';
   ss12:='.';
   ss13:='.';
   matrix:=nil;
   ddii:=nil;
   xxlist:=nil;
   yylist:=nil;
   an9:=nil;
   bn9:=nil;
   an8:=nil;
   bn8:=nil;
   while ss1<>'0' do
   begin
      clrscr;
```

```
writeln('0 : ÇIKIS');
if ss2=' ' then
writeln('1 : EGITILECEK VEKTÖR ÇIFTLERINI GIRME');
if ss11=' ' then
writeln('8 : DOSYADA DEPOLANAN VEKTÖR ÇIFTLERINI KULLANMA');
if ss3=' ' then
writeln('2 : ILIŞKI MATRISI YARATMA');
if ss4=' ' then
writeln('3 : GARANTILI HATIRLAMA IÇIN ÇOKLU EGITIM');
if ss6=' ' then
writeln('5 : HATIRLAMA IÇIN BAŞLAMA VEKTÖR ÇIFTINI GIRME');
if ss9=' ' then
writeln('9 : HATIRLAMA IÇIN DOSYADA DEPOLANMIŞ VEKTÖR ÇIFTINI KULLANMA'
if ss7=' ' then
writeln('6 : TÜM VEKTÖR ÇIFTLERI IÇIN ENERJI HESABI');
if ss12=' ' then
writeln('4 : HATIRLAMA IÇIN YÜKSEK-DERECE METODU');
if ss13=' ' then
begin
   writeln('7 : HATIRLAMA IÇIN DOSYADA DEPOLANMIŞ VEKTÖR ÇIFTINI');
   writeln('    KULLANMA (YÜKSEK-DERECE)');
end;
readln(ss1);
if ss1='0' then                                    { çıkış }
disp(matrix.xxlist.yylist.an9.bn9)
else if ((ss1='1') and (ss2=' ')) then   { eğitilecek vektör }
begin                          { çiftlerini girme, onları giriş }
   ss2:='.';                   { dosyasına yazma }
   ss3:=' ';
   ss4:=' ';
   ss12:=' ';
   ss11:='.';
   fla2:=false;
   rewrite(vectors1);
   rewrite(vec_flag1);
   writeln('EGITIM ÇIFTI SAYISINI GIRIN');
   read(cchh);
   input_check(cchh,jj);
   write(vectors1,jj);
   writeln('A VEKTÖRÜNÜN UZUNLUGUNU GIRIN');
   read(cchh);
   input_check(cchh,oo);
   write(vectors1,oo);
   writeln('B VEKTÖRÜNÜN UZUNLUGUNU GIRIN');
   read(cchh);
   input_check(cchh,ooo);
   write(vectors1,ooo);
   writeln('A yı girin');
   rread1(jj,oo);
   writeln('B yi girin');
   rread1(jj,ooo);
   close(vectors1);
   reset(vectors1);
   read(vectors1,jj);
```

```
   read(vectors1,oo);
   read(vectors1,ooo);
   new(xxlist);
   ff:=xxlist;
   listvector(jj,oo,xxlist);   { Xvektör listesini yaratma }
   xxlist:=ff;
   new(yylist);
   ff:=yylist;
   listvector(jj,ooo,yylist);   { Y vektör listesini yaratma }
   close(vectors1);
   fla1:=true;
   write(vec_flag1,fla1);
   close(vec_flag1);
   yylist:=ff;
   clrscr;
   writeln(arr9);
   wr_arr(arr9);
   ff:=xxlist;
   wwrite(ff);           { X vektörlerini display birimine yazma }
   clrscr;
   writeln(arr10);
   wr_arr(arr10);
   ff:=yylist;
   wwrite(ff);           { Y vektörlerini display birimine yazma }
end
else if ((ss1='2') and (ss3=' ')) then
begin
   ss3:='.';        { Kosko algoritmasını kullanarak ilişki }
   ss6:=' ';        { matrisini yaratma }
   ss7:=' ';
   if fla2=true then
   begin
      ss9:=' ';
      ss13:=' ';
   end;
   new(matrix);
   ff:=matrix;
   wmatrix(xxlist,yylist,matrix);  { ilişki matrisini yaratma }
   clrscr;
   writeln(arr1);
   wr_arr(arr1);
   wwrite(ff);                     { ilişki matrisini yazma }
end
else if ((ss1='3') and (ss4=' ')) then
begin
   ss4:='.';        { Çoklu eğitim algoritmasını kullanarak }
   ss6:=' ';        { ilişki matrisini yaratma }
   ss7:=' ';
   if fla2=true then
   begin
      ss9:=' ';
      ss13:=' ';
   end;
   if ss3=' ' then
```

```
   begin
      new(matrix);
      ff:=matrix;
      wmatrix(xxlist,yylist,matrix);  { iliski matrisini yaratma }
      ss3:='.'
   end;
   multi_process(xxlist,yylist,matrix);
   writeln(arr7);
   wr_arr(arr7);
   wwrite(matrix);                    { iliski matrisini yazma }
end
else if ((ss1='5') and (ss6=' ')) then
begin
   ss9:=' ';    { baslama vektörü verildiğinde. Kosko veya Çoklu }
   rewrite(vectors2); { egitim algoritmasını kullanarak olusturulmus }
   rewrite(vec_flag2); { iliski matrisiyle vektör çiftini hatırlama }
   writeln(' A VEKTÖRÜNÜ GIRIN : ');
   write(vectors2.xxlist^.vv^.col);
   rec_read1(xxlist^.vv^.col);       { klavyeden vektör okuma }
   write(vectors2.yylist^.vv^.col);
   close(vectors2);
   fla2:=true;
   write(vec_flag2,fla2);
   close(vec_flag2);
   new(an9);
   bn9:=nil;
   ffn:=matrix;
   rec_read(xxlist^.vv^.col,yylist^.vv^.col,ffn,an9,bn9); { çifti }
end                                                       { hatırlama }
else if ((ss1='4') and (ss12=' ')) then
begin
   if ((ss3='.') or (ss4='.')) then     { Yüksek-derece metoduyla }
   ss9:=' ';                            { klavyeden girilen vektör }
   rewrite(vectors2);                 { için vektör çiftini hatırlama }
   rewrite(vec_flag2);
   writeln('A VEKTÖRÜNÜ GIRIN :');
   write(vectors2.xxlist^.vv^.col);
   rec_read1(xxlist^.vv^.col);
   write(vectors2.yylist^.vv^.col);
   close(vectors2);
   fla2:=true;
   write(vec_flag2,fla2);
   close(vec_flag2);
   new(an9);
   bn9:=nil;
   high_recall(xxlist^.vv^.col,yylist^.vv^.col,xxlist,yylist,an9,bn9);
end
else if ((ss1='6') and (ss7=' ')) then
begin
   c:=xxlist;  { tüm vektör çiftlernin enerjilerinin hesaplanması }
   e:=yylist;
   coun:=1;
   while c<>nil do
   begin
```

```
      lp:=c^.vv;
      lo:=e^.vv;
      energy(lp,lo,ff,eee);
      writeln('Enerji ',coun,' = ',eee);
      c:=c^.nnext;
      e:=e^.nnext;
      coun:=coun+1;
   end;
   readln;
end
else if ((ss1='8')and(ss11=' ')) then
begin
   ss2:='.';          { giris dosyasında depolanmış vektör }
   ss3:=' ';                { ciftini kullanma }
   ss4:=' ';
   ss12:=' ';
   ss11:='.';
   if fla2=true then
   ss13:=' ';
   reset(vectors1);
   read(vectors1,jj);    { vektör çifti sayısını okuma }
   read(vectors1,oo);    { vektörlerin uzunluğunu okuma }
   read(vectors1,ooo);
   new(xxlist);
   ff:=xxlist;
   listvector(jj,oo,xxlist);  { X vektör listesini olusturma }
   xxlist:=ff;
   new(yylist);
   ff:=yylist;
   listvector(jj,ooo,yylist);  { Y vektör listesini olusturma }
   yylist:=ff;
   close(vectors1);
   clrscr;
   writeln(arr9);
   wr_arr(arr9);
   ff:=xxlist;
   wwrite(ff);                { X vektörlerini yazma }
   clrscr;
   writeln(arr10);
   wr_arr(arr10);
   ff:=yylist;
   wwrite(ff);                { Y vektörlerini yazma }
end
else if ((ss1='9')and(ss9=' ')) then
begin
   new(an9); { Kosko veya Çoklu eğitim algoritmasıyla olusturulmus  }
   bn9:=nil; { iliski matrisiyle, giris dosyasında depolanmıs }
   ffn:=matrix;  { baslama vektör ciftini kullanarak hatırlama }
   rec_read(xxlist^.vv^.col,yylist^.vv^.col,ffn,an9,bn9); {hatırlama}
end
else if ((ss1='7')and(ss13=' ')) then
begin
   new(an9); { Yüksek-derece metoduyla, giris dosyasında depolanmıs }
   bn9:=nil; { baslama vektörü yardımıyla, vektör ciftini hatırlama }
```

```
      high_recall(xxlist^.vv^.col,yylist^.vv^.col,xxlist,yylist,an9,bn9);
    end
    else
    begin
      writeln('YANLIŞ SEÇIM');
      readln
    end

  end
id;
gin  { ana program }
  arr1:=' İLİŞKİ MATRİSİ (KOSKO ALGORİTMASI)       ';  { stringleri yükleme
  arr7:=' İLİŞKİ MATRİSİ (ÇOKLU EĞİTİM)            ';
  arr2:=' A VEKTÖRLERİ                             ';
  arr3:=' B VEKTÖRLERİ                             ';
  arr4:=' A BAŞLAMA VEKTÖRÜ :                      ';
  arr5:=' HATIRLANAN A VEKTÖRÜ:                    ';
  arr6:=' HATIRLANAN B VEKTÖRÜ:                    ';
  arr9:=' EĞİTİLECEK "A" VEKTÖRLERİ (BIPOLAR FORM)';
  arr10:=' EĞİTİLECEK "B" VEKTÖRLERİ (BIPOLAR FORM)';
  arr11:=' İTERASYON SAYISI :                      ';
  arr12:=' YÜKSEK-DERECELİ ÇYİB                    ';
  arr13:=' YÜKSEK DERECELİ ÇYİB SONUÇLARI :        ';
  assign(out,'out492.dat');       { çıkış dosyası, 'out492.dat' }
  assign(vectors1,'vec1.dat'); { vektör çiftleri için giriş dosyası }
  assign(vectors2,'vec2.dat'); { başlama vektörü için giriş dosyası }
  assign(vec_flag1,'vec_f1.dat');
  assign(vec_flag2,'vec_f2.dat');
  rewrite(out);
  menu;
  close(out);
id. { ana program }
```

KAYNAKLAR

1- B. Kosko 'Bidirectional Associative Memory' IEEE Trans. Syst. Man and Cybern. Vol 18, No 1, pp 49-60, Jan/Feb 1988

2- Y-F Wang, J.B. Cruz, J.H. Mulligan 'Two Coding Strategies for Bidirectional Associative Memory' IEEE Trans. Neural Networks, Vol 1, No 1, pp 81-91, March 1990

3- Y-F Wang, J.B. Cruz, J.H. Mulligan 'Guaranteed Recall of All Training Pairs for Bidirectional Associative Memory' IEEE Trans. Neural Networks, Vol 2, No 6, pp 559-567, November 1991

4- Y-F Wang, J.B. Cruz, J.H. Mulligan 'On Multiple Training for Bidirectional Associative Memory' IEEE Trans. Neural Networks, Vol 1, No 3, pp 275-276, September 1990

5- H-M Tai, C-H Wu, T-L Jong 'High-order Bidirectional Associative Memory' Electronics Letters, Vol 25, No 21, pp 1424-1425, 12th October 1989

6- Darpa, Neural Networks Study, 1988

7- A. Johannet, L. Personnaz, G. Dreyfus, J-D Gascuel, M. Weinfeld 'Specification and Implementation of a Digital Hopfield-Type Associative Memory With On-Chip Training' IEEE Trans. Neural Networks, Vol 3, No 4, July 1992

8- R.P. Lippmann 'An Introduction to Computing with Neural Nets' IEEE ASSP Magazine, April 1987

9- Wang, Bo-yeun 'Fuzzy Associative Memories: Identification and Control of Complex Systems' Gerogia Institute of Technology 1991

10- Mehmet Kuntalp, B.Tarık oranç 'Yapay Nöron Ağları' Elektrik Müh. 4. Ulusal Kongresi İzmir, sf 785-788

11- Venugopal Srinivasan and Chen-Siang Chia 'Improving Bidirectional Associative Memory Performance by Unlearning' IEEE 1991, pp 2472-2477

12- Metin Kasımoğlou 'Three Coding Strategies for Bidirectional Associative Memory' Boğaziçi University Computer Engineering, 1992

13- Fikret Gürgen, Yapay Nöron Ağları Ders Notları, Boğaziçi Üniversitesi Bilgisayar Müh. Bölümü, 1992

ÖZGEÇMIŞ

29 Eylül 1970 tarihinde Mersin'de doğdu. İlkokulu Mersin Barboros İlkokulunda, ortaokulu ve liseyi Mersin Dumlupınar Lisesinde okuyarak 1987 yılında mezun oldu. Aynı yıl Yıldız Üniversitesi Mühendislik Fakültesi Elektronik ve Haberleşme Mühendisliği Bölümüne girmeye hak kazandı. 4 yıllık lisans öğrenimimi tamamlayarak 1990-1991 öğretim yılında mezun oldu. 1991-1992 öğretim yılında ise yüksek lisans öğrenimime başladı. Halen Yıldız Teknik Üniversitesi Elektrik-Elektonik Fakültesi, Elektronik ve Haberleşme Mühendisliği Bölümünde araştırma görevlisi olarak çalışmaktadır.