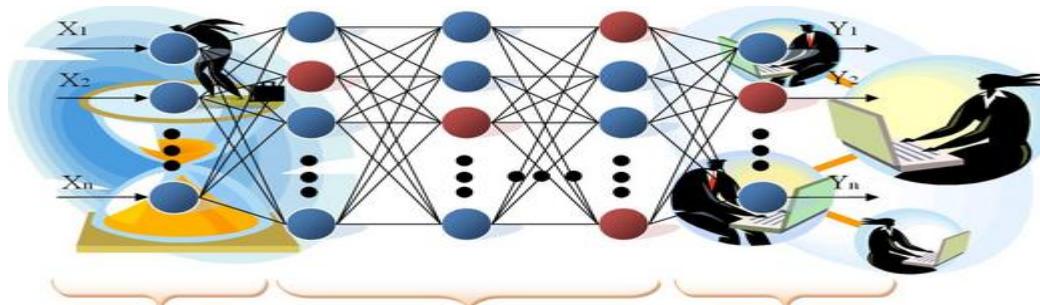




Mekatronik Mühendisliği Uygulamalarında Yapay Zekâ

Ders 3- Yapay Sinir Ağları

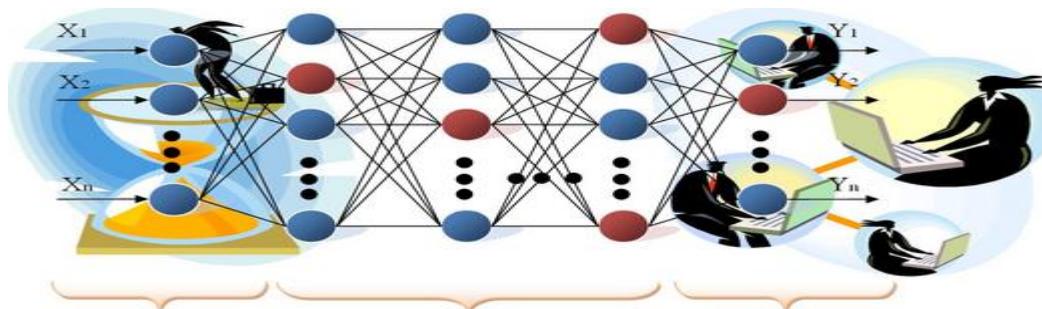
Erhan AKDOĞAN, Ph.D.





Ders 3-1

Yapay Sinir Ağlarına Giriş



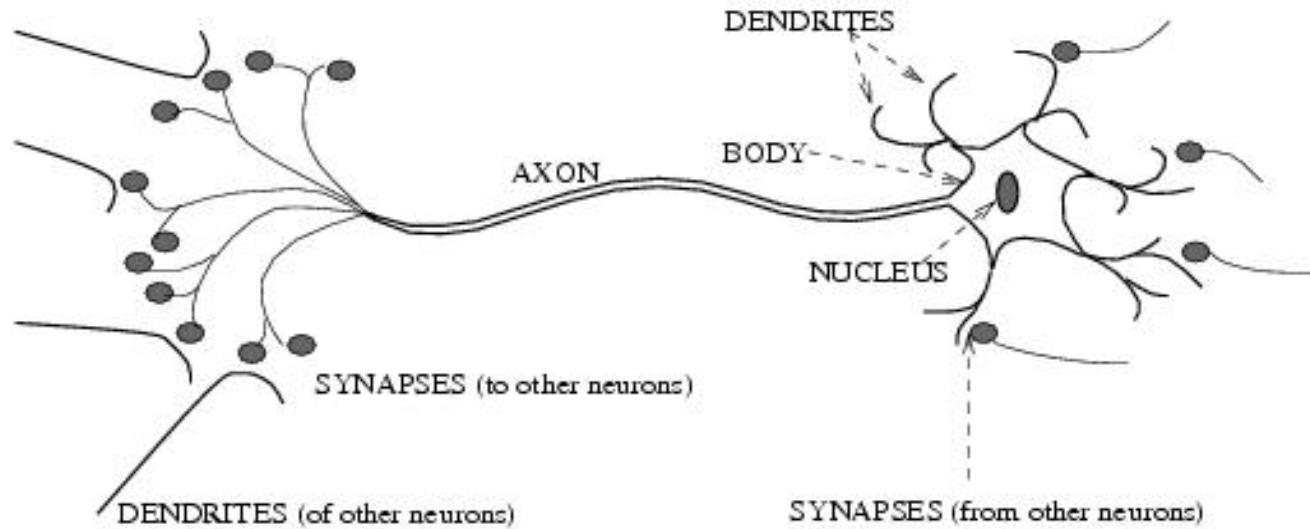
Giriş

- YSA, insan beyninin özelliklerinden olan **öğrenme** yolu ile
 - yeni bilgiler türetebilme,
 - yeni bilgiler oluşturabilme ve
 - keşfedebilme

gibi yetenekleri herhangi bir yardım almadan otomatik olarak gerçekleştirmek amacı ile geliştirilen bilgisayar yazılımlarıdır.



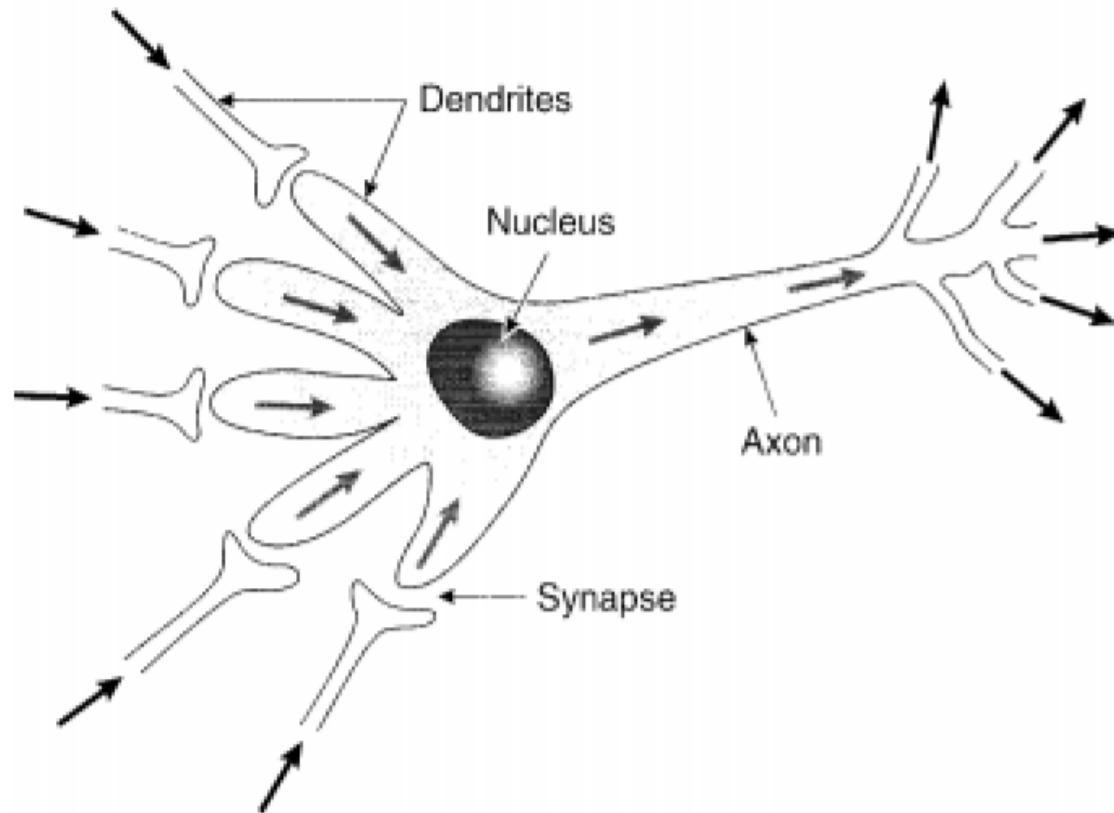
- YSA, insanlar tarafından gerçekleştirilmiş örnekleri kullanarak olayları **öğrenebilen**, çevreden gelen oylara karşı nasıl tepkiler üretileceğini **belirleyebilen** bilgisayar yazılımlarıdır.



- Biyolojik bir sinir hücresi; bir gövde, bir akson, çok sayıda sinir ucu (dendrit) ve akson ile diğer sinir hücresinin sinir ucu arasında kalan ince uzantılar (sinaps) olmak üzere dört bölümden oluşmaktadır.
- Dendritler, gelen sinyalleri çekirdeğe ileter. Çekirdek dendritten gelen sinyalleri bir araya toplar ve aksona ileter. Toplanan bu sinyaller, akson tarafından işlenerek sinapslara gönderilir. Sinapslar da yeni üretilen sinyalleri diğer sinir hücrelerine ileter.



Biyolojik nöron yapısı

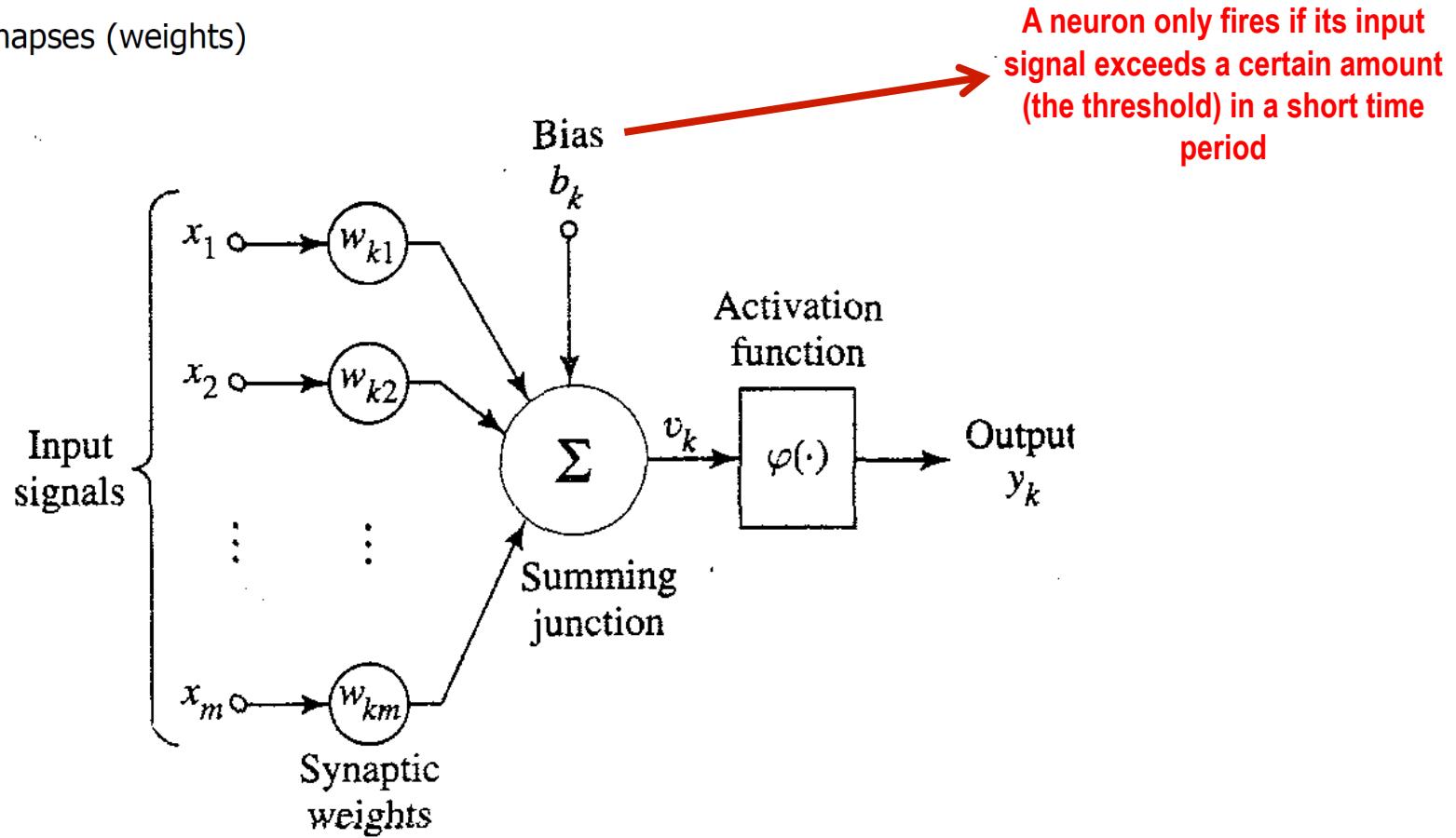


Ref: http://www.iitmandi.ac.in/ciare/files/7_Anand_ANN.pdf



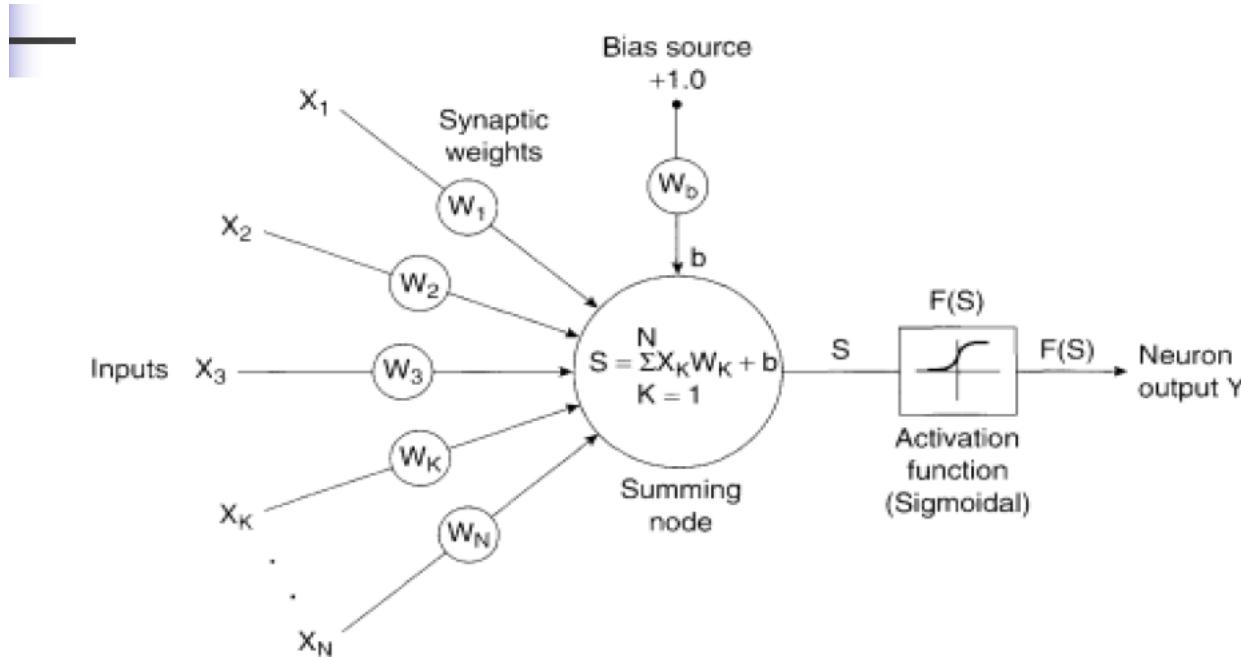
Yapay nöron

1. Neurones (nodes)
2. Synapses (weights)

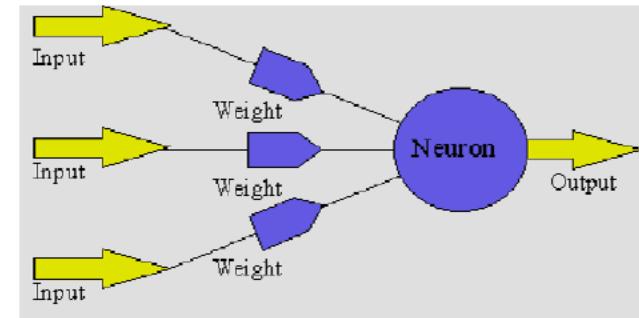
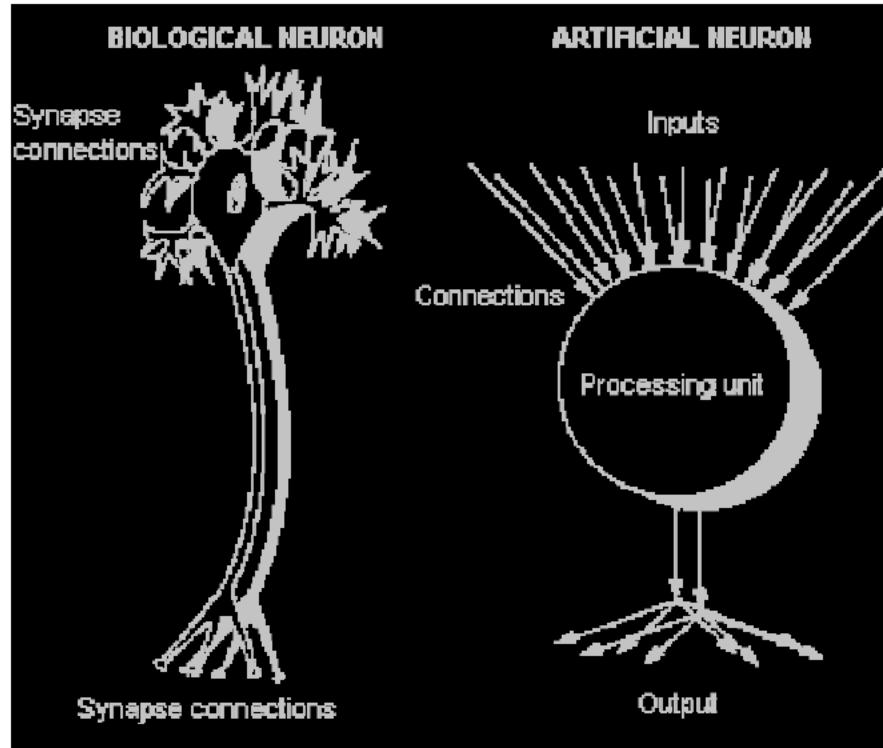




Yapay Nöron



Ref: http://www.iitmandi.ac.in/ciare/files/7_Anand_ANN.pdf

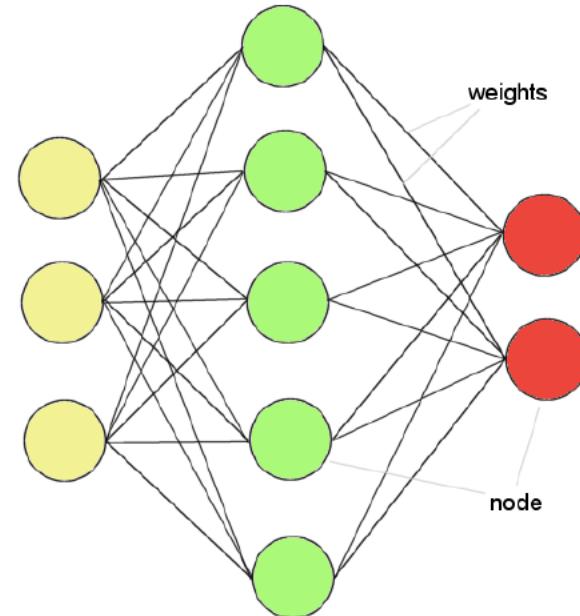


Ref: http://www.iitmandi.ac.in/ciare/files/7_Anand_ANN.pdf



Temel YSA yapısı

1. Neurones (nodes)
2. Synapses (weights)



Tarihsel gelişim süreci:

- **Pre-1940: von Hemholtz, Mach, Pavlov, etc.**
 - General theories of learning, vision, conditioning
 - No specific mathematical models of neuron operation
- **1940s: Hebb, McCulloch and Pitts**
 - Mechanism for learning in biological neurons
 - Neural-like networks can compute any arithmetic function
- **1950s: Rosenblatt, Widrow and Hoff**
 - First practical networks and learning rules
- **1960s: Minsky and Papert**
 - Demonstrated limitations of existing neural networks, new learning algorithms are not forthcoming, some research suspended
- **1970s: Amari, Anderson, Fukushima, Grossberg, Kohonen**
 - Progress continues, although at a slower pace
- **1980s: Grossberg, Hopfield, Kohonen, Rumelhart, etc.**
 - Important new developments cause a resurgence in the field

YSA'nın kabiliyetleri:

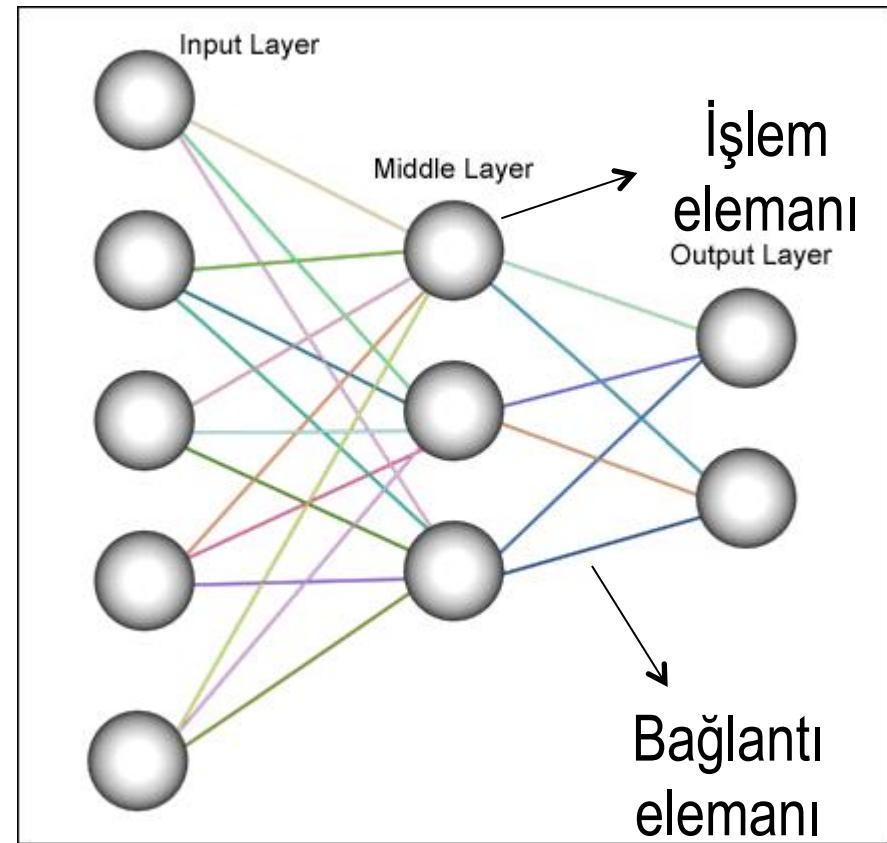
- Öğrenme
- İlişkilendirme
- Sınıflandırma
- Genelleme
- Özellik belirleme
- Optimizasyon



- YSA, örneklerden elde ettiği bilgiler ile kendi deneyimlerini oluşturur ve daha sonra benzer konularda benzer kararlar verir.
- YSA, olaylar hakkında bilgilerin olmadığı fakat örneklerin bulunduğu durumlarda çok etkin olarak kullanılabilir.



- YSA birbirine bağlı ve paralel çalışabilen yapay hücrelerden meydana gelir
- Bu hücrelere **İşlem elemanı (=nöron)** adı da verilir.





- Her hücrenin birbirine olan bağlantılarının bir değere sahip olduğu kabul edilir.
- Bilginin öğrenme yolu ile elde edildiği ve bu bilgilerin bağlantılarında saklandığı kabul edilir.

YSA'nın temel işlevi:

- Kendisine gösterilen bir girdi setine karşılık gelebilecek bir çıktı seti belirlemektir.
- Bunu yapabilmek için ağ, ilgili olayın örnekleri ile eğitilerek (öğrenme) genelleme yapabilecek yeteneğe kavuşturulur.
- Bu genelleme ile benzer olaylara karşılık gelen çıktı setleri belirlenir.

YSA'ya verilen diğer adlar

- Bağlantılı ağlar (connectionist networks)
- Paralel dağıtılmış ağlar (parallel distributed networks)
- Nöromorfik sistemler (neuromorphic systems)

Yapay Sinir Ağlarının Genel Özellikleri

- Makine öğrenmesi gerçekleştirirler.
- Bilgi veri tabanında değil ağ üzerinde saklanır.
- Örnekleri kullanarak öğrenirler. Örnekler gerçekleşmiş olaylardır. Elde edilen örneklerin olayı tamamı ile gösterebilmesi önemlidir.
- YSAların güvenle kullanılabilmesi için önce eğitilmeleri ve test edilmeleri gereklidir.
- YSAların çıkışlarının kontrol altına alınması için ek sınırlar koymak gerekebilir.



- Ağın eğitilmesinden sonra ağın hiç görmediği örnekler sisteme verilir ve cevaplara bakılır. Eğer ağ hiç görmediği örneklerle doğru cevaplar veriyor ise performansı kabul edilir.
- Gerekirse çevrime alınarak (on-line) da çalıştırılabilir.
- Gerekirse ağ yeni örneklerle tekrar eğitilebilir.



- Algılamaya yönelik olaylarda kullanılabilir.
- Şekil (örüntü) ilişkilendirme ve sınıflandırma yapabilirler.
- Eksik bilgi tamamlayabilirler.
- Kendi kendine organize etme ve öğrenebilme özellikleri vardır (adaptasyon)
- Eksik bilgi ile çalışabilirler. Ağın durumuna göre eksik bilginin ağı için ne kadar önemli olduğu tespit edilebilir.
- Bu durum hataya karşı toleranslarını yükseltir.



- Belirsizlikleri işleyebilirler.
- Dereceli olarak bozulabilirler.

YSA ların dezavantajları:

- Paralel çalışabilen donanımlara ihtiyaç duyar.
- Bir ağın nasıl oluşturulması gerektiğini belirleyen kurallar yoktur. Uygun ağ deneme yanılma yolu ile bulunur.
- YSA lar kabul edilebilir çözümler üretebilir. Ama bunun optimum çözüm olduğu test edilmeden iddia edilemez.

YSA ların dezavantajları

- Ağ parametrelerinin seçimine ilişkin de herhangi bir kural yoktur. Her problem için ayrı ayrı değerlendirilmesi gereklidir.
- Ağın öğreneceği problemin ağa gösterilmesi önemli bir problemdir. YSA sadece nümerik değerler ile çalışır. Bu nedenle problemin nümerik değerlere dönüşümü gereklidir.

YSA ların dezavantajları

- Ağın ne kadar eğitilmesi gerekiğine ilişkin bir metod yoktur. Ağın örnekler üzerindeki hatasının belirli bir değerin altına indirilmesi eğitimin tamamlanması için yeterli görülmektedir.
- Ağın davranışının açıklanması önemli bir problemdir. Bir probleme çözüm üretildiğinde bunun nasıl ve neden üretildiği konusunda bir bilgi bulmak mümkün değildir.



Bir problemin YSA ile çözülmesi için şu şartlardan birini taşıması gereklidir

- Sadece YSA ile probleme pratik çözümler üretme durumu söz konusu olmalıdır,
- Başka çözüm yolları olmasına rağmen YSA ile daha kolay, etkin ve hızlı çözümler üretilebilmeli.
 - Örnek: Matematiksel olarak çözülüyor ise YSA ile olan karşılaştırması yapılmalıdır

YSA'nın kullanılabileceği en etkin problemler

- Doğrusal olmayan
- Çok boyutlu
- Gürültülü
- Karmaşık
- Kesin olmayan
- Eksik
- Kusurlu
- Matematiksel modeli olmayan
- Algoritması bulunmayan

YSA'nın konuları:

- Olasılıksal fonksiyon kestirimleri
- Sınıflandırma
- İlişkilendirme veya şekil eşleştirme
- Zaman serileri analizi
- Sinyal filtreleme
- Veri sıkıştırma
- Örüntü tanıma

YSA'nın konuları:

- Doğrusal olmayan sinyal işleme
- Doğrusal olmayan sistem modelleme
- Optimizasyon
- Zeki ve doğrusal olmayan kontrol

Uygulamalar

- Veri madenciliği
- Optik karakter tanıma
- Ürünün Pazar performansı tahmini
- Kredi kartları hileleri
- Robotlarda ve zeki araçlarda optimum yörüğe belirleme
- Konuşma, parmak izi ve iris tanıma

Uygulamalar

- Kalite kontrol
- İletişimde geçersiz eko filtreleme
- Haberleşmede trafik yoğunluğunun kontrol etme ve anahtarlama
- Radar ve sonar sinyal sınıflandırma
- Hastalık teşhisi (MR, ultrason)
- Beyin modellenmesi

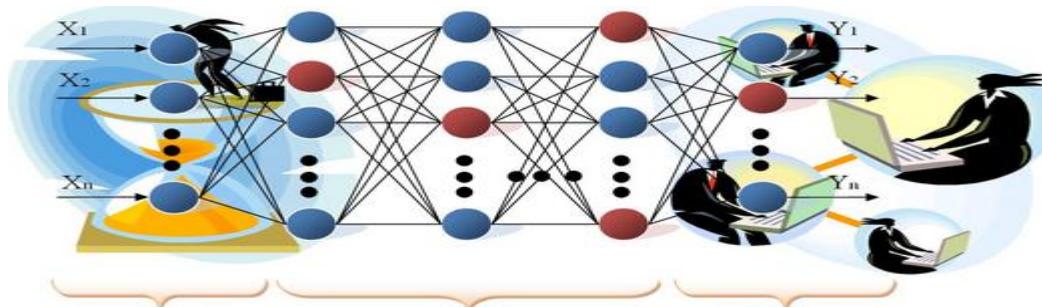
Referanslar

- E.Öztemel, Yapay Sinir Ağları, Papatya Yayıncılık, 2003
- [http://bm.bilecik.edu.tr/Dosya/Arsiv/odevnot/
yapay_sinir_aglari.pdf](http://bm.bilecik.edu.tr/Dosya/Arsiv/odevnot/yapay_sinir_aglari.pdf)



Ders 3-2

Yapay Sinir Ağları Tasarımı



Yapay Sinir Ağı Tasarımında Başarı Ölçütleri:

- Uygun yapı seçimi
- Bu yapıya uygun öğrenme algoritmasını belirleme
- Uygun algoritma parametreleri seçimi
- Seçilen yapıya uygun giriş, arakatman ve çıkış sayısı belirleme
- Eğitim ve test setlerini belirleme

YSA sistem karmaşıklığı

Sistem karmaşıklığı = f (toplam hesaplama karmaşıklığı)

Toplam hesaplama karmaşıklığı = f (yapısal karmaşıklık)



Toplam hesaplama karmaşıklığı parametreleri

- YSA tepki süresi
- Hafıza



YSA performansı ve genelleme kabiliyetine etki eden faktörler

- Ağ büyüklüğü
- Öğrenme seviyesi
- Ağırlıkların sınırlandırılması



YSA parametrelerinin seçiminde karşılaşılan güçlükler

- Probleme uygun mimari seçimi
- Problemin çözümü için YSA giriş ve çıkış sayılarının en uygun sayıda seçimi
- Arakatman nöron sayılarının en uygun sayıda belirlenmesi
- Arakatman sayısının seçimi
- Kullanılacak öğrenme algoritmasının YSA yapısına uygun olması



- Seçilen veri kodlama yapısı
- Veri normalizasyon yaklaşımı
- Seçilen transfer fonksiyonunun yapısı
- Uygun performans fonksiyonu seçimi
- Uygun iterasyon sayısı seçimi

YSA'nın Mühendislik Uygulamaları:

- **Tahmin:** Hastalık riski
- **Sınıflandırma:** Arıza sınıflandırma
- **Veri ilişkilendirme:** Taranan bir dokümandaki karakterleri algılama
- **Veri yorumlama:** Bir veri tabanında birbirine benzer verileri gruplandırma
- **Veri filtreleme:** Gürültü ayıklama

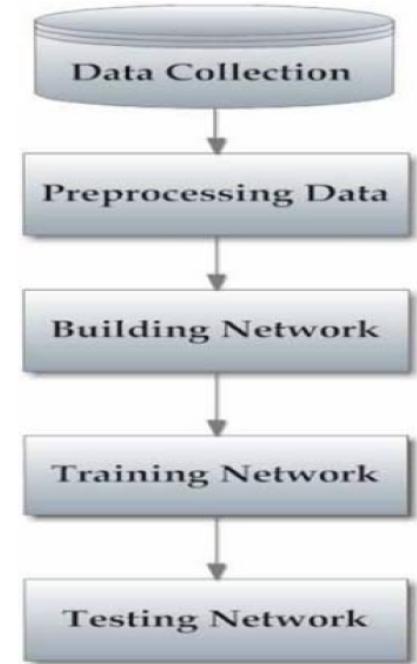
YSA Tasarım Adımları:

- Collect data
- Create the network
- Configure the network
- Initialize the weights and biases
- Train the network
- Validate the network
- Use the network

Ref: <http://www.mathworks.com/help/nnet/gs/neural-network-design-steps.html>

YSA Tasarım Adımları:

- (1) collecting data,
- (2) preprocessing data
- (3) building the network
- (4) train, and
- (5) test performance of model



Ref: http://www.iitmandi.ac.in/ciare/files/7_Anand_ANN.pdf

YSA'da giriş-çıkış-aktivasyon fonk.

- Ağın giriş sayısı=problemin giriş sayısı
- Ağın çıkış sayısı=problemin çıkış sayısı
- Aktivasyon fonksiyonu probleme özel belirlenir.

YSA yapısı seçimi

- Multi layer perceptron
- LVQ (Learning vector quantization) yapısındaki ağlar
- RBFNN-Radial Basis ağlar
- Re-current (yinelemeli) ağlar
-

- Katman Sayısı
- Nöron sayısı

Ağ Yapısı Seçimi

| Kullanım amacı | Ağ türü | Ağın kullanımı |
|---------------------|---|--|
| Tahmin | MLP | Ağın girdilerine karşılık bir çıktı değerinin tahmin edilmesi |
| Sınıflandırma | <ul style="list-style-type: none">•LVQ•ART•Counterpropagation•Olasılıksal sinir ağı | Girdilerin hangi sınıf'a ait olduklarının belirlenmesi |
| Veri ilişkilendirme | <ul style="list-style-type: none">•Hopfield•Boltzman machine•Bidirectional associative memory | Girdilerin içindeki hatalı bilgilerin bulunması ve eksik bilgilerin tamamlanması |

Öğrenme Algoritmaları Seçimi

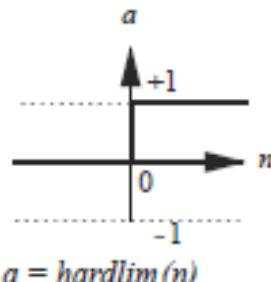
- Problemi öğrenme başarısı, kullanılabilirliğini belirleyen **genelleme başarısı** ile belirlenir.
- Genelleme başarısı, **gerçekleştirilen testlerle sınanmalıdır**.
- En uygun öğrenme seviyesi, eğitim süreci boyunca **performans fonksiyonun** izlenmesi ve sık sık genelleme testleri yapılması ile belirlenebilir.

Aktivasyon fonksiyonu seçimi:

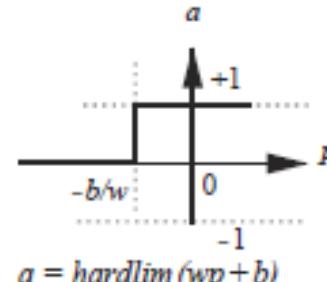
- Problem davranışına uygun bir transfer fonksiyonu seçilmelidir.
- Bazı durumlarda yeni bir transfer fonksiyonu üretilebilir.



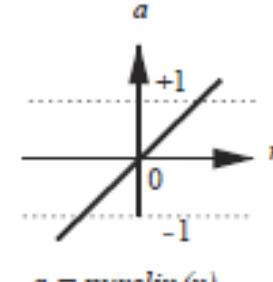
Aktivasyon Fonksiyonları



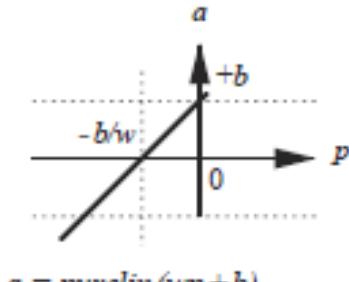
$a = \text{hardlim}(n)$
Hard Limit Transfer Function



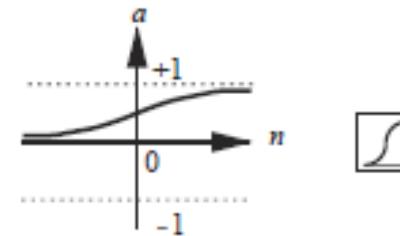
$a = \text{hardlim}(wp + b)$
Single-Input hardlim Neuron



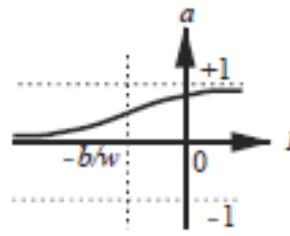
$a = \text{purelin}(n)$
Linear Transfer Function



$a = \text{purelin}(wp + b)$
Single-Input purelin Neuron



$a = \text{logsig}(n)$
Log-Sigmoid Transfer Function



$a = \text{logsig}(wp + b)$
Single-Input logsig Neuron

| Name | Input/Output Relation | Icon | MATLAB Function |
|-----------------------------|---|------|-----------------|
| Hard Limit | $a = 0 \quad n < 0$ $a = 1 \quad n \geq 0$ | | hardlim |
| Symmetrical Hard Limit | $a = -1 \quad n < 0$ $a = +1 \quad n \geq 0$ | | hardlims |
| Linear | $a = n$ | | purelin |
| Saturating Linear | $a = 0 \quad n < 0$ $a = n \quad 0 \leq n \leq 1$ $a = 1 \quad n > 1$ | | satlin |
| Symmetric Saturating Linear | $a = -1 \quad n < -1$ $a = n \quad -1 \leq n \leq 1$ $a = 1 \quad n > 1$ | | satlins |
| Log-Sigmoid | $a = \frac{1}{1 + e^{-n}}$ | | logsig |
| Hyperbolic Tangent Sigmoid | $a = \frac{e^n - e^{-n}}{e^n + e^{-n}}$ | | tansig |
| Positive Linear | $a = 0 \quad n < 0$ $a = n \quad 0 \leq n$ | | poslin |
| Competitive | $a = 1 \quad \text{neuron with max } n$ $a = 0 \quad \text{all other neurons}$ | | compet |

Ön ve son veri işleme

Ön veri işleme:

- Dinamik yapıya sahip YSA uygulamalarında önemlidir.
- Verinin en doğru , en kısa ve en hızlı şekilde işlenmesi için hazırlanmasını ifade eder. Çeşitli veri geliştirme aşamalarında yararlanılabilir (dönüştürücüler, kodlama algoritmaları gibi)

Ön ve son veri işleme

Son veri işleme:

- YSA çıkışının ne anlama geldiğinin yorumlandığı ve bu yorumun bilgiye veya başlangıç aşamasındaki veri tipine dönüştürüldüğü bir süreçtir.

Normalizasyon

- Ağ çıkışının doğru şekilde yorumlanabilmesi için normalizasyon işlemine ihtiyaç duyulur.
- Bipolar çıkış fonksiyonlarında $[-1, +1]$, aksi durumlarda $[0, 1]$ aralığında normalizasyon yapılır.



X veri kümесинin [a,b] aralığıda normalizasyonу için

$$X' = \frac{(X - X_{\min})}{X_{\max} - X_{\min}} (b - a) + a$$



Arakatman(hidden layer) sayısı belirleme

- Genel olarak en fazla iki arakatmanla problemin çözümü önerilir.
- Derin öğrenmede bu sayının arttığı gözlenmektedir.

Veri kodlama

- Dilsel verilerin kodlanması
- Sayısal verilerin kodlanması

Dilsel veri kodlama

- Bu tür için üyelik fonksiyonları olarak adlandırılan bir sosyal kavramın alt ve üst kabul edilebilirlik sınırlarının tanımlanabilmesini sağlayan bulanık mantık araçlarından yararlanılır.
- Nöro-fuzzy teknikler....

Sayısal veri kodlama

- Kayıplı ve kayıpsız veri kodlama teknikleri kullanılır:
- Kayıpsızda, asıl veri kodlanmış olarak tekrar elde edilebilir.(bir girişin 8 bitle kodlanması gibi)
- Kayıplıda, kabul edilebilir bir hata ile veri tekrar elde edilir.



Yaygın kullanılan kayıplı veri kodlama araçları:

- Ayrık kosinüs dönüşümü (DCT)
- Hızlı fourier dönüşümü (FFT)
- Dalgacık dönüşümü (wavelet)
- Harr dönüşümü
- Sinüs dönüşümü

Performans fonksiyonları:

- Karesel ortalama hata MSE
- Toplam karesel hata SSE
- Karesel ortalama hata karekökü RMS

YSA Yapıları

- Adaline
- Çok katlı perseptron
- Radyal tabanlı ağlar
- LVQ
- Hopfield
- Elman ve Jordan
- Kohonen



- ART (adaptive resonance theory) ağı
- Olasılık tabanlı ağlar
- Genel regresyon
- Çoklu ve modüler YSAlar
- Mantıksal YSAlar
- GMDH(Group method of data handling)

YSA Öğrenme kuralları

- Hebb
- Hopfield
- Delta
- Kohonen

Öğrenme alg.nın sınıflandırılması

- Danışmanlı (supervised)
- Danışmansız (unsupervised)
- Destekli (reinforcement)

Uygulamaya göre sınıflandırma

- Off-line
- On-line

Diğer yaklaşımlar:

- Desen tabanlı eğitim
- Grup eğitimi
- Artırımlı öğrenme
- Yapısal öğrenme
- Ardışıl öğrenme

Öğrenme algoritmaları

- Backpropagation
- Esnek yayılım
- Delta-bar-delta
- Hızlı yayılım
- Genetik alg.
- Yönlendirilmiş rasgele arama



- Levenberg-Marquart
- Eşleştirmeli eğim
- Kuasi-Newton
- Tek adım sekant

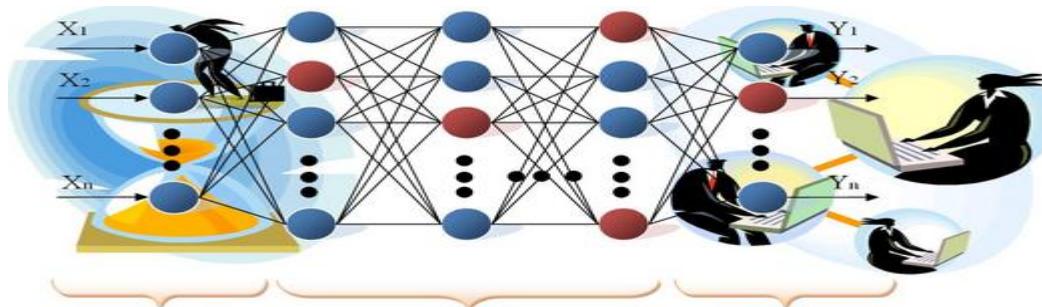
Referanslar

- Mühendislikte yapay zeka uygulamaları, Ş.Sağiroğlu, E.Beşdok, M.Erlер, Ufuk Yayınevi, 2003.
- Neural Network Design, M. Hagan, H. Demuth, M. Beale, PWS Publishing Comp., 2002.



Ders 3-3

Yapay Sinir Ağ Yapıları



YSA Donanımları

- Analog (çarpıcı-toplayıcı-transistör)
- Sayısal
- Melez (Hibrid) sistemler

Donanım özellikleri

- Yüksek seviyeli gürültüden etkilenmezler
- Yüksek işlem hızları
- Yüksek işlem hassasiyeti
- Mevcut sistemler ile tasarlanabilir
- Programlanabilir bileşenler içerir

- Isıl kararlılık düşük
- Gürültülü
- Analog bağlantı problemleri
- Sınırlı doğruluk
- Test problemleri

Ağ Yapısı Seçimi

| Kullanım amacı | Ağ türü | Ağın kullanımı |
|---------------------|---|--|
| Tahmin | MLP | Ağın girdilerine karşılık bir çıktı değerinin tahmin edilmesi |
| Sınıflandırma | <ul style="list-style-type: none">•LVQ•ART•Counterpropagation•Olasılıksal sinir ağı | Girdilerin hangi sınıf'a ait olduklarının belirlenmesi |
| Veri ilişkilendirme | <ul style="list-style-type: none">•Hopfield•Boltzman machine•Bidirectional associative memory | Girdilerin içindeki hatalı bilgilerin bulunması ve eksik bilgilerin tamamlanması |

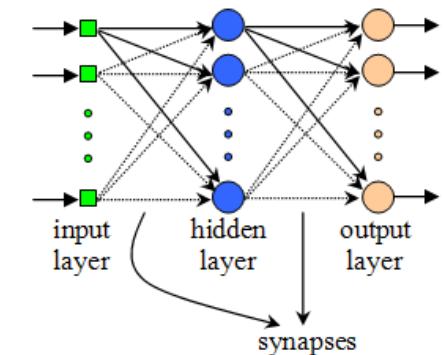
1. ADALINE Adaptif lineer eleman

- Her iterasyonda ortalama karesel hatayı (Mean Square Error) azaltarak ağırlıkları ayarlayan ve sınıflandırmayı sağlayan basit bir perseptrondur.
- Danışmanlı öğrenme kullanır.
- Lineer çalışma uzayı ile sınırlıdır.
- Lineer transfer fonksiyonu kullanırlar.
- Özellikle ses sinyalleri üzerindeki gürültü giderimi için kullanılmaktadır.



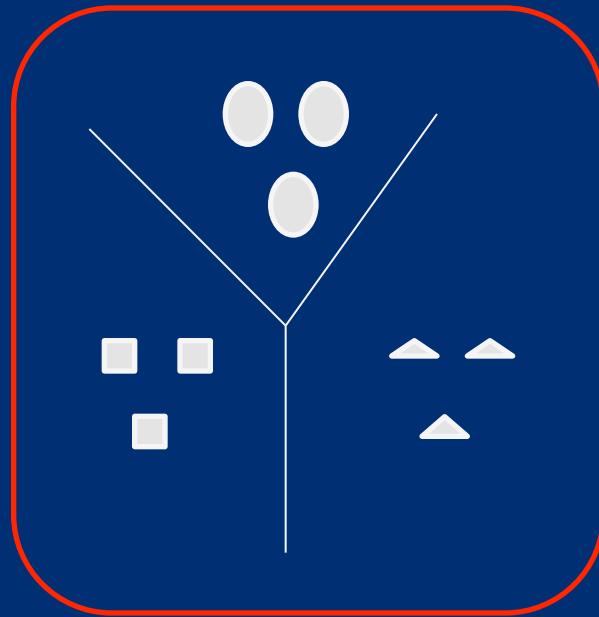
2. Çok katmanlı perceptron (MLP)

- İleri beslemeli sinir ağı modeli olarakta anılır.
- Geri besleme yoktur
- En çok kullanılan yapıdır
- Birçok öğretme algoritması bu yapıda kullanılabilir.
- En genel anlamda giriş uzayı ile çıkış uzayı arasında statik haritalama yaparlar.

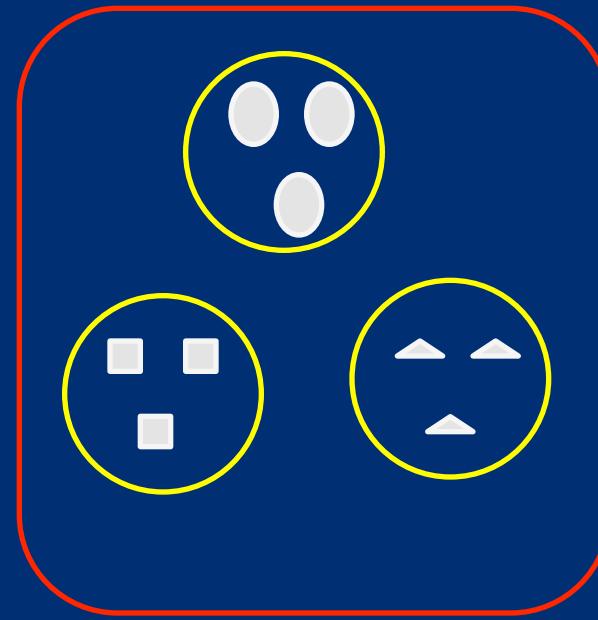


3. Radyal tabanlı ağlar

- Çok değişkenli modelleme ve yakınsamalarda kullanılır.
- Ara katmandaki işlemci elemanlar girişlerin ağırlıklandırılmış şeklini kullanmamakta ve ara katmandaki işlemci elemanların çıkışları YSA girişleri ile temel fonksiyonun merkezi arasındaki uzaklığa göre belirlenmektedir.
- En genel anlamda, radyal olarak simetrik olan ara katman işlemci elemanları içeren bir yapıdır.



MLP



RBF

RBF'İN MLP'ye ÜSTÜNLÜKLERİ

- Daha hızlı öğrenirler
- Karar sınırları ve sınıflandırmada daha iyi sonuç verir
- Arakatman elemanları ile giriş uzayının bir fonksiyon ile temsil edilebilemişi

Zayıflıkları

- Eğitme aşamasında, merkez vektörünün belirlenmesi için kontrollsüz öğrenme yönteminin kullanılması durumunda önemli ayırtırma bilgilerinin kaybolması
- Regresyon problemlerinin çözümünde sınırlandırılmamış aktivasyon fonksiyonlarına ihtiyaç duymaları
- MLP gibi farklı yapısal formda olamamaları

4. LVQ (Learning Vector Quantisation) Kohonen Ağları

- Arakat ile çıkış katı arasındaki ağırlıklar 1 değerine sabitlenir.
- Giriş ile arakat arasındaki nöron bağlantı ağırlıkları referans vektör olarak isimlendirilir.
- Her bir nöron için bir referans vektörü tanımlanır. Bu ağırlıklar ağın eğitilmesi sırasında değiştirilir.



- Diğer bütün nöronların çıkışlarının “0” olması için bütün nöronlara baskı uygulanır.
- Çıkış nöronuna bağlı olarak, kazanan nöronu içeren arakat nöron grubuna “1” ve diğer bütün çıkış nöronları “0” değerleri atanır. “1” değerini üreten çıkış nöronu, her farklı sınıfı ifade eden her bir giriş nöronu giriş vektörünün sınıfını verir.



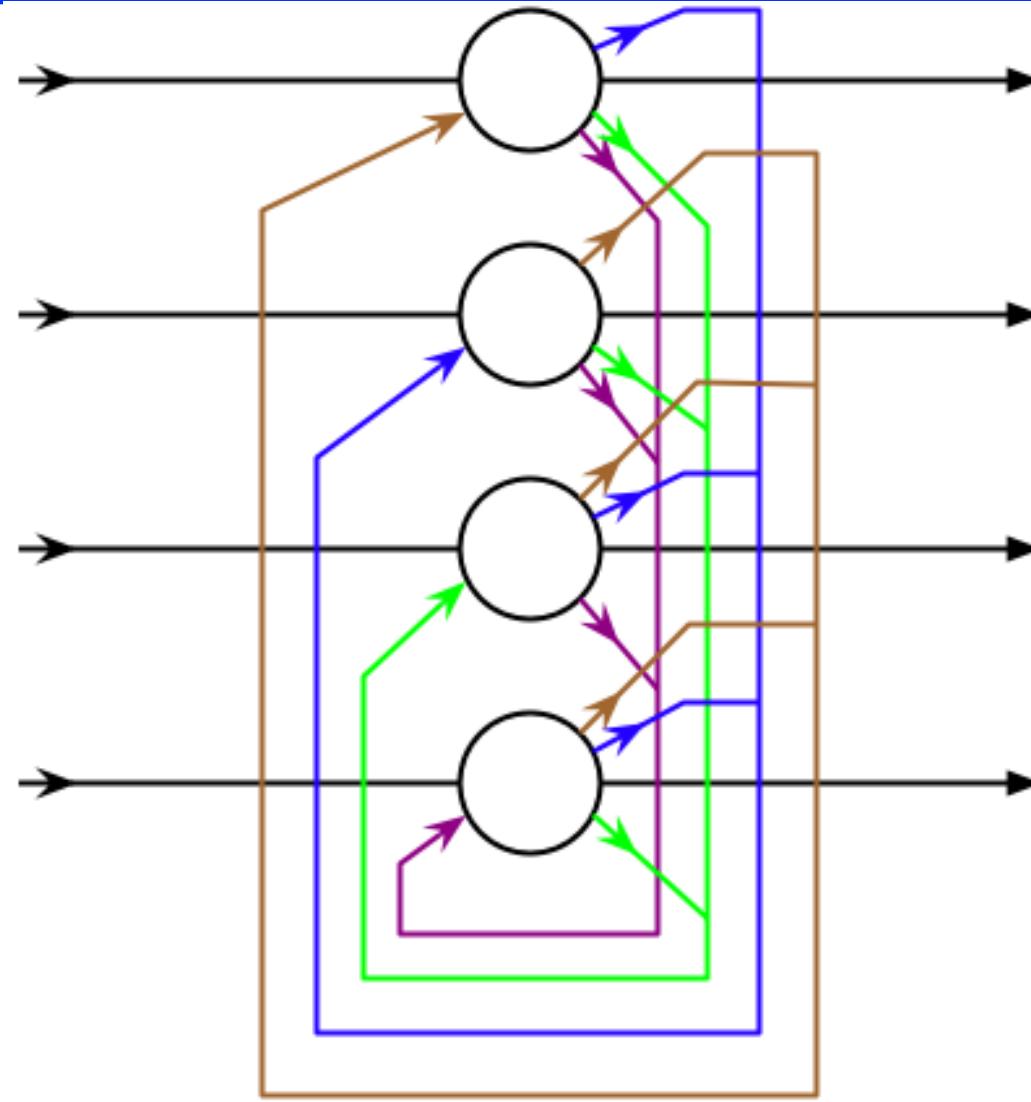
5. Hopfield

- Genelde ikili (0-1) ve bipolar (-1, +1) girişleri kabul eder.
- Tek katman olarak işlem görür.
- Her nöron çıkışı diğer nöronlarla bağlantılıdır.
- Geri beslemeli bir yapıya sahiptir.
- Tek adımda eğitilir.
- Gürültülü verilerden orijinal verilerin eldesinde kullanılır.

- Eğitme esnasında örnek giriş grupları seçilir, bunlar ağ ağırlıklarının başlangıç değerlerini hesaplamak için kullanılır.
- Bir kere yapılan bu işlemden sonra herhangi bir giriş ağa verilir. Bu da girişe en çok benzeyen örnek girişlerden biri ile sonuçlandırılır.
- Çıkış, birimlerin durumlarına bakılarak ağdan okunabilir.

- Birimler arasındaki etkileşim ağ bağlantıları üzerindeki ağırlıklar tarafından yönlendirilir.

- Ağırlık değerleri öyle ayarlanmalıdır ki sonuç olarak elde edilen kararlı durumlar örnek desenlerden birini temsil etsin.





$$w_{ij} = 0, i = j$$

$$w_{ij} = \frac{1}{N} \sum_{c=1}^P x_i^c x_j^c, i \neq j$$

- w_{ij} : i . İşlem elemanından j. İşlemci elemanına olan bağlantının ağırlığını,
- x_i^c : c sınıfı için eğitme giriş deseninin i. Elemanını
- P: sınıf sayısını
- N: işlemci eleman sayısını



- Bilinmeyen bir bilgi ağa girildiğinde , ağın çıkışları bilinmeyen bir desen elemanlarına eşitlenir.

$$y_i(0) = x_i \quad 1 \leq i \leq N$$

- Bu başlangıç değerleri ile başlayarak ağ aşağıda verilen denklemi kullanarak minimum enerji durumuna geçmek için döngüye girer.



$$y_i(k+1) = f \left[\sum_{j=1}^N w_{ij} y_j(k) \right] \quad 1 \leq i \leq N$$

- Burada *f hardlim fonksiyonudur (-1,1)*

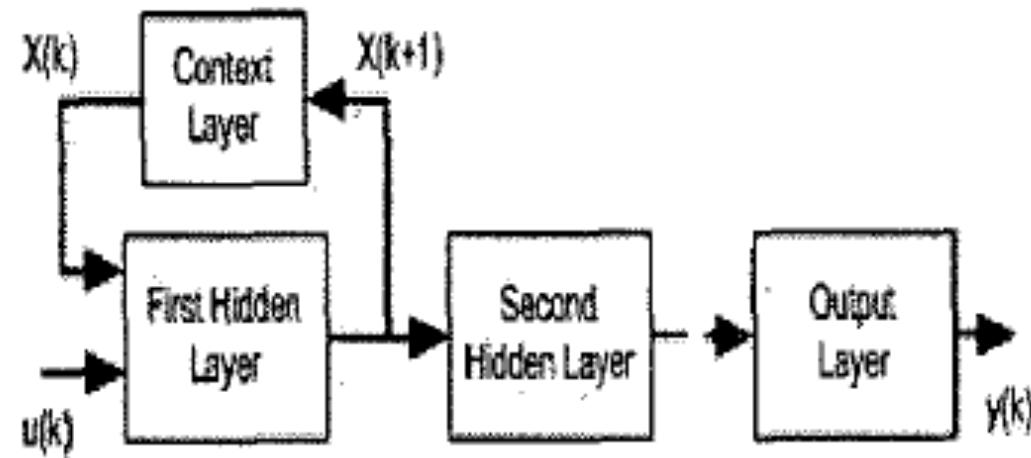


6. Elman ve Jordan Ağları

- Çok katlı yapıya sahiptirler.
- Gizli katmana ek olarak durum katmanı adı verilen bir katman vardır. Bu katman gizli katmandan veya çıkış katmanından geri besleme bilgisi alır.
- Durum tabakasının çıkışları ileriye doğru gizli katmanın girişleri olarak verilir.



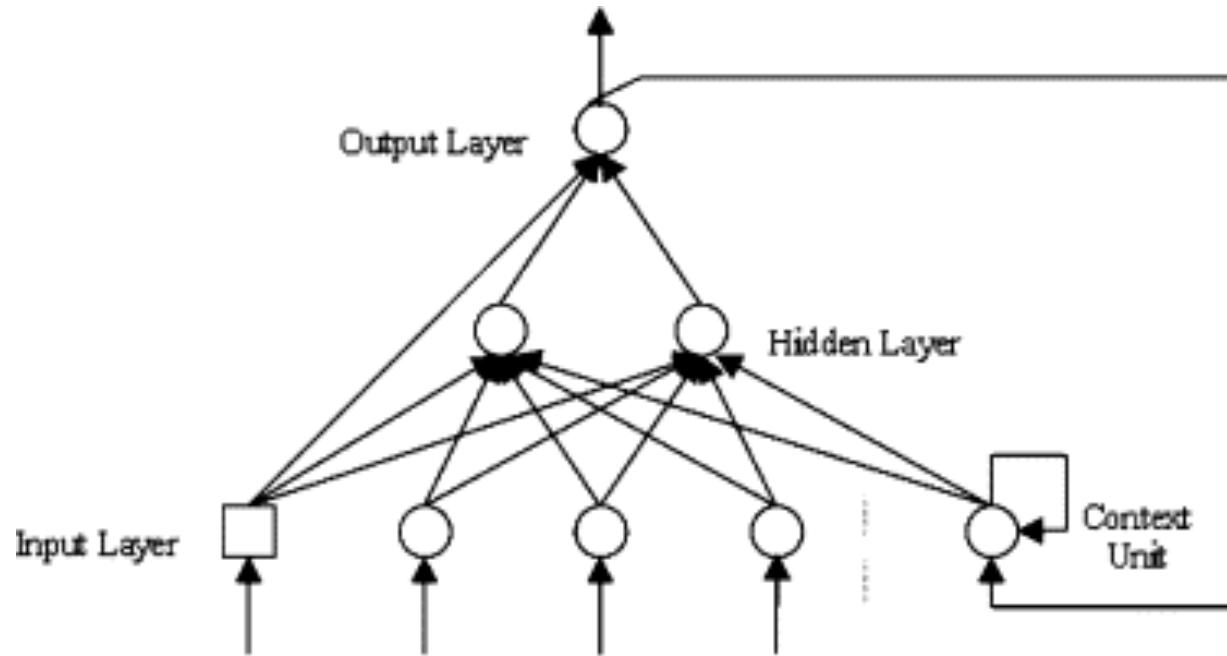
Elman ağ yapısı



Elman Network

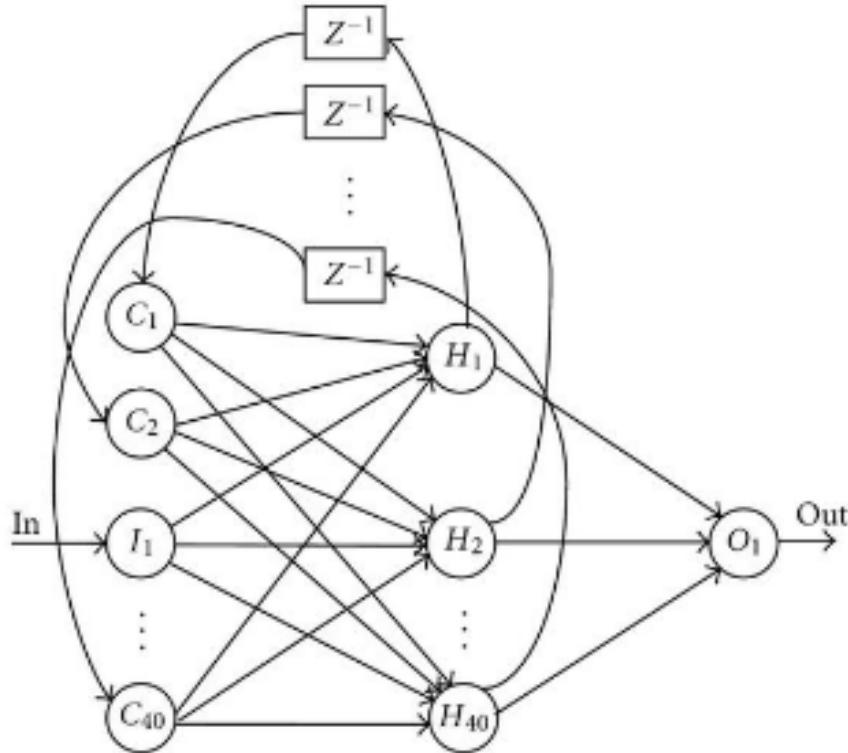
Jordan ağ yapısı:

- Jordan ağda durum katmanındaki her elemanın kendine de dönüşü vardır.

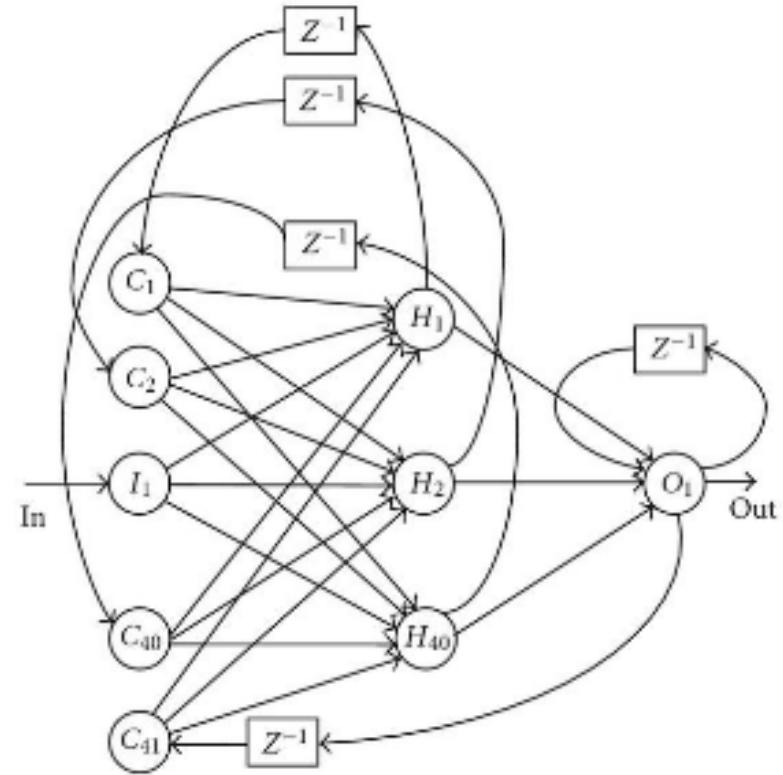




Elman & Jordan



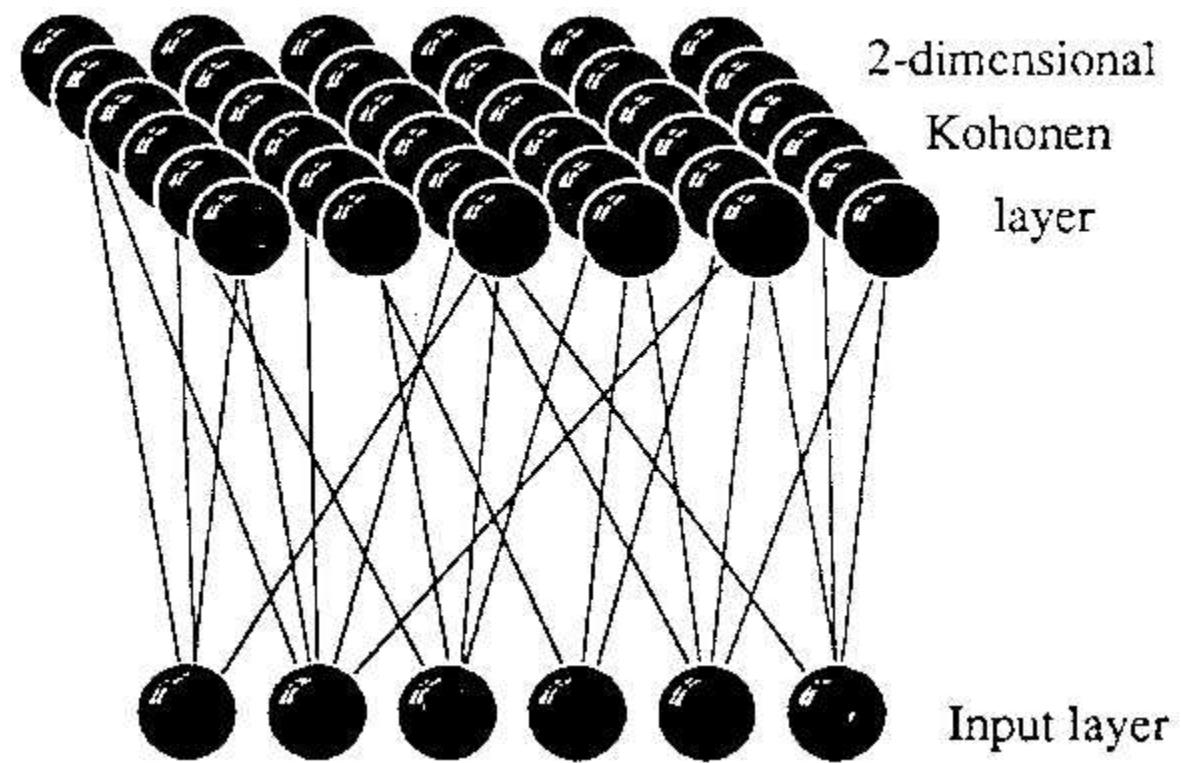
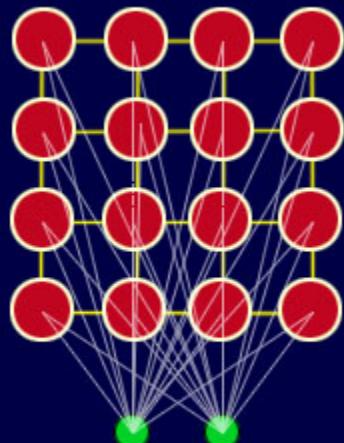
Elman Neural Network



Jordan Neural Network

7. Kohonen Ağı

- Bir giriş ve bir çıkış katmanından oluşur.
- Çıkış tabasındaki işlemci elemanlar genellikle düzenli iki boyutlu aralıklar olarak düzenlenir.
- Çıkıştaki her işlem elemanı, bütün giriş işlemci elemanlarına bağlıdır.
- Bağlantıların ağırlıkları, verilen çıkış işlemci elemanı ile ilgili olan referans vektörünün elemanlarını oluşturur.
- Kümeleme problemlerinde sıkça kullanılır.



Sınıflandırma ve kümeleme nedir ve farkı nedir?

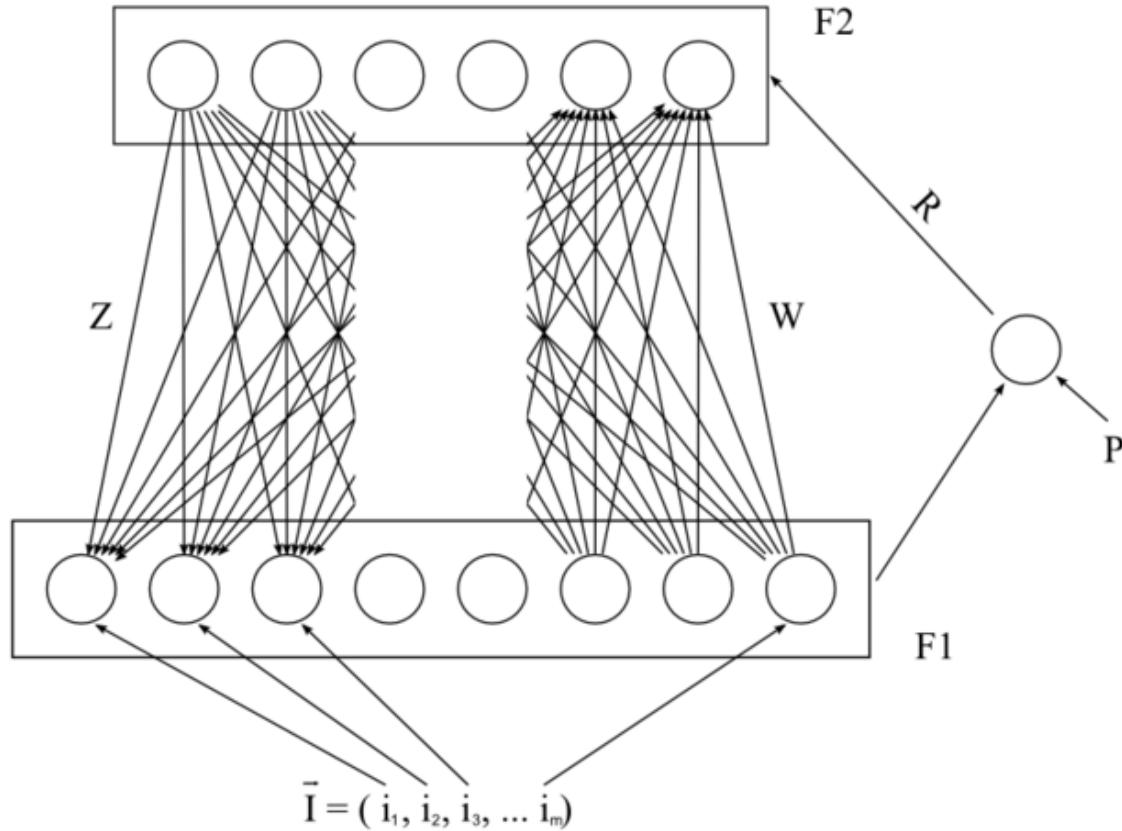
- Sınıflandırma, gözetimli(supervised); kümeleme gözetimsiz(unsupervised) öğrenme metodudur.
- **Sınıflandırmada**
 - Verilerin etiketi vardır.
 - Verileri bir gruba dahil etmek için bir kural oluşturulmasını bekler.
 - Veri setini eğitim ve test olarak ayırmak gereklidir.
- **Kümelemede,**
 - Verilerin etiketi yoktur.
 - Verileri bir gruba yakınlığa/benzerliğe/alakaya/hiyerarşiye göre dahil eder.
 - Verideki örüntülerini ve yapıları tespit eder.

8. ART (Adaptive Resonance Theory) Ağı

- ART1-ART2 , Fuzzy ART gibi tipleri vardır.
- Kümeleme problemlerinde sıkça kullanılır.
- Öğreticisiz öğrenme ağıdır.
- ART1 giriş ve çıkış olmak üzere iki katman vardır. İki tabakada birbiriyle çift yönlü iletişim içindedir.

- İleri yöndeki bağlantıların vektörlerinin toplamı ağın uzun-dönem hafızasını oluşturur.
- Bu vektörler giriş desenine bezerlikleri ölçüsünde çıkış işlemci elemanlarını belirlerler.

- Çıkış işlemci elemanlarının, geri bağlantılarının vektörleri ise, giriş deseninin hafızadaki desenine yeterince benzeyip benzemediğini (o sınıfı ait olup olmadığını) kesin olarak belirlemek için bir çeşit test amacı ile kullanılır.
- Bu vektörle ağın kısa dönem hafızasını oluşturur.



Olasılık Tabanlı YSA

Genel Regresyon YSA

Çoklu ve Modüler YSA

Mantıksal YSA

GMDH (Group Method of Data Handling)

ÖDEV 3

- Bir öğrenme algoritmasını ve bir ağ yapısını araştırarak açıklayınız.
- Bir uygulama örneği yapınız veya hazır bir uygulama örneğini tanıtınız.

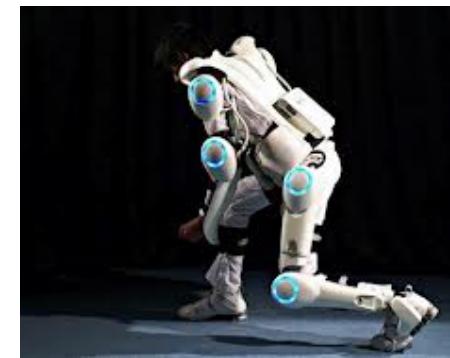


Mekatronik Mühendisliği Uygulamalarında Yapay Zekâ

Öğrenme parametreleri ile YSA

Erhan AKDOĞAN, Ph.D.

Bu ders notu http://bilgisayar.kocaeli.edu.tr/files/62_DM-YSA.pptx dosyasından hazırlanmıştır.



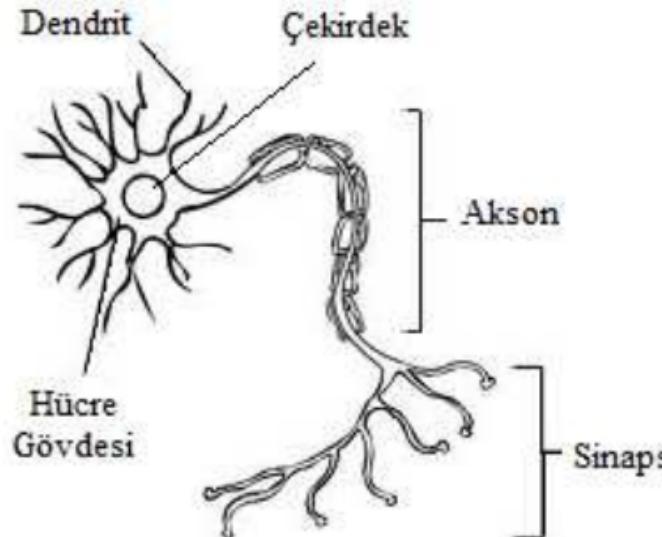
İnsan Beyninin Modellenmesi

- Yapay sinir ağları (YSA) insan beyinden esinlenmiştir.
- Öğrenme sürecinin matematiksel olarak modellenmesi temeline dayanmaktadır.
- YSA çalışmaları ilk olarak beynin biyolojik üniteleri olan nöronların modellenmesi ile başlamıştır.
- Günümüzde ise pek çok alanda uygulanmaktadır.



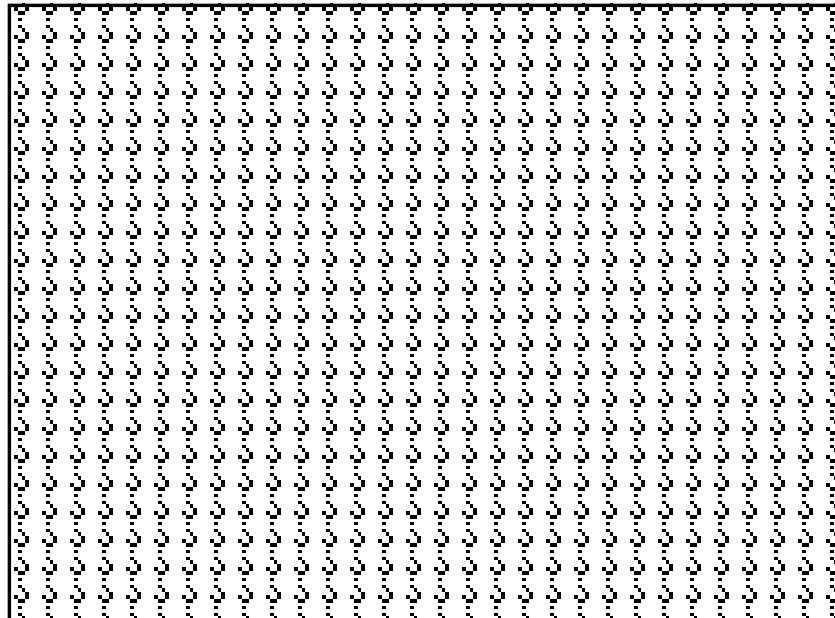
Biyolojik Sinir Hücresi

- Biyolojik bir sinir hücresi; bir gövde, bir akson, çok sayıda sinir ucu (dendrit) ve akson ile diğer sinir hücresinin sinir ucu arasında kalan ince uzantılar (sinaps) olmak üzere dört bölümden oluşmaktadır.
- Dendritler, gelen sinyalleri çekirdeğe ileter. Çekirdek dendritten gelen sinyalleri bir araya toplar ve aksona ileter. Toplanan bu sinyaller, akson tarafından işlenerek sinapslara gönderilir. Sinapslar da yeni üretilen sinyalleri diğer sinir hücrelerine ileter.
- Yapay sinir hücreleri, gerçek sinir hücrelerinin simule edilmesiyle gerçekleştirilir



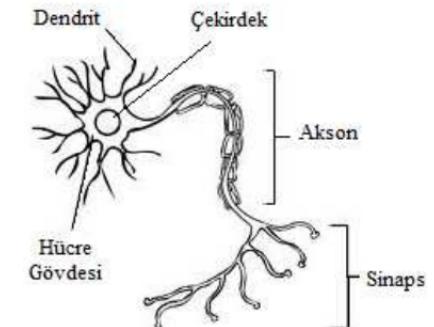
YSA Genel Yapısı ve Öğrenme

- Dış ortamdan veya diğer hücrelerden alınan girdiler, ağırlıklar yardımıyla hücreye bağlanır. Toplama fonksiyonu ile net girdi hesaplanır. Net girdinin aktivasyon fonksiyonundan geçirilmesiyle net çıktı hesaplanır. Bu işlem aynı zamanda bir hücrenin çıkışını verir.
- Her bağlantının bir ağırlık değeri vardır.
- YSA, kendisine örnekler gösterildikçe bu ağırlık değerlerini değiştirir.
- Hedef, ağa gösterilen örnekler için doğru çıkışları verecek ağırlıkları bulmaktadır.



Biyolojik ve Yapay Sinir Hücreleri

- **Biyolojik sinir hücresi** → ■ **Yapay sinir hücresi**
- Akson
- Dentrit
- Çekirdek
- Sinaps
- ■ Çıktı
- ■ Toplama fonksiyonu
- ■ Aktivasyon fonksiyonu
- ■ Ağırlıklar



YSA Algoritması

- Örnekler çok sayıda nitelik çiftleri ile temsil edilmektedir. Öğrenilecek olan hedef fonksiyonu önceden tanımlanmış olan özellikler vektörü ile tanımlanabilir.
- Hedef fonksiyon ayrık değerli, reel değerli yada ayrık ve reel değerlerden oluşan vektör şeklinde olabilir.
- Eğitim verileri hata içerebilir. YSA öğrenme metotları eğitim verisindeki gürültülere dayanıklıdır.
- Uzun süren eğitim zamanları olabilir.
- Öğrenme süresi uzundur fakat, öğrenilen durumun yeni örneklerde test edilmesi yani değerlendirilmesi hızlıdır.
- Hedef fonksiyonu öğrenme yeteneğini anlamak önemli değildir. YSA tarafından öğrenilen ağırlıkların yorumlanması zordur.

YSA'da Bilgi

- Bilgi ağın bağlantı ağırlıkları ile temsil edilir.
- YSA'nın zekası ağırlıklardır.
- Ağın sahip olduğu ağırlık değerlerinin doğru olduğu ölçüde ağın performansı fazladır.
- Bilgi dağıtılmıştır, ağırlık değerlerinin bazıları kaybolsa dahi ağ çalışmasını sürdürübilir.

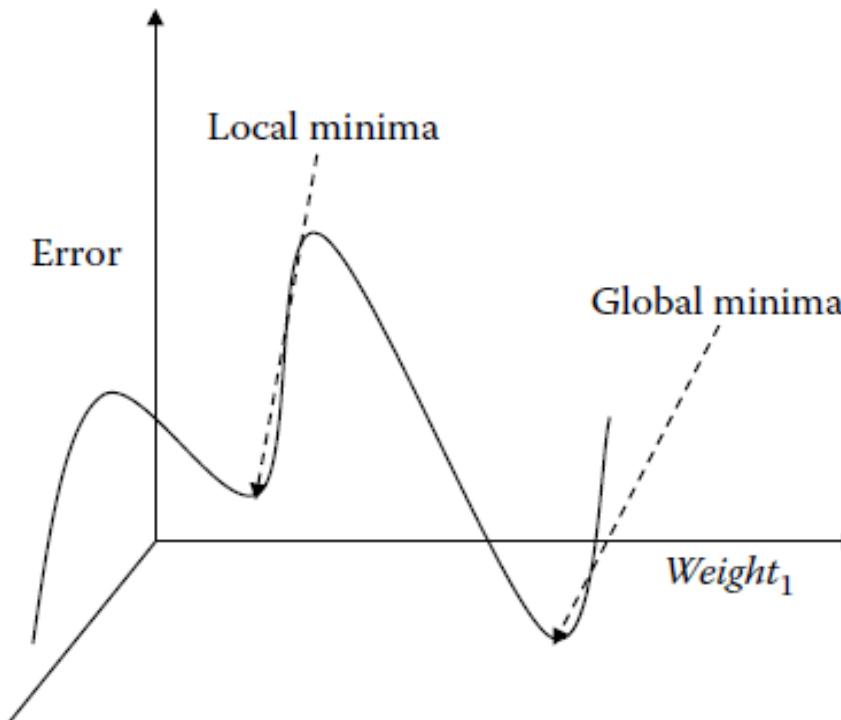


ANN Öğrenme Özellikleri / Hata Fonksiyonu

- İyi bir ANN minimum hataya sahiptir.
- İyi bir öğrenme fonksiyonu ANN'yi yüksek hatalı bir konfigürasyondan düşük hatalıya taşıır.
- ANN'deki hata, hata ölçüm fonksiyonları ile ölçülür.
- Bu hata fonksiyonları eğitim kümesindeki örneklerin hatalarının tümünü temsil eder. Problemin tipine ve tasarımcının seçimine göre farklı hata fonksiyonları kullanılabilir.
- En sık kullanılanları verilmiştir:
 - Mean absolute error (MAE)
 - Mean squared error (MSE)
 - Sum squared error (SSE)



- ANN'deki hata, çıkış değerlerinin ve bias değerinin çıkışa yansımasıdır.



Hata grafiği

ANN Öğrenme Özellikleri / Epoch

- ANN eğitimi batch processing(çevrim içi) modda yapılır.
- Tüm eğitim verileri sisteme verilir. Sistem tüm bu girişleri işlemeyi bitirdikten sonra, başka bir batch proses olan ağırlıkların ve bias'ın güncellenmesi için başlar.



Epoch: tüm giriş verilerinin işlenmesindeki tek bir iterasyon.

- Her epoch'da ANN öğrenir.
- Aynı girişleri çok kere uygulamanın sonucu olarak, sistem kendini az bir hata ile eğitebilir hale gelir.
- Çok sayıda epoch ile sistem tam olarak eğitilmiş kabul edilir.
- Epoch'lar sistemin kendini eğitim verisine göre eğitebilmesi için gereklidir.

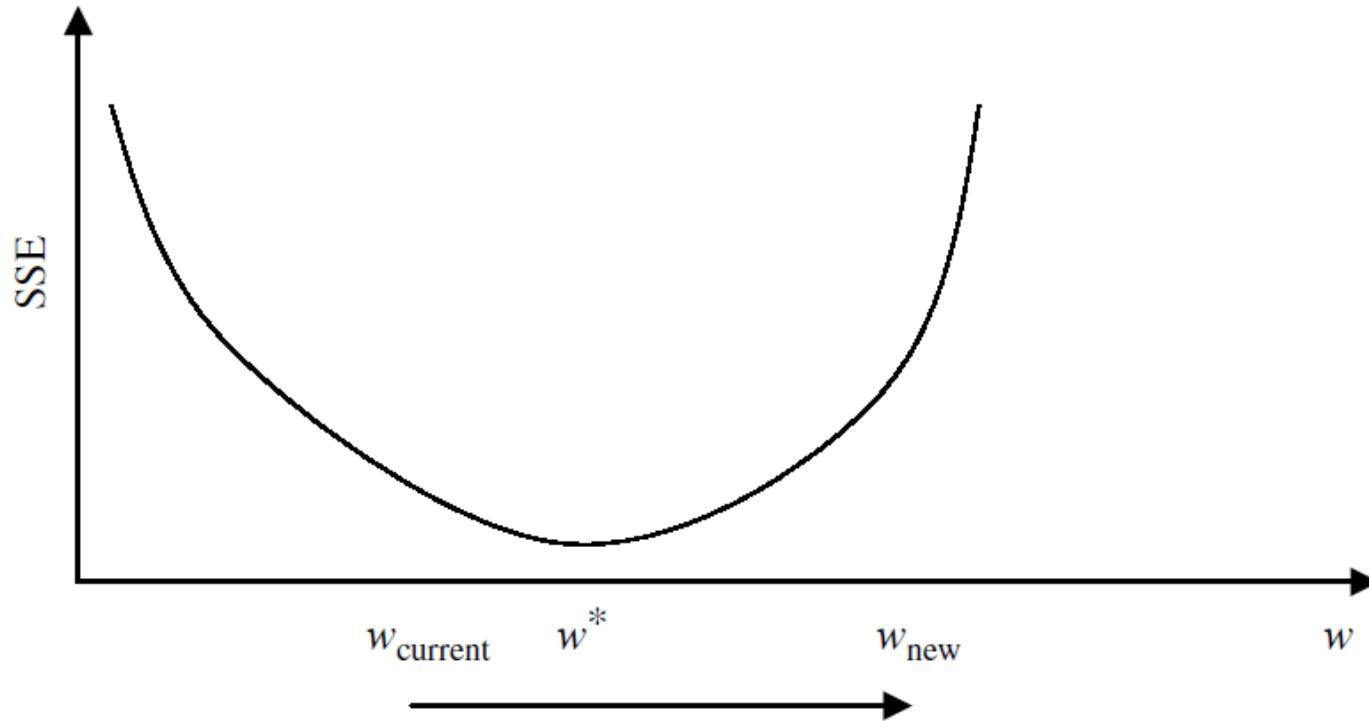


ANN Öğrenme Özellikleri / Learning Rate

- Büyük öğrenme oranı: sistem veriyi çok hızlı öğrenir, toplam hata artar.
- Düşük öğrenme oranı: sistem çok yavaş öğrenir, eğitim zamanı artar ancak hata azalır.
- η ile temsil edilir.



...



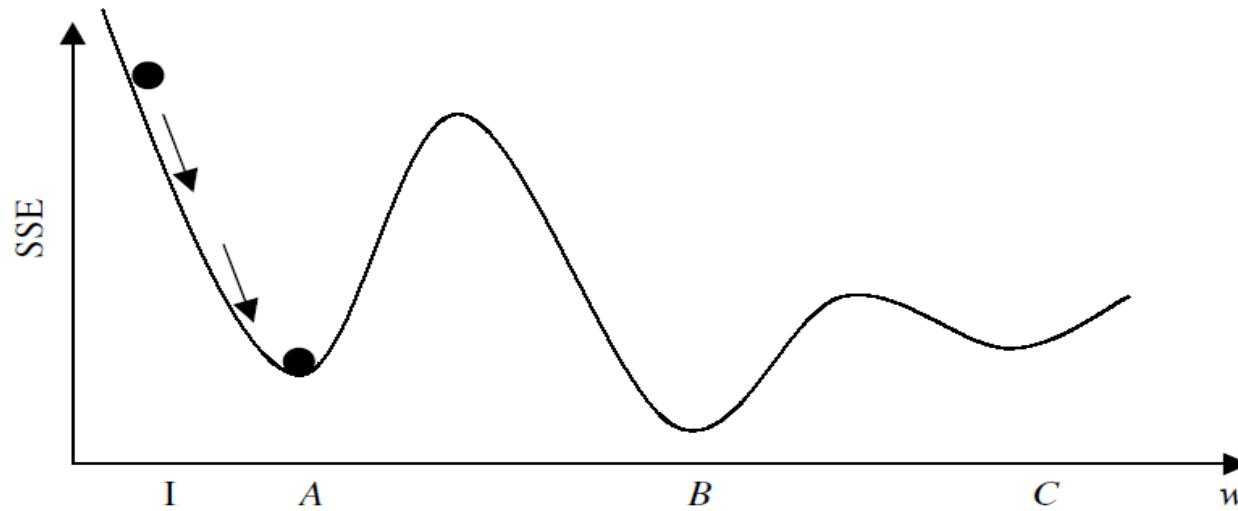
Büyük öğrenme oranı global minimumun aşılması sonucunu doğurabilir.

ANN Öğrenme Özellikleri / Momentum

- ANN her adımda daha az hata değerine sahip bir noktaya gelmek isteyecektir.
- Birden çok iterasyon sonucunda sistem minimum hatalı olan noktaya erişecektir.
- Eğitim sırasında, ANN hatanın azaldığı yerde durmayı sürdürmek ister.
- ANN için local minima tuzağına düşmek doğaldır ve bu durum global minimuma ulaşmasına engel olabilir.
- Momentum öğrenme algoritmasını local minimumdan kaçacak şekilde önceki yönde tutmaya çalışır.



- En iyi sonuçlar alınmadan birçok deneme yapılmalıdır.
- 0.6-0.8 aralığında değerler önerilir.



Küçük öğrenme oranı global minimumuma ulaşamamaya neden olabilir.



- **zaman** eşik değerine ulaştığında bitebilir
- önceden tanımlı **epoch** değeri vardır algoritma bu epoch a ulaşlığında bitebilir.
- önceden tanımlı **hata değerine** eriştiğinde bitebilir.
- **iki epoch arasındaki hata değeri** azaldığında eğitim bitebilir. Eğitim devam etse bile performansta çok fazla değişiklik olmayacağı hesap edilir.

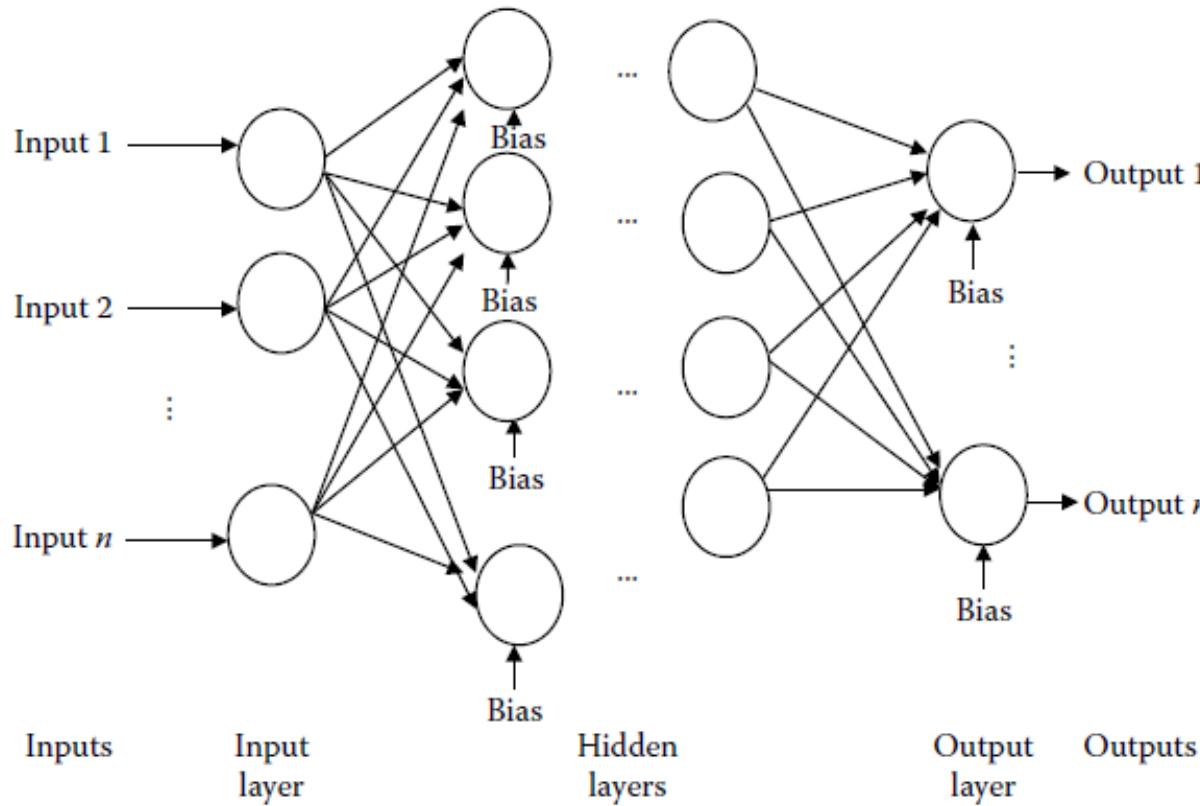


ANN Öğrenme Özellikleri / Örneklerin ağa sunulması

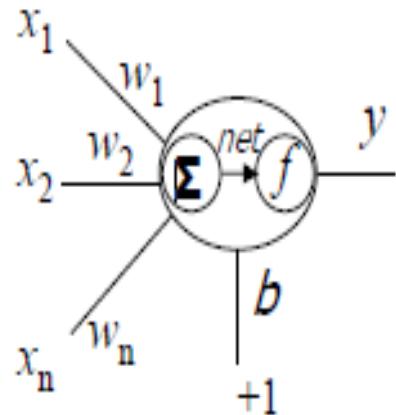
- Sıralı: n örnek sırası ile ağa sunulurlar.
 - 1. iterasyonda 1. örnek,
 - 2. iterasyonda 2. örnek vs şeklinde devam eder.
- Hepsi tek tek sunulduktan sonra başa dönerek sırası ile yeniden ağa sunulurlar. Bu işlem öğrenme sağlanana kadar devam eder.
- Rastgele: örnekler eğitim kümelerinden rastgele seçilirler. Ya seçilen örnek eğitim verisine atılıp yeniden seçilir yada seçilen örnek yeniden seçilemez. Tüm örnekler rastgele ağa sunulduktan sonra, öğrenme sağlanan kadar yeniden sunulmaya devam ederler.



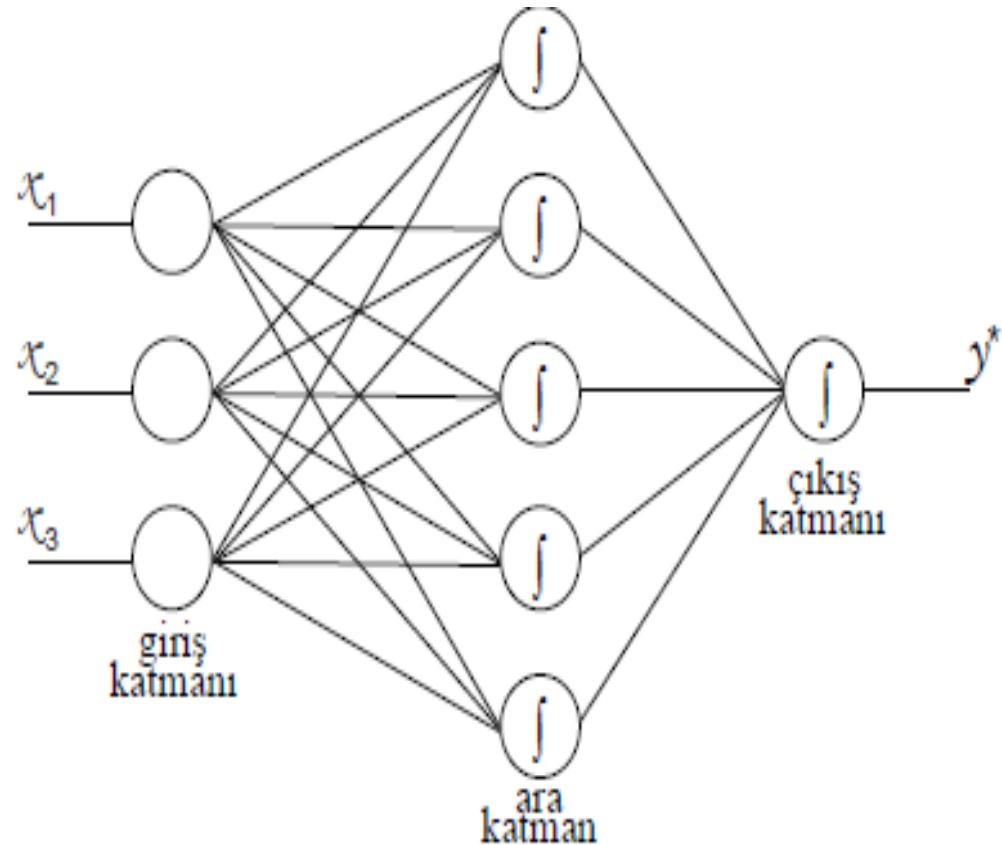
Multilayer Perceptron (MLP) Genel Yapısı



YSA, yapay sinir hücrelerinin (perceptron) birbirlerine bağlanması sonucu oluşan yapılardır



MLP ağlarında kullanılan bir nöronun genel yapısı.



Bir MLP ağının genel mimarisi.

Katmanlar

- ANN mimarisi katmanlardan kurulmuştur.
- Katmanların sayısı ANN'nin işlemsel karmaşıklığını ifade etmektedir.
- Daha çok sayıda katmandan oluşan ANN'ler her zaman daha az katmandan oluşanlara göre daha fazla işlem zamanı talep edeceklerdir.
 - Giriş katmanı (1),
 - Gizli katman (n),
 - Çıkış katmanı (1).

Giriş Katmanı:

YSA'ya dış dünyadan bilgilerin geldiği katmandır. Bu katmandaki nöronların sayısı giriş sayısı kadardır. Her nöron bir girişe aittir. Girdiler herhangi bir işleme uğramadan gizli katmana iletilirler.

Gizli Katman:

- Giriş katmanından aldığı bilgiyi işleyerek bir sonraki katmana iletir.
- Bu katman asıl bilgi işlemenin gerçekleştirildiği katmandır.
- Her gizli katmandaki nöron sayısı değişkenlik gösterebilir.
- Genelde bu katmandaki nöronların sayısı giriş ve çıkış katmanındaki nöron sayılarından fazladır.
- Gizli katman sayısı ağır karmaşıklığını da kontrol altında tutmaya yarar.

Çıkış katmanı:

- Gizli katmandan gelen bilgiyi işler ve giriş katmanına gelen girdiye uygun olarak üretilen çıktıyı dış dünyaya gönderir.
- Nöronların sayısı sistemden beklenen çıkışlar kadardır.
- Her nöron tek bir çıkışa aittir.

Ağırlıklar

- ANN eğitimi ağırlıklardaki değişim demektir.
- Ağırlıkların farklı kombinasyonları ANN'nin farklı performanslarını temsil edecektir.
- Ağırlıklar ANN zekasını oluşturur. ANN'nin öğrenme yeteneği direk olarak ağırlıklar ile ilişkilidir.

Bias (eşik):

- Sistem performansını etkiler.
- Bias öğrenmede bir ağırlık gibi alınır
- Giriş sinyallerinin toplamı 0 olduğunda öğrenme gerçekleşmez. Çıkış değerleri hep 0'dan büyük olan bias nöronları, sonradan bağlı nöronların giriş sinyallerinin sürekli sıfırdan farklı olmasını sağlar.
- Öğrenmeyi hızlandırırken yerel optimum değerlere takılmayı güçleştirirler.

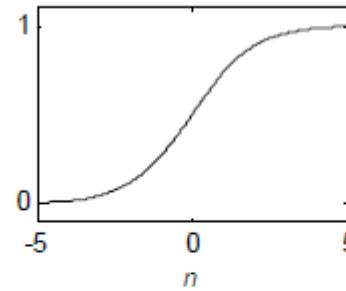
Aktivasyon Fonksiyonu

- Aktivasyon fonksiyonu: bir değişkeni farklı bir boyuta taşıyan doğrusal veya doğrusal olmayan bir fonksiyondur.
- Aktivasyon fonksiyonunun türevinin kolay alınabilmesi eğitim hızını artıran bir özelliktir.
- Sık kullanılan üç aktivasyon fonksiyonu şunlardır:
 - Sigmoid,
 - Hipерbolik Tanjant
 - Basamak

Aktivasyon Fonksiyonu

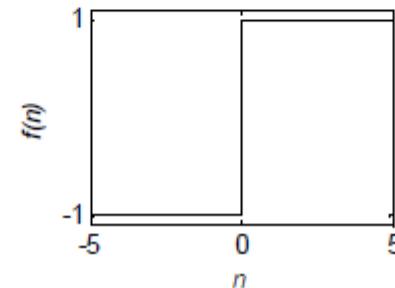
Sigmoid

$$y = \frac{1}{1 + e^{-net}}$$



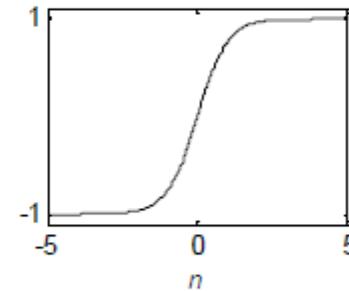
Adım basamak

$$y = \begin{cases} 1 & net \geq 0 \\ -1 & net < 0 \end{cases}$$



Hiperbolik tantant

$$y = \frac{1 - e^{-2net}}{1 + e^{2net}}$$





Aktivasyon Fonksiyonları

TABLE 2.1
Common Activation Functions

| No. | Function Name | Function Equation |
|-----|---------------|---|
| 1. | Identity | $f(x) = x$ |
| 2. | Logistic | $f(x) = \frac{1}{1 - e^{-x}}$ |
| 3. | Sigmoid | $f(x) = \frac{1}{1 + e^{-x}}$ |
| 4. | Tanh | $f(x) = \tanh(x/2)$ |
| 5. | Signum | $f(x) = \begin{cases} +1 & x > 0 \\ -1 & x < 0 \\ \text{undefined} & x = 0 \end{cases}$ |
| 6. | Hyperbolic | $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ |
| 7. | Exponential | $f(x) = e^{-x}$ |
| 8. | Softmax | $f(x) = \frac{e^x}{\sum_i e^{x_i}}$ |
| 9. | Unit sum | $f(x) = \frac{x}{\sum_i x_i}$ |
| 10. | Square root | $f(x) = \sqrt{x}$ |
| 11. | Sine | $f(x) = \sin(x)$ |
| 12. | Ramp | $f(x) = \begin{cases} -1 & x \leq 0 \\ x & -1 < x < 1 \\ +1 & x \geq 1 \end{cases}$ |
| 13. | Step | $f(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 0 \end{cases}$ |

YSA Çalışma Adımları

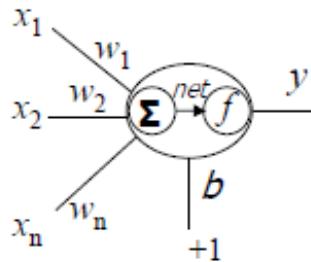
- Örneklerin belirlenmesi
- Ağın topolojisinin belirlenmesi
 - Girdi ve çıktı sayısının belirlenmesi
- Ağın öğrenme parametrelerinin belirlenmesi
 - öğrenme katsayısı ve sabitlerin belirlenmesi
- Ağın başlangıç değerlerinin atanması
- Epoch sayısı kadar
 - Eğitim setindeki tüm örnekler için
 - Örnek ağa gösterilir
 - Hatanın hesaplanması
 - Bulunan hataya göre ağırlıkların güncellenmesi
- Sistemin toplam hatası hesaplanır.

YSA Sınıflandırılması

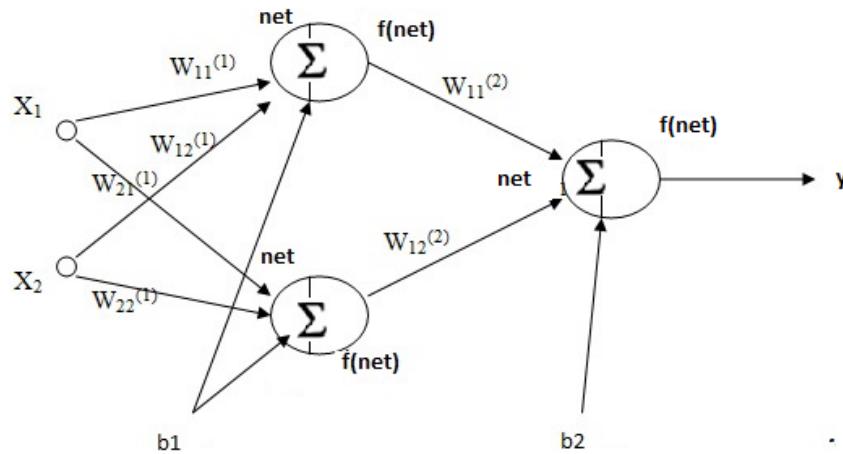
Yapılarına göre:

- İleri Beslemeli
 - Hücreler, girişten çıkışa doğru düzenli katmanlar şeklindedir. Ağa gelen bilgiler giriş katmanına daha sonra sırasıyla gizli katmanlardan ve çıkış katmanından işlenerek geçer ve sonra dış dünyaya çıkar.
- Geri Beslemeli
 - Bir hücrenin çıktısı sadece kendinden sonra gelen katmana girdi olarak verilmez. Kendinden önceki katmanda veya kendi katmanında bulunan herhangi bir hücreye girdi olarak verilebilir.

Geriye Yayılım Algoritması



MLP ağlarında kullanılan bir nöronun genel yapısı.



$$E = \frac{1}{2} \sum_j e_j^2 = \frac{1}{2} \sum_j (d_j - y_j)^2$$

$$\begin{aligned}\Delta w_{ij} &= -\eta \frac{\partial E}{\partial w_{ij}} = -\eta \left[\frac{\partial E}{\partial y_j} \right] \left[\frac{\partial y_j}{\partial \text{net}_i} \right] \left[\frac{\partial \text{net}_i}{\partial w_{ij}} \right] \\ &= -\eta [-e_j] \left[\frac{\partial y_j}{\partial \text{net}_i} \right] [x_i]\end{aligned}$$

Aktivasyon fonksiyonu olarak sigmoid kullanılırsa;

$$\Delta w_{ij} = -\eta [-e_j] [(1 - y_j)y_j] [x_i] = \eta e_j (1 - y_j)y_j x_i$$

- **E hata formülündeki d_j : beklenen çıkış, y_j : YSA çıkışı; $j:1,\dots,m$**
- **W_{ij} : j girişini, ara katmandaki i. Nörona bağlayan ağırlık değeri**

Geriye Yayılım Algoritması

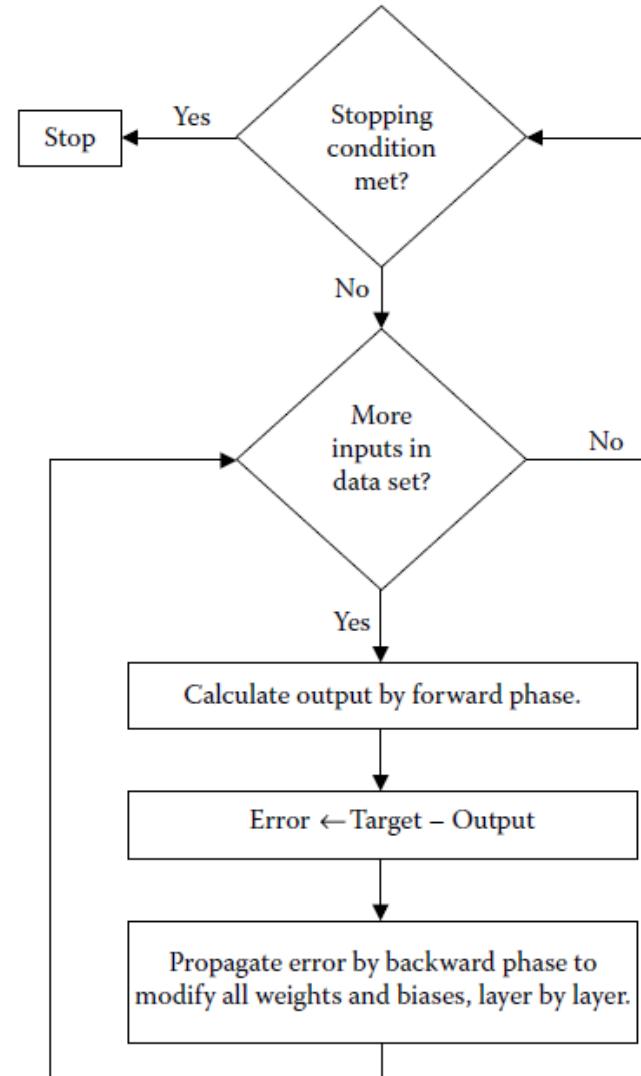
- Yerel minimuma yakalanma olasılığı yüzünden YSA eğitiminde momentum katsayısı kullanılır.
- Bunun için eğim azaltma yönteminin önceki adımlardaki değişim değerlerinin belirli oranda (α) katkısını sağlayan momentum etkisi kullanılır.
- Backpropagation'da ağırlık güncellemede en yaygın şekilde kullanılan formül:

$$\Delta w_{\text{current}} = -\eta \frac{\partial \text{SSE}}{\partial w_{\text{current}}} + \alpha \Delta w_{\text{previous}}$$

$$\Delta w_{ij}(t+1) = -\eta \frac{\partial E}{\partial w_{ij}} + \alpha \Delta w_{ij}(t)$$



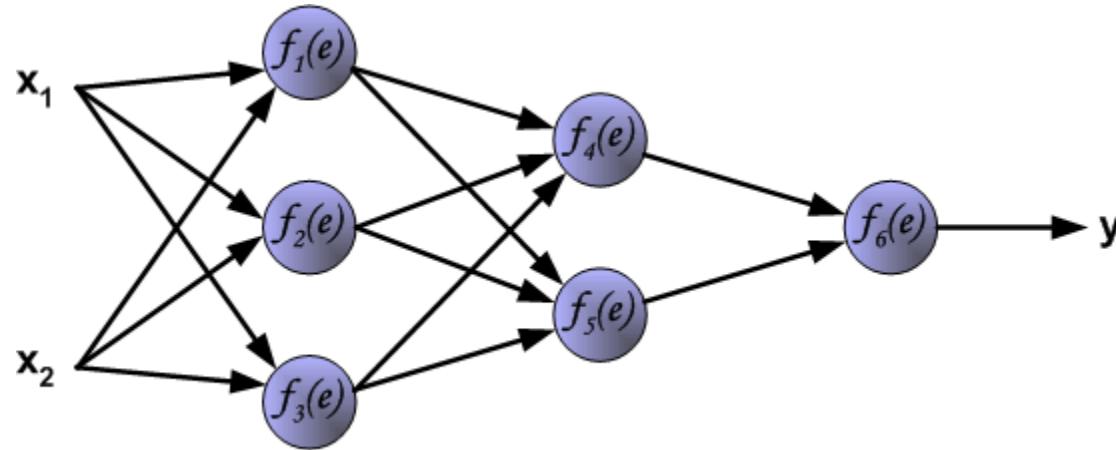
Akış Şeması



Backpropagation Algoritması

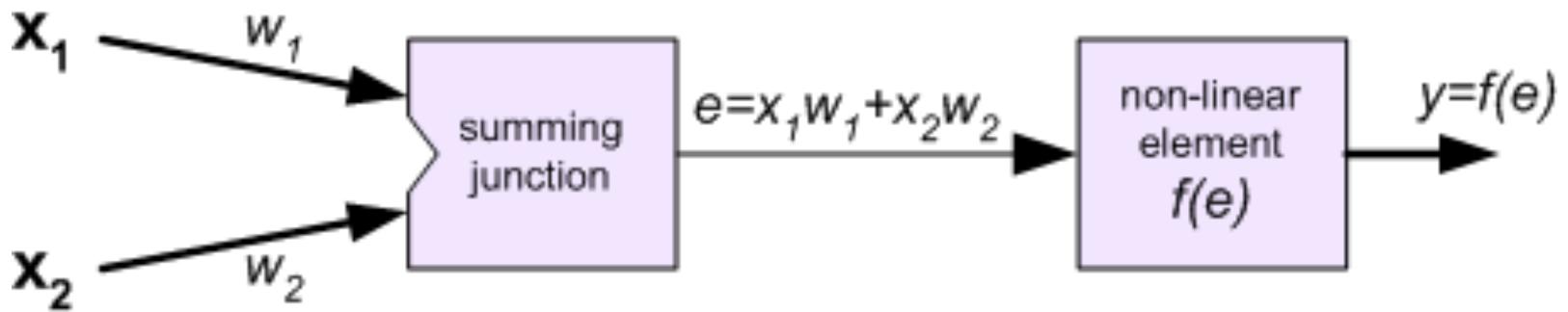
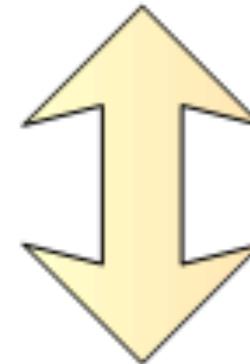
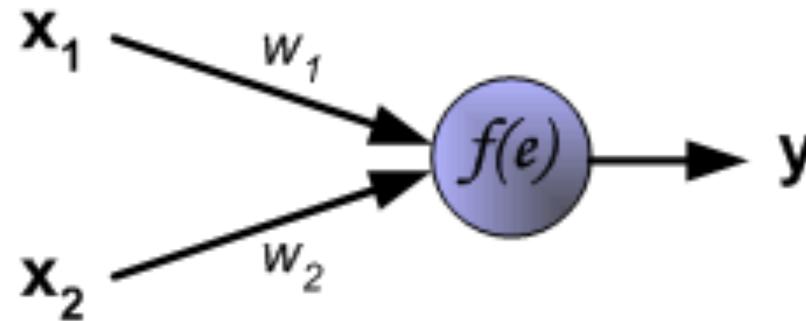
- Backpropagation daki ağırlık güncelleme döngüsü binlerce kez iteratif olarak devam edebilir.
- Belirlili bir iterasyon sonucunda döngü sonlandırılabilir yada hata değerine göre sonlandırılabilir.
- Bu seçim kritiktir çünkü az sayıda iterasyon hatayı yeterince düşürmez, çok sayıda iterasyon ise overfitting'e sebep olabilir.

Backpropagation Adımları



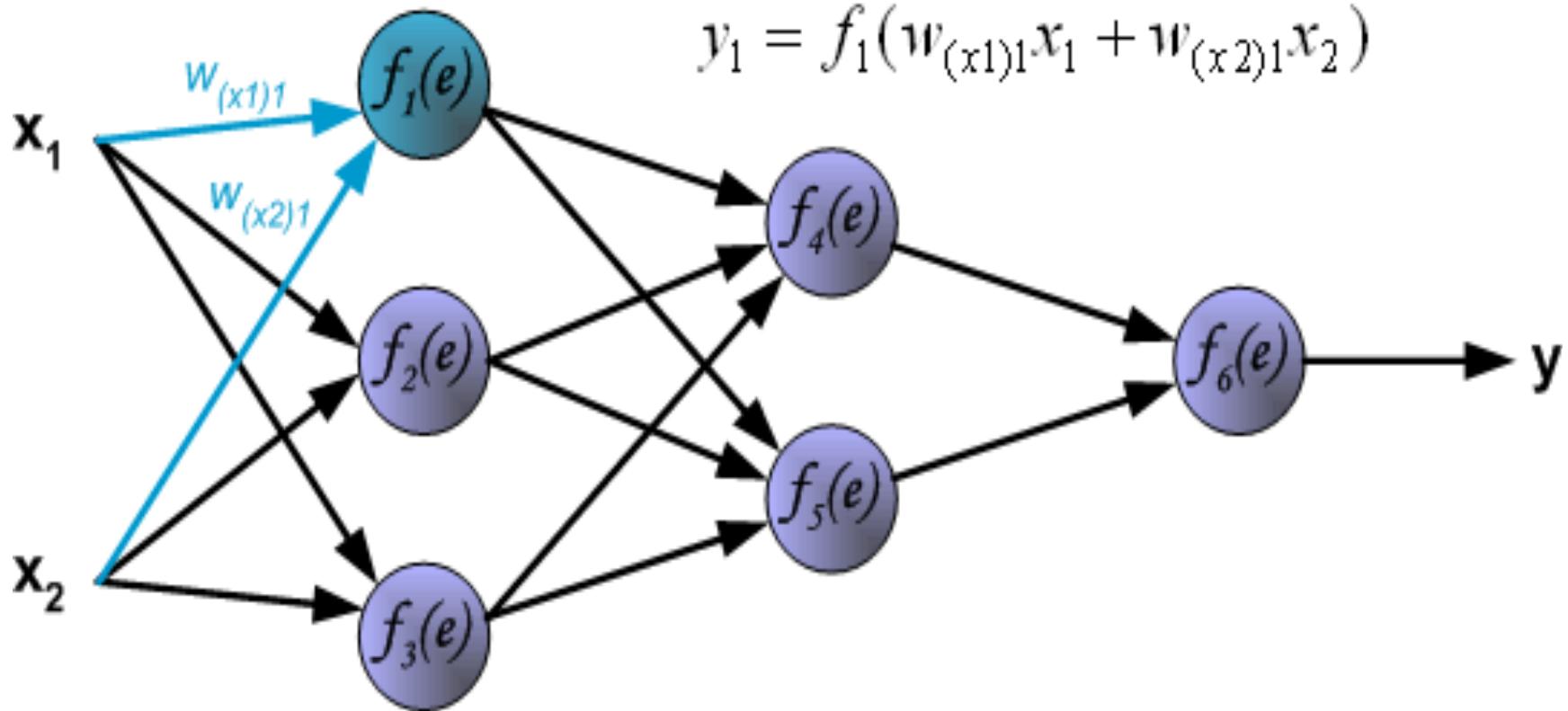


Backpropagation Adımları



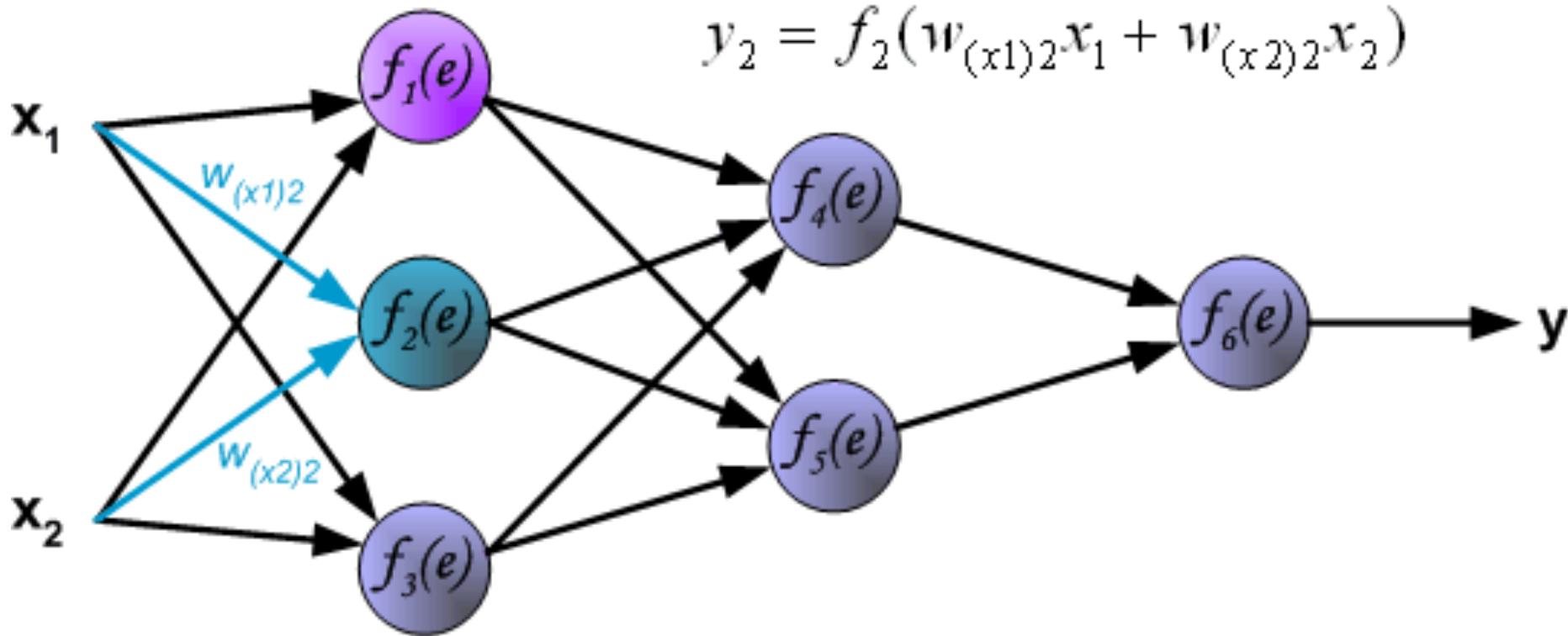


...



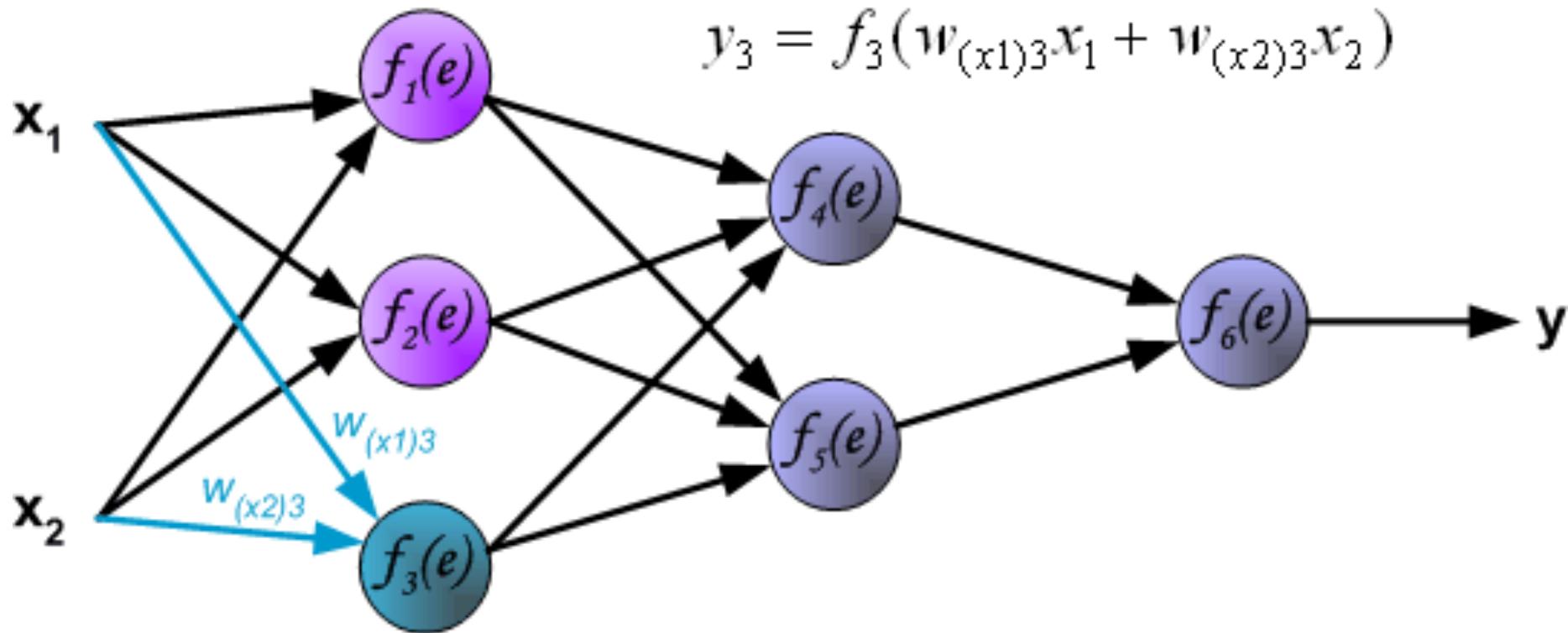


...



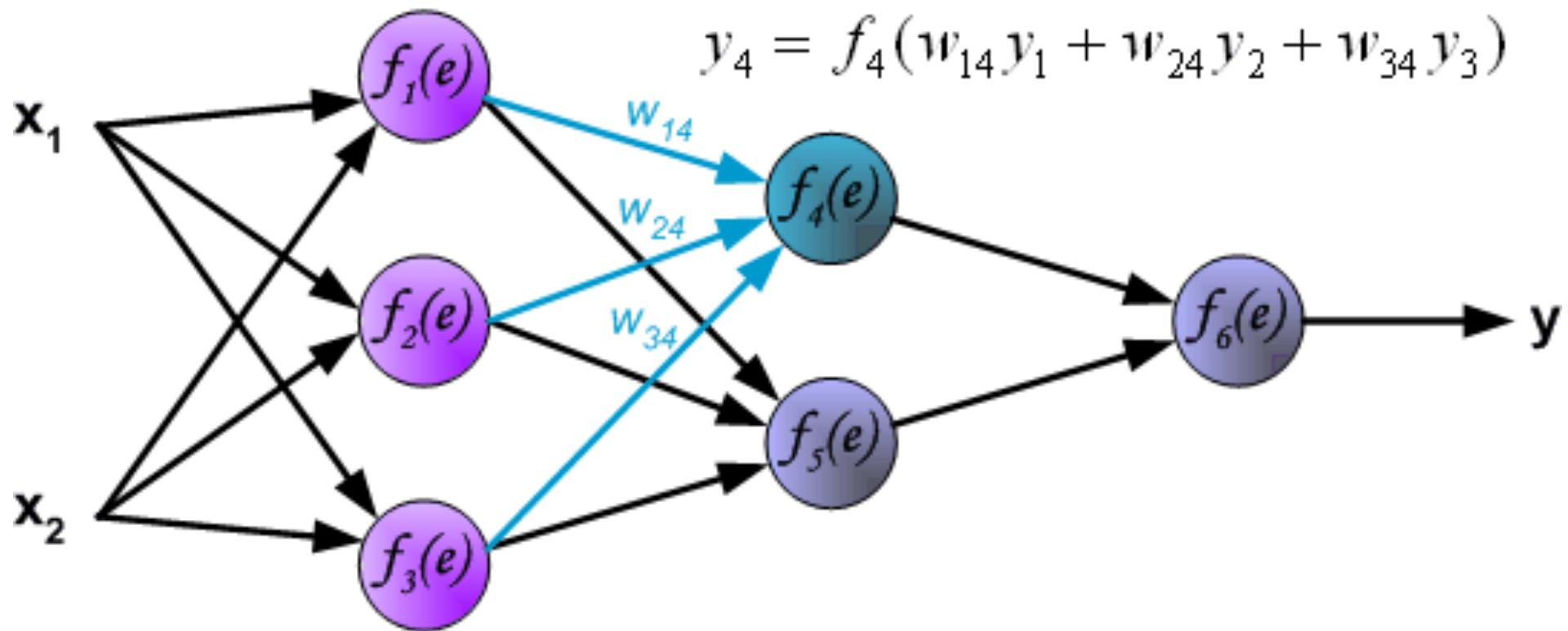


...



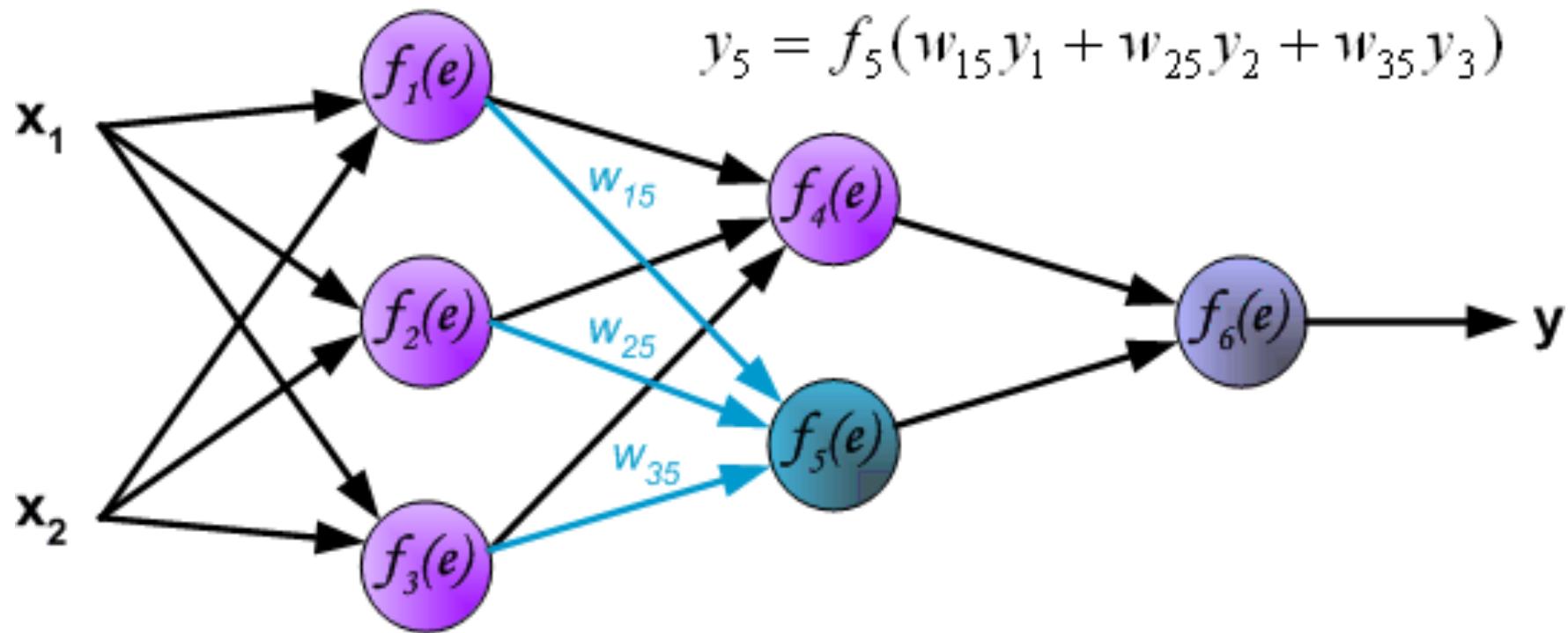


...



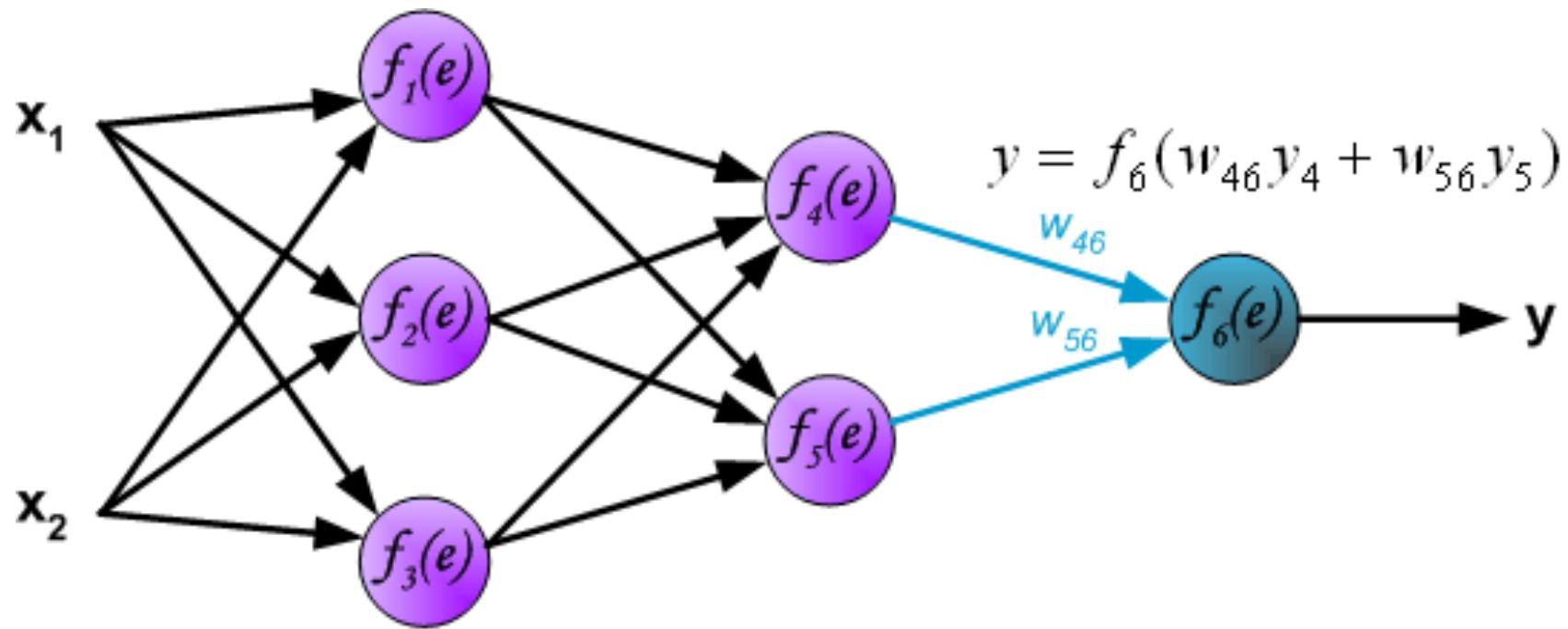


...



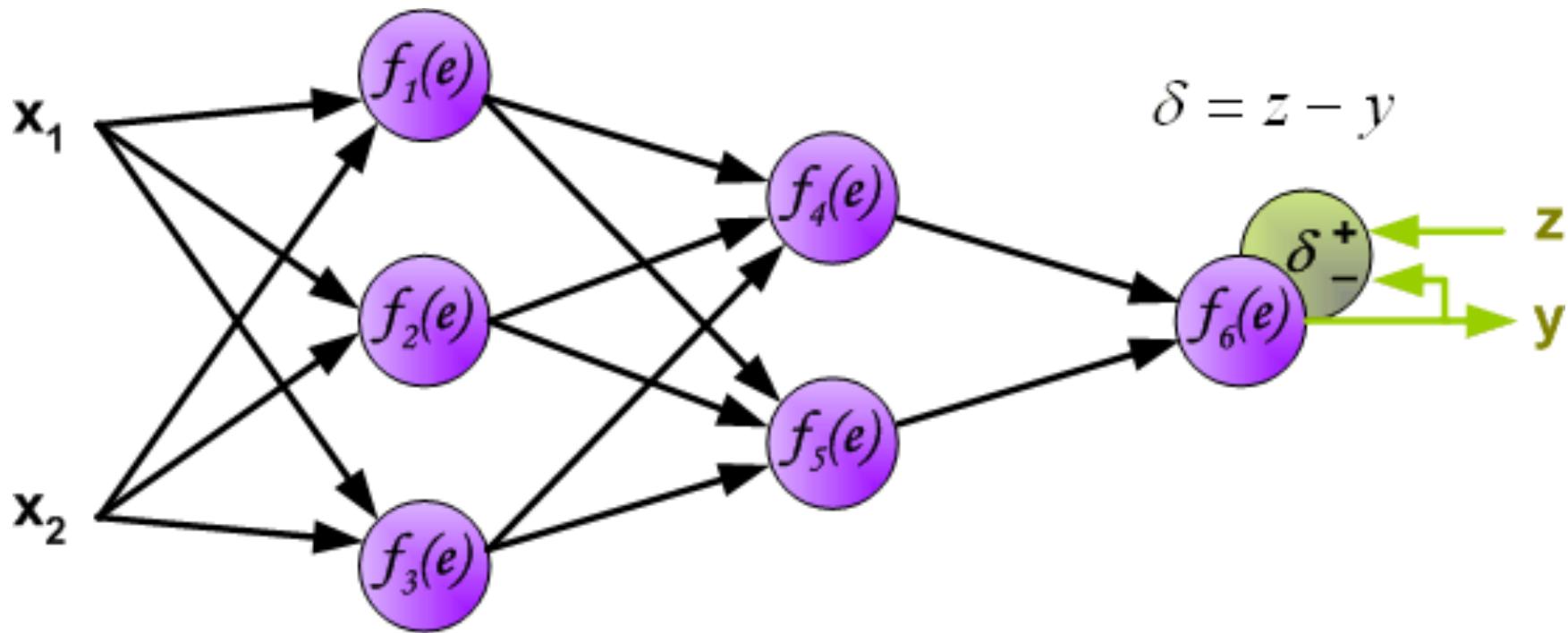


...



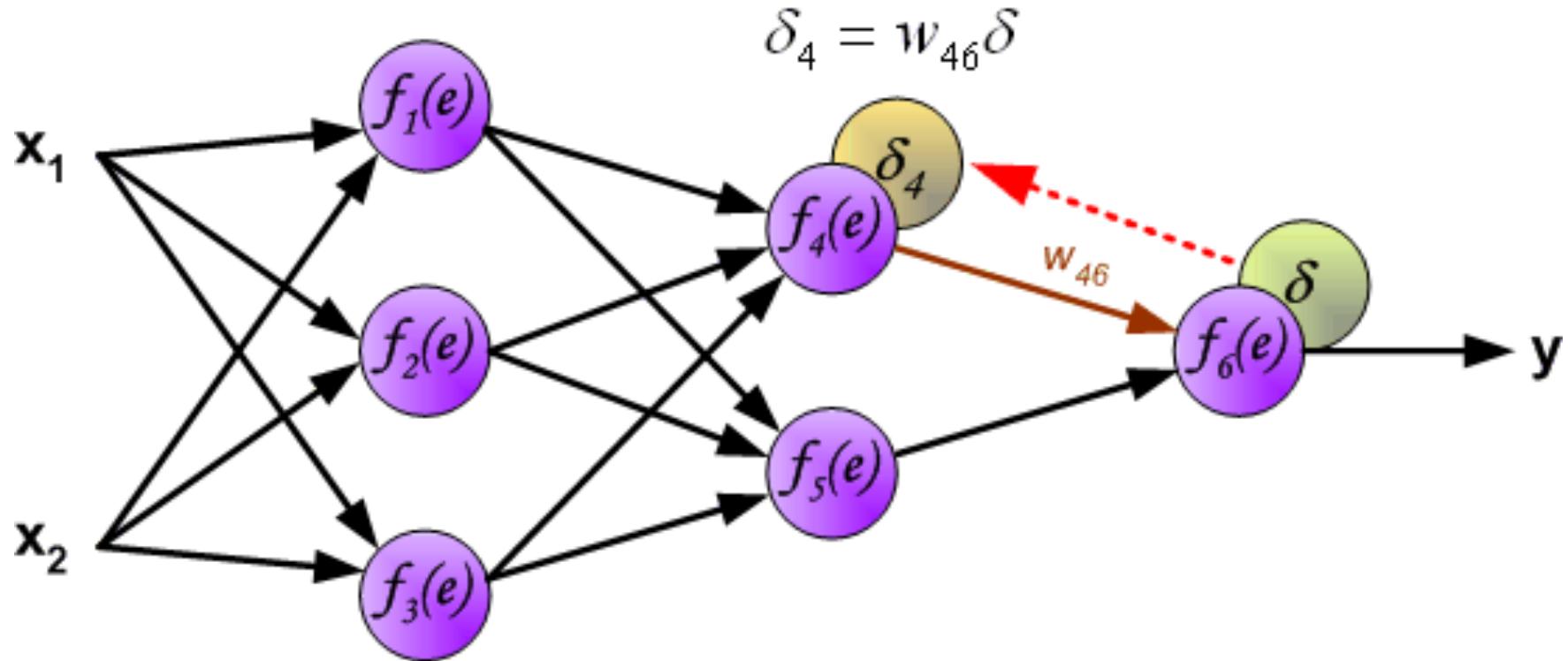


...



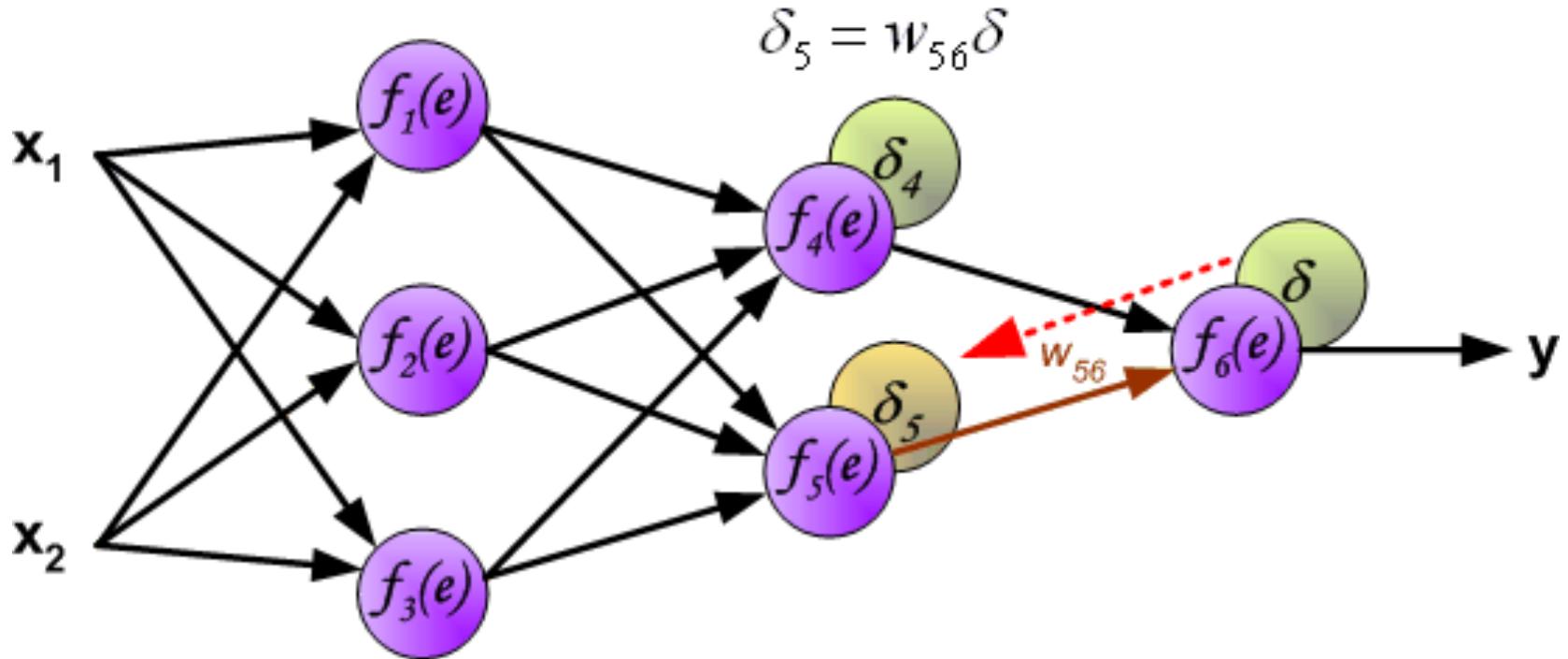


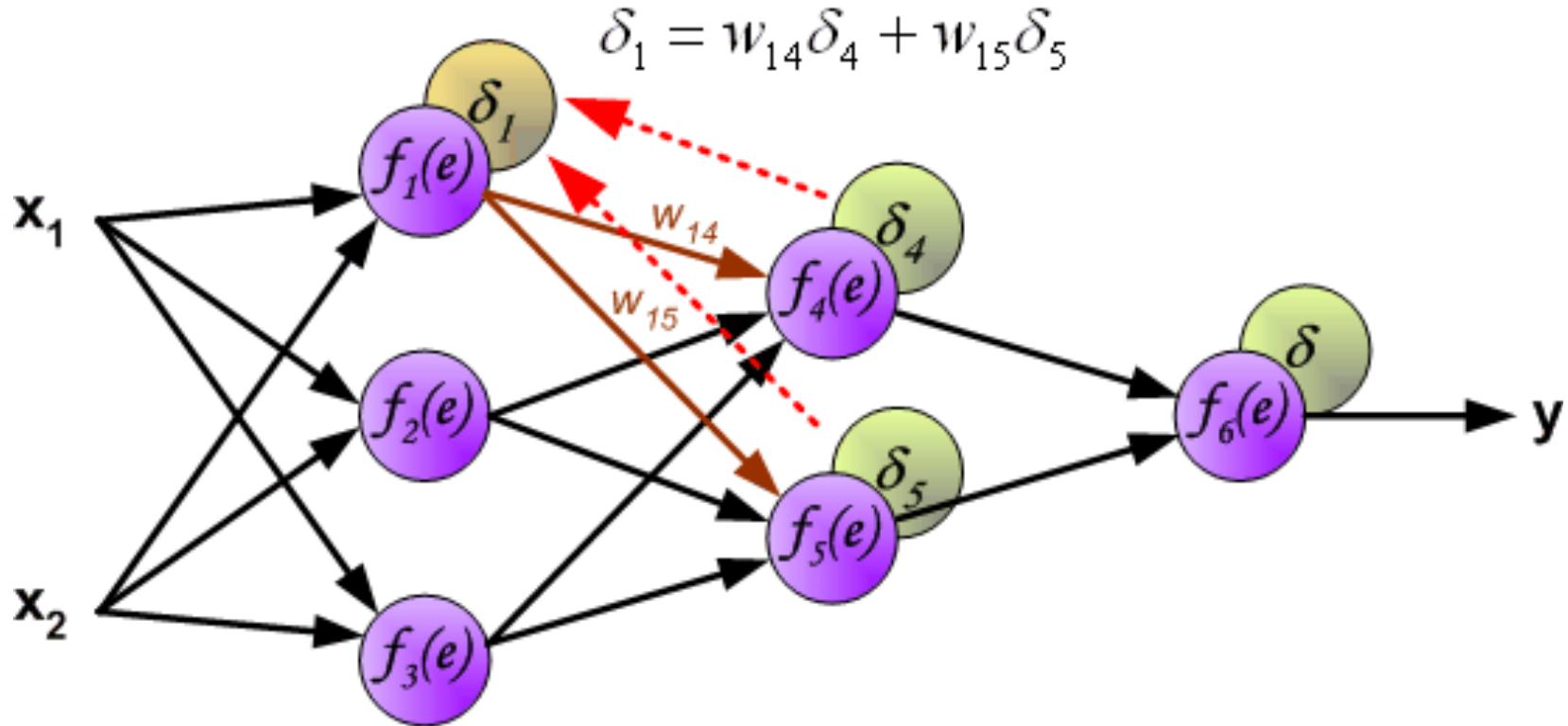
...





...

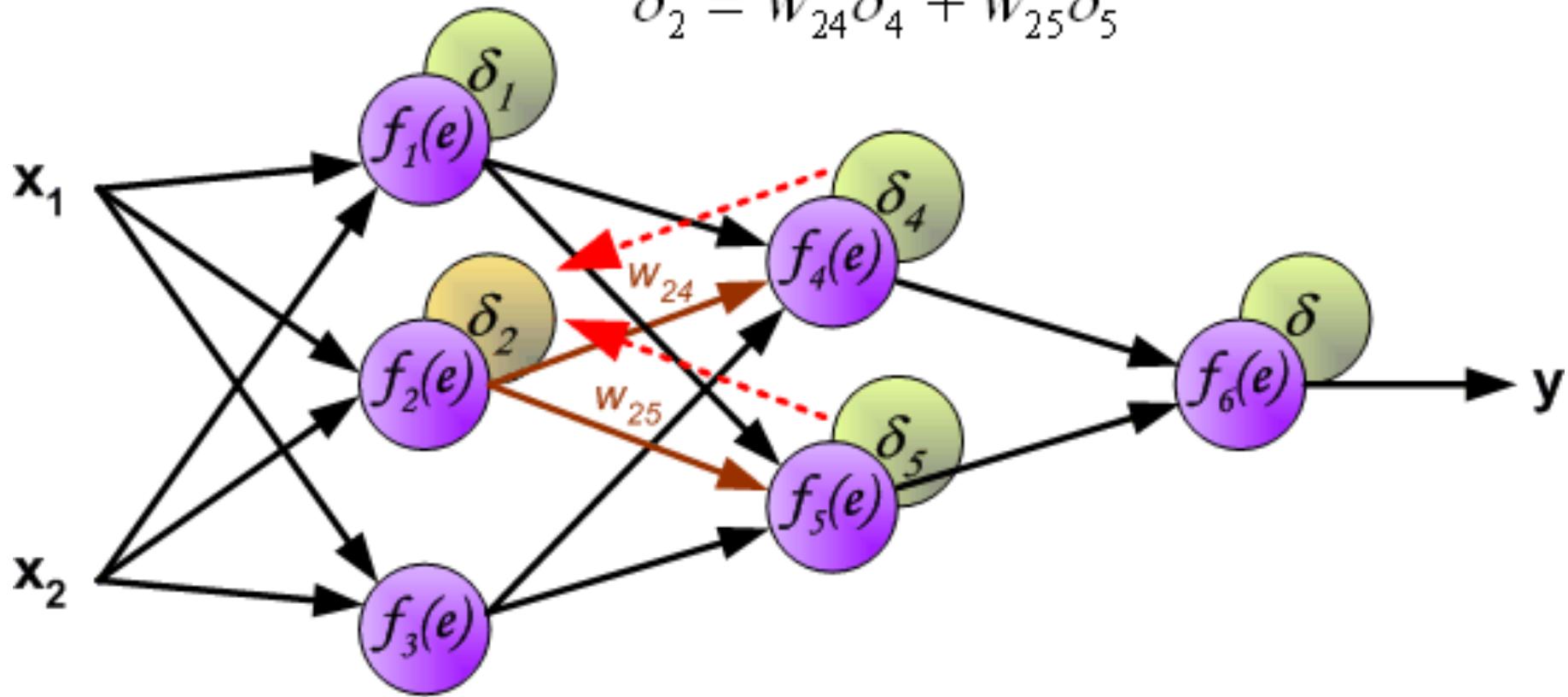






...

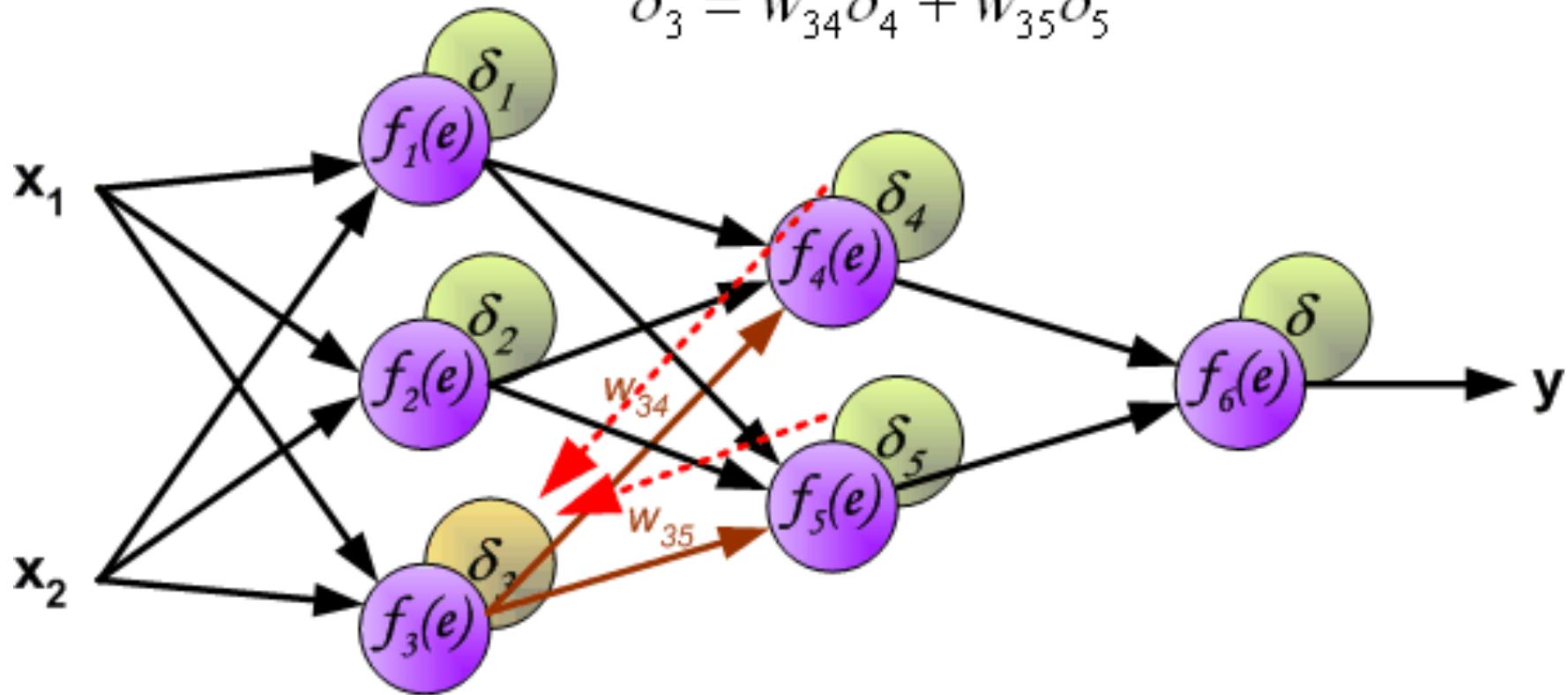
$$\delta_2 = w_{24}\delta_4 + w_{25}\delta_5$$





...

$$\delta_3 = w_{34}\delta_4 + w_{35}\delta_5$$

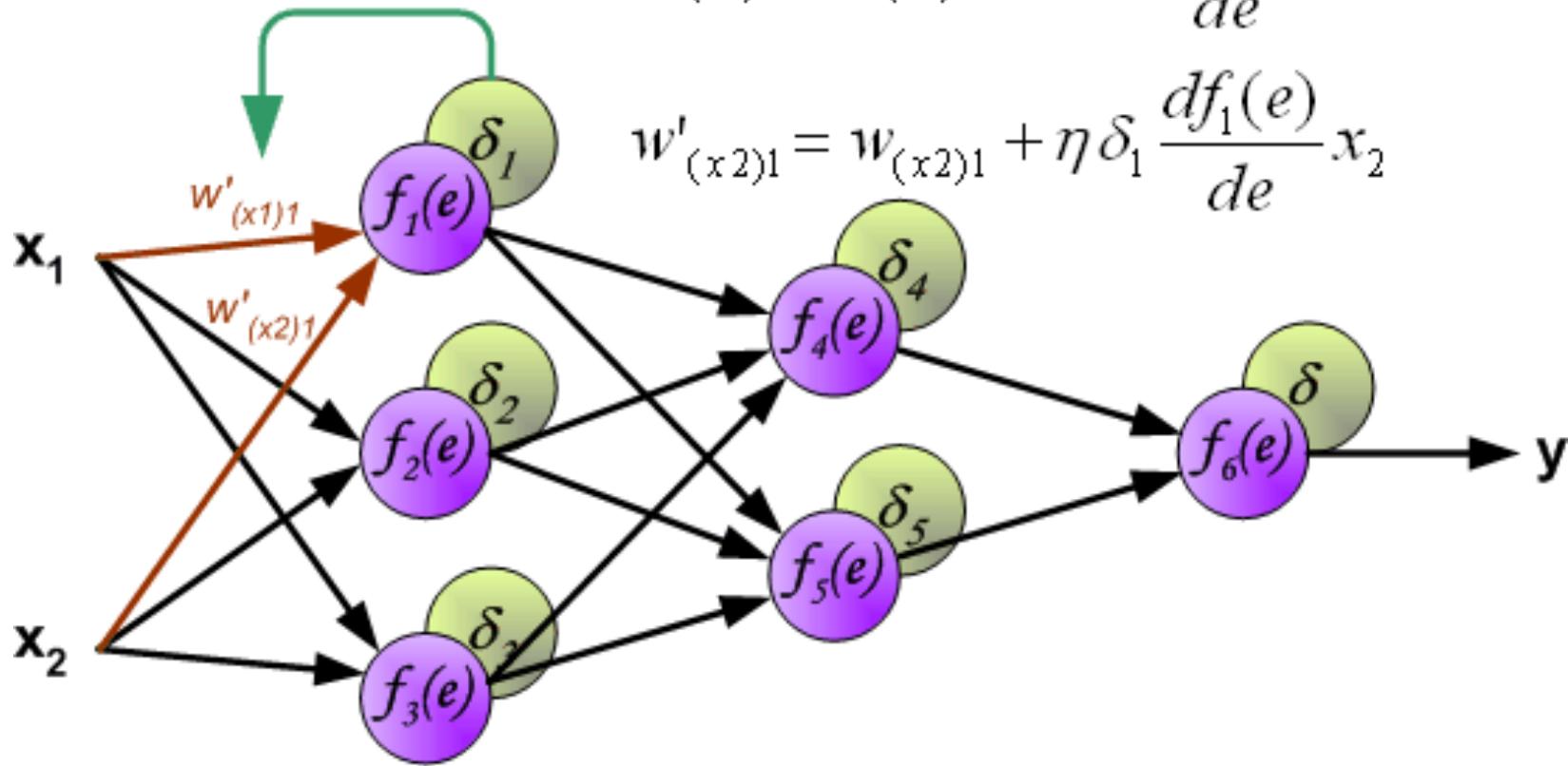




...

$$w'_{(x1)1} = w_{(x1)1} + \eta \delta_1 \frac{df_1(e)}{de} x_1$$

$$w'_{(x2)1} = w_{(x2)1} + \eta \delta_1 \frac{df_1(e)}{de} x_2$$

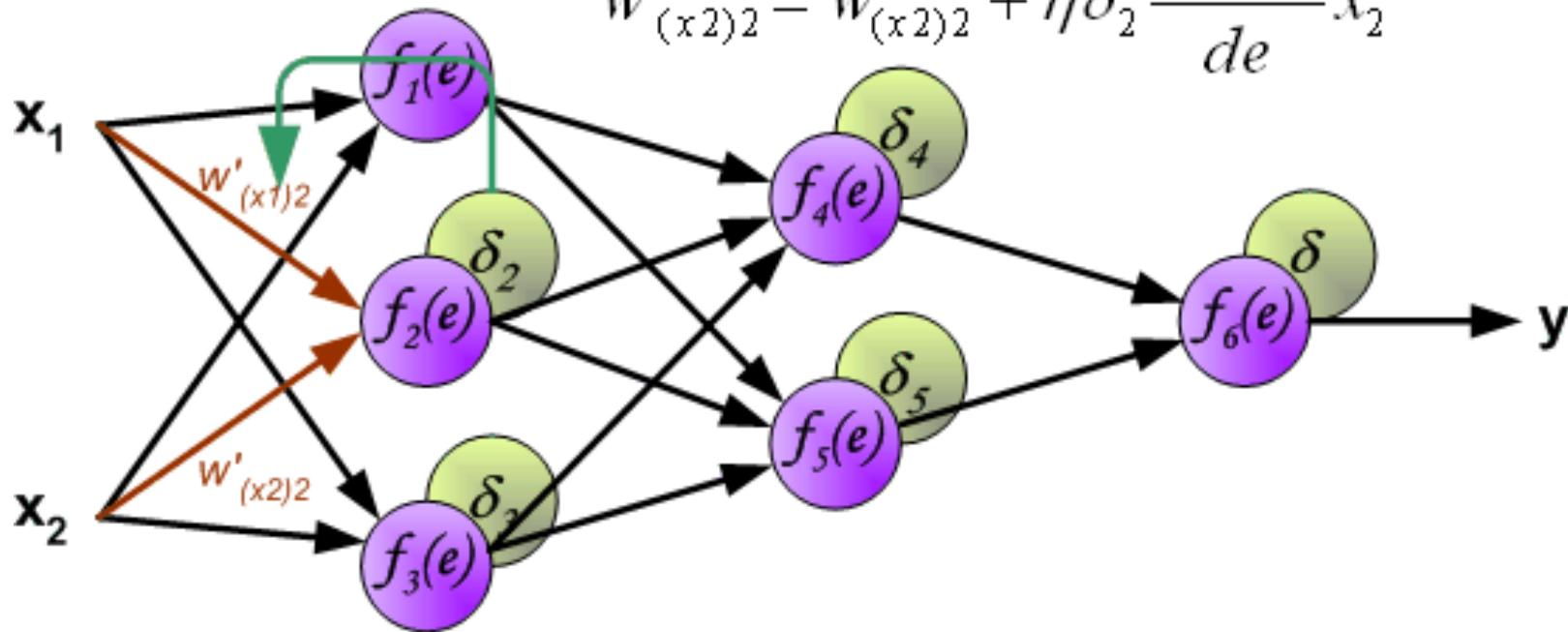




...

$$w'_{(x1)2} = w_{(x1)2} + \eta \delta_2 \frac{df_2(e)}{de} x_1$$

$$w'_{(x2)2} = w_{(x2)2} + \eta \delta_2 \frac{df_2(e)}{de} x_2$$

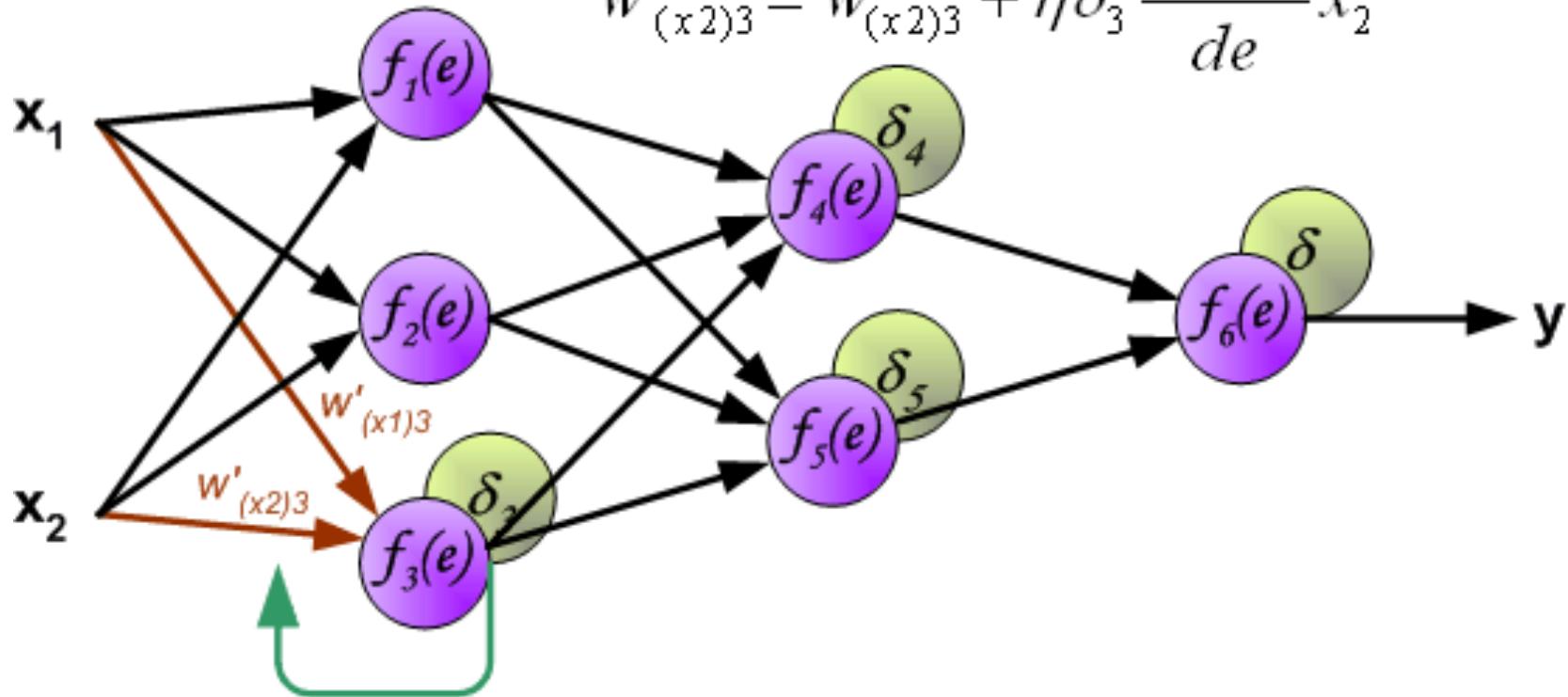




...

$$w'_{(x1)3} = w_{(x1)3} + \eta \delta_3 \frac{df_3(e)}{de} x_1$$

$$w'_{(x2)3} = w_{(x2)3} + \eta \delta_3 \frac{df_3(e)}{de} x_2$$



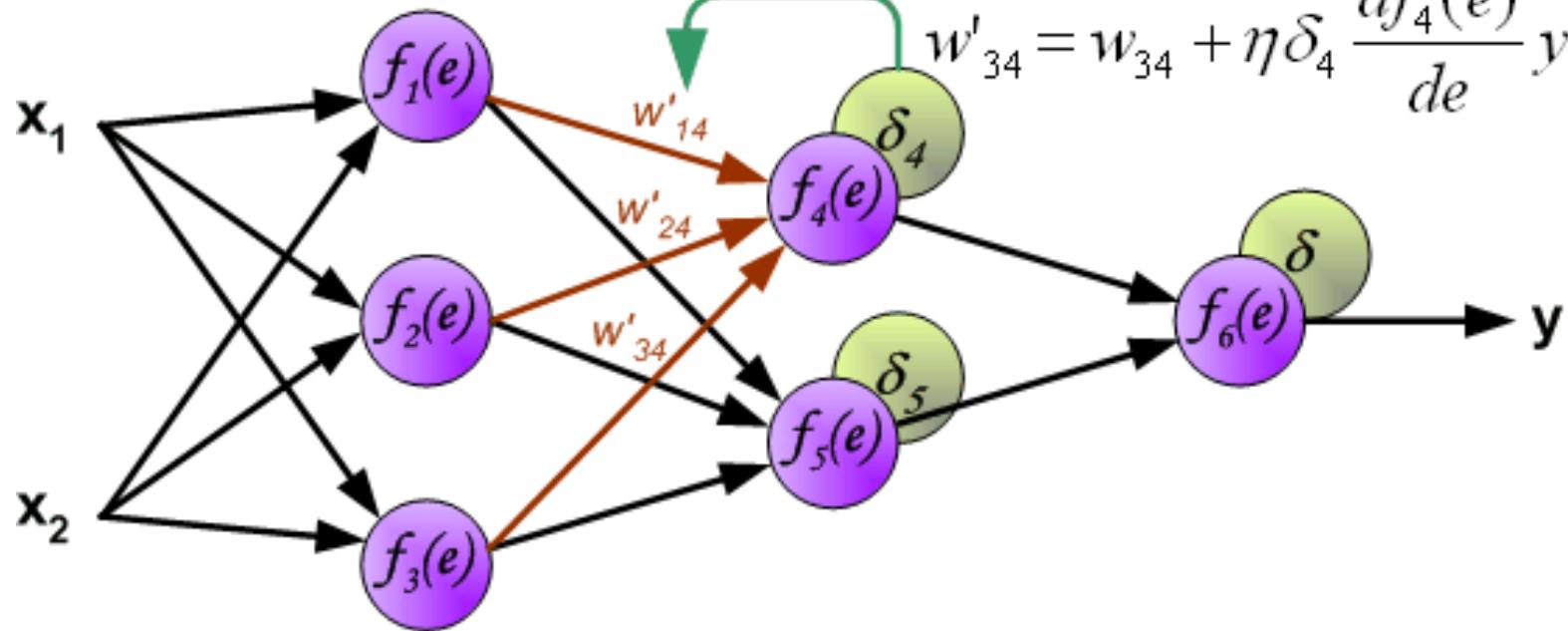


...

$$w'_{14} = w_{14} + \eta \delta_4 \frac{df_4(e)}{de} y_1$$

$$w'_{24} = w_{24} + \eta \delta_4 \frac{df_4(e)}{de} y_2$$

$$w'_{34} = w_{34} + \eta \delta_4 \frac{df_4(e)}{de} y_3$$



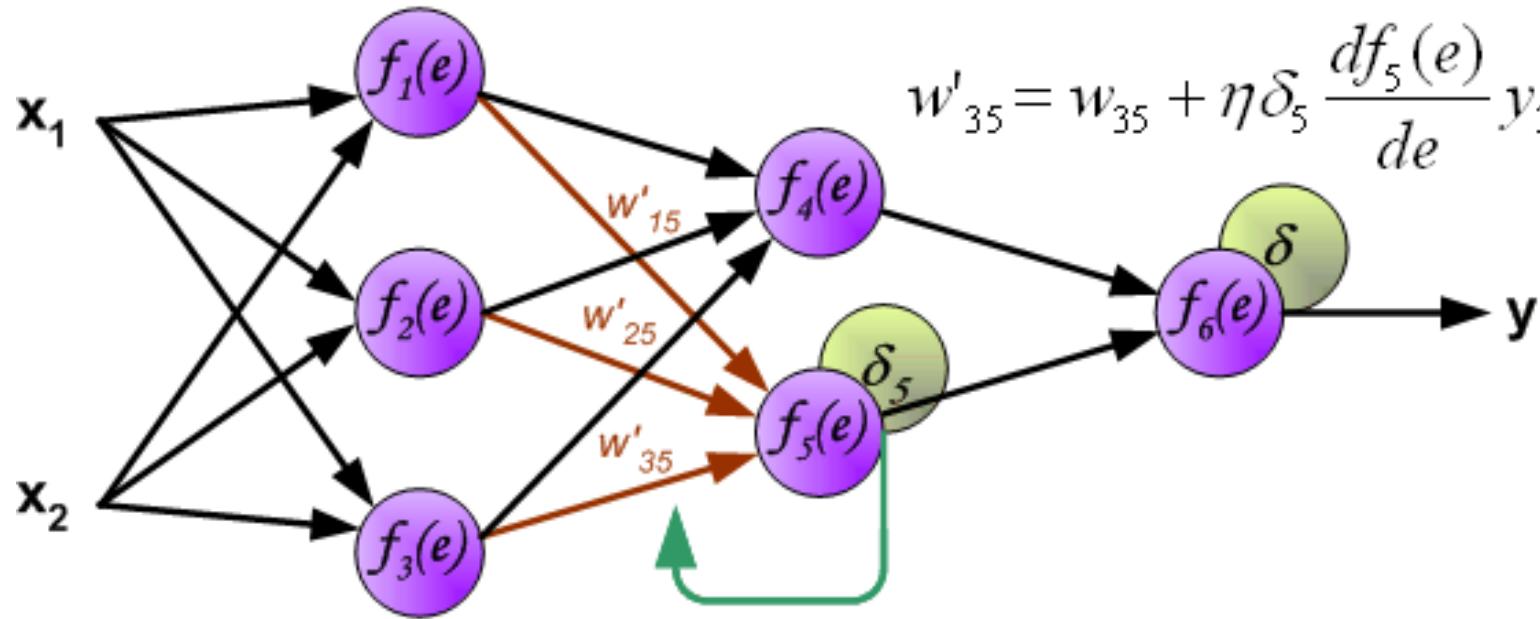


...

$$w'_{15} = w_{15} + \eta \delta_5 \frac{df_5(e)}{de} y_1$$

$$w'_{25} = w_{25} + \eta \delta_5 \frac{df_5(e)}{de} y_2$$

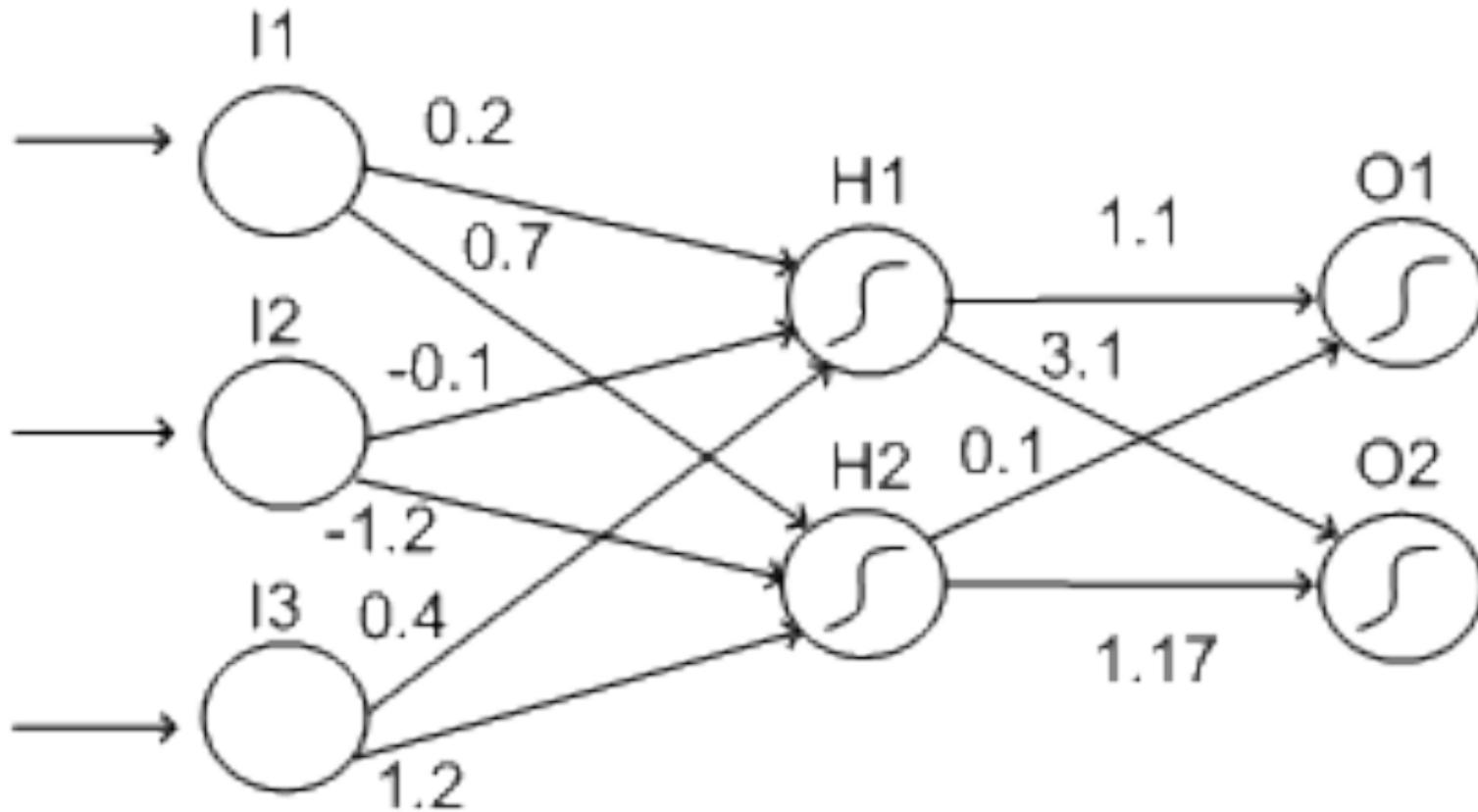
$$w'_{35} = w_{35} + \eta \delta_5 \frac{df_5(e)}{de} y_3$$







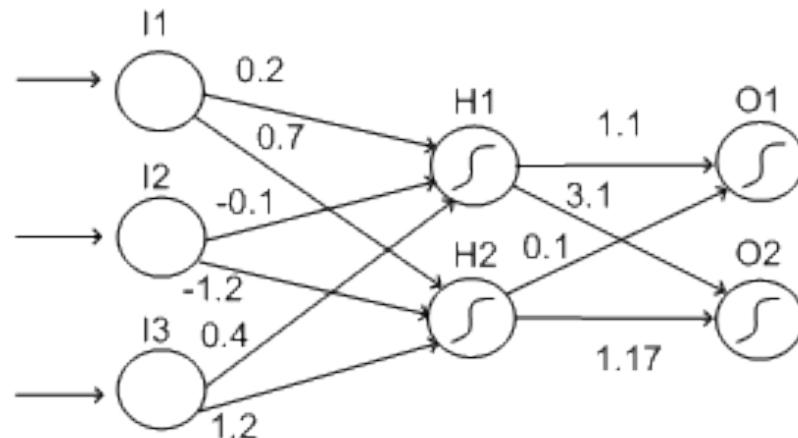
MLP Örnek



Örnekleri ileri sürmek

Ağa sunulan örnekler: E(10,30,20)

- Gizli katmanlarda ağırlıklı toplamlar:
 - $S_{H1} = (0.2*10) + (-0.1*30) + (0.4*20) = 2-3+8 = 7$
 - $S_{H2} = (0.7*10) + (-1.2*30) + (1.2*20) = 7-6+24 = -5$
- Bir sonraki adımda gizli katman çıkışları hesaplanır:
 - $\sigma(S) = 1/(1 + e^{-S})$
 - $\sigma(S_{H1}) = 1/(1 + e^{-7}) = 1/(1+0.000912) = 0.999$
 - $\sigma(S_{H2}) = 1/(1 + e^5) = 1/(1+148.4) = 0.0067$





- Ağırlıklı toplamları çıkış katmanında hesapla:

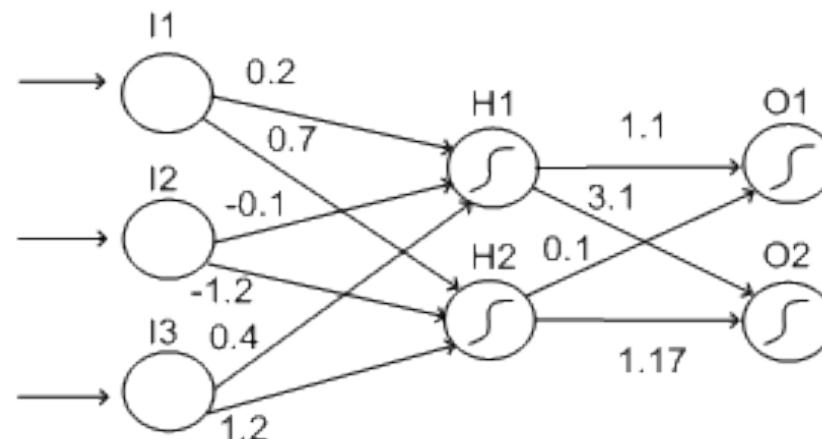
- $S_{O1} = (1.1 * 0.999) + (0.1 * 0.0067) = 1.0996$

- $S_{O2} = (3.1 * 0.999) + (1.17 * 0.0067) = 3.1047$

- Son olarak ANN çıkışını hesapla:

- $\sigma(S_{O1}) = 1/(1+e^{-1.0996}) = 1/(1+0.333) = 0.750$

- $\sigma(S_{O2}) = 1/(1+e^{-3.1047}) = 1/(1+0.045) = 0.957$

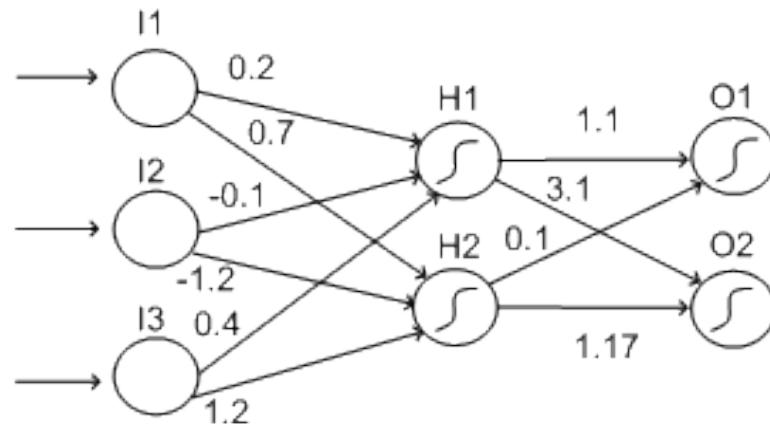




Örnekleri ileri doğru sürdürüğümüzde elde edilen hesaplamalar

Ağa sunulan örnekler
Öğrenme oranı= 0.1

| Input units | | Hidden units | | | Output units | | |
|-------------|--------|--------------|--------------------|--------|--------------|--------------------|--------|
| Unit | Output | Unit | Weighted Sum Input | Output | Unit | Weighted Sum Input | Output |
| I1 | 10 | H1 | 7 | 0.999 | O1 | 1.0996 | 0.750 |
| I2 | 30 | H2 | -5 | 0.0067 | O2 | 3.1047 | 0.957 |
| I3 | 20 | | | | | | |



Çıkış Birimlerinin Hata Değerleri

- $o_1(E)=0.750, o_2(E)=0.957$

Ağdaki her k çıkış birimi için, bu çıkış biriminin δ_k hata terimini hesapla.

- $\delta_k = o_k(1 - o_k)(t_k - o_k)$

$$\delta_{O1} = o_1(E)(1 - o_1(E))(t_1(E) - o_1(E)) = 0.750(1-0.750)(1-0.750) = 0.0469$$

$$\delta_{O2} = o_2(E)(1 - o_2(E))(t_2(E) - o_2(E)) = 0.957(1-0.957)(0-0.957) = -0.0394$$

Gizli Birimler için Hata Değerleri

ağdaki her h gizli katmanı için, gizli birim δ_k hatasını hesapla

$$\circ \quad \delta_h = o_h(1 - o_h) \sum_{k \in outputs} w_{kh} \delta_k$$

- $\delta_{O1} = 0.0469$ ve $\delta_{O2} = -0.0394$
- $h_1(E) = 0.999$ ve $h_2(E) = 0.0067$
- H1 için:
 - $(w_{11} * \delta_{O1}) + (w_{12} * \delta_{O2}) = (1.1 * 0.0469) + (3.1 * -0.0394) = -0.0706$
 - Ve bu değeri, $h_1(E)(1-h_1(E))$ ile çarpalım, sonuç:
 - $-0.0706 * (0.999 * (1-0.999)) = 0.0000705 = \delta_{H1}$
- H2 için:
 - $(w_{21} * \delta_{O1}) + (w_{22} * \delta_{O2}) = (0.1 * 0.0469) + (1.17 * -0.0394) = -0.0414$
 - Ve bu değeri, $h_2(E)(1-h_2(E))$ ile çarpalım, sonuç:
 - $-0.0414 * (0.067 * (1-0.067)) = -0.00259 = \delta_{H2}$

Ağırlık güncellemleri için hesaplamalar

- Giriş ve gizli katman arasındaki ağırlık değerleri:

| Input unit | Hidden unit | η | δ_H | x_i | $\Delta = \eta * \delta_H * x_i$ | Old weight | New weight |
|------------|-------------|--------|------------|-------|----------------------------------|------------|------------|
| I1 | H1 | 0.1 | -0.0000705 | 10 | -0.0000705 | 0.2 | 0.1999295 |
| I1 | H2 | 0.1 | -0.00259 | 10 | -0.00259 | 0.7 | 0.69741 |
| I2 | H1 | 0.1 | -0.0000705 | 30 | -0.0002115 | -0.1 | -0.1002115 |
| I2 | H2 | 0.1 | -0.00259 | 30 | -0.00777 | -1.2 | -1.20777 |
| I3 | H1 | 0.1 | -0.0000705 | 20 | -0.000141 | 0.4 | 0.39999 |
| I3 | H2 | 0.1 | -0.00259 | 20 | -0.00518 | 1.2 | 1.1948 |

her ağ bağlantısı w_{ji} 'yi güncelle

- $w_{ji} = w_{ji} + \Delta w_{ji}$
- $\Delta w_{ji} = \eta \delta_j x_{ji}$

- Gizli ve çıkış katmanı arasındaki ağırlık değerleri:

| Hidden unit | Output unit | η | δ_O | $h_i(E)$ | $\Delta = \eta * \delta_O * h_i(E)$ | Old weight | New weight |
|-------------|-------------|--------|------------|----------|-------------------------------------|------------|------------|
| H1 | O1 | 0.1 | 0.0469 | 0.999 | 0.000469 | 1.1 | 1.100469 |
| H1 | O2 | 0.1 | -0.0394 | 0.999 | -0.00394 | 3.1 | 3.0961 |
| H2 | O1 | 0.1 | 0.0469 | 0.0067 | 0.00314 | 0.1 | 0.10314 |
| H2 | O2 | 0.1 | -0.0394 | 0.0067 | -0.0000264 | 1.17 | 1.16998 |

her ağ bağlantısı w_{ji} 'yi güncelle

- $w_{ji} = w_{ji} + \Delta w_{ji}$
- $\Delta w_{ji} = \eta \delta_j x_{ji}$

Referanslar

- T.M. Mitchell, Machine Learning, McGraw Hill, 1997.
- E.Alpaydin, Introduction to Machine Learning, MIT Press, 2010.
- Daniel T. Larose, Discovering Knowledge in Data, Wiley, 2005.
- <http://galaxy.agh.edu.pl/~vlsi/AI/backprop.html>