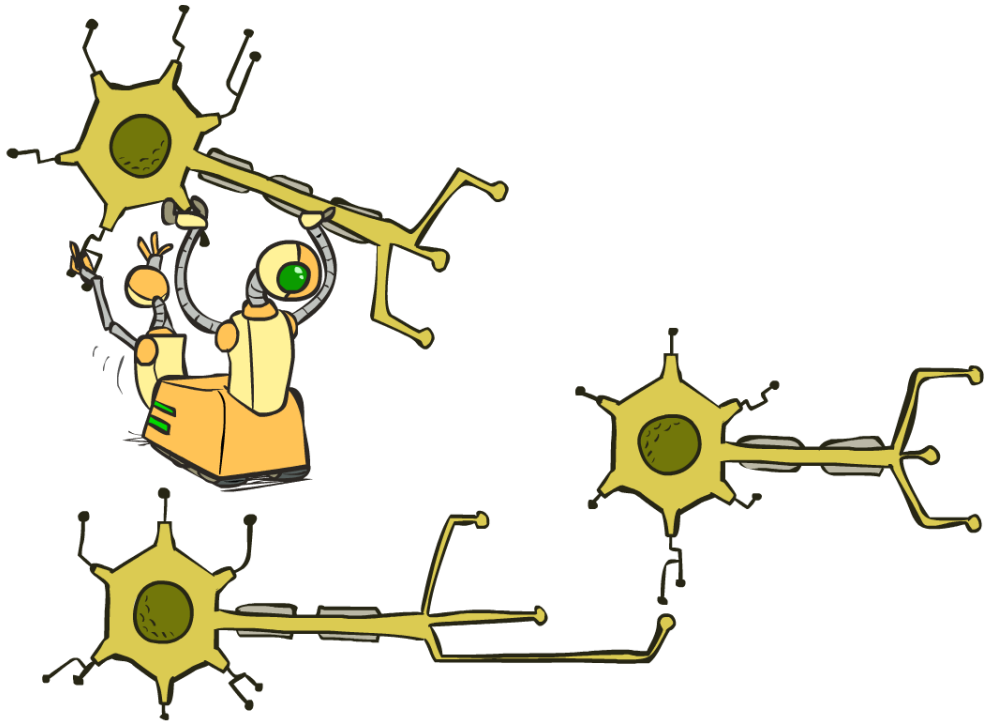
Yapay Zeka

Yapay Sinir Ağları

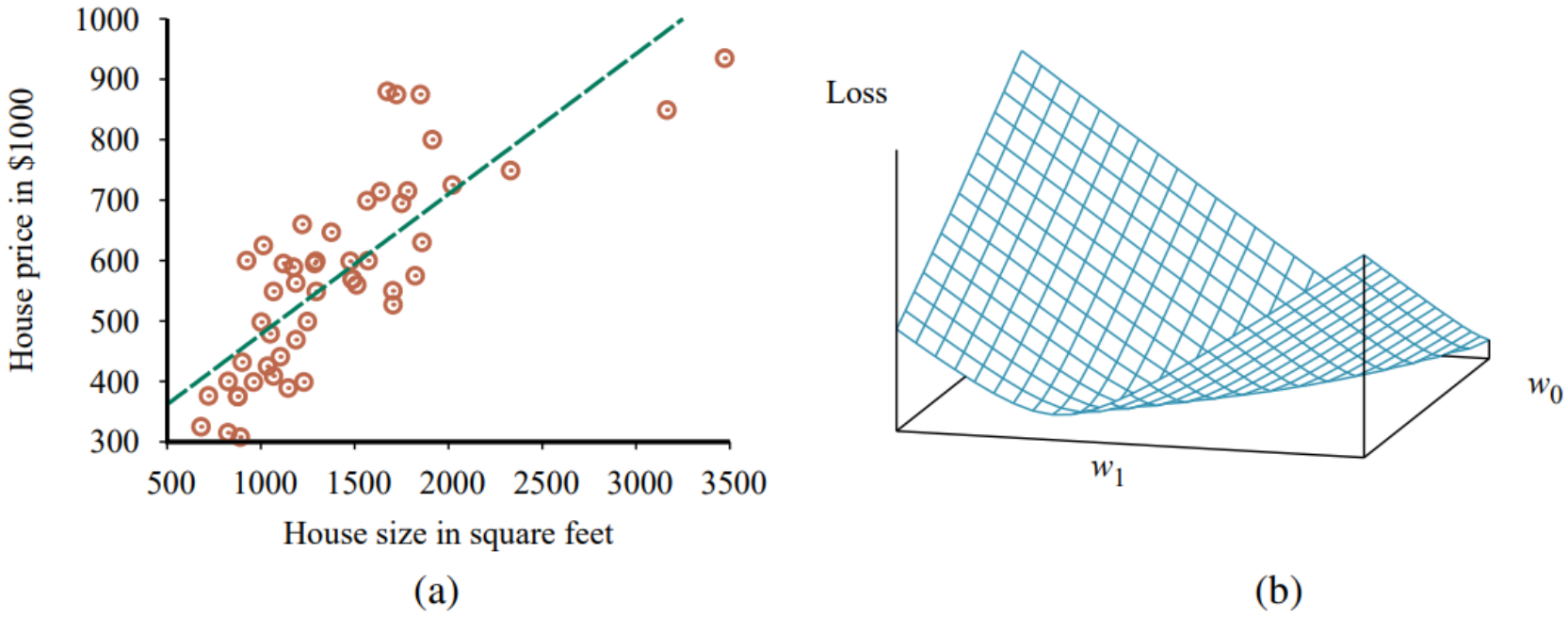


Prof. Sevinç İlhan Omurca / Dr. Öğretim Üyesi Fidan Kaya Gülağız

Kocaeli Üniversitesi

Doğrusal Modeller ile Regresyon ve Sınıflandırma

- $y = 0.232x + 246$
- Kayıp fonksiyonu: $\sum_j (y_j - w_1 x_j + w_0)^2$
- $w_1 = 0.232$, $w_0 = 246$



Tek Değişkenli Doğrusal Regresyon

- Hipotez: $h_{\mathbf{w}}(x) = w_1 x + w_0$
- Kayıp Fonksiyonu: $Loss(h_{\mathbf{w}}) = \sum_{j=1}^{N} L_2(y_j, h_{\mathbf{w}}(x_j)) = \sum_{j=1}^{N} (y_j - h_{\mathbf{w}}(x_j))^2 = \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2$
- Optimum Hipotez: $\mathbf{w}^* = \text{argmin}_{\mathbf{w}} \, Loss(h_{\mathbf{w}})$
- Çözüm: $\frac{\partial}{\partial w_0} \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 = 0$ and $\frac{\partial}{\partial w_1} \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 = 0$

$$w_1 = \frac{N(\sum x_j y_j) - (\sum x_j)(\sum y_j)}{N(\sum x_j^2) - (\sum x_j)^2}; \quad w_0 = (\sum y_j - w_1(\sum x_j))/N$$

- Gradyan Azalma: $\mathbf{w} \leftarrow$ any point in the parameter space

  loop until convergence **do**

  for each w_i **in w do**

  $$w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss(\mathbf{w})$$

Tek Değişkenli Doğrusal Regresyon

- Yalnızca bir eğitim örneği (x,y) ile çözüm:

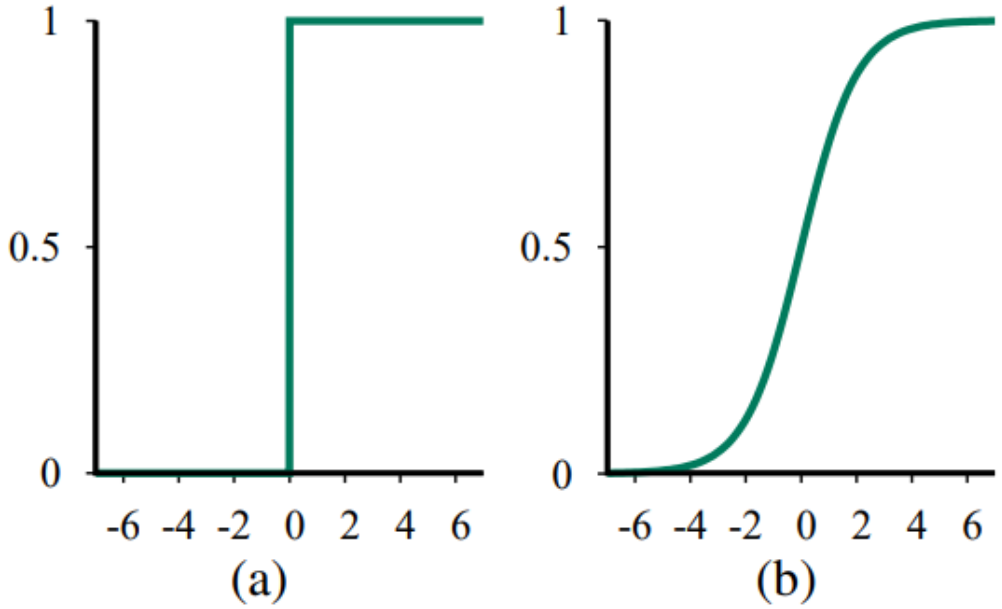
$$\begin{aligned}\frac{\partial}{\partial w_i} Loss(\mathbf{w}) &= \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(x))^2 \\ &= 2(y - h_{\mathbf{w}}(x)) \times \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(x)) \\ &= 2(y - h_{\mathbf{w}}(x)) \times \frac{\partial}{\partial w_i}(y - (w_1 x + w_0)) \,,\end{aligned}$$

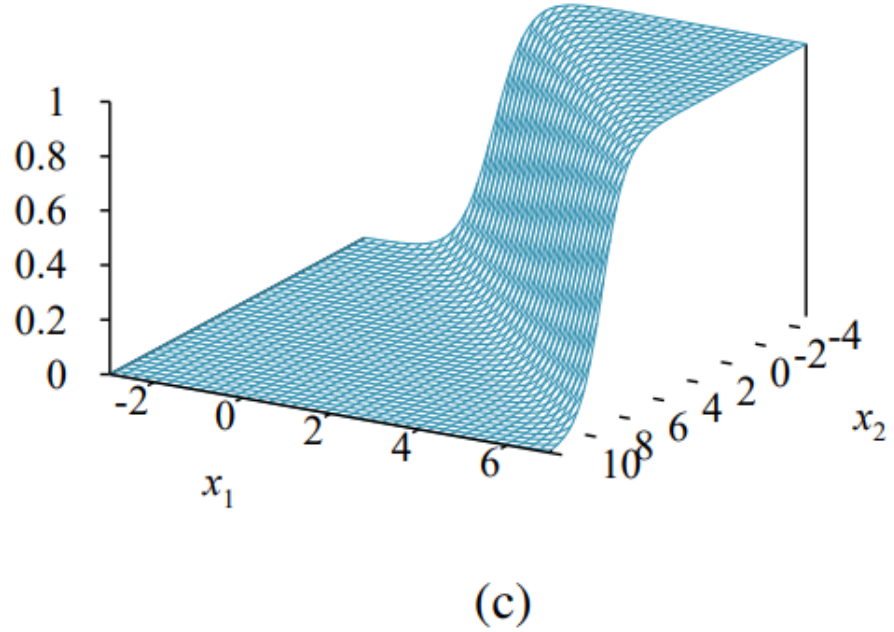
$$\frac{\partial}{\partial w_0} Loss(\mathbf{w}) = -2(y - h_{\mathbf{w}}(x)) \,; \qquad \frac{\partial}{\partial w_1} Loss(\mathbf{w}) = -2(y - h_{\mathbf{w}}(x)) \times x$$

- Ağırlık Güncelleme Kuralı: $w_0 \leftarrow w_0 + \alpha\,(y - h_{\mathbf{w}}(x)) \,; \quad w_1 \leftarrow w_1 + \alpha\,(y - h_{\mathbf{w}}(x)) \times x$

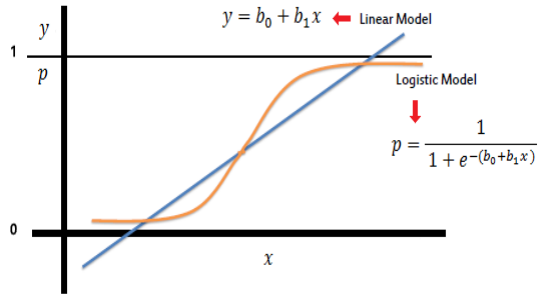
- N adet eğitim verisi için: $w_0 \leftarrow w_0 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j)) \,; \quad w_1 \leftarrow w_1 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j)) \times x_j$

Lojistik Regresyon ile Doğrusal Sınıflandırma





$$h_{\mathbf{w}}(\mathbf{x}) = Logistic(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}}$$



https://www.saedsayad.com/logistic_regression.htm

Lojistik Regresyon ile Doğrusal Sınıflandırma

$$h_{\mathbf{w}}(\mathbf{x}) = Logistic(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}}$$

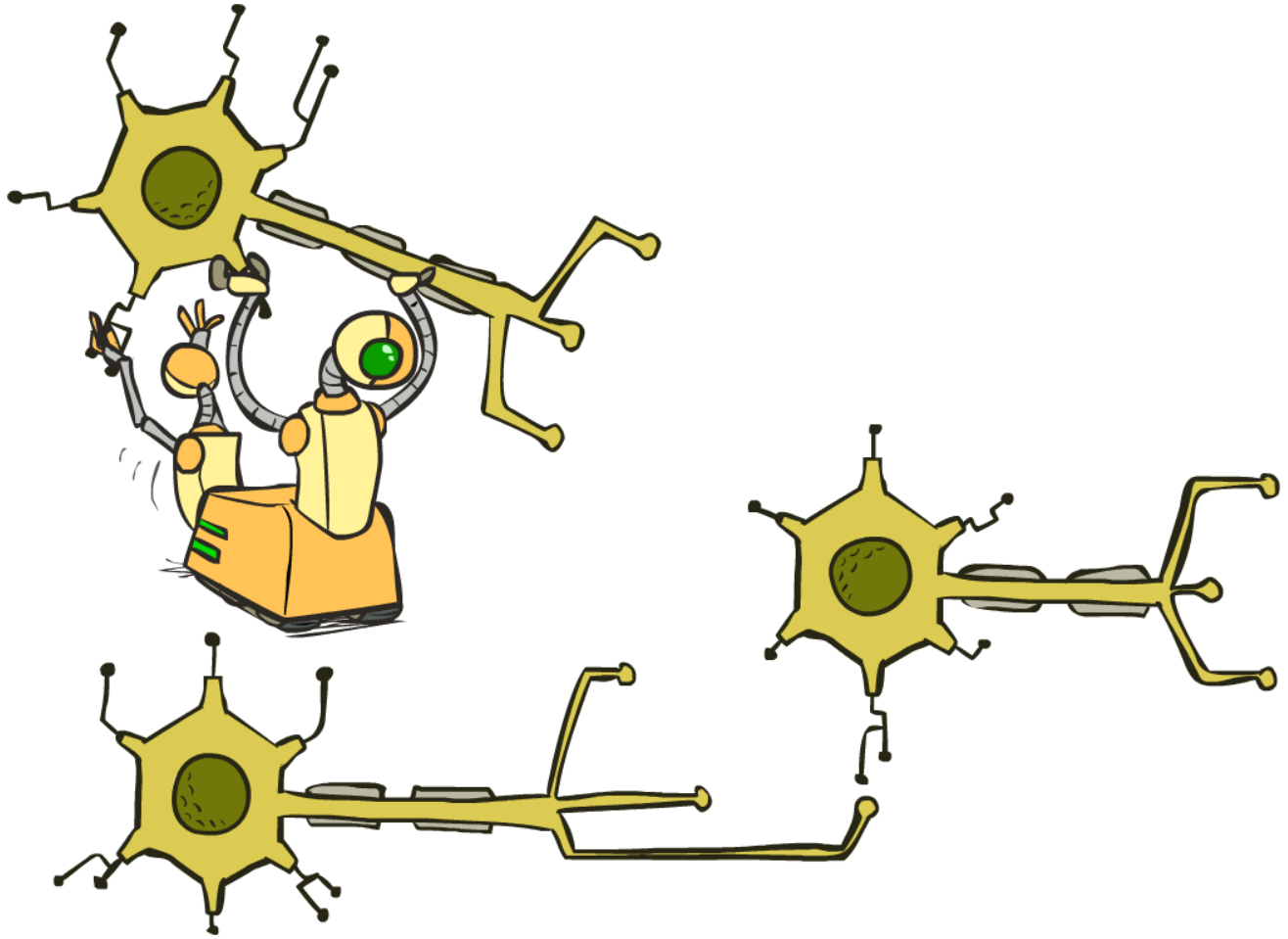
$$\begin{aligned}
\frac{\partial}{\partial w_i} Loss(\mathbf{w}) &= \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(\mathbf{x}))^2 \\
&= 2(y - h_{\mathbf{w}}(\mathbf{x})) \times \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(\mathbf{x})) \\
&= -2(y - h_{\mathbf{w}}(\mathbf{x})) \times g'(\mathbf{w} \cdot \mathbf{x}) \times \frac{\partial}{\partial w_i}\mathbf{w} \cdot \mathbf{x} \\
&= -2(y - h_{\mathbf{w}}(\mathbf{x})) \times g'(\mathbf{w} \cdot \mathbf{x}) \times x_i \ .
\end{aligned}$$

- Lojistik fonksiyonun türevi g' : $g'(z) = g(z)(1 - g(z))$

$$g'(\mathbf{w} \cdot \mathbf{x}) = g(\mathbf{w} \cdot \mathbf{x})(1 - g(\mathbf{w} \cdot \mathbf{x})) = h_{\mathbf{w}}(\mathbf{x})(1 - h_{\mathbf{w}}(\mathbf{x}))$$

- Kaybı en aza indirgeyecek ağırlık güncellemesi: $w_i \leftarrow w_i + \alpha\,(y - h_{\mathbf{w}}(\mathbf{x})) \times h_{\mathbf{w}}(\mathbf{x})(1 - h_{\mathbf{w}}(\mathbf{x})) \times x_i$

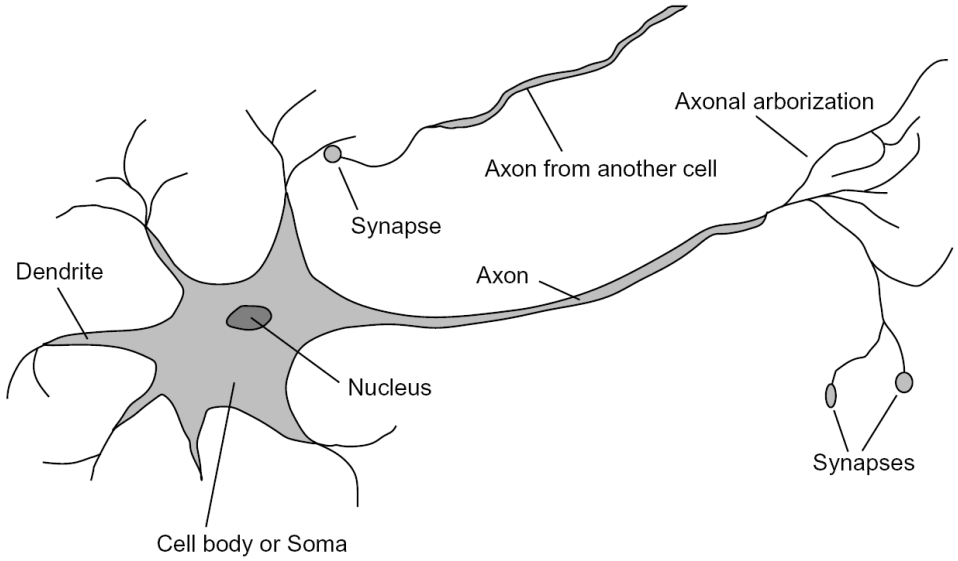
Yapay Sinir Ağları

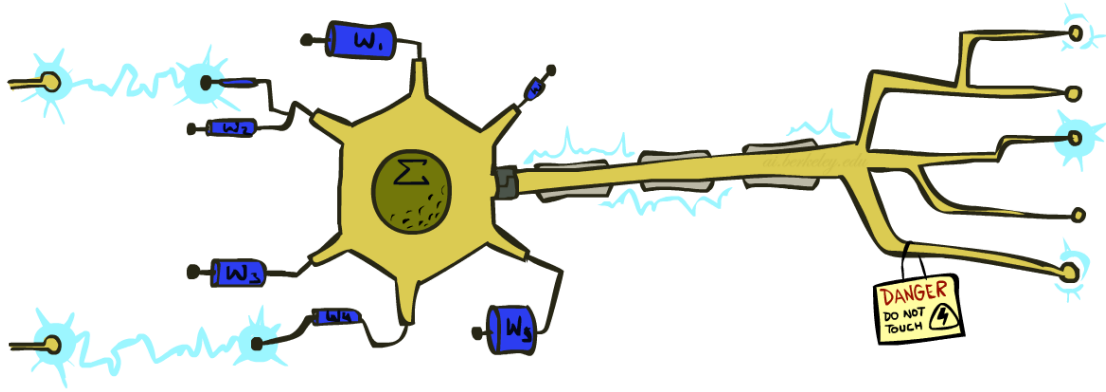


Biyolojik Sinir Hücresi

- Yapay sinir hücreleri, gerçek sinir hücrelerinin simule edilmesiyle gerçekleştirilir

- Biyolojik sinir hücresi:
  - Gövde
  - Akson
  - Sinir ucu (dendrit)
  - sinaps

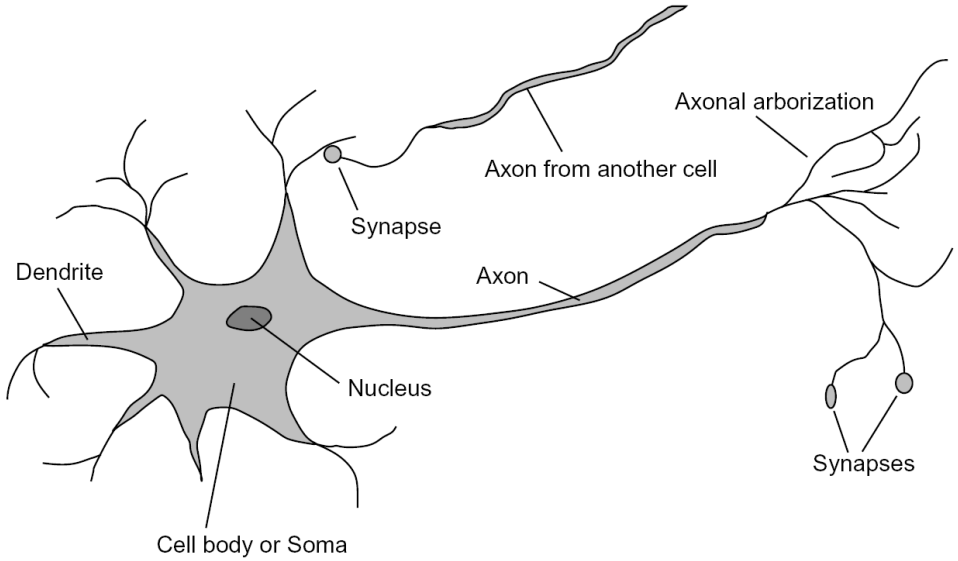


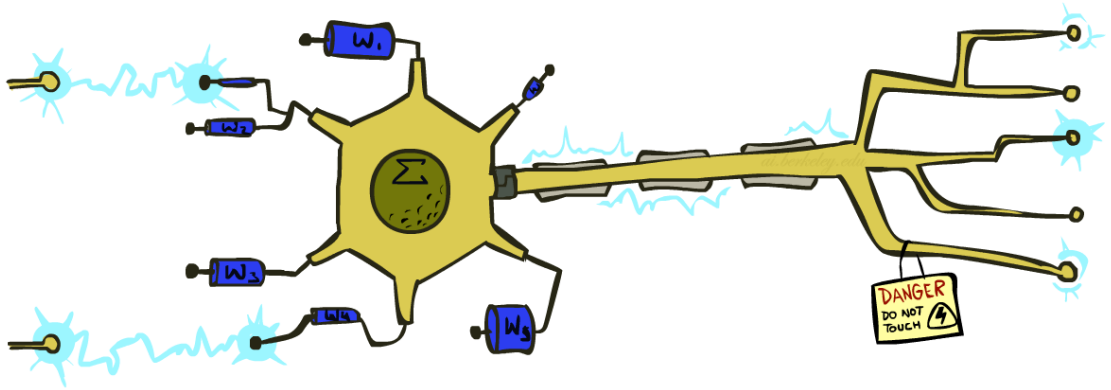


Biyolojik vs. Yapay Sinir Hücresi

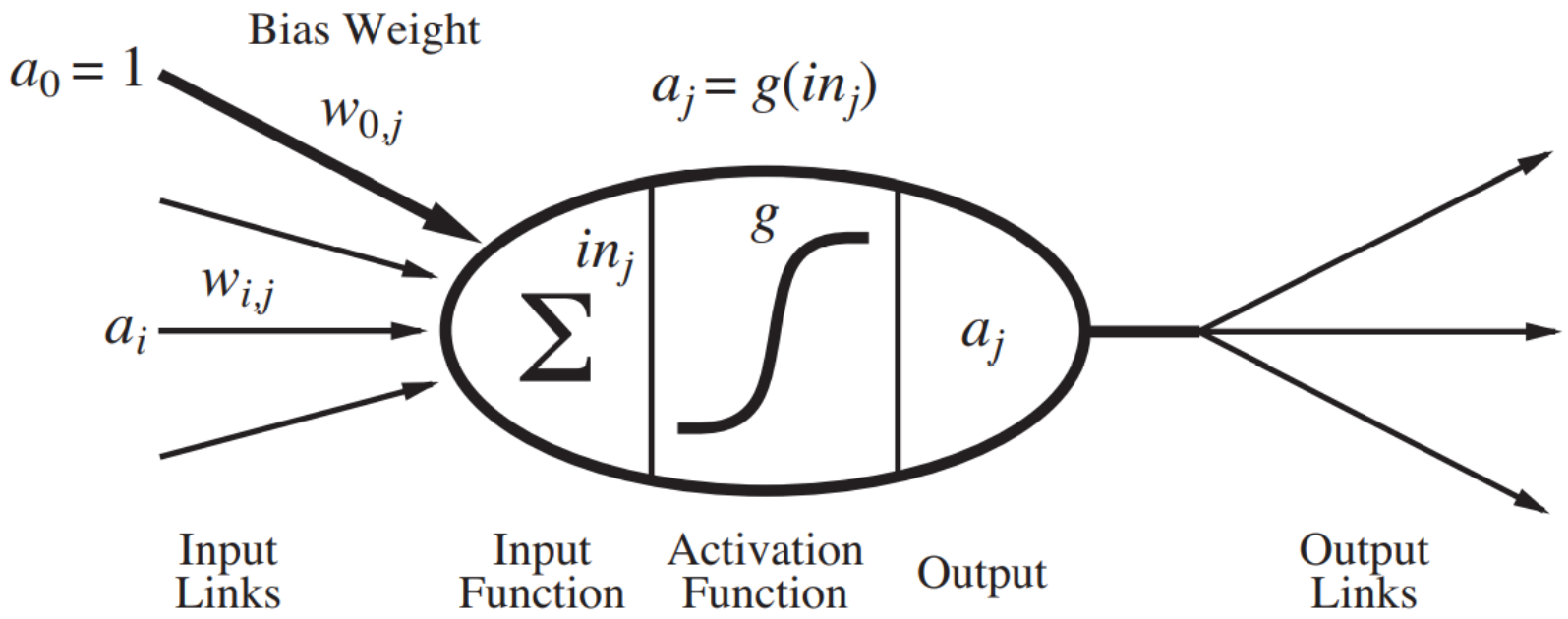
- Biyolojik sinir hücresi:
  - Akson
  - Sinir ucu (dendrit)
  - Çekirdek
  - Sinaps

- Yapay sinir hücresi:
  - Çıktı
  - Toplama fonksiyonu
  - Aktivasyon Fonksiyonu
  - Ağırlıklar





Perceptron

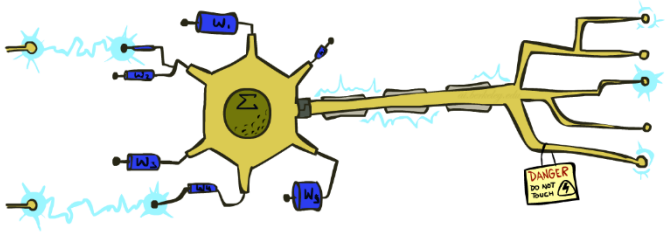


$$in_j = \sum_{i=0}^{n} w_{i,j} a_i$$

$$a_j = g(in_j) = g\left(\sum_{i=0}^{n} w_{i,j} a_i\right)$$

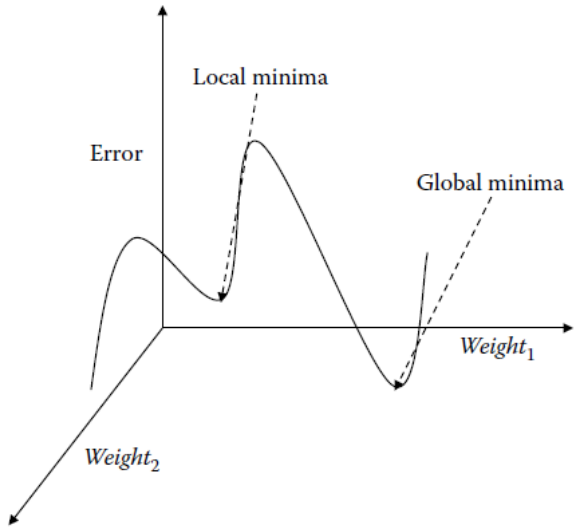
YSA Algoritması Özellikleri

- Örnekler nitelikler ile temsil edilir.
- Hedef fonksiyonu özellik vektörü ile tanımlanabilir.
- Hedef fonksiyon:
  - Ayrık değerli
  - Reel değerli
  - Ayrık ve reel değerli
- YSA öğrenme metotları eğitim verisindeki gürültülere dayanıklıdır
- Uzun süren eğitim zamanları olabilir.
  - Test süreci hızlıdır.
- YSA tarafından öğrenilen ağırlıkların yorumlanması zordur.
- **BİLGİ: Ağın bağlantı ağırlıkları ile temsil edilir.**



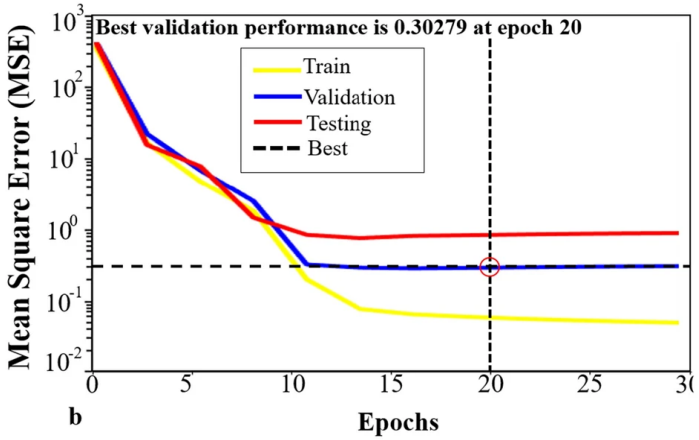
YSA Özellikleri (Hata Fonksiyonu)

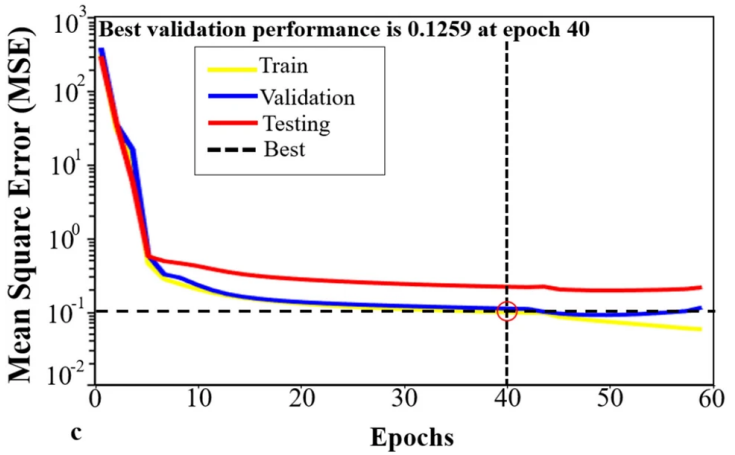
- İyi bir YSA minimum hataya sahiptir.
- İyi bir öğrenme fonksiyonu YSA'yı yüksek hatalı bir konfigürasyondan düşük hatalıya taşır.
- YSA'daki hata, hata ölçüm fonksiyonları ile ölçülür.
- Problemin tipine ve tasarımcının seçimine göre farklı hata fonksiyonları kullanılabilir.
  - Mean absolute error
  - Mean squared error
  - Sum squared error
- ANN'deki hata, çıkış değerlerinin ve bias değerinin çıkışa yansımasıdır.

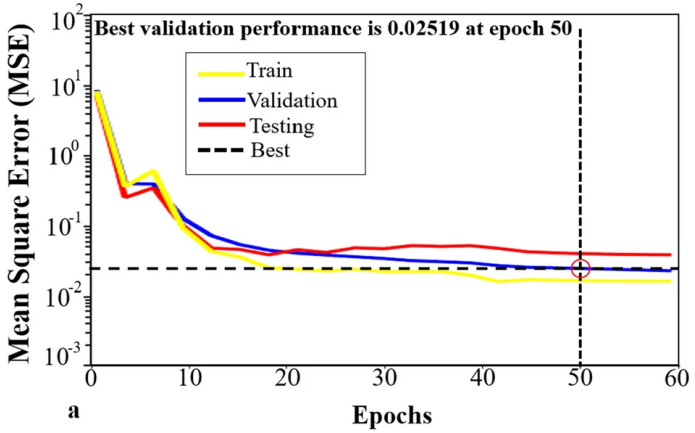


YSA Özellikleri (Epoch)

- Epoch: tüm giriş verilerinin işlenmesindeki tek bir iterasyon.
  - Her epoch'da YSA öğrenir.
  - Aynı girişleri çok kereler uygulamanın sonucu olarak, sistem kendini az bir hata ile eğitebilir hale gelir.
  - Çok sayıda epoch ile sistem tam olarak eğitilmiş kabul edilir.
  - Epoch sistemin kendini eğitim verisine göre eğitebilmesi için gereklidir.



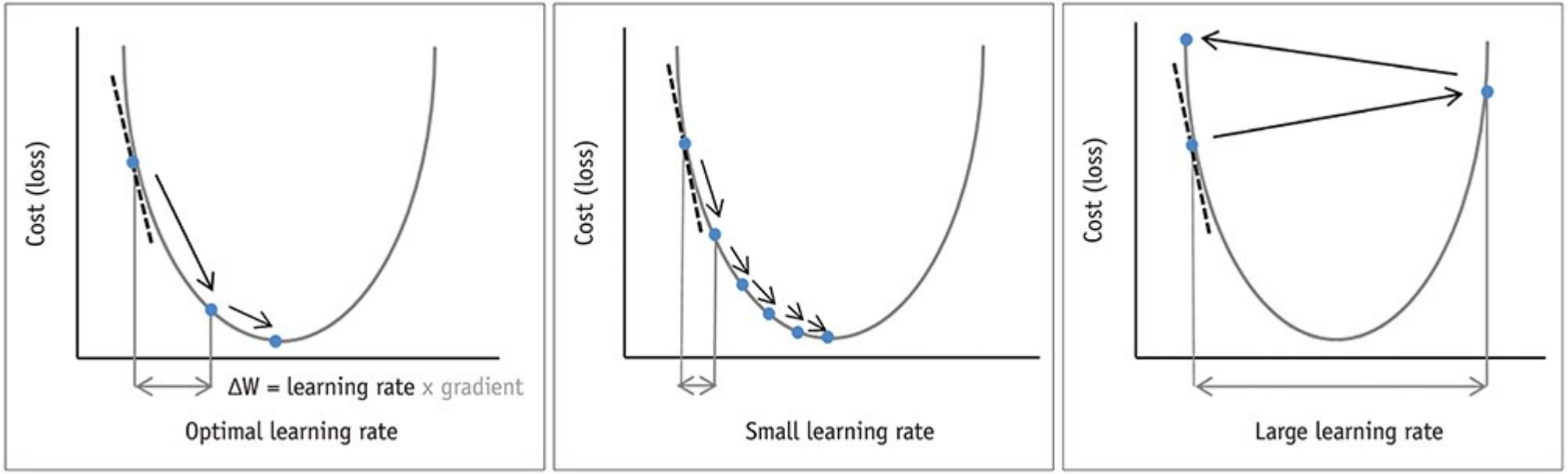




* https://www.nature.com/articles/s41598-020-73931-2/figures/5

YSA Özellikleri (Learning Rate)

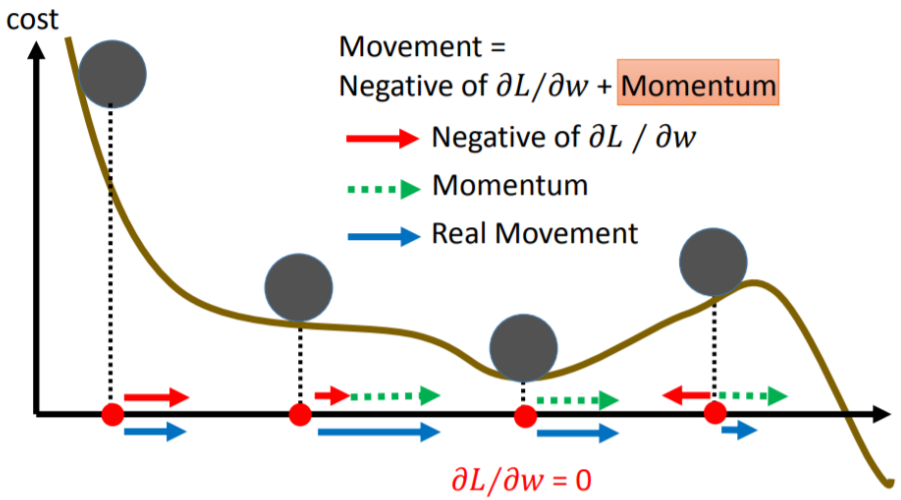
- Büyük öğrenme oranı: Sistem veriyi çok hızlı öğrenir, toplam hata artar.
- Düşük öğrenme oranı: Sistem çok yavaş öğrenir, eğitim zamanı artar ancak hata azalır.
- η ile temsil edilir.

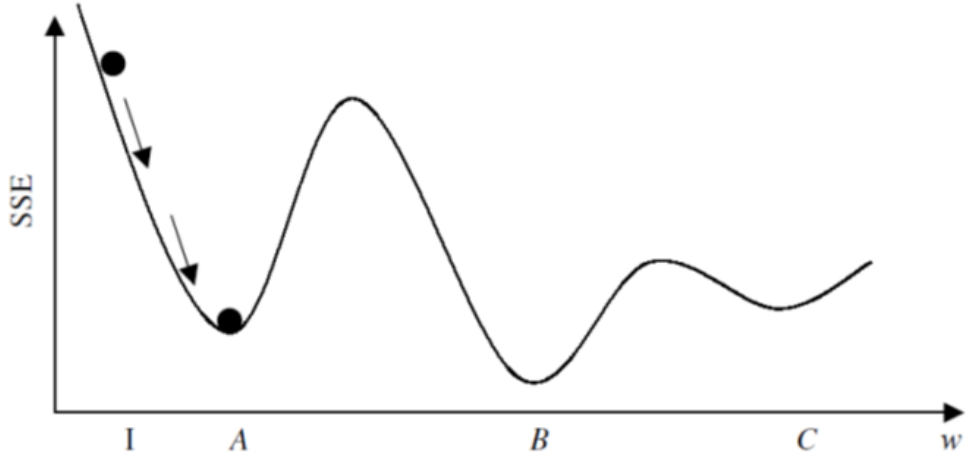


* https://www.kjronline.org/ViewImage.php?Type=F&aid=658625&id=F7&afn=68_KJR_21_1_33&fn=kjr-21-33-g007_0068KJR

YSA Özellikleri (Momentum)

- YSA her adımda daha az hata değerine sahip bir noktaya gelmek isteyecektir.
- Eğitim sırasında, YSA hatanın azaldığı yerde durmayı sürdürmek ister.
- Momentum öğrenme algoritmasını yerel minimumdan kaçacak şekilde önceki yönde tutmaya çalışır.
- En iyi sonuçlar alınmadan birçok deneme yapılmalıdır.
- A simgesi ile gösterilir
  - 0.6-0.8 aralığında değerler önerilir.



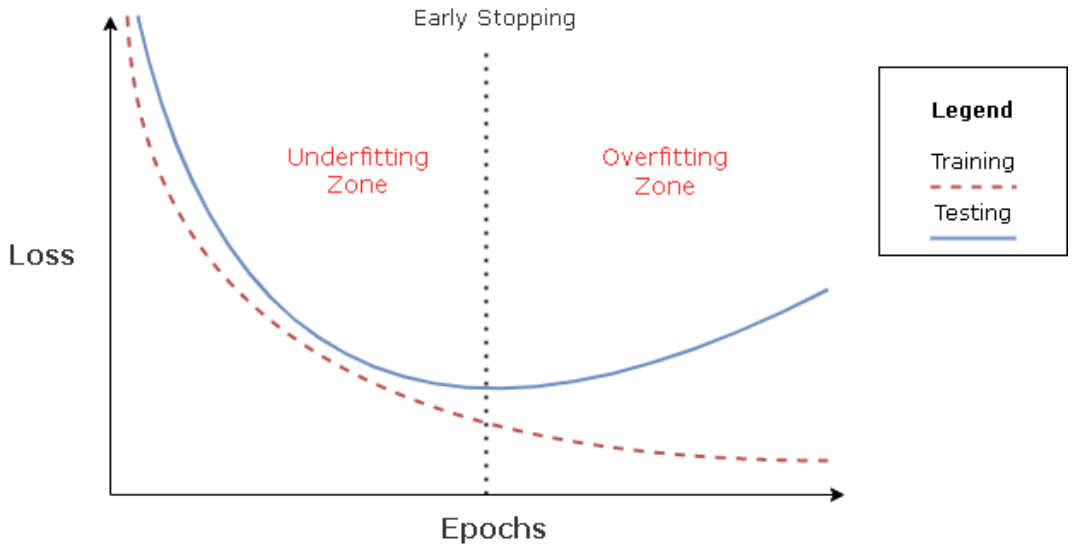


Small momentum α may cause algorithm to undershoot global minimum.

* https://ithelp.ithome.com.tw/articles/10236554

YSA Özellikleri (Durdurma Kriteri)

- **Durdurma kriteri:**
  - **Zaman** eşik değerine ulaştığında eğitim bitebilir
  - Önceden tanımlı **epoch** değerine ulaştığında eğitim bitebilir.
  - Önceden tanımlı **hata değeri**ne eriştiğinde eğitim bitebilir.
  - **İki epoch arasındaki hata değeri** azaldığında eğitim bitebilir.
    - Eğitim devam etse bile performansta çok fazla değişiklik olmayacağı hesap edilir.



* https://towardsdatascience.com/the-million-dollar-question-when-to-stop-training-deep-learning-models-fa9b488ac04d

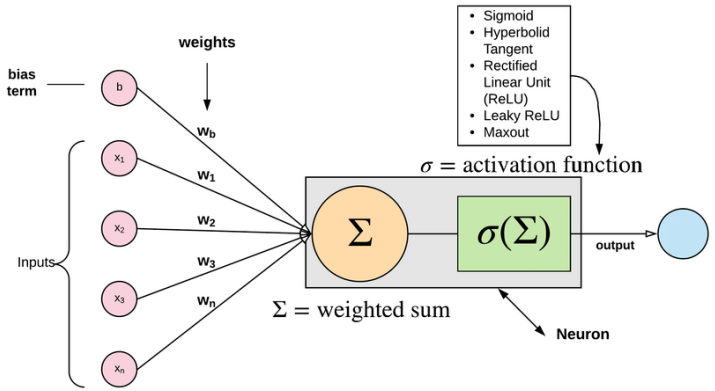
YSA – Örneklerin Ağa Sunulması

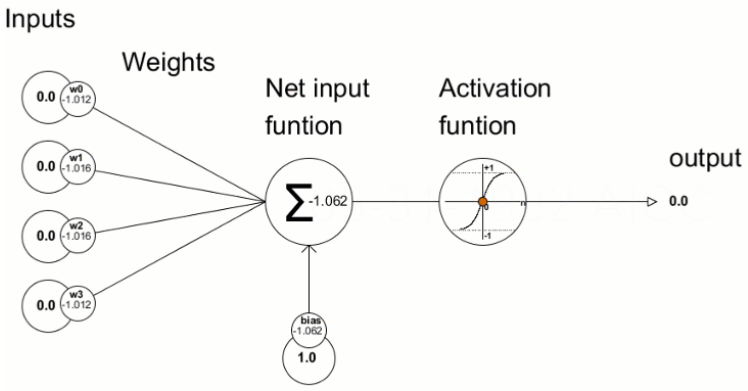
- Sıralı:
  - n örnek sırası ile ağa sunulurlar.
  - 1. iterasyonda 1. örnek,
  - 2. iterasyonda 2. örnek vs şeklinde devam eder.
  - Tüm örnekler sunulduktan sonra başa dönerek sırası ile yeniden ağa sunulurlar.
  - Bu işlem öğrenme sağlanana kadar devam eder.

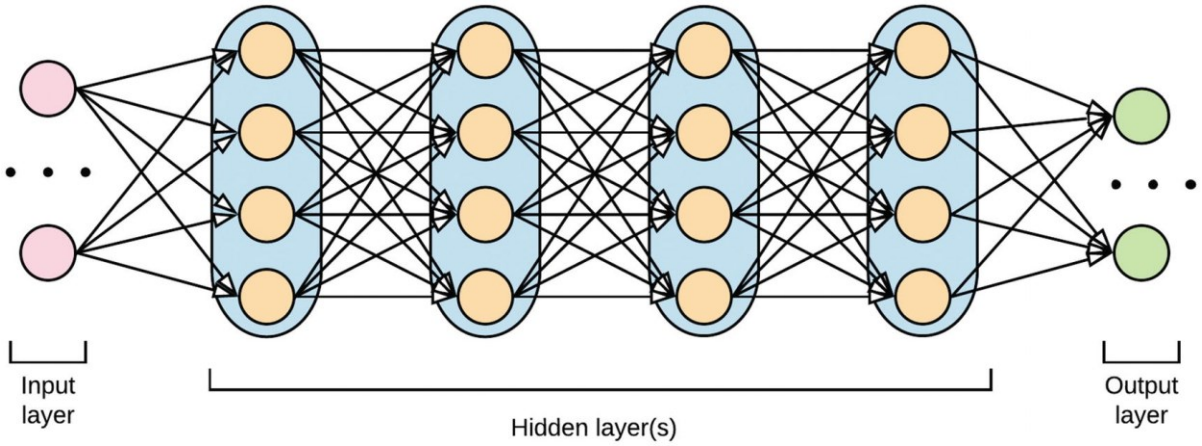
- Rastgele:
  - Örnekler eğitim kümesinden rastgele seçilirler.
  - Tüm örnekler rastgele ağa sunulduktan sonra, öğrenme sağlanan kadar yeniden sunulmaya devam ederler.

Multilayer Perceptron (MLP)

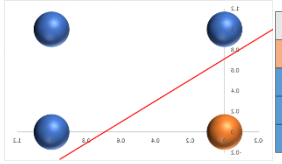
- YSA, yapay sinir hücrelerinin (perceptron) birbirlerine bağlanması sonucu oluşan yapılardır



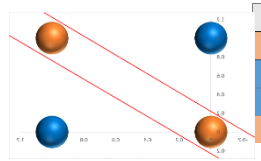




Bir MLP ağının genel mimarisi



A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1



A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

https://www.researchgate.net/figure/Activation-Functions-in-ANN-Source-Google-Images-2019_fig3_338014968
https://link.springer.com/chapter/10.1007/978-1-4842-4470-8_31
https://towardsdatascience.com/building-a-simple-neural-network-from-scratch-a5c6b2eb0c34
https://medium.com/@ugurcan.soruc/xor-problemi-nedir-ve-%C3%A7%C3%B6z%C3%BCm%C3%BC-197f9b110053

YSA – Katmanlar & Ağırlıklar

- YSA mimarisi katmanlardan kurulmuştur.
- Katmanların sayısı YSA'nın işlemsel karmaşıklığını ifade etmektedir.
- 3 çeşit katman vardır.
  - Giriş Katmanı
  - Gizli katmanlar
  - Çıkış katmanı

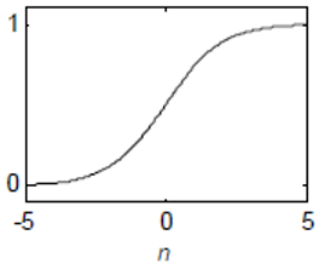
- ANN eğitimi ağırlıklardaki değişim demektir.
- Ağırlıkların farklı kombinasyonları ANN'nin farklı performanslarını temsil edecektir.
- Bias:
  - Sistem performansını etkiler
  - Bias da öğrenmede bir ağırlık gibi alınır
  - Giriş sinyallerinin toplamı 0 olduğunda öğrenme gerçekleşmez.
  - Çıkış değerleri hep 1 olan bias nöronları, nöronların giriş sinyallerinin sürekli sıfırdan farklı olmasını sağlarlar.
  - Öğrenmeyi hızlandırırken yerel optimum değerlere takılmayı güçleştirirler.

YSA – Aktivasyon Fonksiyonu

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

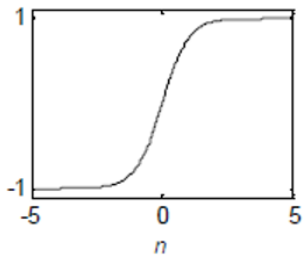
Sigmoid

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



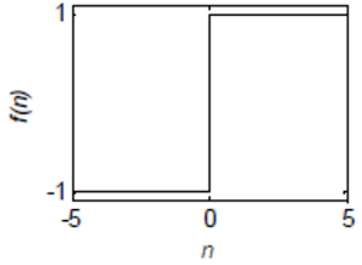
Hiperbolik tanjant

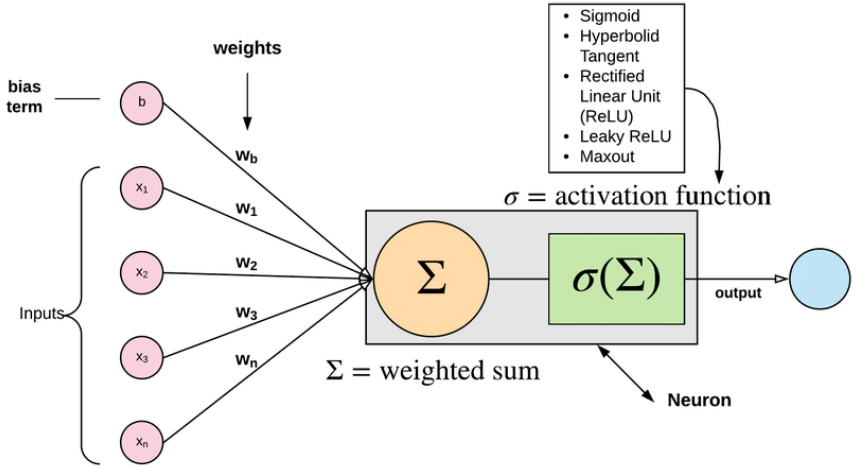
$$y = \frac{1-e^{-2net}}{1+e^{2net}}$$



Adım basamak

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



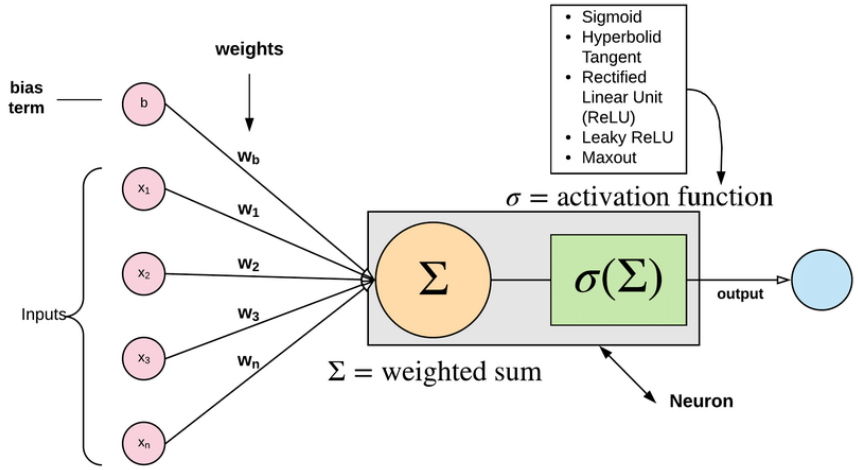


https://www.researchgate.net/figure/Activation-Functions-in-ANN-Source-Google-Images-2019_fig3_338014968

Aktivasyon Fonksiyonları

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	$f(x) = \begin{cases} -1 & x \le 0 \\ x & -1 < x < 1 \\ +1 & x \ge 1 \end{cases}$
13.	Step	$f(x) = \begin{cases} 0 & x < 0 \\ 1 & x \ge 0 \end{cases}$



https://www.researchgate.net/figure/Activation-Functions-in-ANN-Source-Google-Images-2019_fig3_338014968

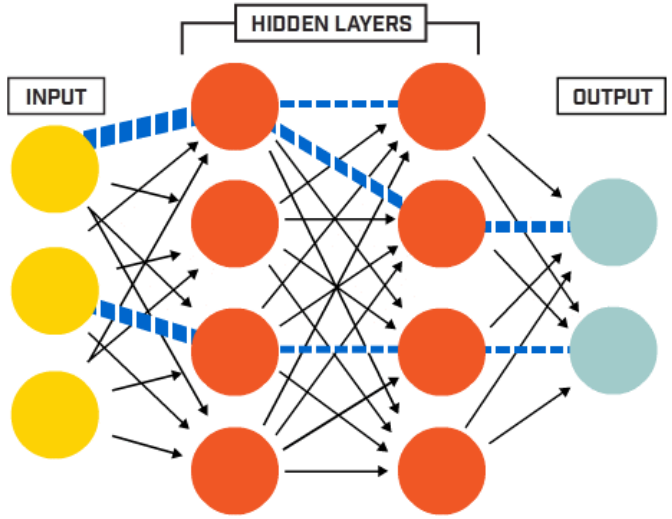
YSA – Çalışma Adımları

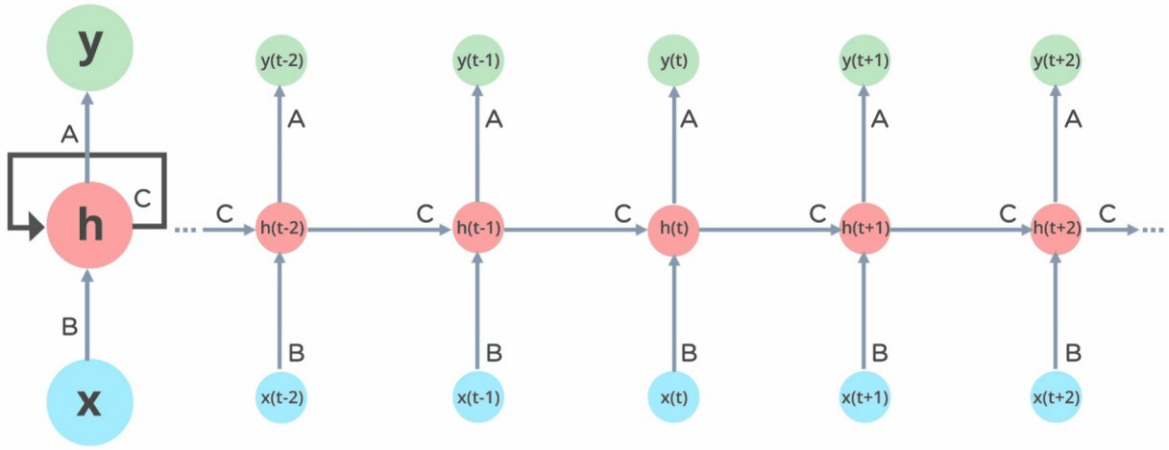
- Örneklerin belirlenmesi
- Ağın topolojisinin belirlenmesi
  - Girdi ve çıktı sayısının belirlenmesi
- Ağın öğrenme parametrelerinin belirlenmesi
  - öğrenme katsayısı ve sabitlerin belirlenmesi
- Ağın başlangıç değerlerinin atanması
- Epoch sayısı kadar
  - Eğitim setindeki tüm örnekler için
    -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

YSA Sınıflandırma

- İleri Beslemeli
  - Ağa gelen bilgiler giriş katmanına daha sonra sırasıyla gizli katmanlardan ve çıkış katmanından işlenerek geçer ve sonra dış dünyaya çıkar

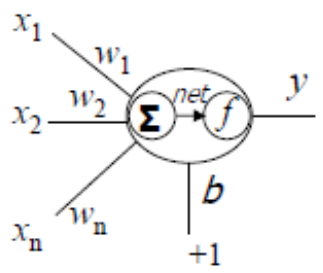
- Geri Geslemeli
  - Bir hücrenin çıktısı sadece kendinden sonra gelen katmana girdi olarak verilmez. Kendinden önceki katmanda veya kendi katmanında bulunan herhangi bir hücreye girdi olarak verilebilir





https://cdn-images-1.medium.com/max/550/1*pO5X2c28F1ysJhwnmPsy3Q.gif

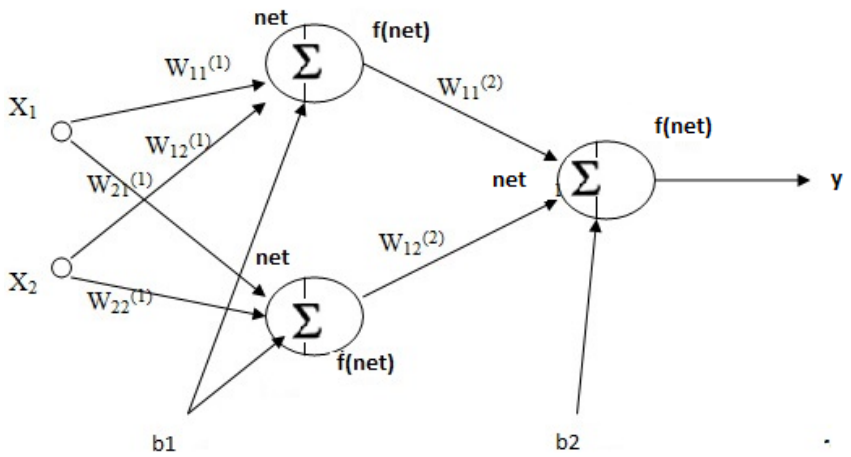
Geri Yayılım Algoritması



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

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = -\eta \left[\frac{\partial E}{\partial y_j}\right]\left[\frac{\partial y_j}{\partial net_i}\right]\left[\frac{\partial net_i}{\partial w_{ij}}\right]$$

$$= -\eta \left[-e_j\right]\left[\frac{\partial y_j}{\partial net_i}\right]\left[x_i\right]$$



Aktivasyon fonksiyonu olarak sigmoid kullanılırsa;

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

- Hata formülündeki d_j: beklenen çıkış, y_j:YSA çıkışı; j:1,…,m
- W_{ij}: x_j girişini, ara katmandaki i. Nörona bağlayan ağırlık değeri

Geri Yayılım Algoritması

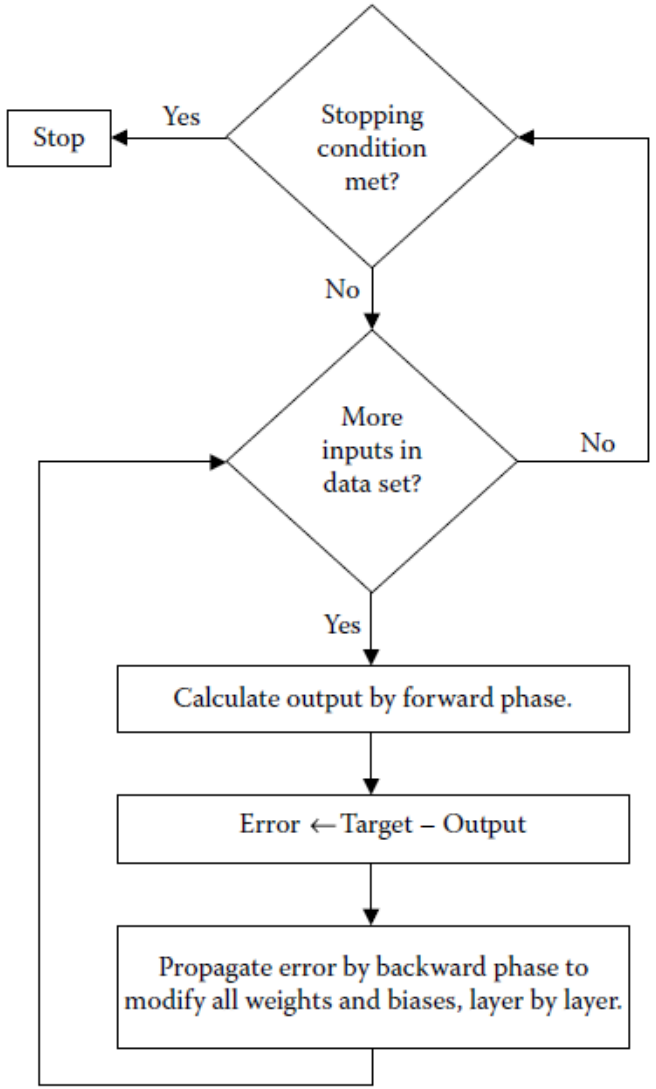
- Yerel minimuma yakalanma olasılığı yüzünden momentum katsayısı kullanılır.
- Backpropagation'da ağırlık güncellemede en yaygın şekilde kullanılan formül

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

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

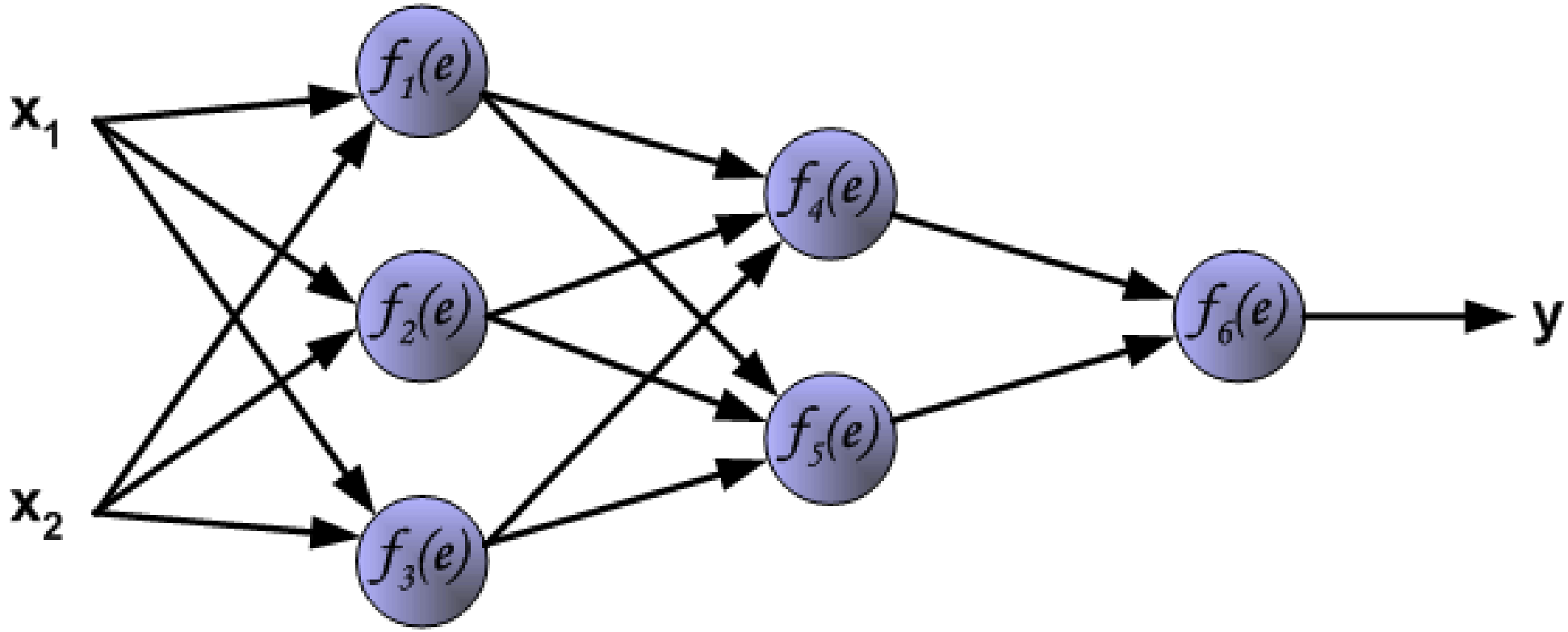
Geri Yayılım Algoritması

- Akış Şeması:

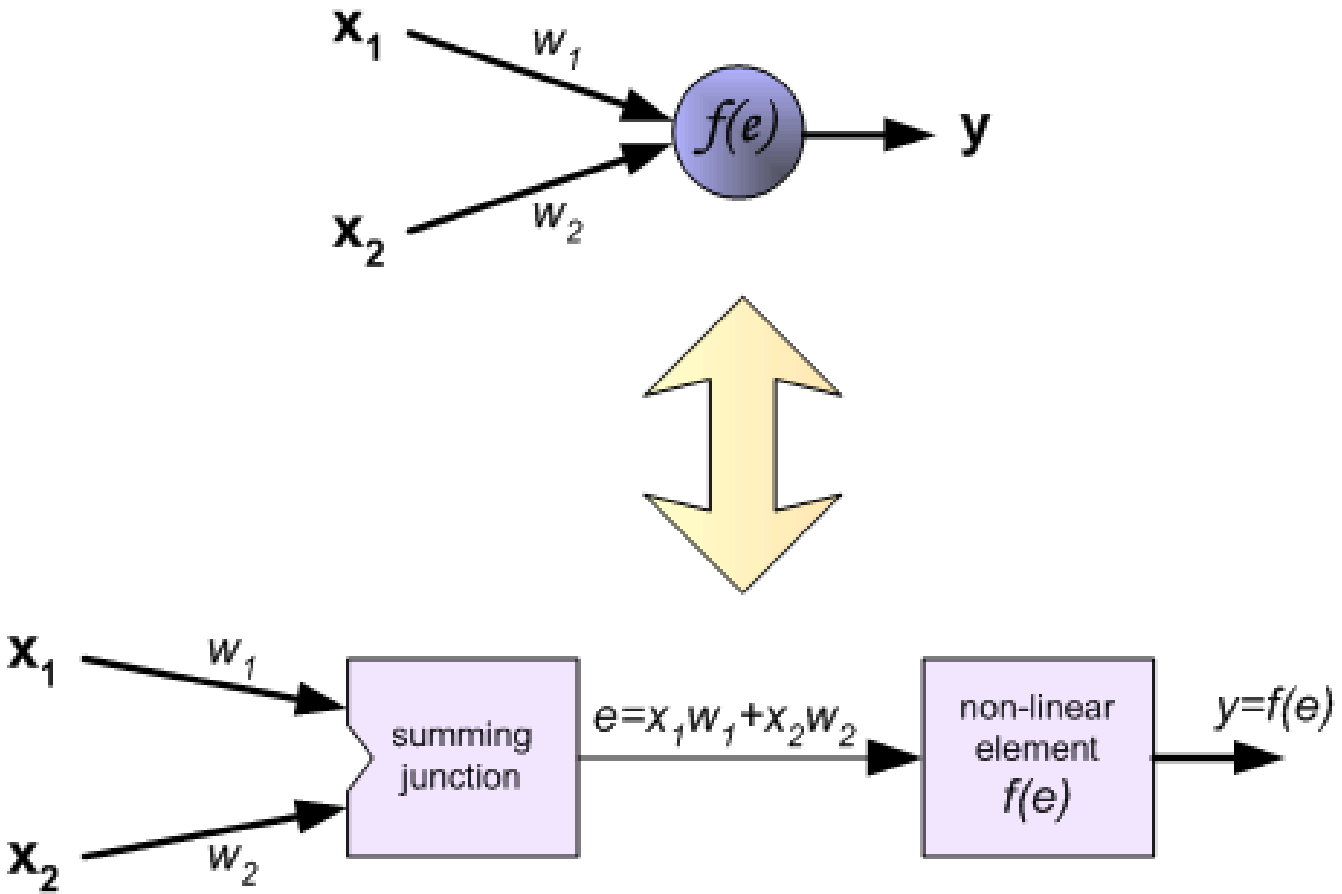


- Sonlanma koşulu sağlanan kadar, yap
  - Her $\langle \vec{x}, \vec{t} \rangle$ eğitim verisi çifti için, yap
    - $\vec{x}$ girişini ağa sun ve ağdaki her birim (u) için o_u çıkışını hesapla.
    - Ağdaki her k çıkış birimi için, bu çıkış biriminin δ_k hata terimini hesapla.
      - $\delta_k = o_k(1-o_k)(t_k - o_k)$
    - ağdaki her h gizli katmanı için, gizli birim δ_k hatasını hesapla
      - $\delta_h = o_h(1-o_h)\sum_{k \in outputs} w_{kh}\delta_k$
    - her ağ bağlantısı w_{ji}'yi güncelle
      - $w_{ji} = w_{ji} + \Delta w_{ji}$
      - $\Delta w_{ji} = \eta \delta_j x_{ji}$

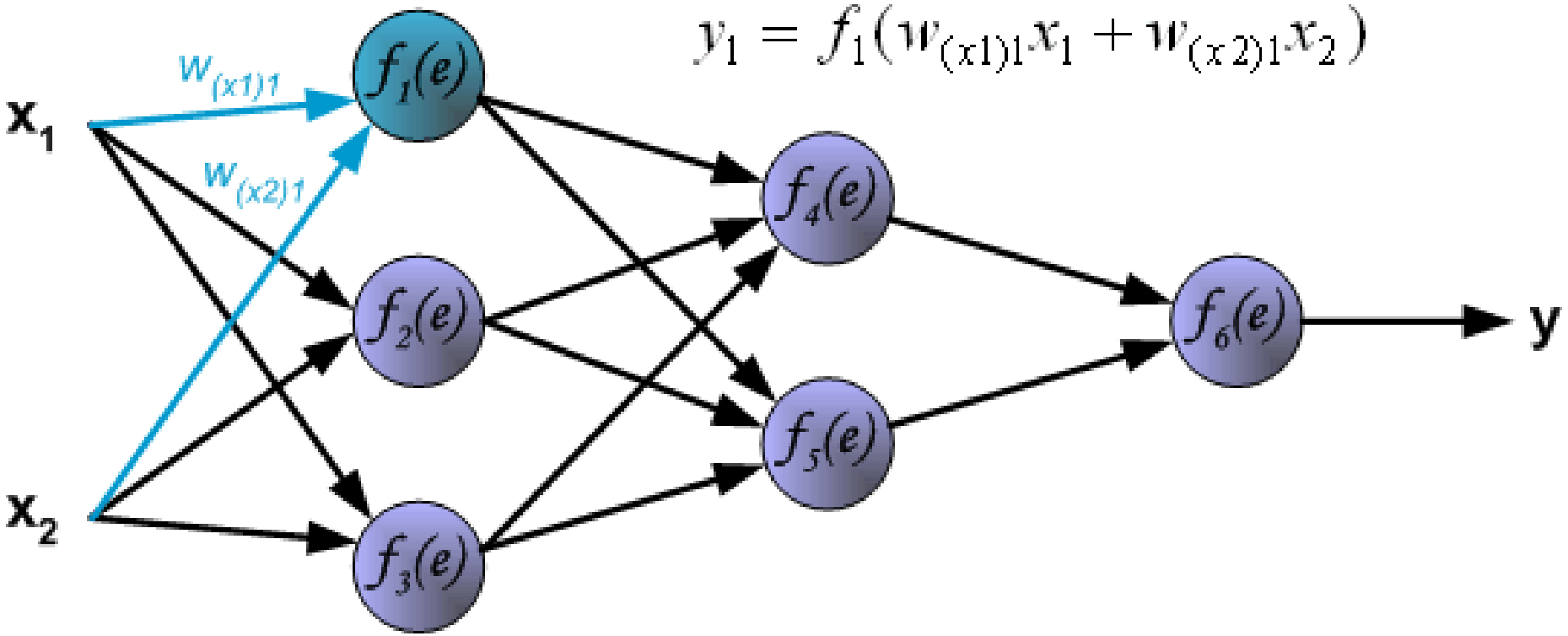
Backpropagation Adımları



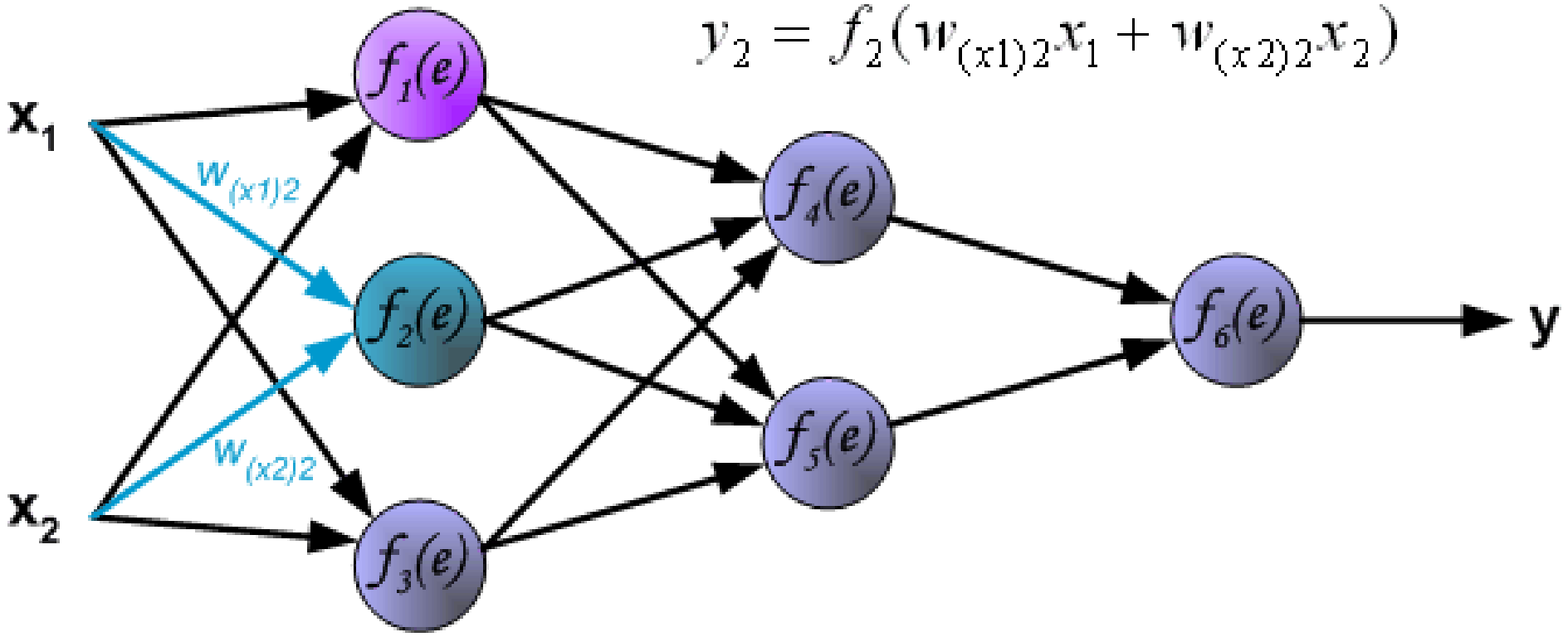
Backpropagation Adımları



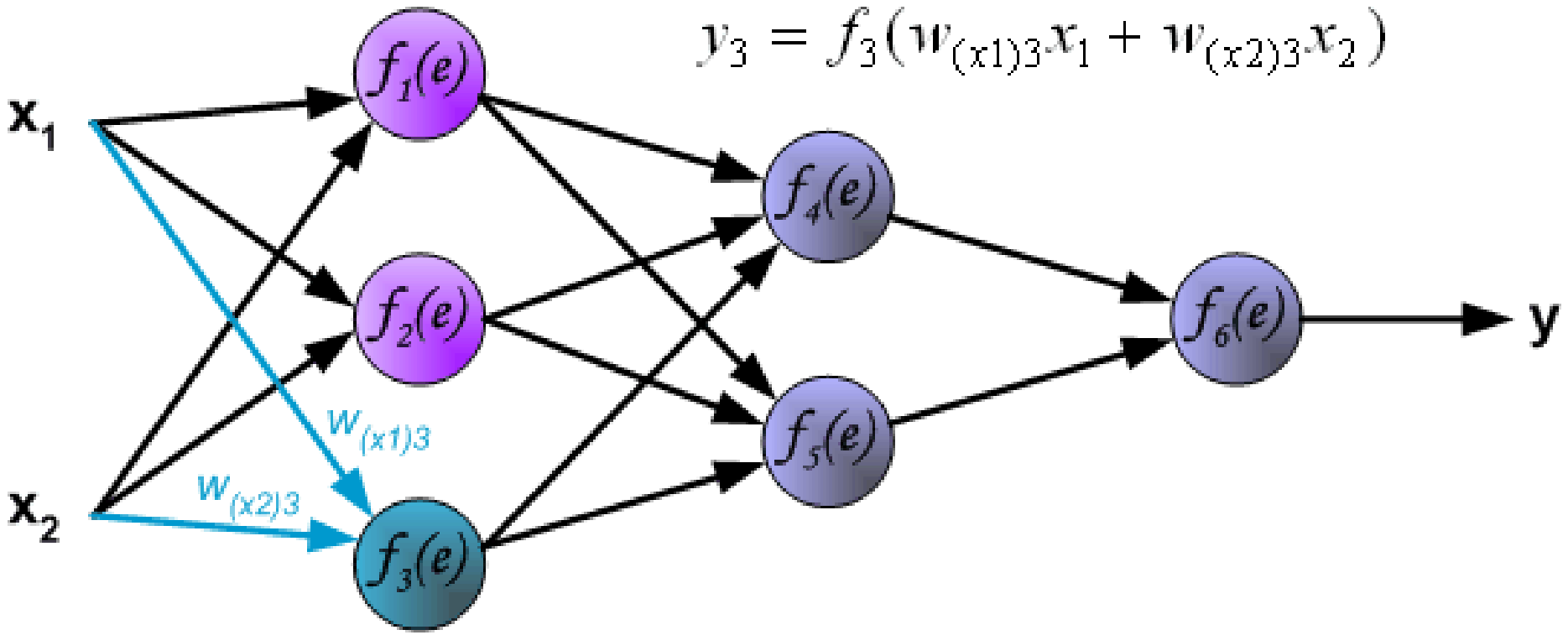
Backpropagation Adımları



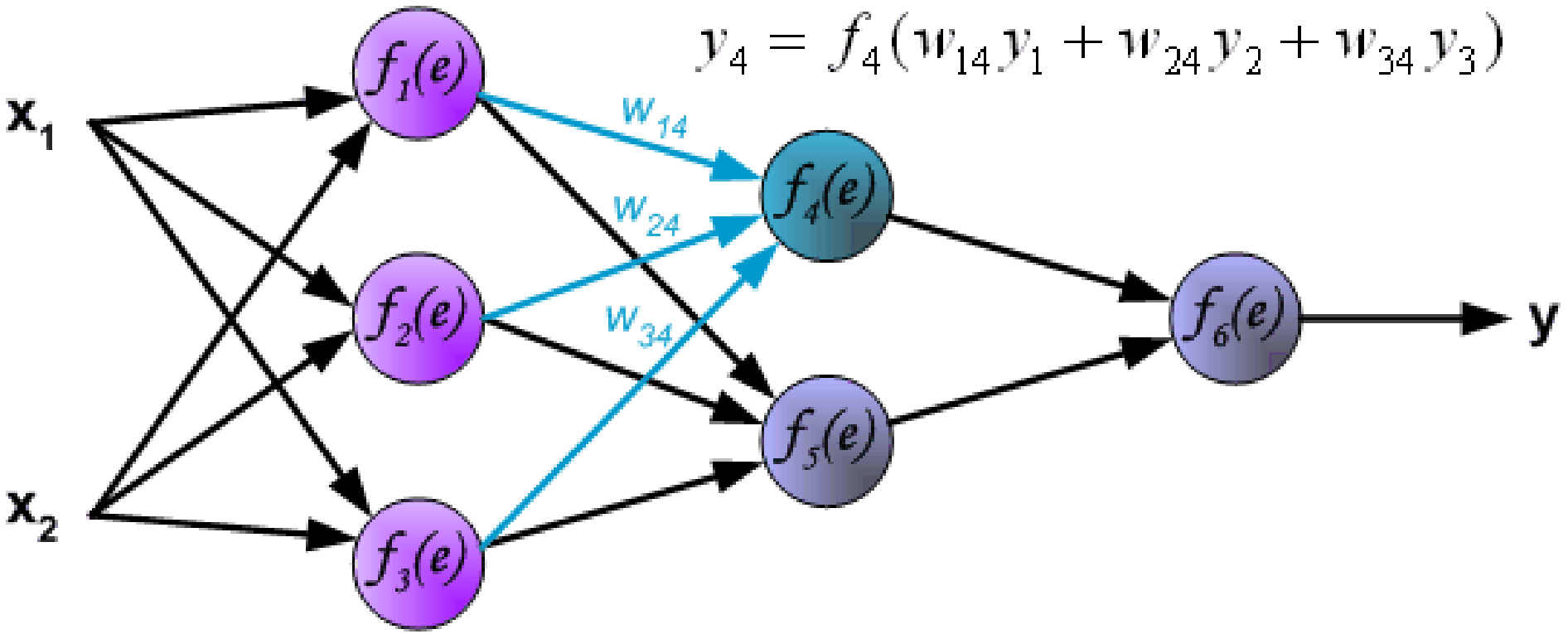
Backpropagation Adımları



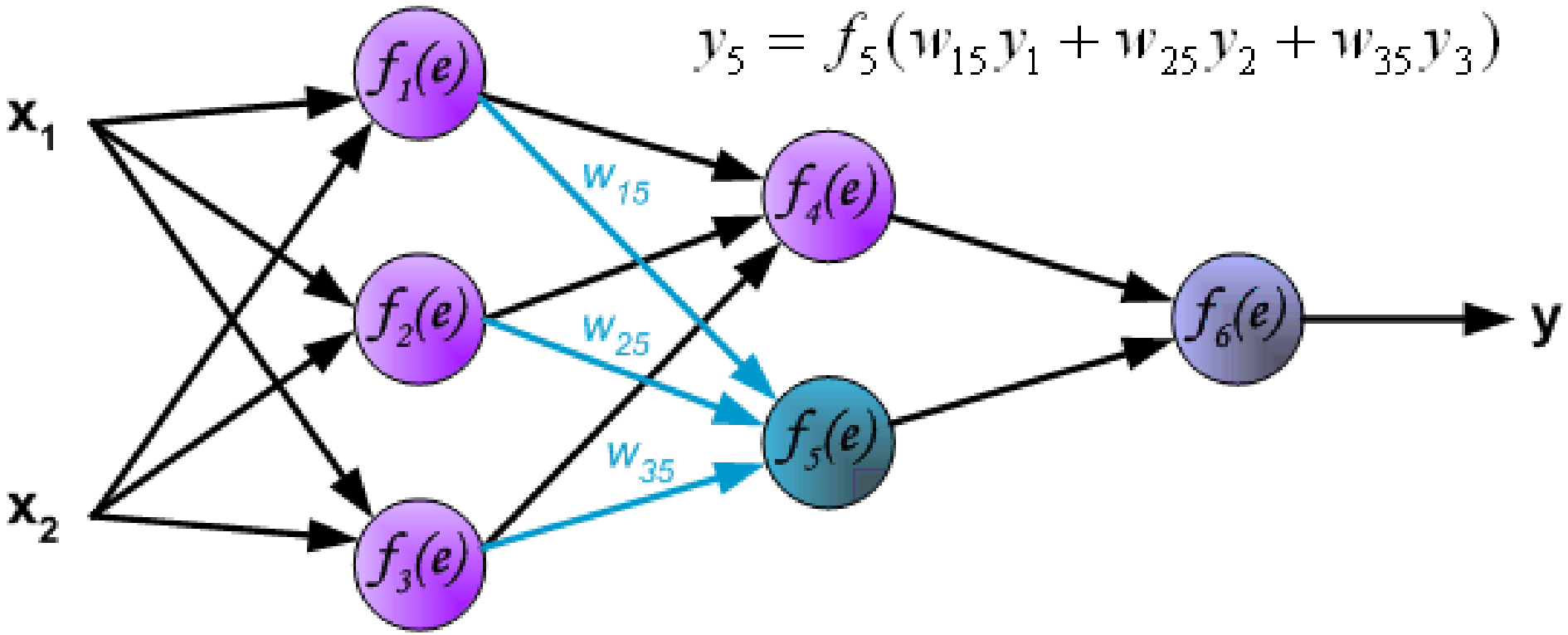
Backpropagation Adımları



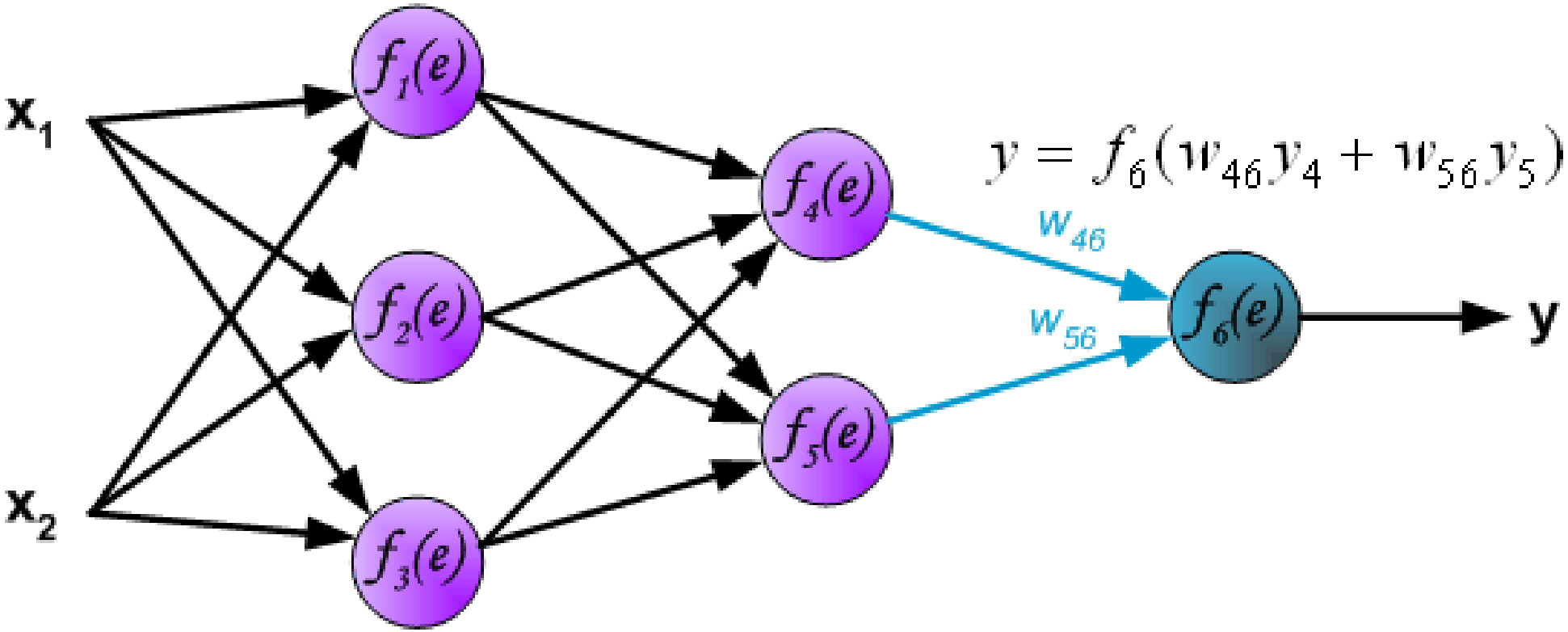
Backpropagation Adımları



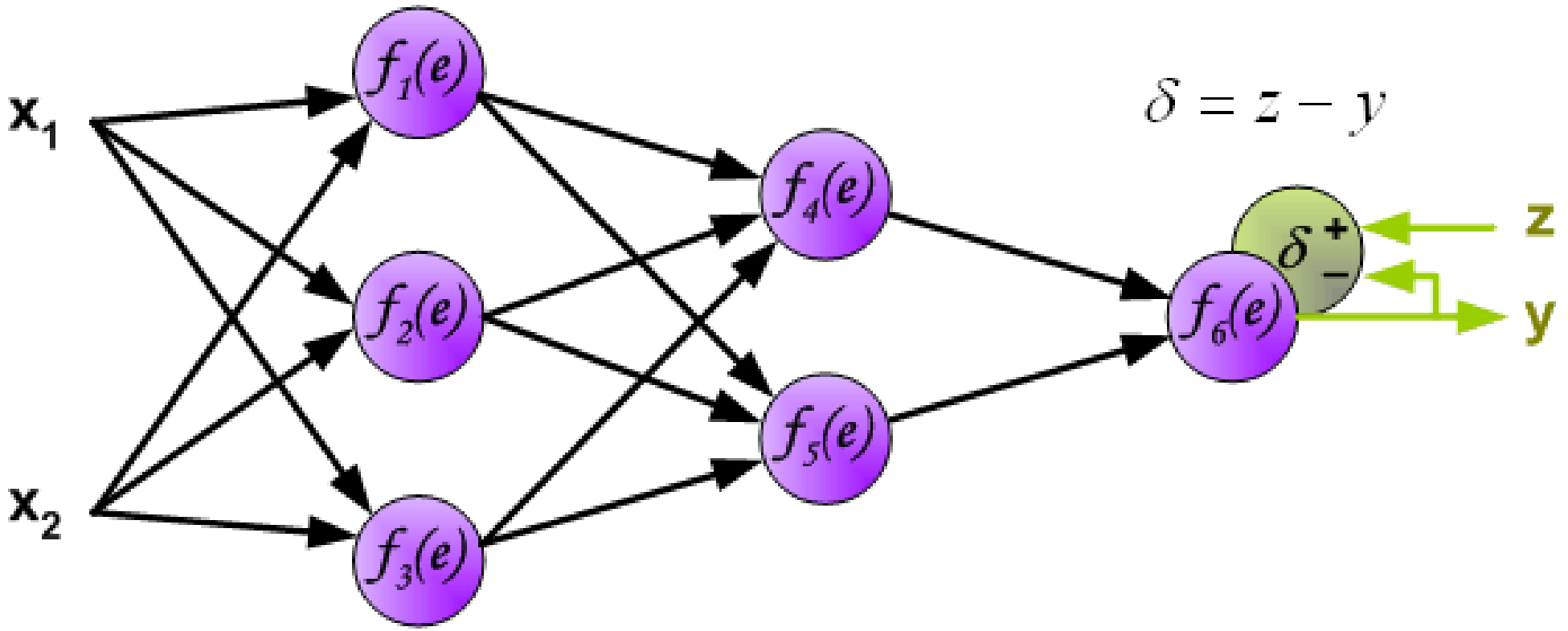
Backpropagation Adımları



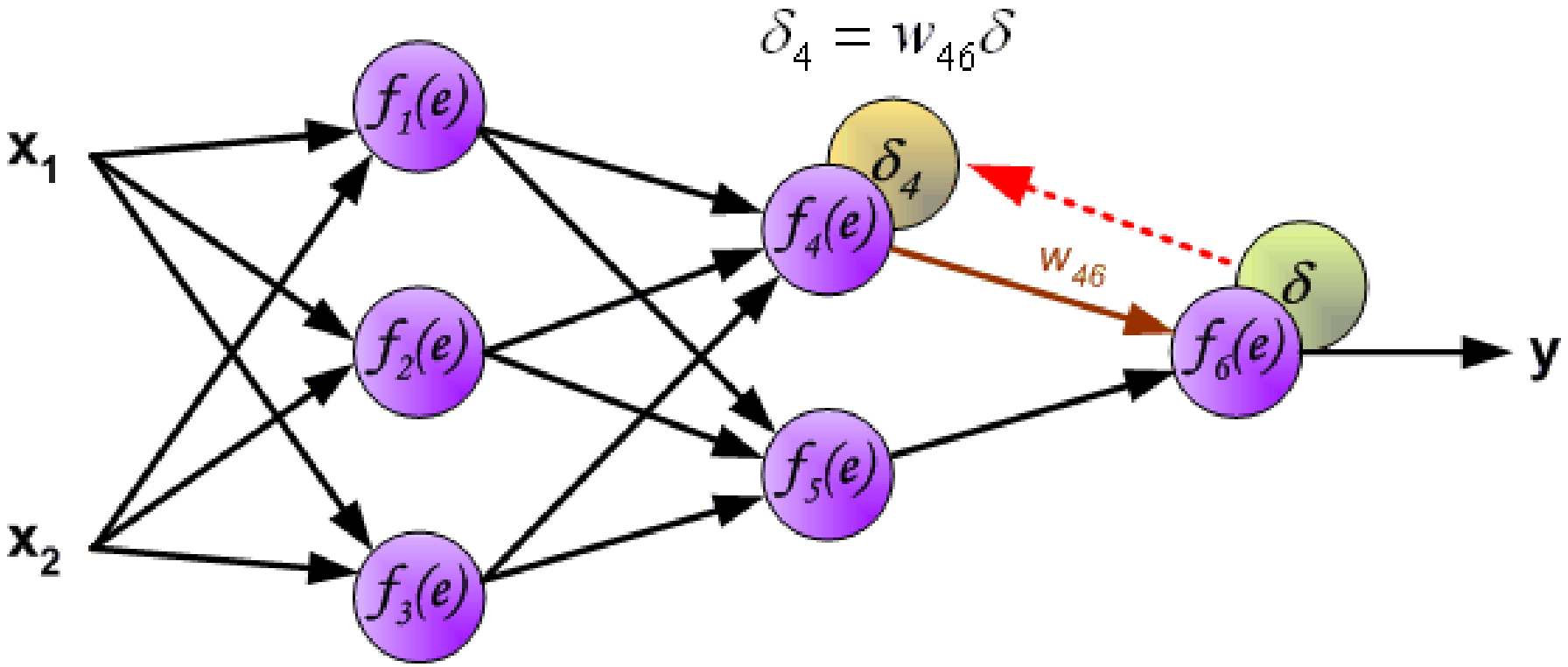
Backpropagation Adımları



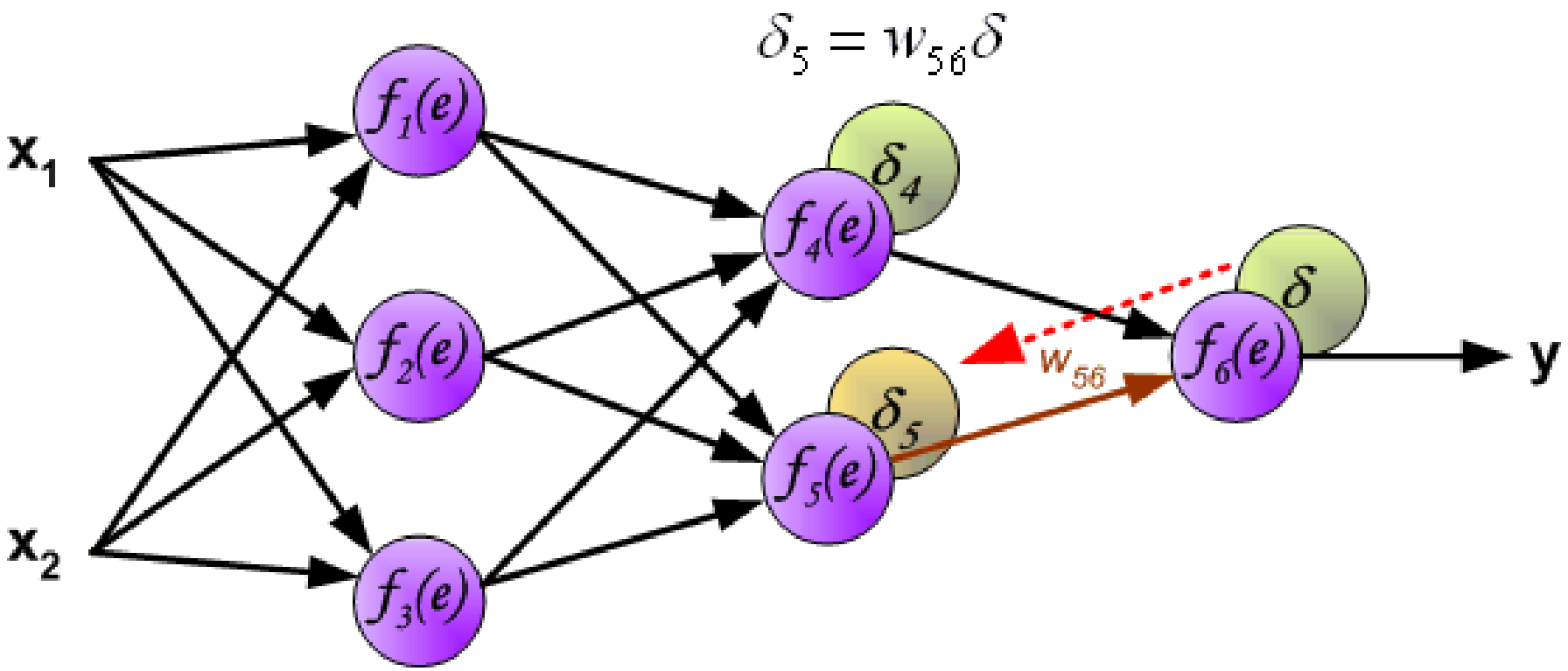
Backpropagation Adımları



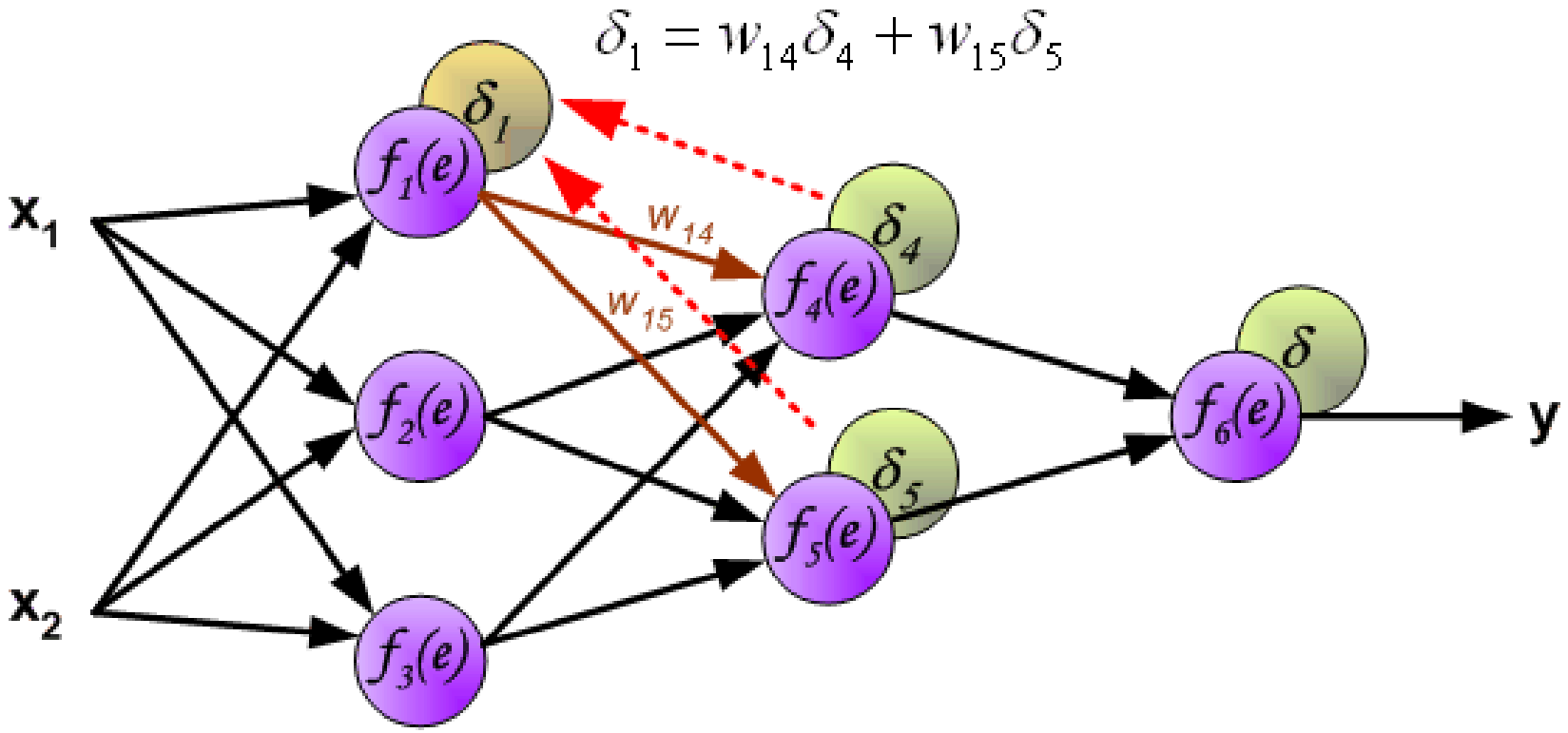
Backpropagation Adımları



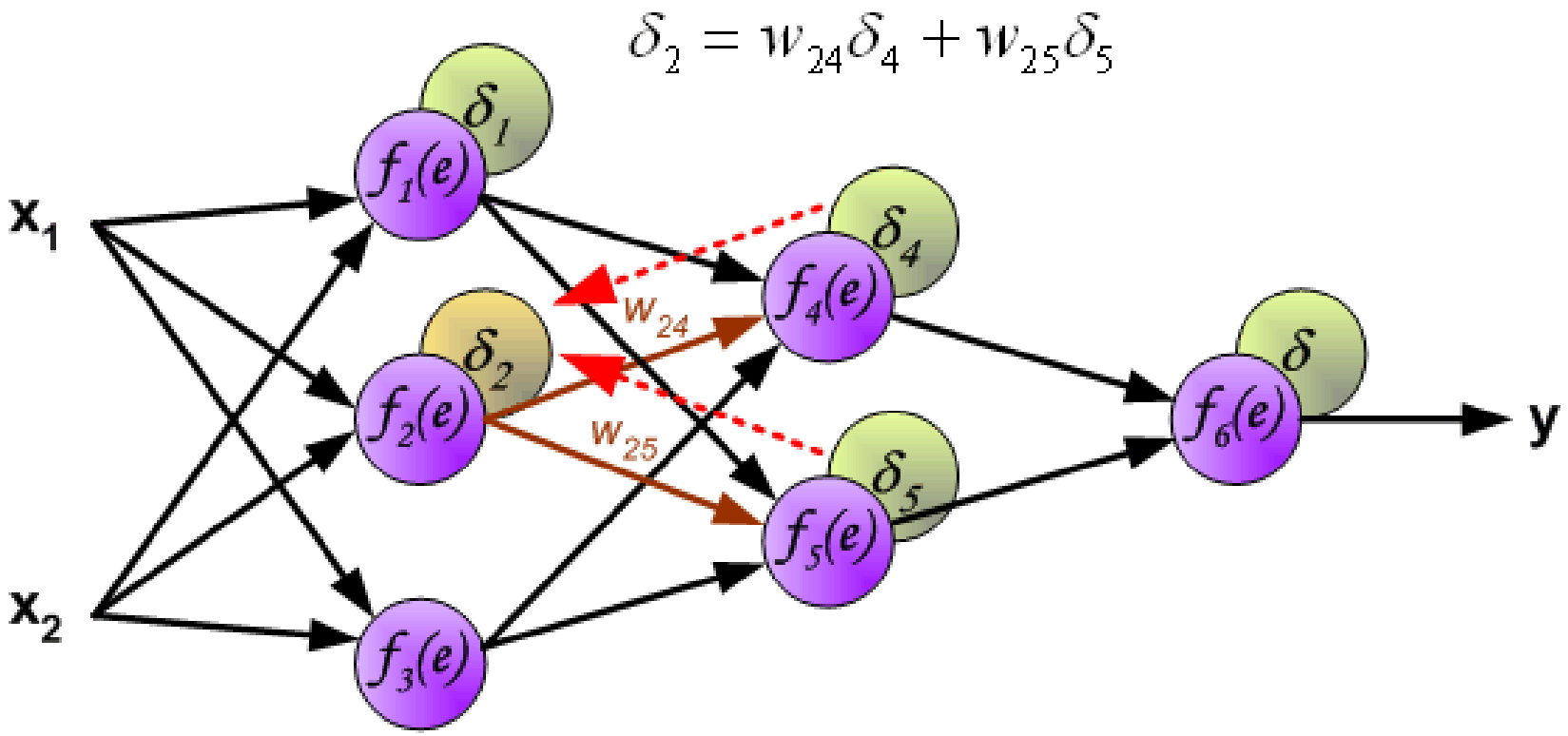
Backpropagation Adımları



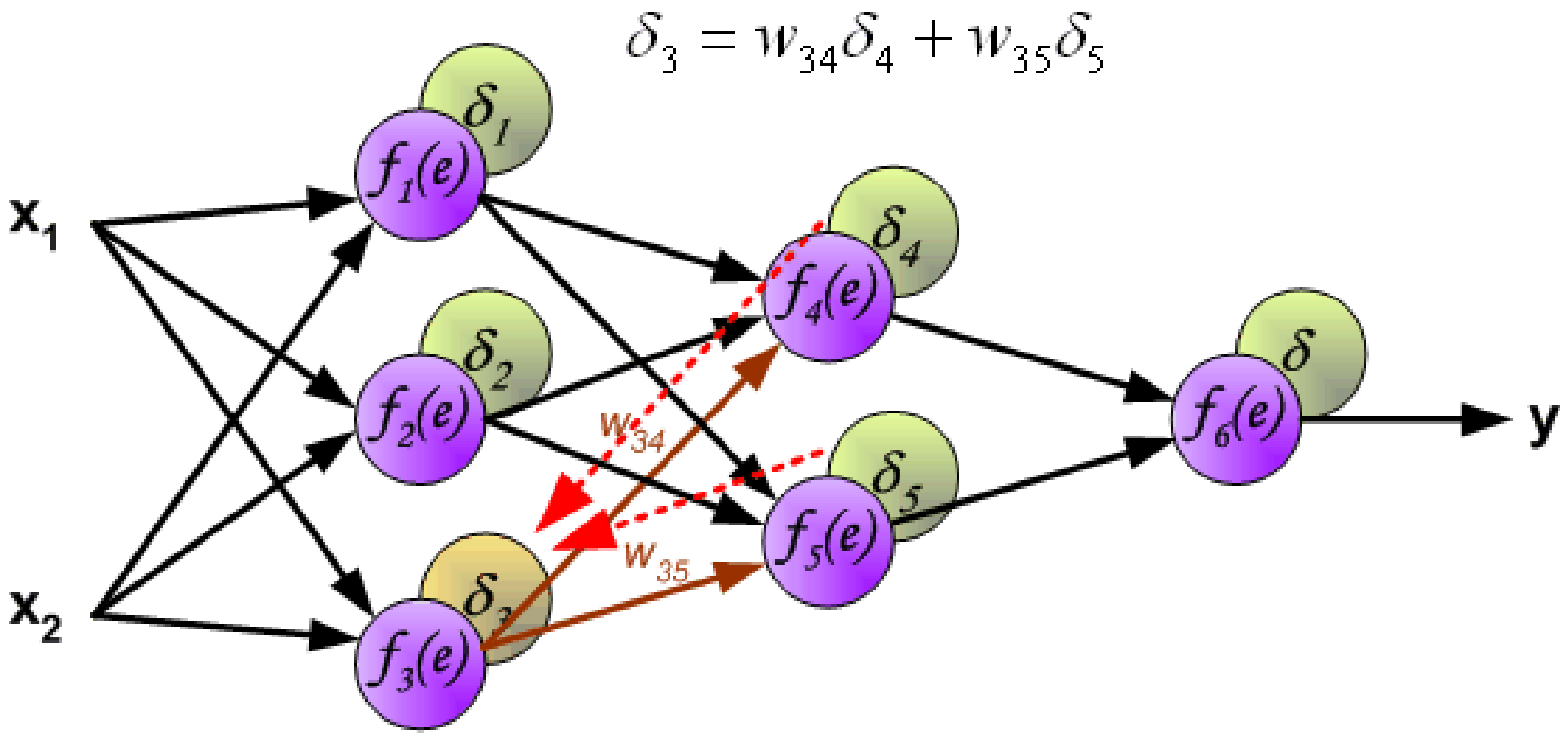
Backpropagation Adımları



Backpropagation Adımları



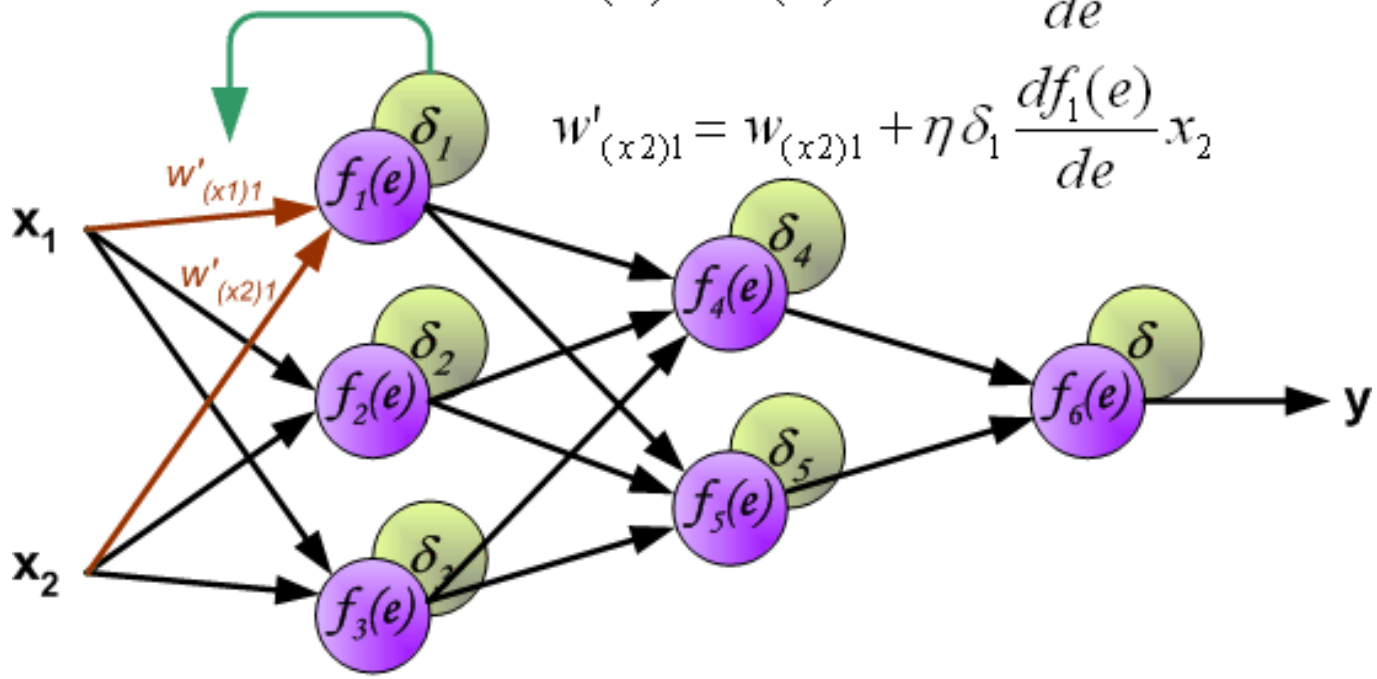
Backpropagation Adımları



Backpropagation Adımları

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

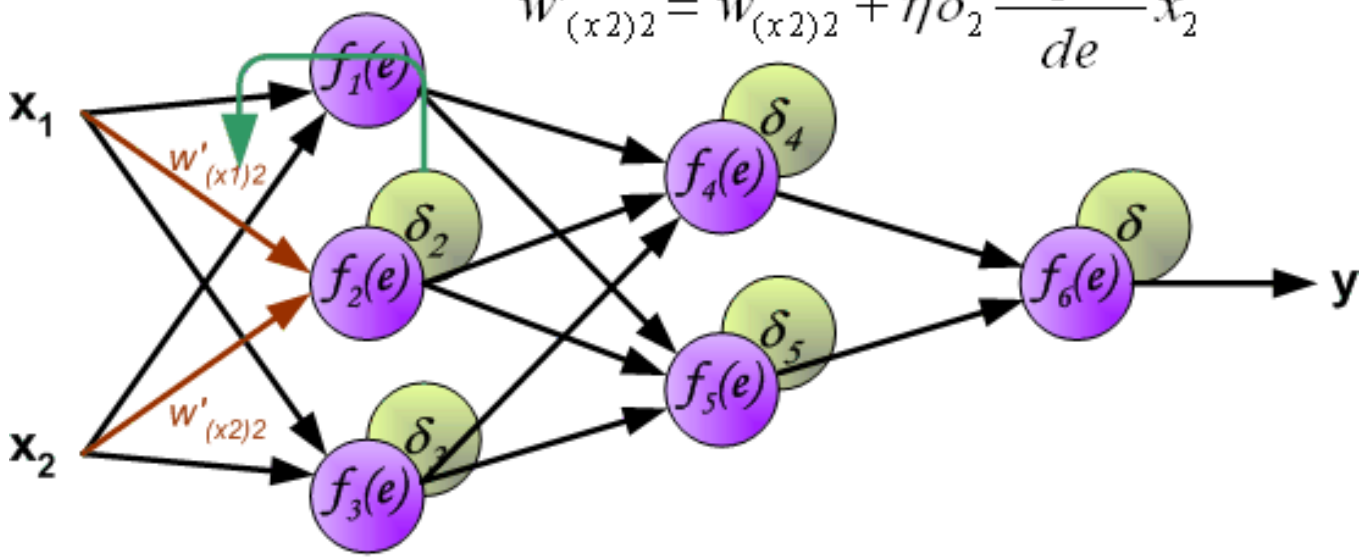
$$w'_{(x2)1} = w_{(x2)1} + \eta \delta_1 \frac{df_1(e)}{de} x_2$$



Backpropagation Adımları

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

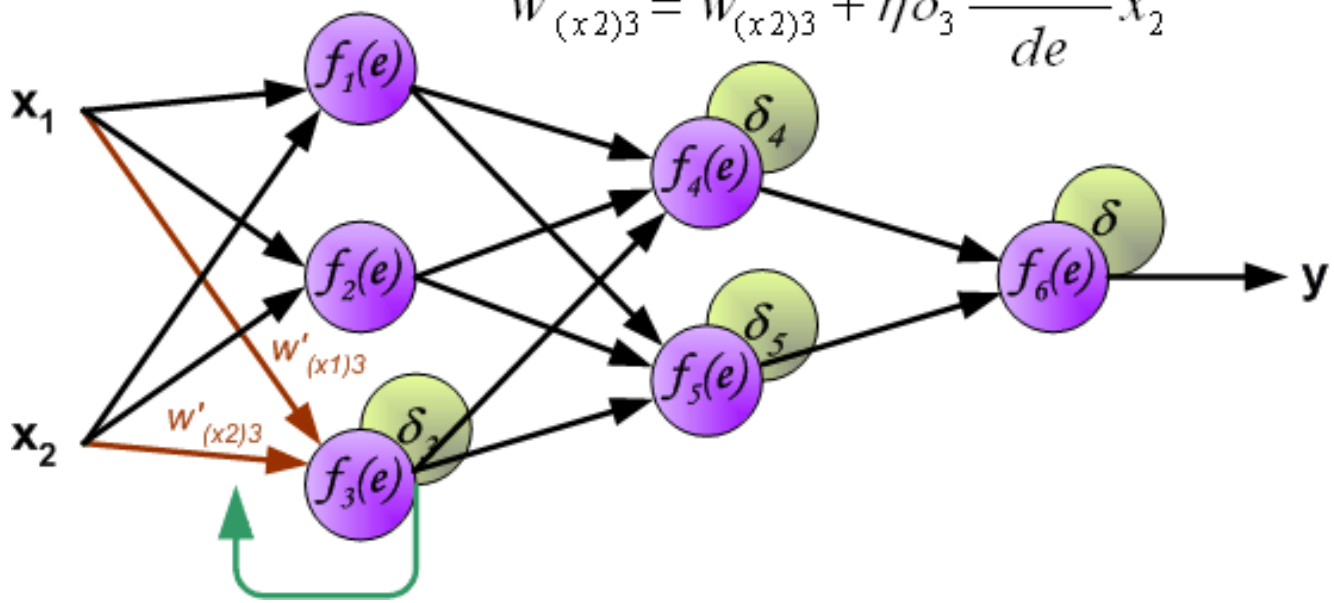
$$w'_{(x2)2} = w_{(x2)2} + \eta \delta_2 \frac{df_2(e)}{de} x_2$$



Backpropagation Adımları

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

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

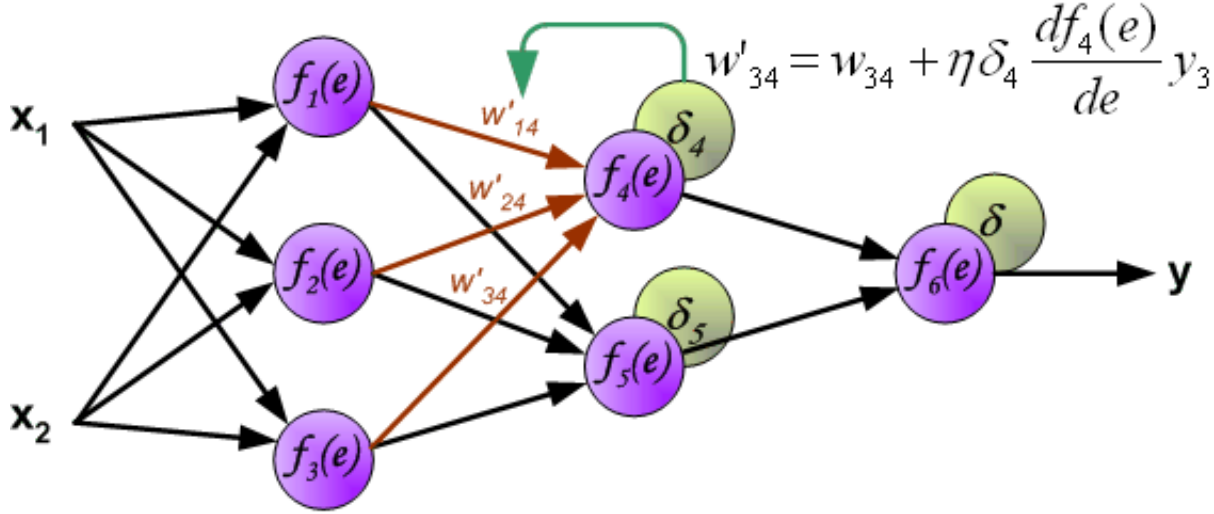


Backpropagation Adımları

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

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

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

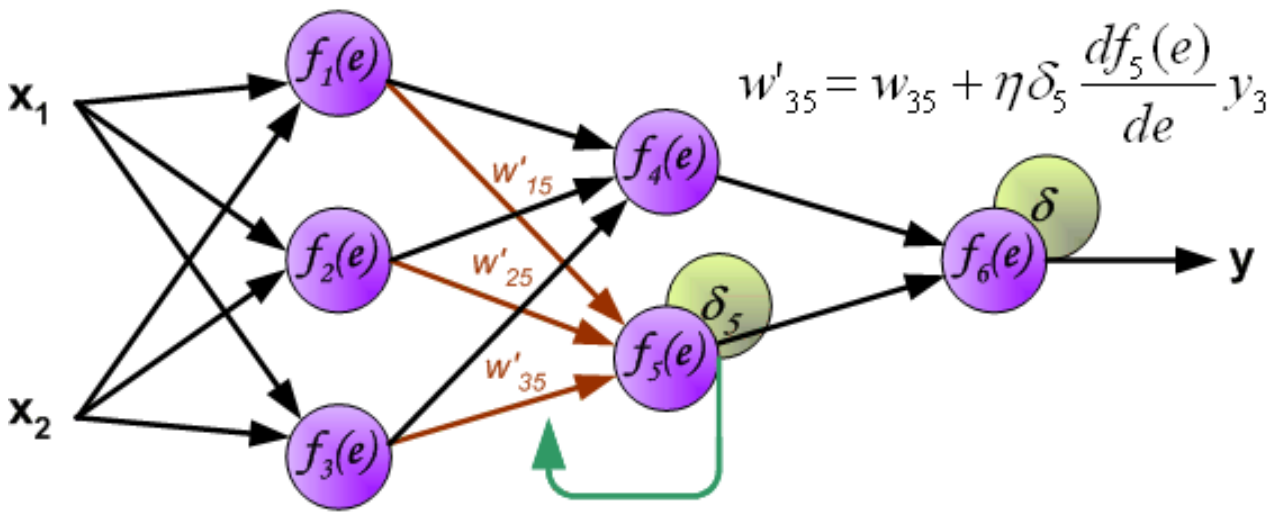


Backpropagation Adımları

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

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

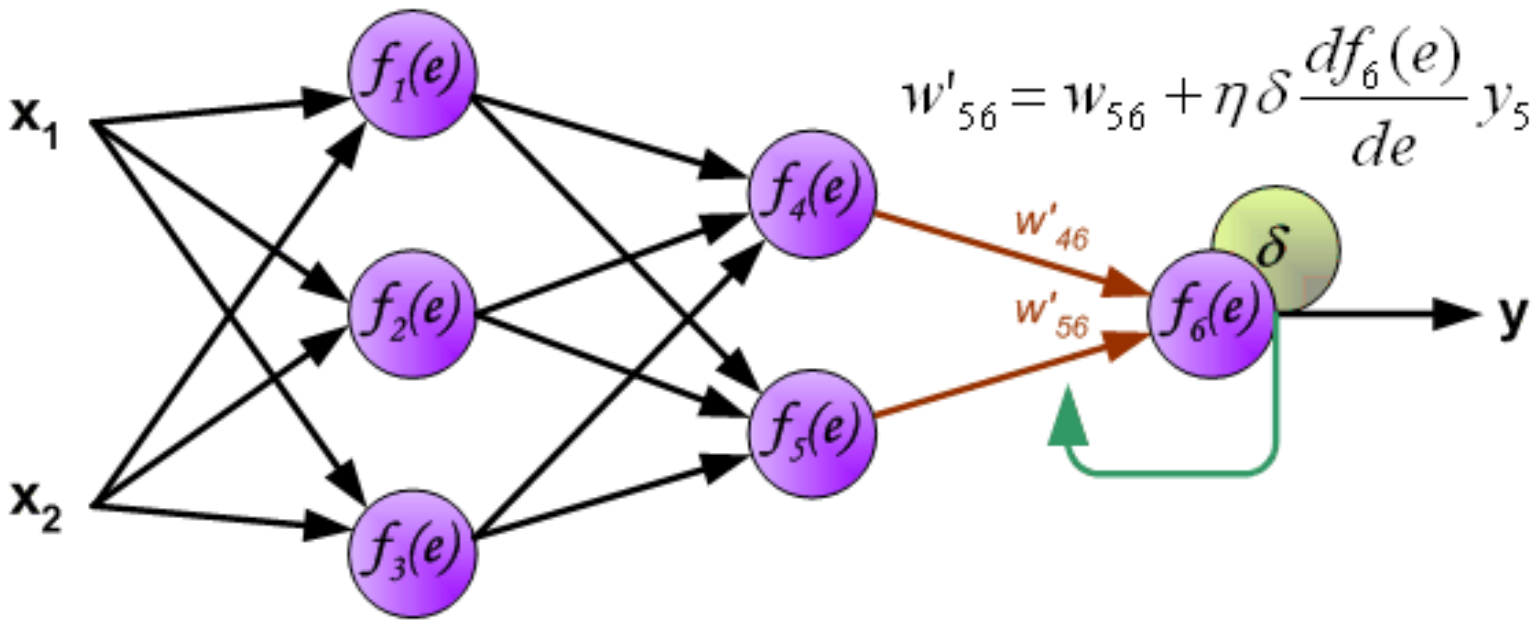
$$w'_{35} = w_{35} + \eta \delta_5 \frac{df_5(e)}{de} y_3$$



Backpropagation Adımları

$$w'_{46} = w_{46} + \eta\,\delta\,\frac{df_6(e)}{de}\,y_4$$

$$w'_{56} = w_{56} + \eta\,\delta\,\frac{df_6(e)}{de}\,y_5$$



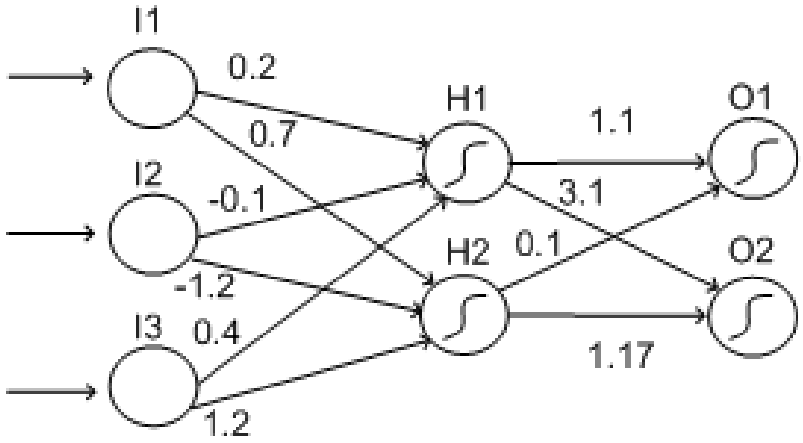
MLP Örnek

- Örnekleri ileri doğru sürdüğümüzde

  - Gizli katmanlarda ağırlıklı toplamlar:
    - $S_{H1} = (0.2*10) + (-0.1*30) + (0.4*20) = 2-3+8 = 7$
    - $S_{H2} = (0.7*10) + (-1.2*30) + (1.2*20) = 7-6+24= -5$

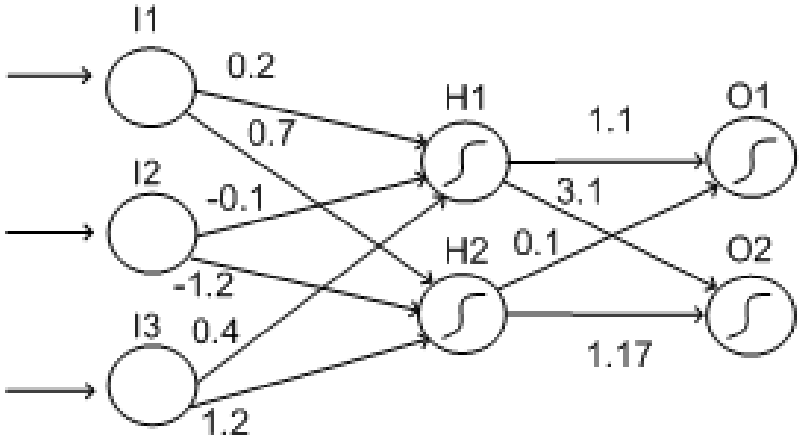
  - Bir sonraki adımda gizli katman çıkışları hesaplanır:
    - $\sigma(S) = 1/(1 + e^{-S})$
    - $\sigma(S_{H1}) = 1/(1 + e^{-7}) = 1/(1+0.000912) = 0.999$
    - $\sigma(S_{H2}) = 1/(1 + e^{5}) = 1/(1+148.4) = 0.0067$

Ağa sunulan örnekler: E(10,30,20)



MLP Örnek

- 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:
  - $\sigma(S_{O1}) = 1/(1+e^{-1.0996}) = 1/(1+0.333) = 0.750$
  - $\sigma(S_{O2}) = 1/(1+e^{-3.1047}) = 1/(1+0.045) = 0.957$



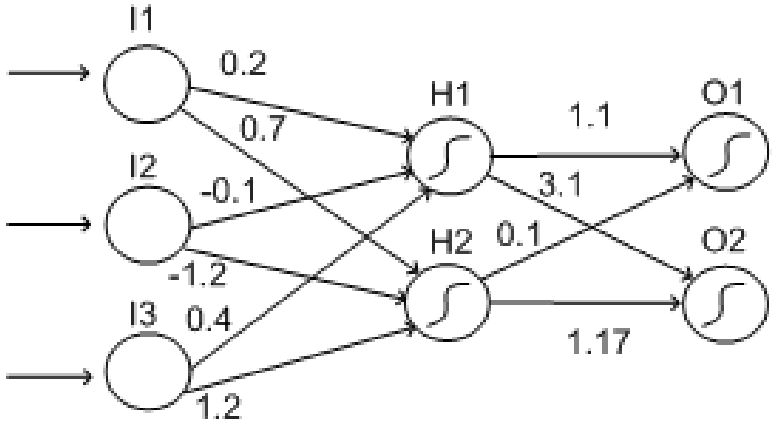
MLP Örnek

- Örnekleri ileri doğru sürdüğümüzde elde edilen hesaplamalar:

Ağa sunulan örnekler E(10,30,20)

η = 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						

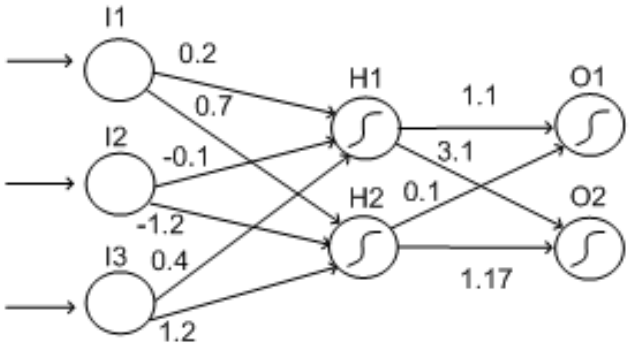


MLP Örnek

- Çıkış birimlerinin hata degerleri: $\delta_k = o_k(1-o_k)(t_k-o_k)$
  - o1(E)=0.750, o2(E)=0.957

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

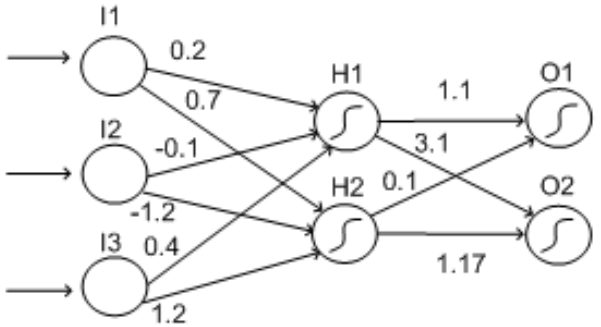
$$\delta_{O2} = o_2(E)(1 - o_2(E))(t_2(E) - o_2(E)) = 0.957(1-0.957)(0-0.957) = -0.0394$$



MLP Örnek

- Gizli birimlerinin hata degerleri: $\delta_h = o_h(1-o_h)\sum_{k \in outputs} w_{kh}\delta_k$

  - $\delta_{O1} = 0.0469$ ve $\delta_{O2} = -0.0394$
  - $h_1(E) = 0.999$ ve $h_2(E) = 0.0067$
  - H1 için:
    - $(w_{11}*\delta_{01}) + (w_{12}*\delta_{O2}) = (1.1*0.0469)+(3.1*-0.0394)=-0.0706$
    - 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:
    - $(w_{21}*\delta_{01}) + (w_{22}*\delta_{O2}) = (0.1*0.0469)+(1.17*-0.0394) = -0.0414$
    - Ve bu değeri, $h_2(E)(1-h_2(E))$ ile çarpalım, sonuç:
      - $-0.0414 * (0.067 * (1-0.067)) = -0.00259 = \delta_{H2}$



MLP Örnek

- Ağırlık güncellemeleri için hesaplamalar:
  - Giriş ve gizli katman arasındaki ağırlık değerleri:

her ağ bağlantısı w_{ji} 'yi güncelle

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

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

MLP Örnek

- Ağırlık güncellemeleri için hesaplamalar:
  - Gizli ve çıkış katman arasındaki ağırlık değerleri:

her ağ bağlantısı w_{ji} 'yi güncelle

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

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

References

- Artificial Intelligence A Modern Approach, Stuart Russell and Peter Norvig, Prentice Hall Series in Artificial Intelligence.
- 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/backp_t_en/backprop.html
- http://www.sfu.ca/iat813/lectures/lecture8.html