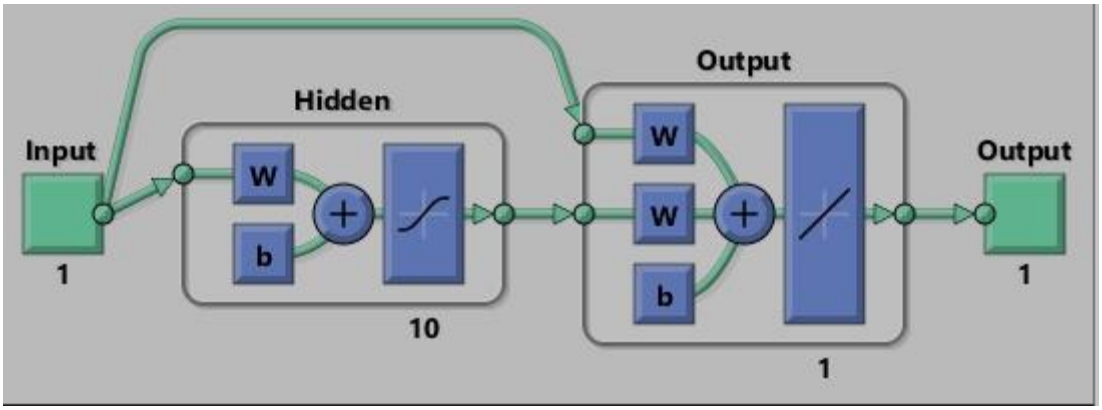
Cascadeforwardnet

Art arda ileten ağlar(ascade-forward networks), İleri beslemeli ağlarda(feed-forward networks) benzer, ama giriş ve her önceki katmandan sonraki katmanlara bir bağlantı içerir. İleri beslemeli ağlarda olduğu gibi, iki veya daha fazla katmanlı bir cascad ağı, yeterince gizli nöronlar verildiğinde keyfi olarak sonlu giriş-çıkış ilişkilerini öğrenebilir.



Competitive

bu öğrenme tipinde yaklaşımımız olasılıksal değildir ve kaynakların dağılımı için olasılıksal bir model beklemeyeceğiz. Bir başka fark da bu bölümde tartışacağımız öğrenme yöntemlerinin çevrimiçi olmasıdır. öğrenme sırasında elimizde tüm öğrenme kümesi olmayabilir, bu durumda örnekler birer birer geldikçe model parametrelerini ayarlamamız gerekecek. sanki bu öbekler girdiyi Betimlemek için kendi aralarında yarışma yaparlarmış gibi yarışmacı öğrenme deyimini kullanacağız . Sanki öbekler Yarışıyor biri kazanıyor ve yalnızca kazanın parametreleri güncelleniyor diye düşünüldüğünden böylesi yaklaşıma bazen "kazanan Hepsini alır" denir. Rekabetçi katmanlar, girdi vektörlerini, sınıflar arasında eşit sayıda vektör tercih edilerek, vektörler arasındaki benzerliğe göre belirli sayıda sınıfa sınıflandırmayı öğrenir.

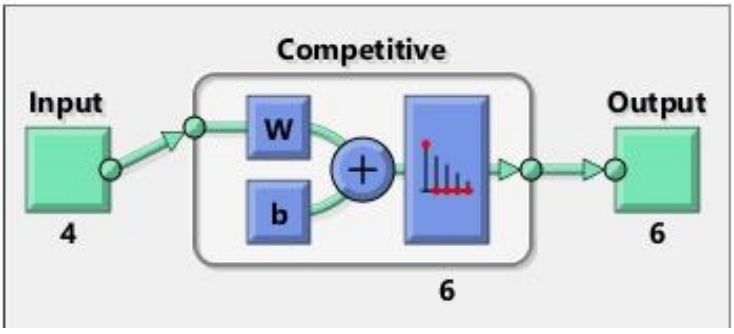
ör: Burada 150 iris çiçeği 6 sınıfa ayırmak için rekabetçi bir katman eğitildi.

ör: İşte bazı girdi verileri içinde üç küme bulmak için basit bir rekabetçi öğrenme algoritması.

1. (Kurulum.) Bir dizi sensörün üç farklı düğüme beslenmesine izin verin, böylece her düğüm her sensöre bağlanabilir. Her bir düğümün sensörlerine verdiği ağırlıkların 0,0 ile 1,0 arasında rastgele ayarlanmasına izin verin. Her bir düğümün çıkışı tüm sensörlerinin toplamı olsun, her bir sensörün sinyal gücü ağırlıkla çarpılır.

2. Ağ bir girdi gösterildiğinde, en yüksek çıkışa sahip düğüm kazanan olarak kabul edilir. Giriş, bu düğüme karşılık gelen küme içinde olarak sınıflandırılır.

3. Kazanan, ağırlıklarının her birini güncelleyerek ağırlığı daha zayıf sinyaller veren bağlantılardan daha güçlü sinyaller veren bağlantılara taşır. Böylece, daha fazla veri alındıkça, her bir düğüm temsil ettiği geldiği kümenin merkezinde birleşir ve bu kümedeki girdiler için daha güçlü ve diğer kümelerdeki girdiler için daha zayıf bir şekilde etkinleştirilir.



Elman neural network

Elman ağları, ileri besleme ağlarıdır (ileri beslemeli ağ) ve gecikmeli kademe gecikmeli bağlantılar eklenir. Tam dinamik türev hesaplarının (fpderiv ve bttderiv) kullanılabilirliği ile Elman ağı artık tarihsel ve araştırma amaçları dışında önerilmemektedir. Daha doğru öğrenme için zaman gecikmesi (timedelaynet), katman tekrarlayan (layrecnet), NARX (narxnet) ve NAR (narnet) sinir ağlarını deneyin. Bir veya daha fazla gizli katmana sahip Elman ağları, gizli katmanlarda yeterli nöron verildiğinde, herhangi bir dinamik giriş-çıkış ilişkisini keyfi olarak öğrenebilir. Ancak, Elman ağları daha az güvenilir öğrenme pahasına basitleştirilmiş türev hesaplamaları (gecikmeli bağlantıları yok sayan staticderiv kullanarak) kullanır. Jeffrey Elman tarafından önerilen sinir ağı mimarilerinin çoğu tekrarlıydı ve sıralı veya zamanla değişen modelleri öğrenmek için tasarlandı. Bu şekilde, algoritmalar öğrenilen değer veya olay serilerini tanıyabilir ve tahmin edebilir. Elman'ın bu mimariye olan birincil ilgisi dil işleme algoritmaları içindi, ancak dizileri içeren hemen hemen her şey için yararlı olduğunu öne sürdü. Elman'ın bir bağlam tanımı önceki iç durumlar etrafında dönmüştür ve böylece standart bir ileri beslemeli ağa bir "bağlam birimleri" katmanı eklemiştir. Bu şekilde, gizli birimlerin durumları, bir sonraki girdi aşaması sırasında gizli birimlere geri beslenebilir. Elman (1990) bunu en iyi açıklar: "Hem giriş birimleri hem de bağlam birimleri gizli birimleri etkinleştirir ve sonra gizli birimler çıktı birimlerini etkinleştirmek için ileri beslenir.Gizli birimler de içerik birimlerini etkinleştirmek için geri beslenir. Bu, ileri etkinleştirmeyi oluşturur. Göreve bağlı olarak, orada Öyleyse, çıktı bir öğretmen girdisi ile karşılaştırılır ve bağlantı güçlerini kademeli olarak ayarlamak için hatanın geri yayılımı kullanılır Tekrarlayan bağlantılar 1.0'a sabitlenir ve ayarlamaya tabi değildir. t + 1 adımının bir sonraki basamağı tekrarlanır. Bu kez içerik birimleri t zamanında tam olarak gizli birim değerleri olan değerler içerir. Bu bağlam birimleri böylece ağa bellek sağlar (s. 4-5). "

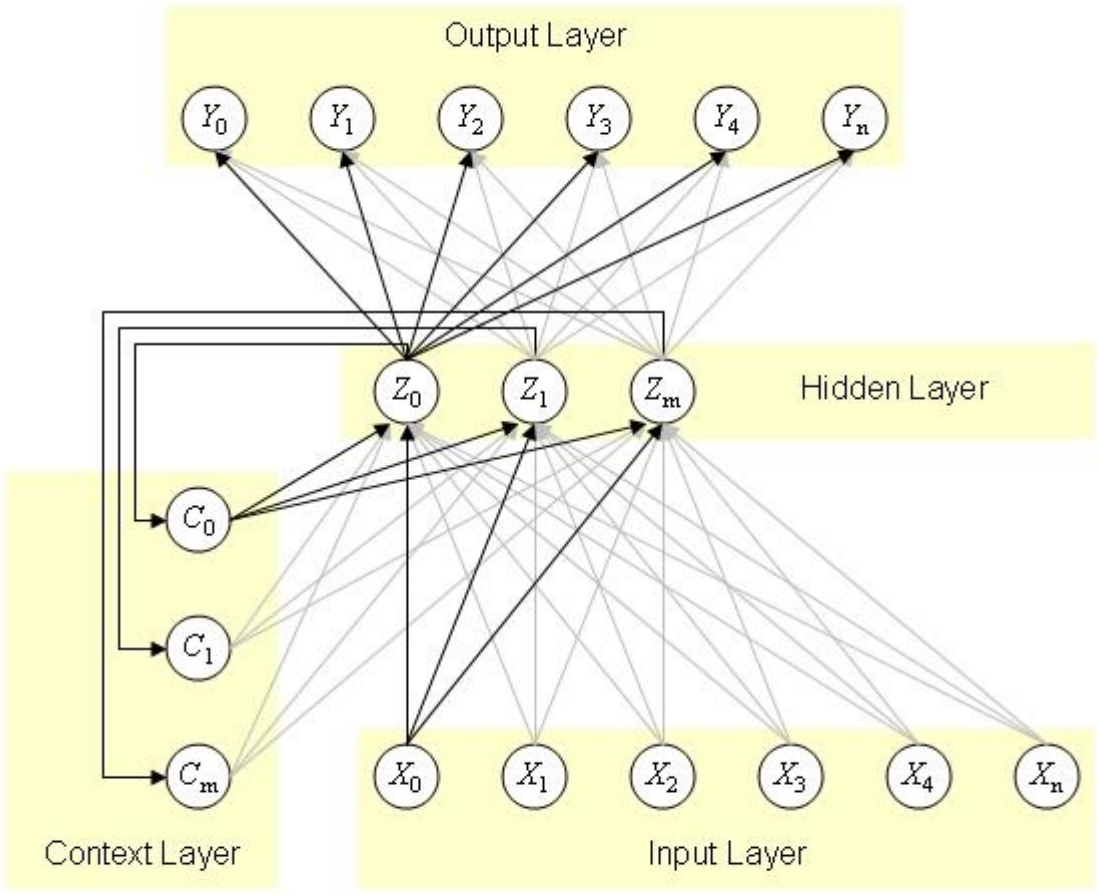


Figure 1. A common structure of an Elman network. Note the context layer, which receives input from, and returns values to, the hidden layer.

Şekil 1'de, değişen uzunluktaki sınırsız sayıda diziyi öğrenme potansiyeli olan basit bir tekrarlayan ağ tasarımına bir örnek verilmektedir. Elman ve diğerleri, giriş ve çıkış birimlerinin, bir harf dizisindeki bir sonraki harfi tahmin etmek için ağın eğitilebileceği ayrı harfleri temsil etmesini amaçladı. Bununla birlikte, bu tür tekrarlayan ağın doğasını anlamak daha kolaydır (en azından benim için), ünitenin seri sıradaki pozisyonuna bakarak (yani; Y0, Y1, Y2, Y3, ...). Bu çizimin amacı için, sadece 0, 3, 5, 2, 0 gibi sayı dizelerini kullanacağım, burada 0 Y0, 3 Y3, 5 Y5, vb. bir terminal sembolü ile biter; Bu örnek için 0 kullanacağım.Terminal sembolü ile başlamak ve bitirmek dışında, dize içindeki sayıların sırası herhangi bir sıra olabilir. Ve ağda yeterli birimle, dize uzunluğu sınırsızdır.Bu örnek için nöral ağ mimarisinde altı giriş ve altı çıkış birimi vardır (1'den 5'e kadar sayılar için artı terminal sembolü için 0). Üç gizli birim vardır, bunlara karşılık gelen üç içerik birimi vardır. X0 terminal sembolünü temsil ettiğinden, girişi 1'dir, diğer tüm girişler 0'dır. Terminal sembolü vektöre (1, 0, 0, 0, 0, 0) karşılık gelir. Gerisi bu kalıbı izler:

1 = 0, 1, 0, 0, 0, 0

2 = 0, 0, 1, 0, 0, 0

3 = 0, 0, 0, 1, 0, 0

4 = 0, 0, 0, 0, 1, 0

5 = 0, 0, 0, 0, 0, 1

Belirli bir dize için ağın eğitimi, dizenin uzunluğuna bağlı olarak sayı olmak üzere birkaç adım içerir. Eğitimin başında, bağlam birimlerinin aktivasyonları 0,5 olarak ayarlanmıştır. Terminal sembolü ilk olarak giriş birimlerine sunulur ve ağ halefi tahmin eder. Hata (eğitim dizisi tarafından belirtilen tahmini ve gerçek halef arasındaki fark) belirlenir ve geriye doğru yayılır ve ağırlıklar ayarlanır. Bağlam birimleri gizli birimin aktivasyonlarının bir kopyasını alır ve eğitim dizesindeki bir sonraki sembol (eğitimin ilk adımındaki çıktı birimleri için hedef olan) giriş birimlerine sunulur. Terminal sembolü (0) başka bir örneğine ulaşılana kadar eğitim bu şekilde devam eder.

ör:Burada basit bir zaman serisi problemini çözmek için bir Elman sinir ağı kullanılır.

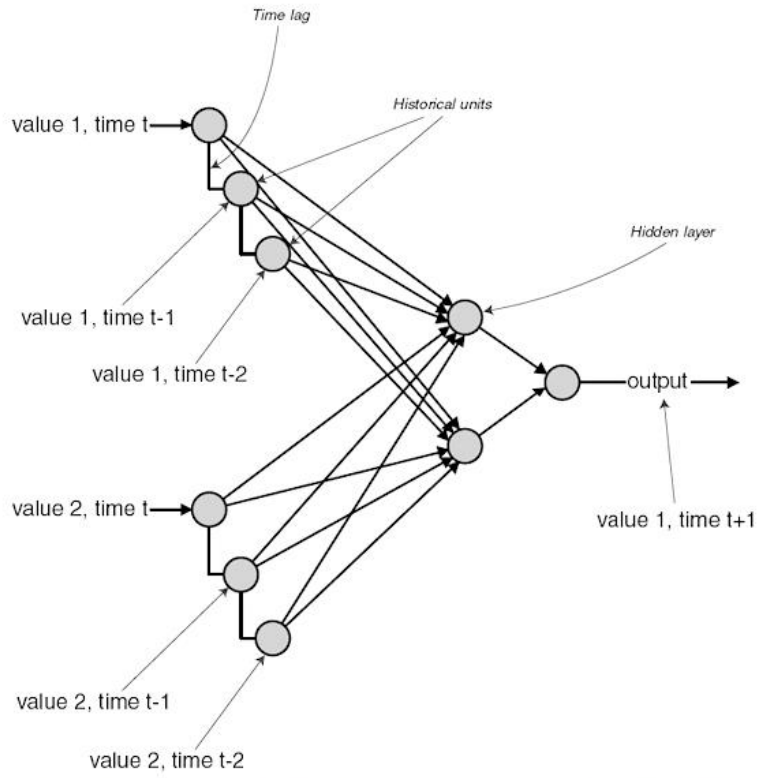
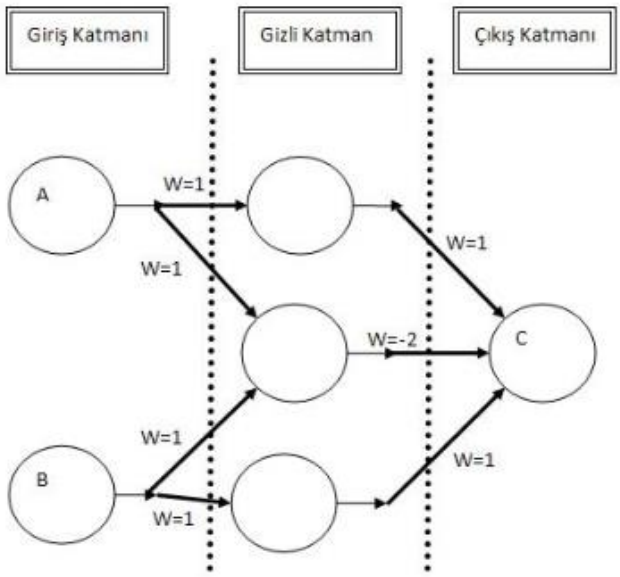


Figure 2. A time-delay neural network remembers the previous few training examples and uses them as input into the network. The network then works like a feed-forward, back propagation network.

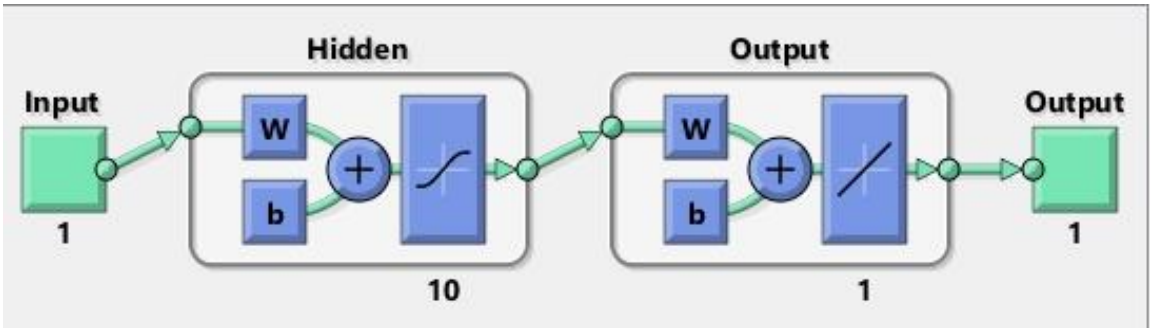
Feedforwardnet

İleri beslemeli ağlar bir dizi katmandan oluşur. İlk katmanın ağ girişinden bir bağlantısı vardır. Sonraki her katmanın bir önceki katmandan bağlantısı vardır. Son katman ağın çıktısını üretir. İleri beslemeli ağlar çıkış eşlemesine her türlü giriş için kullanılabilir. Bir gizli katman ve gizli katmanlarda yeterli nöron içeren ileri beslemeli bir ağ, herhangi bir sonlu girdi-çıktı haritalama problemine uyabilir. İleri beslemeli ağın özel sürümleri, uygun (fitnet) ve kalıp tanıma (patternnet) ağlarını içerir. İleri beslemeli ağdaki bir varyasyon, girdiden her katmana ve her bir katmandan sonraki tüm katmanlara ek bağlantılara sahip basamaklı ileri ağdır (kaskadforwardnet). İleri beslemeli yapay sinir ağlarında temel olarak 3 çeşit katman (layer) bulunur. Giriş, gizli ve çıkış katmanları sırasıyla yapay sinir ağına giren verileri tutan giriş katmanı, işlemlerin yapıldığı ve istenilen sonuca göre kendisini eğiten gizli katman yada katmanlar ve son olarak çıkış değerlerini gösteren çıkış katmanıdır. Bir gizli katmanın kaç seviye olacağı tamamen probleme göre belirlenmektedir. Her katman ve seviyede 1 veya daha çok sayıda sinir hücresi (nöron) bulunabilir. Aşağıda örnek bir tek gizli katmanı bulunan ileri beslemeli ağ tasviri bulunmaktadır:



Yukarıdaki ağda dikkat edilirse bütün sinapsis yönleri (yani verinin akışı) giriş katmanından çıkış katmanına doğrudur. Bir ileri beslemeli yapay sinir ağında her katmanda ne kadar sinir hücresi (neuron) olacağına aşağıdaki basit bir iki kurala göre karar verilebilir: Öncelilkle giriş katmanı için nöron sayısına sistemin girdisi olan verinin sayısına göre kolayca karar verilebilir. Örneğin sistemimizin öğrenmesini ve daha sonra sınıflandırmasını istediğimiz örüntünün (pattern) kaç veri ünitesinden oluştuğuna (örneğin bit) göre giriş katmanındaki sinir hücresi sayısı belirlenebilir. Kısaca giriş katmanındaki her sinir hücresi, sonucu etkilemesi istenen bir değişkene karşılık gelmektedir. Benzer uygulama çıkış katmanı için de kullanılabilir. Buna göre çıkış bilgisinin nasıl gösterilmesi istendiğine karar verildikten sonra bu çıkışta bulunması istenen her değişken için bir sinir hücresi bulundurulması gerekir. Örneğin bir sınıflandırma problemi için çıkış katmanında farklı sınıfların gösterilmesini sağlamaya yeterli miktarda Sinir hücresi bulunması yeterlidir. Veya bir filtreleme problemi için giriş ve çıkıştaki nöronların sayısı genelde eşit olur.

ör: basit bir sorunu çözmek için ileri beslemeli sinir ağının nasıl kullanılacağını gösterir.



Time delay

neural network

zaman içindeki bir diziyi tanımanın en kolay yolu bekletip uzay içindeki bir diziye çevirmektir. bu yapıldığında Şu ana dek gördüğümüz herhangi bir yöntem sınıflandırma için kullanılabilir zaman gecikmeli sinir anında önceki girdiler geciktirip en son girdiği ile aynı anda bir yöney olarak verilebilir hata geri yayma kullanılarak ağırlıklar güncellenir. Zaman içinde yerel öznitelikleri öğrenmek için yapısal bağlantılar ve zamanda taşımaya karşı değişmezlik için hazırlık paylaşımı kullanılabilir. Bu yapının en önemli kısıtı geçmişi saklayan pencerenin büyüklüğünün sabit olması ve baştan tanımlanmasının gerekmesidir.

ÖR: Bir konuşma sinyali durumunda, girdiler zaman içindeki spektral katsayılardır. Kritik akustik-fonetik özellikleri (örneğin, geçici geçişler, patlamalar, sürtünme, vb.) Önce kesin yerelleştirme gerektirmeden öğrenmek için, TDNN zaman kaydırma-değişmez olarak eğitilir. Zaman kaydırma değişmezliği, eğitim sırasında zaman boyunca ağırlık paylaşımı ile elde edilir: TDNN'nin zaman kaydırmalı kopyaları giriş aralığında yapılır (Şekil 1'de soldan sağa). Geriye yayılma daha sonra genel bir sınıflandırma hedef vektöründen yapılır (bakınız TDNN diyagramı, çıktı katmanında üç fonetik sınıf hedefi (/ b /, / d /, / g /) gösterilir), sonuçta her biri için genellikle değişecek gradyanlar elde edilir. zaman kaydırmalı ağ kopyaları. Ancak bu tür zaman kaydırmalı ağlar yalnızca kopya olduğundan, konum bağımlılığı ağırlık paylaşımı ile kaldırılır. Bu örnekte, ağırlık güncellemesi gerçekleştirilmeden önce her bir zaman kaydırmalı kopyadaki gradyanların ortalaması alınarak yapılır. Konuşmada, zaman kaydırmalı değişmez eğitimin, girdinin kesin konumlandırmasından bağımsız olan ağırlık matrislerini öğrendiği gösterilmiştir. Ağırlık matrislerinin, aynı zamanda, eski geçişler, patlama, vb. Gibi insan konuşma algısı için önemli olduğu bilinen önemli akustik-fonetik özellikleri saptadığı da gösterilebilir. TDNN'ler ayrıca ön eğitim yoluyla birleştirilebilir veya büyütülebilir.

ÖRNEKLER2:

Konuşma tanıma

Büyük kelime konuşma tanıma

Konuşmacı bağımsızlığı

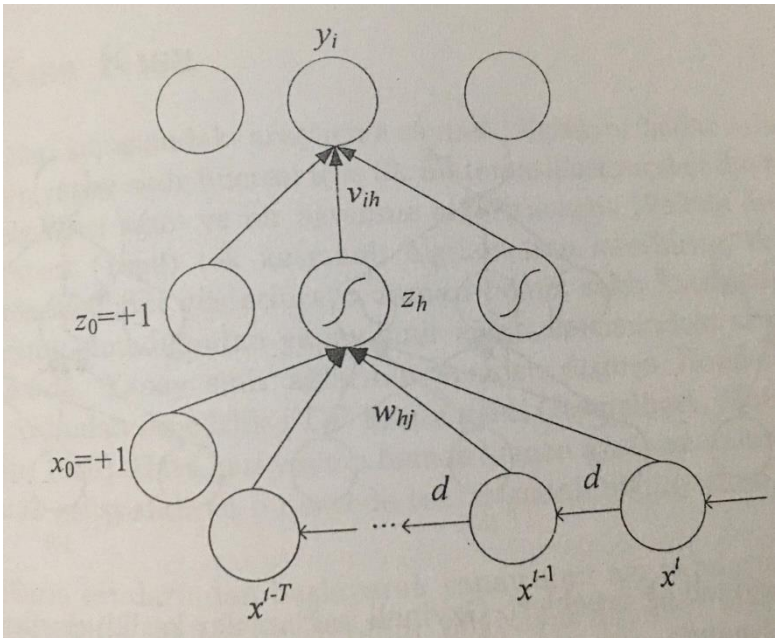
Yankılanma

Dudak okuma - görsel-işitsel konuşma

El yazısı tanıma

Video analizi

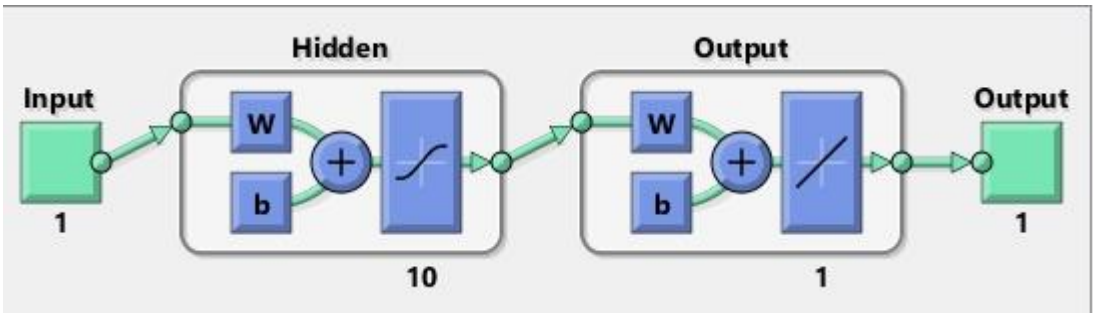
Görüntü tanıma

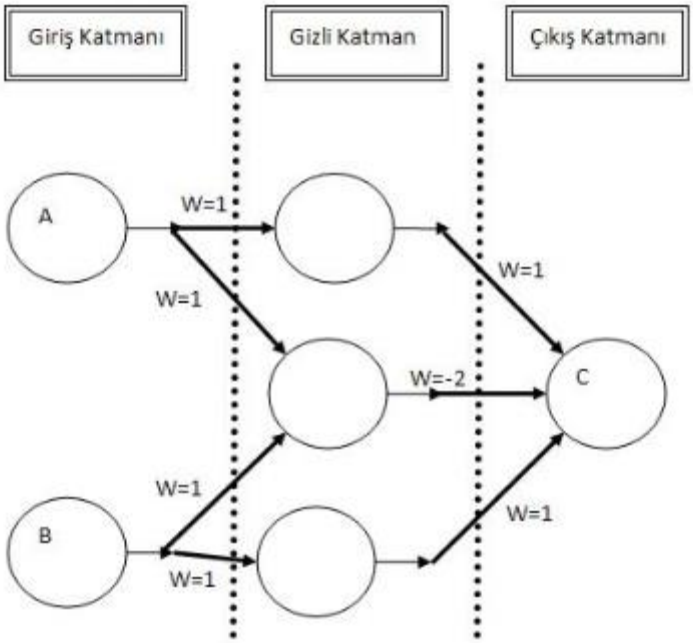


Feedforward distributed time delay

Dağıtılmış gecikme ağları, her giriş ve katman ağırlığının kendisiyle ilişkilendirilmiş bir musluk gecikme çizgisine sahip olması dışında, ileri beslemeli ağlara benzer. Bu, ağın zaman serisi giriş verilerine sonlu bir dinamik yanıt almasını sağlar. Bu ağ aynı zamanda sadece giriş ağırlığı üzerinde gecikmelere sahip olan zaman gecikmesi sinir ağına (zaman gecikmesi ağı) benzer.

ör:Burada basit bir zaman serisi problemini çözmek için dağıtılmış bir gecikme sinir ağı kullanılır.





Design generalized regression neural network

Genelleştirilmiş regresyon sinir ağları (grnns), genellikle işlev yaklaşımı için kullanılan bir tür radyal temel ağdır. grnns çok hızlı bir şekilde tasarlanabilir.

Özellikleri

newgrnn iki katmanlı bir ağ oluşturur. İlk katman radbas nöronlarına sahiptir ve netprod ile dist ve net girdili ağırlıklı girişleri hesaplar. İkinci tabaka purelin nöronlarına sahiptir, normprod ile ağırlıklı girişi ve netsum ile net girdileri hesaplar. Sadece ilk katmanın önyargıları vardır.newgrnn ilk katman ağırlıklarını P 'olarak ayarlar ve ilk

katman sapmalarının tümü 0.8326 / yayılma olarak ayarlanır, bu da +/– yayılma ağırlıklı girişlerinde 0,5'i geçen radyal temel işlevleriyle sonuçlanır. İkinci katman ağırlıkları W2, T'ye ayarlanmıştır.

---Avantajlar ve dezavantajlar---

RBFNN'e benzer şekilde, GRNN aşağıdaki avantajlara sahiptir:

Tek geçişli öğrenme, böylece backpagagation gerekmez.

Gauss fonksiyonlarını kullandığı için tahminlerde yüksek doğruluk.

Girişlerdeki sesleri işleyebilir.

Yalnızca daha az sayıda veri kümesi gerektirir.

GRNN'nin ana dezavantajları şunlardır:

Boyutu büyük olabilir, bu da hesaplama açısından pahalı hale getirir.

Geliştirmek için en uygun yöntem yoktur.

ör:

Burada, P girişleri ve T hedefleri göz önüne alındığında radyal bir ağ tasarlarsınız.

P = [1 2 3];

T = [2.0 4.1 5.9];

net = newgrnn(P,T);

Ağ, yeni bir giriş için simüle edilir.

P = 1.5;

Y = sim(net,P)

$$Y(x) = \frac{\sum_{k=1}^{N} y_k K(x, x_k)}{\sum_{k=1}^{N} K(x, x_k)}$$

where:

- $Y(x)$ is the prediction value of input x
- y_k is the activation weight for the pattern layer neuron at k
- $K(x, x_k)$ is the Radial basis function kernel (Gaussian kernel) as formulated below.

Gaussian Kernel [edit]

$$K(x, x_k) = e^{-d_k/2\sigma^2}, \qquad d_k = (x - x_k)^T (x - x_k)$$

where d_k is the squared euclidean distance between the training samples x_k and the input x.

Layer recurrent neural network

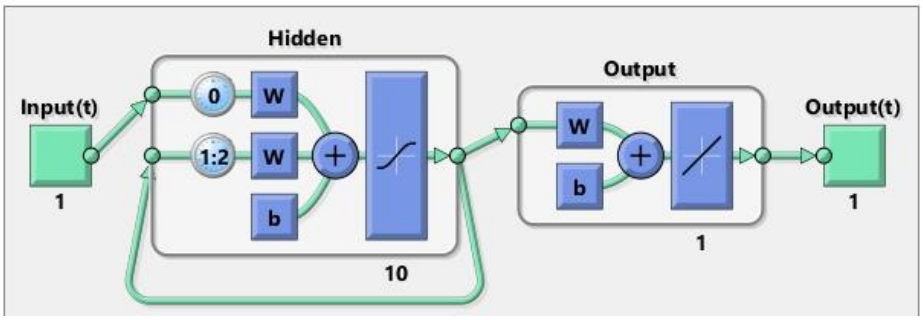
Katman tekrarlayan sinir ağları, ileri beslemeli ağlara benzer, ancak her katmanın kendisiyle ilişkilendirilmiş bir dokunma gecikmesi ile tekrarlayan bir bağlantısı vardır. Bu, ağın zaman serisi giriş verilerine sonsuz dinamik yanıt vermesini sağlar. Bu ağ, sınırlı giriş yanıtı olan zaman gecikmesi (zaman gecikmesi ağı) ve dağıtılmış gecikme (distdelaynet) sinir ağlarına benzer.

ör:Basit bir zaman serisi problemini çözmek için katman tekrarlayan sinir ağı kullanın.

```
[X,T] = simpleseries_dataset;

net = layrecnet(1:2,10);

[Xs,Xi,Ai,Ts] = preparets(net,X,T);

net = train(net,Xs,Ts,Xi,Ai);

view(net)

Y = net(Xs,Xi,Ai);

perf = perform(net,Y,Ts)
```

sonuç:

perf =6.1239e-11



Linear layer

Doğrusal katmanlar, doğrusal nöronların tek katmanlarıdır. Statik, 0 giriş gecikmeleriyle veya dinamik, giriş gecikmeleri 0'dan büyük olabilirler. Basit doğrusal zaman serisi problemleri üzerinde eğitilebilirler, ancak konuşlandırılırken değişikliklere uyum sağlayabilmeleri için konuşlandırılırken öğrenmeye devam etmek için adaptif olarak kullanılırlar. girişler ve çıkışlar arasındaki ilişki. Doğrusal olmayan bir zaman serisi ilişkisini çözmek için bir ağ gerekiyorsa, denemek için daha iyi ağlar timedelaynet, narxnet ve narnet'i içerir.

ör:Burada doğrusal bir katman basit bir zaman serisi problemi üzerine eğitilmiştir.

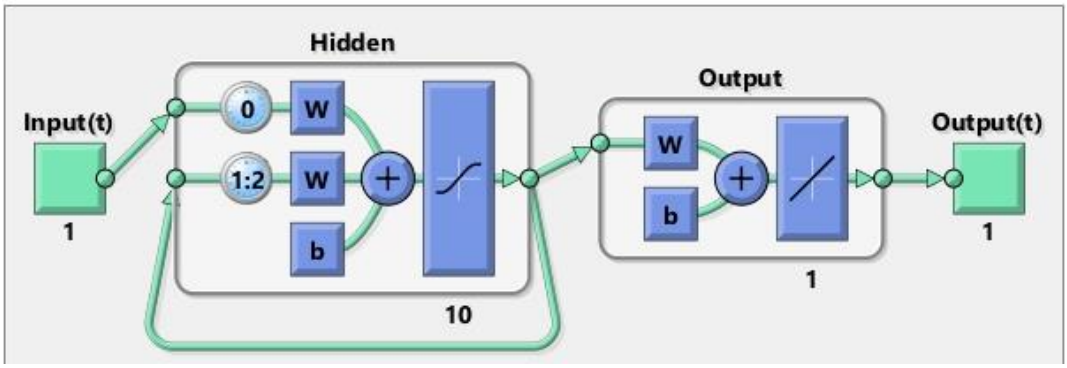
```
x = {0 -1 1 1 0 -1 1 0 0 1};

t = {0 -1 0 2 1 -1 0 1 0 1};

net = linearlayer(1:2,0.01);

[Xs,Xi,Ai,Ts] = preparets(net,x,t);

net = train(net,Xs,Ts,Xi,Ai);

view(net)
```

Y = net(Xs,Xi);

perf = perform(net,Ts,Y)

sonuç

perf =0.2396



Hopfield

8.2.2. *Hopfield* Ağı

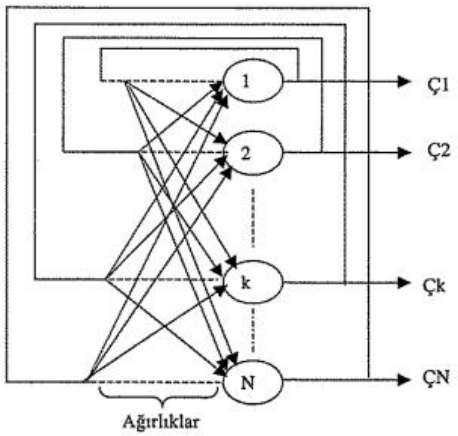
Hopfield ağı tek katmanlı ve geri dönüşümlü bir ağdır. Proses elemanlarının tamamı hem girdi hem de çıktı elemanlarıdır. Ağın bağlantı değerleri bir enerji fonksiyonu olarak saklanmaktadır. *Hopfiel* tarafından geliştirilen *Hopfield* ağı hakkında ayrıntılı bilgi [5] nolu referansta bulunabilir...

Günümüzde geliştirilmiş iki tür *Hopfield* ağı ardır:

- Kesikli (*discrete*) *Hopfield* Ağı: Bu ağlar çağrışımlı bellek (*associative memory*) olarak kullanılırlar.
- Sürekli (*continuous*) *Hopfield* Ağı: Bu ağlar ise daha çok kombinatoriyel optimizasyon problemlerinin çözümünde kullanılmaktadırlar.

8.2.2.1. Kesikli *Hopfield* Ağı

Bu ağdaki her hücrenin iki değeri vardır. Hücre *on* (+1) veya *off* (-1) olabilir. *N* proses elemanından oluşan ağ Şekil-8.4'de verilmiştir:



Şekil-8.4. *Hopfiel* ağının yapısı

Bir proses elemanının *t.* zamandaki girdisi *G(t)* olarak gösterilirse bu;

$$G_k(t) = \sum_{j \neq k}^{N} A_{jk} Ç_j(t-1) - \theta_k$$

formülü ile hesaplanmaktadır. Burada kullanılan *A* ağırlık değerini, *Ç(t-1)* proses elemanının bir önceki zaman dilimindeki çıktısını θ ise sabit eşik değerini göstermektedir.

Aynı proses elemanının çıktısı; *Ç(t)* ise şöyle hesaplanır:

$$Ç(t) = \text{sgn}(G(t))$$

Buradaki *sgn signum* fonksiyonunu göstermektedir. Yani,

$$Ç_k(t) = \begin{cases} +1 & \text{Eğer } G_k(t) > U_k \\ -1 & \text{Eğer } G_k(t) > U_k \\ Ç_k(t-1) & \text{Aksi halde} \end{cases}$$

Burada kullanılan U değeri de bir eşik değeridir. Pratikte 0 değeri seçilmekle beraber böyle bir zorunluluk yoktur. Çağrışımlı bellekte de olduğu gibi *Hopfield* ağının eğitilmesinde de ÇKA ağlarında olduğu gibi iki faz vardır:

- Ağırlıkları belirleme ve saklama fazı
- Bilgilere ulaşma fazı.

Ağırlıkların Belirlenmesi ve Bellekte Saklanması

Ağırlıkların belirleme fazı ağın öğrenme aşamasını göstermektedir. Ağın eğitimi bir defada öğrenme prensibine göre aşağıdaki formül kullanılarak gerekleşmektedir:

$$A_{ij} = \begin{cases} \frac{1}{N} \sum_{p=1}^{M} \chi_i^p \chi_j^p & \text{Eğer } j \neq k \text{ ise} \\ 0 & \text{Aksi halde} \end{cases}$$

Burada *M* öğrenilecek (saklanacak) örnek sayısının göstermektedir. χ ise bir örneğin *i.* ve *j.* elemanının değerlerini göstermektedir. Bu ağırlıklar hesaplandıktan sonra sabitlenirler. Dikkat edilirse burada A_{ij} ile A_{ji} ağırlıkları aynıdır. Bu da oluşturulan Ağırlık matrisinin simetrik bir matris olması demektir.

Ağın kullanılabilmesi için durağan *(stable)* hale gelmiş olması gerekmektedir. Bu ise şu şekilde sağlanabilir.

Bilgilerin Çağrılması

Bu fazda, ağa daha önce görmediği yani eğitim setinde olmayan bir örnek gösterilir. Bu örnek eksik bilgiler içerebilir. Ağın görevi bu eksiklikleri belirlemek ve örneğin tamamını hafızadan bulmaktır. Bunun için verilen örnek ağa sunulur ve ağın iterasyonlar yaparak durağan hale gelmesi beklenir. Ağ durağan hale gelince ürettiği çıktı, ağın kendisine gösterilen girdiye ürettiği cevabı olarak görülür. Girdi örneği *X* (*X*1, *X*2, *X*3,.....*X*N) başlangıç değerlerine atanmak üzere ağın iterasyonları yukarıda da gösterilen şu formüle göre devam eder.

$$Ç_k(t+1) = \text{sgn}\left(\sum_{j=1,\, j\neq k}^{N} A_{jk} Ç_j(t) - \theta_k \right)$$

Bu formülün çalışabilesi için girdi vektörü, ağın başlangıç çıktı değerleri olarak atanır. Yani,

$$Ç(0) = X = (X1, X2, X3, ...XN)$$

olarak alınır. Ağın durağan hale gelmesi bir enerji fonksiyonunun değerinin en azlanması demektir. Bu enerji fonksiyonu;

$$E(t) = -\frac{1}{2} \sum_{i=1,}^{N} \sum_{j=1, j\neq i}^{N} A_{ij} Ç_i(t) Ç_j(t) + \sum_{i=1}^{N} Ç_i(t) \theta_i$$

şeklinde verilmektedir. Ağ çalışırken bu enerji fonksiyonu ya azalır yada değişmez. Dolayısıyla zaman içinde ağın minimum hata düzeyine ulaşması (durağan duruma geçmesi) her durumda mümkün olmaktadır.

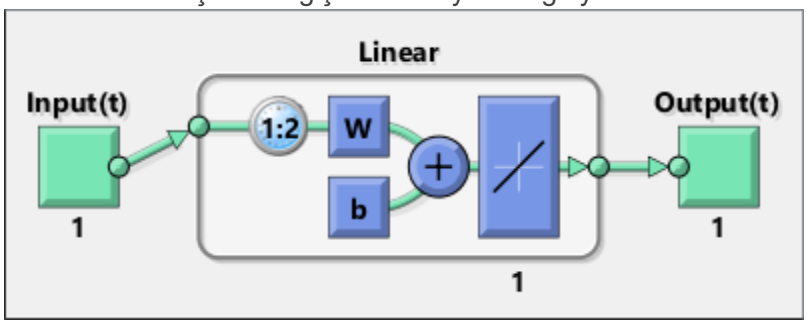
8.2.2.2. Sürekli *Hopfield* Ağı

Bu ağlar kesikli hopfield ağlarının aynısıdır. Aradaki fark ise *signum* fonksiyonunun yerine *sigmoid* fonksiyonunun kullanılmasıdır. Bu durumda ağın çıktı değerleri 0-1 arasında sürekli değerler olabilmektedir.

Hopfield ağının en önemli uygulamalarından birisi geleneksel optimizasyon algoritmaları ile çözümü mümkün olmayan veya çok zor olan gezgin satıcı problemini çözmesidir. Bu problemde bir satıcı *N* adet şehire gitmek zorundadır. Bir şehre bir defa uğramak koşulu ile en kısa zamanda bütün şehirleri gezebilmesi için izlemesi gereken rotanın bulunması istenmektedir. Bu problemin *Hopfield* ağı ile çözülmesi [6] nolu referansta ayrıntılı olarak anlatılmıştır.

Linear Layer - Train (Doğrusal Katman Eğitimi)

Doğrusal katmanlar, doğrusal nöronların tek katmanlarıdır. Statik, 0 giriş gecikmeleriyle veya dinamik, giriş gecikmeleri 0'dan büyük olabilirler. Basit doğrusal zaman serisi problemleri üzerinde eğitilebilirler, ancak konuşlandırılırken öğrenmeye devam etmek için adaptif olarak kullanılırlar, böylece kullanılırken girişler ve çıkışlar arasındaki ilişkide değişikliklere uyum sağlayabilirler.



LVQ Learning Vector Quantization (Vektör Nicemlemenin Öğrenilmesi)

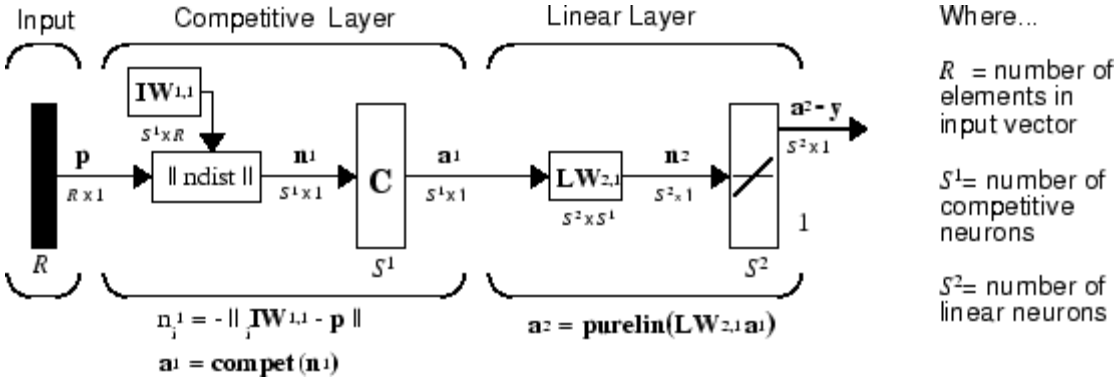
Destekli öğrenme stratejisini kullanırlar. Eğitim sırasında ağa sadece öğrenilmesi istenen girdiler verilmekte ve ağın çıktıyı kendisi üretmesi istenmektedir. Ağ çıktısını ürettikten sonra, ağa sunulan girdi vektörüne karşılık gelen ağın ürettiği çıktının doğru veya yanlış olup olmadığı söylenmektedir. LVQ ağının temel felsefesi ağa sunulan girdi vektörünü problem uzayını temsil eden referans vektörlerinden birisine haritalamaktır. 3 katmandan oluşmaktadır. Bunlar:

• Girdi katmanı

• Kohonen katmanı

• Çıktı katmanı

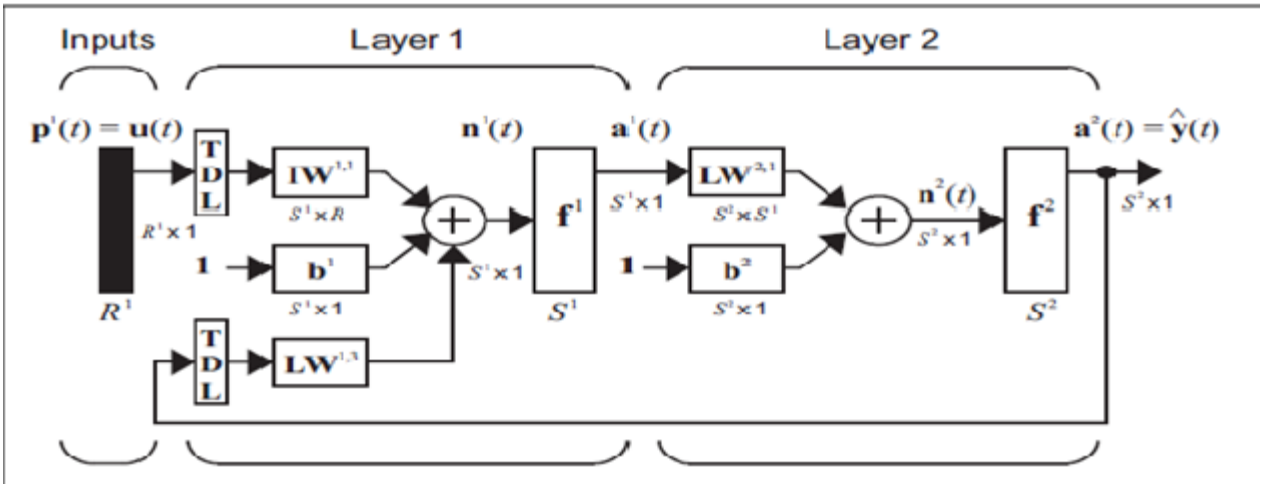
Girdi katmanındaki her eleman Kohonen katmanındaki her elemana bağlıdır. Girdi katmanından, Kohonen katmanına bağlantıların ağırlıkları bir referans vektörünü oluştur. Öğrenme sırasında sadece bu referans vektörlerinin değerleri (ağırlık değerleri) değiştirilirler. Her iterasyonda sadece tek bir vektörün değerleri değiştirilir. Öğrenmenin başarısı ise bu vektörlerin başlangıç değerleri ile yakından ilgilidir. Kohonen katmanındaki elemanların her biri çıktı katmanındaki sadece tek bir elemana bağlıdır. Kohonen katmanı ile Çıktı katmanı arasındaki ağırlıklar sabit olup değerleri 1'dir. Bu ağırlıkların değerleri eğitim sırasında değiştirilmez. LVQ ağlarında öğrenme girdi vektörü ile referans vektörleri arasındaki öklid mesafesine dayanmaktadır. Kohonen katmanındaki her eleman bir referans vektörünü göstermekte olup birbirleri ile yarılırlar. Öklid mesafesi en kısa olan eleman yarışmayı kazanır. Yarışmayı kazanan elemanın çıktısı 1 diğerleri ise 0 değerini alır. Yarışmayı kazanan çıktı elemanı ilgili gir- Papatya Yayıncılık Eğitim bilgi@papatya.gen.tr dinin sınıfını gösterir. Eğer girdi doğru sınıflandırılmış ise ilgili referans vektörü girdi vektörüne yaklaştırılır. Aksi halde ise uzaklaştırılır. Yaklaştırma ve uzaklaştırma öğrenme katsayısı ile gerçekleştirilmektedir. LVQ ağının en önemli problemi aynı vektörün çok sık kazanması ve ağın öğrenme performansının düşük olmasıdır. Bu sorun zamanla öğrenmenin ters yönde öğrenmemeye dönüşmesine neden olmaktadır. Bunu önlemek için öğrenme katsayısı zaman içinde azaltılarak sıfır değerine kadar düşürülmesi ve öğrenme tamamlanınca sıfır değerini alarak öğrenmeyi durdurması sağlanmaktadır. Bu sorunu çözmek için yeterli olmadığından, öğrenme algoritması da geliştirilmiştir. LVQ2 öğrenme yöntemi sınıflar arasında kalan örnekleri sınıflandırabilecek yeteneğe sahiptir.



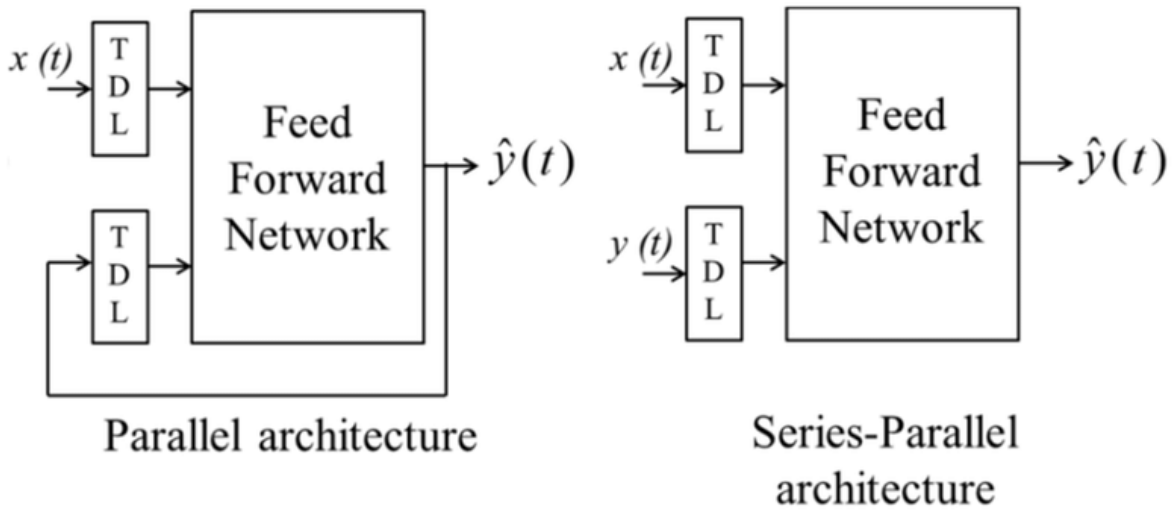
NARX A Nonlinear Autoregressive Exogenous (Doğrusal Olmayan Otoregresif Ekzojen)

Dışsal girişlere (NARX) sahip doğrusal olmayan otoregresif ağ, ağın çeşitli katmanlarını kapsayan geri besleme bağlantılarına sahip tekrarlayan bir dinamik ağdır. NARX modeli, zaman serisi modellemesinde yaygın olarak kullanılan lineer ARX modeline dayanmaktadır. NARX modelini, f işlevine yaklaşmak için ileri beslemeli bir sinir ağı kullanarak uygulayabilirsiniz. NARX ağı için birçok uygulama var. Giriş sinyalinin bir sonraki değerini tahmin etmek için bir yordayıcı olarak kullanılabilir. Hedef çıkışın giriş sinyalinin gürültüsüz bir versiyonu olduğu doğrusal olmayan filtreleme için de kullanılabilir. NARX ağının kullanımı, bir diğer önemli uygulamada, doğrusal olmayan dinamik sistemlerin modellenmesinde gösterilmiştir.

NARX ağının eğitimini göstermeden önce, eğitimde yararlı olan önemli bir konfigürasyonun açıklanması gerekir. NARX ağının çıktısını, modellemeye çalıştığınız doğrusal olmayan dinamik bir sistemin çıktısının bir tahmini olarak düşünebilirsiniz. Çıktı, aşağıdaki sol şekilde gösterildiği gibi, standart NARX mimarisinin bir parçası olarak ileri beslemeli sinir ağı girişine geri beslenir. Gerçek çıktı ağın eğitimi sırasında kullanılabilir olduğundan, seri paralel bir mimari oluşturabilirsiniz, tahmini çıktıyı geri beslemek yerine gerçek çıktı kullanılır. Bunun iki avantajı vardır. Birincisi, ileri beslemeli ağa girişin daha doğru olmasıdır. İkincisi, ortaya çıkan ağın tamamen ileri beslemeli bir mimariye sahip olması ve statik backpropagation'ın eğitim için kullanılabilmesidir.

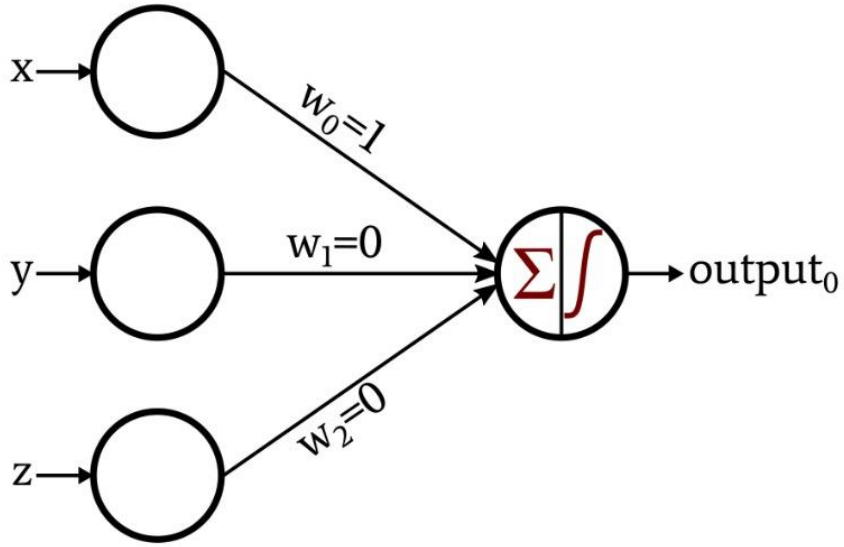


NARX A Nonlinear Autoregressive Exogenous Series – Parallel (Doğrusal Olmayan Otoregresif Ekzojen Seri Paralel)



Perceptron (Algılayıcı)

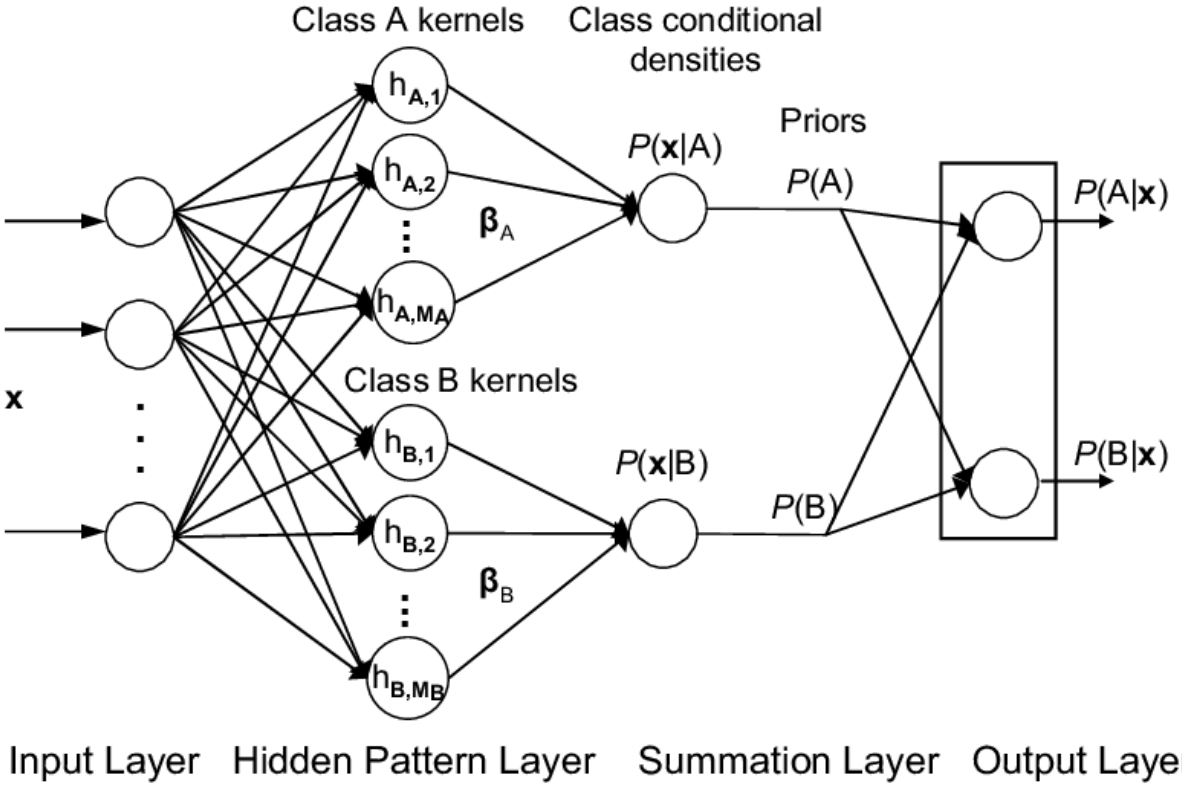
Basit tek katmanlı algılayıcılarda ağırlıklar değiştirilir iken girdilerin öğrenme katsayısı (λ) denilen bir sabit ile çarpılıp ağırlıklara eklenmesi veya çıkartılması ile gerçekleştirilir. Ağa sunulan girdilere dayanarak üretilen çıktının değerine göre ağırlıklar artırılır veya azaltılır



Probabilistic (Olasılık)

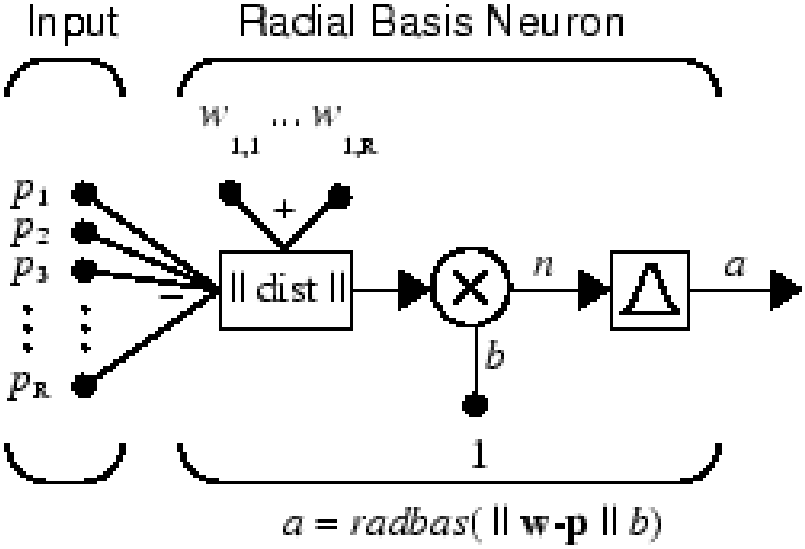
Yapay sinir ağları yaygın sınıflandırılması ve tanıma probleminde kullanılır. PNN algoritmasında, her sınıfın üst olasılık dağıtım işlevi (PDF), bir Parzen penceresi ve parametrik olmayan bir işlevle yaklaşık olarak tahmin edilir . Daha sonra, her sınıfın PDF'sini kullanarak, yeni bir giriş verisinin sınıf olasılığı tahmin edilir ve Bayes kuralı, en yüksek posterior olasılıklı sınıfı yeni giriş verilerine tahsis etmek için kullanılır. Bu yöntemle, yanlış sınıflandırma olasılığı en aza indirilir. Operasyonlar dört katmanlı çok katmanlı bir ileri beslemeli ağ halinde düzenlenir:

- Giriş katmanı
- Desen katmanı
- Toplama katmanı
- Çıktı katmanı



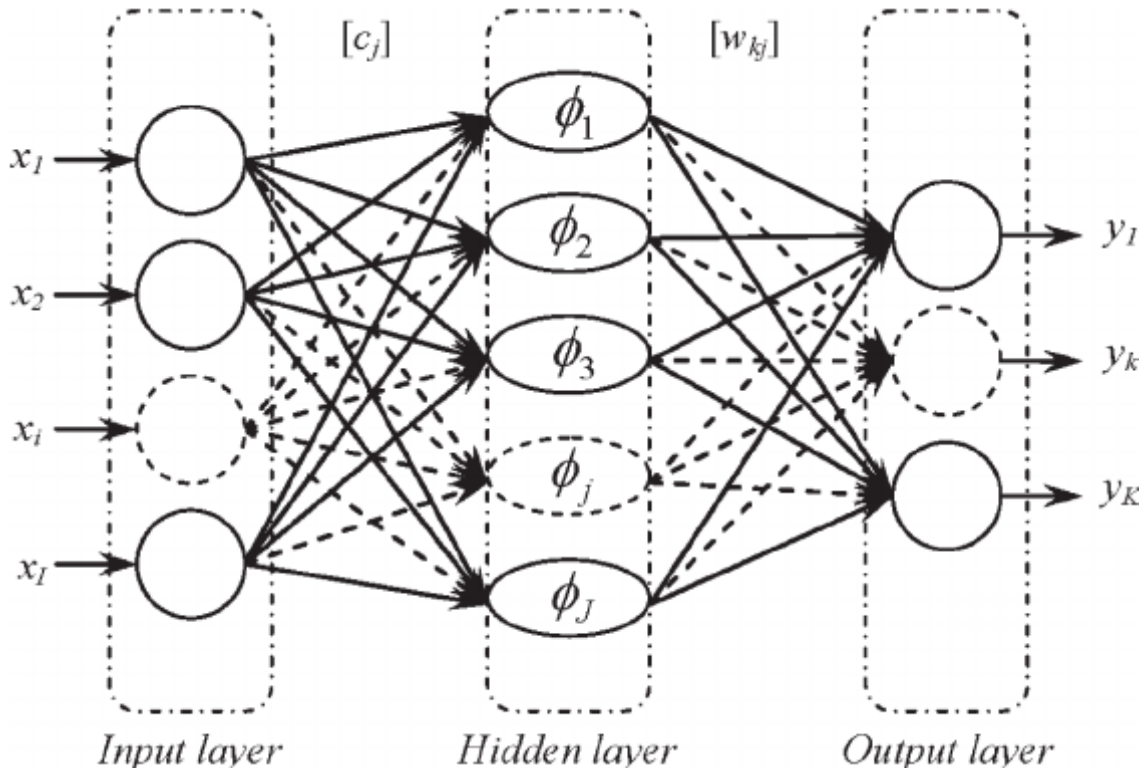
Radial Basis Exact Fit (Radyal Bazlı (Tabanlı) Tam Uyum)

Radyal esaslı fonksiyonların toplamı tipik olarak verilen fonksiyonların yaklaşık değerlerini belirlemek için kullanılır. Bu yaklaşım süreci aynı zamanda basit bir sinir ağı türü olarak da yorumlanabilir. Gerçek değerli bir fonksiyondur. Değeri yalnızca giriş ve başlangıç noktası gibi sabit bir nokta arasındaki mesafeye bağlıdır.



Radial Basis Fewer Neurons (Radyal (Tabanlı) Bazlı Daha Az Nöron)

İşlenmiş Peynir, peynir çeşitleri arasında en popüler çeşitlerden biridir. Radyal Baz (Daha Az Nöron) ve Çoklu Doğrusal Regresyon modelleri, vücut ve doku, aroma ve lezzet, nem, serbest yağ asitleri giriş parametreleri ve duyusal puan alınarak 7-8º C'de saklanan işlenmiş peynirlerin raf ömrünü tahmin etmek için geliştirilmiştir. çıkış parametresi. Geliştirilen modellerin tahmin kapasitesini hesaplamak için ortalama kare hatası, kök ortalama kare hatası, belirleme katsayısı ve Nash-Sutcliffe katsayısı kullanılmıştır. Geliştirilen iki modelin karşılaştırılması, radyal baz (daha az nöron) yapay sinir ağı modelinin performansının, işlenmiş peynirlerin raf ömrünü tahmin etmek için istatistiksel çoklu lineer regresyon modelinden daha iyi olduğunu ortaya koydu.



Self-Organizing Map (Kendi Kendini Düzenleyen Harita)

Bir öz organize haritası (SOM) ya da kendi kendini organize özelliği, harita (SOFM) türüdür yapay sinir ağına kullanılarak eğitilmiştir (YSA) denetimsiz öğrenme düşük boyutlu (genellikle iki boyutlu), discretized temsilini üretmek için eğitim örneklerinin girdi alanı, harita olarak adlandırılır ve bu nedenle boyutsal küçültme yapmak için bir yöntemdir. Bunlar geçerli olarak Özörgütleme haritalar diğer yapay sinir ağları farklı rekabetçi öğrenme hata düzeltme öğrenme aksine (örneğin geri yayılım ile degrade iniş) ve giriş alanının topolojik özelliklerini korumak için bir mahalle işlevi kullandıkları anlamındadır.

ABD Kongresi oylama modellerini gösteren kendi kendini düzenleyen bir harita. Girdi verileri, her Kongre üyesi için bir satır içeren bir tablo ve her üyenin evet / hayır / çekimser oyunu içeren belirli oylar için sütunlardı. SOM algoritması, bu üyeleri benzer üyeleri birbirine yakın yerleştirerek iki boyutlu bir ızgarada düzenledi. İlk grafik, veriler iki kümeye ayrıldığında gruplamayı gösterir. İkinci grafik komşulara ortalama mesafeyi göstermektedir: daha büyük mesafeler daha karanlıktır. Üçüncü arsa Cumhuriyetçi (kırmızı) veya Demokrat (mavi) parti üyeliğini öngörüyor. Diğer parsellerher biri sonuçta ortaya çıkan haritayı girdi boyutunda öngörülen değerlerle kaplar: kırmızı, bu faturada tahmini 'evet' oyu, mavi ise 'hayır' oyu anlamına gelir. Çizim Synapse'de oluşturuldu. Bu, SOM'ları çok boyutlu ölçeklemeye benzer, yüksek boyutlu verilerin düşük boyutlu görünümlerini oluşturarak görselleştirme için kullanışlı hale getirir.