

Makine Öğrenmesi

Yapay Sinir Ağları ve Derin Öğrenme

İlker Birbil ve Utku Karaca

Erasmus Üniversitesi Rotterdam

İstanbul'da Makine Öğrenmesi

27 Ocak – 2 Şubat, 2020



Makine Öğrenmesi

```
graph TD; A[Makine Öğrenmesi] --> B[Doğrusal Bağlanım]; A --> C[Boyut Küçültme ve Düzenleştirme]; A --> D[Tekrar Örnekleme ve Model Değerlendirme]; A --> E[Sınıflandırma ve Ağaçlar]; A --> F[GÜdÜmsüz Öğrenme]; A --> G[Yapay Sinir Ağları ve Derin Öğrenme];
```

Doğrusal Bağlanım

Boyut Küçültme
ve
Düzenleştirme

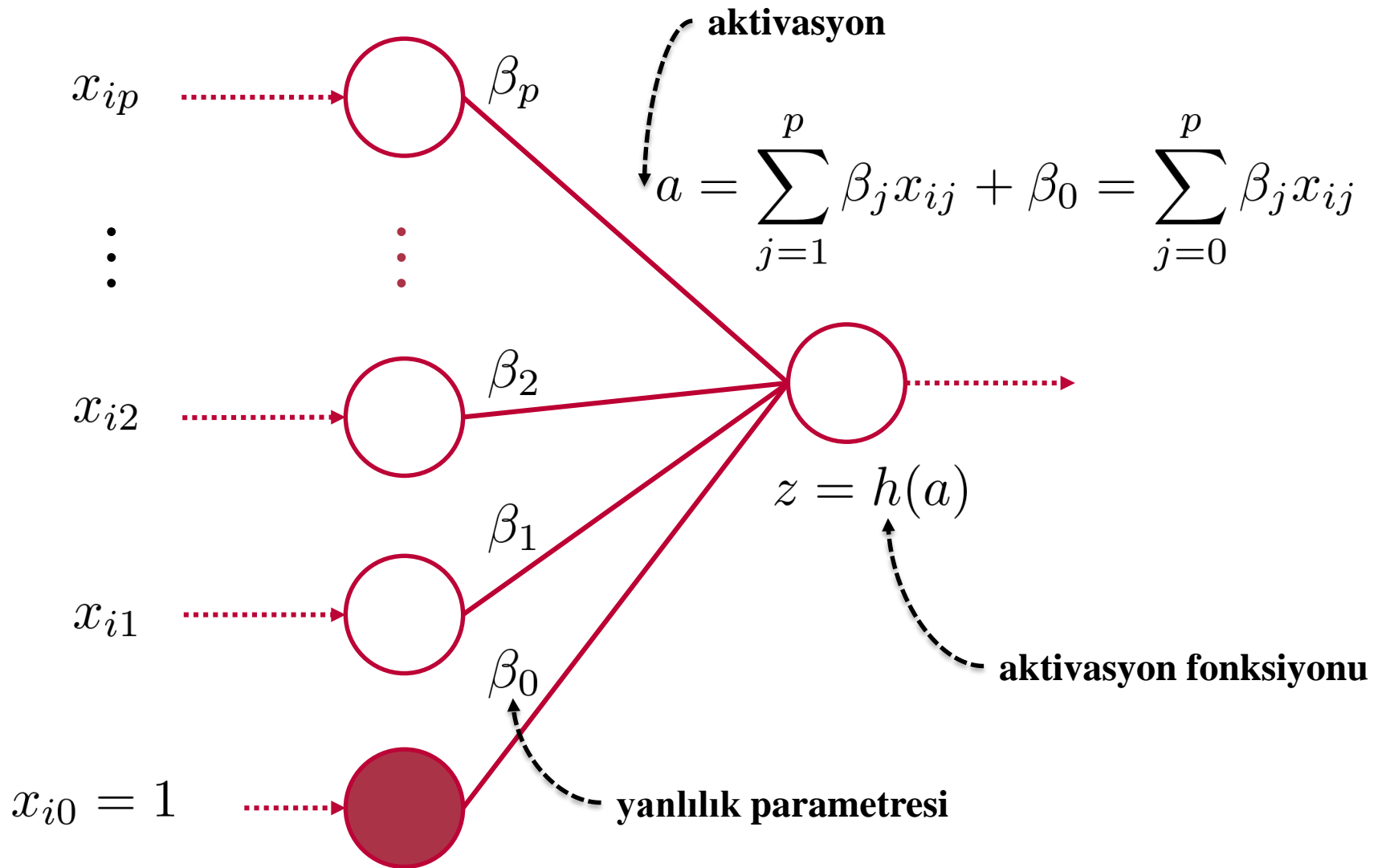
Tekrar Örnekleme
ve
Model Değerlendirme

Sınıflandırma
ve
Ağaçlar

GÜdÜmsüz
Öğrenme

Yapay Sinir Ağları
ve
Derin Öğrenme

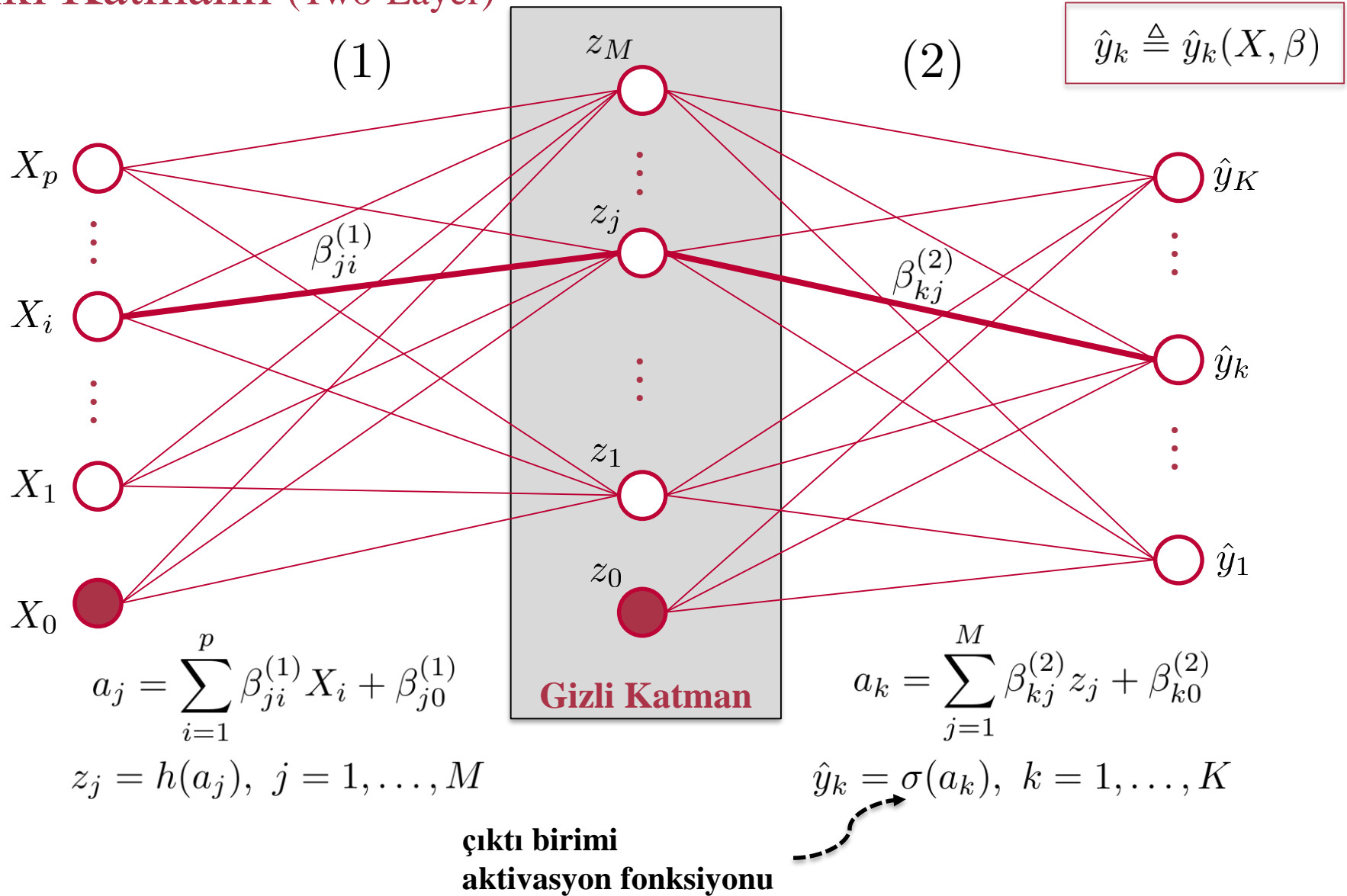
İşlem Birimleri (Processing Units)



Yapay Sinir Ağları

İleribeslemeli Yapay Sinir Ağı (Feed-Forward Neural Network)

İki Katmanlı (Two-Layer)



Aktivasyon Fonksiyonları

$$h(a_j) = \frac{1}{1 + e^{-a_j}}$$

Sigmoid Fonksiyonu

$$h(a_j) = \tanh(a_j)$$

“tanh” Fonksiyonu

$$h(a_j) = \max\{0, a_j\}$$

Rectified Linear Unit (ReLU)

$$\sigma(a_k) = a_k$$

Özdeşlik (Identity) Fonksiyon

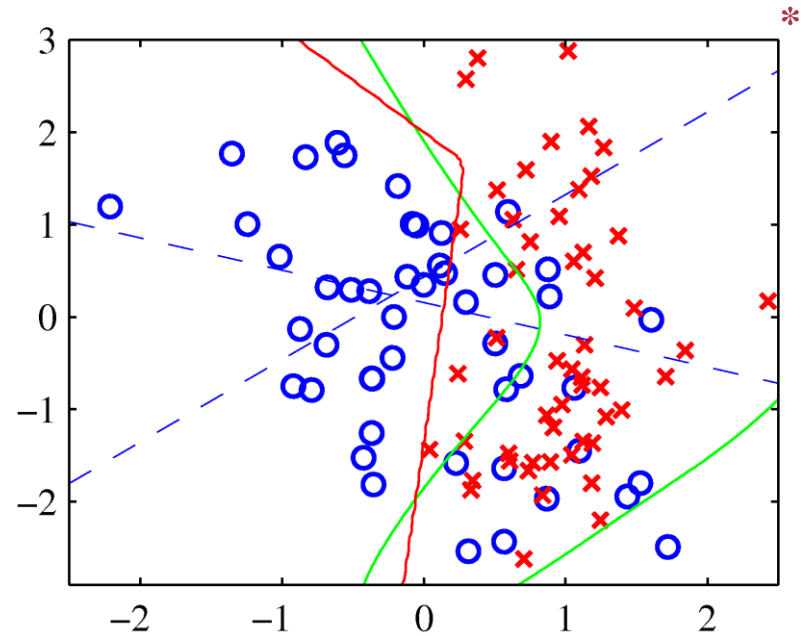
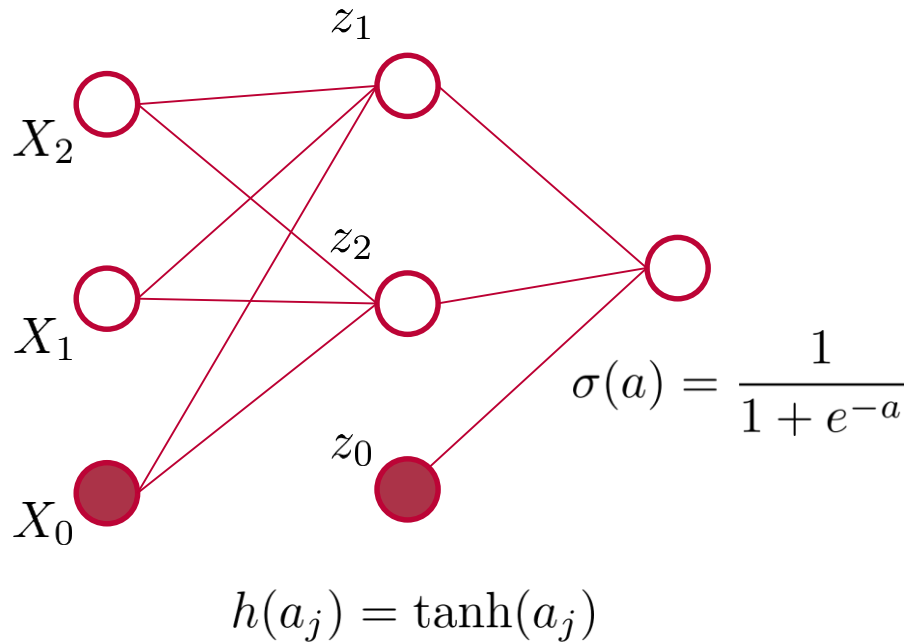
$$\sigma(a_k) = \frac{1}{1 + e^{-a_k}}$$

Sigmoid Fonksiyonu

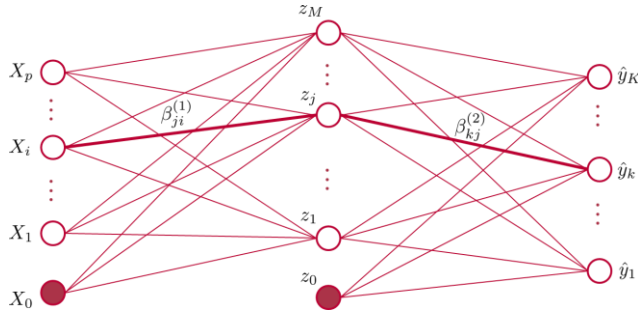
$$\sigma(a_k) = \frac{e^{a_k}}{\sum_{j=1}^K e^{a_j}}$$

Düzgeli Üstel (Softmax) Fonksiyon

İkili Sınıflandırma Örneği



Ağ Fonksiyonu



$$a_j = \sum_{i=1}^p \beta_{ji}^{(1)} X_i + \beta_{j0}^{(1)} \quad a_k = \sum_{j=1}^M \beta_{kj}^{(2)} z_j + \beta_{k0}^{(2)}$$

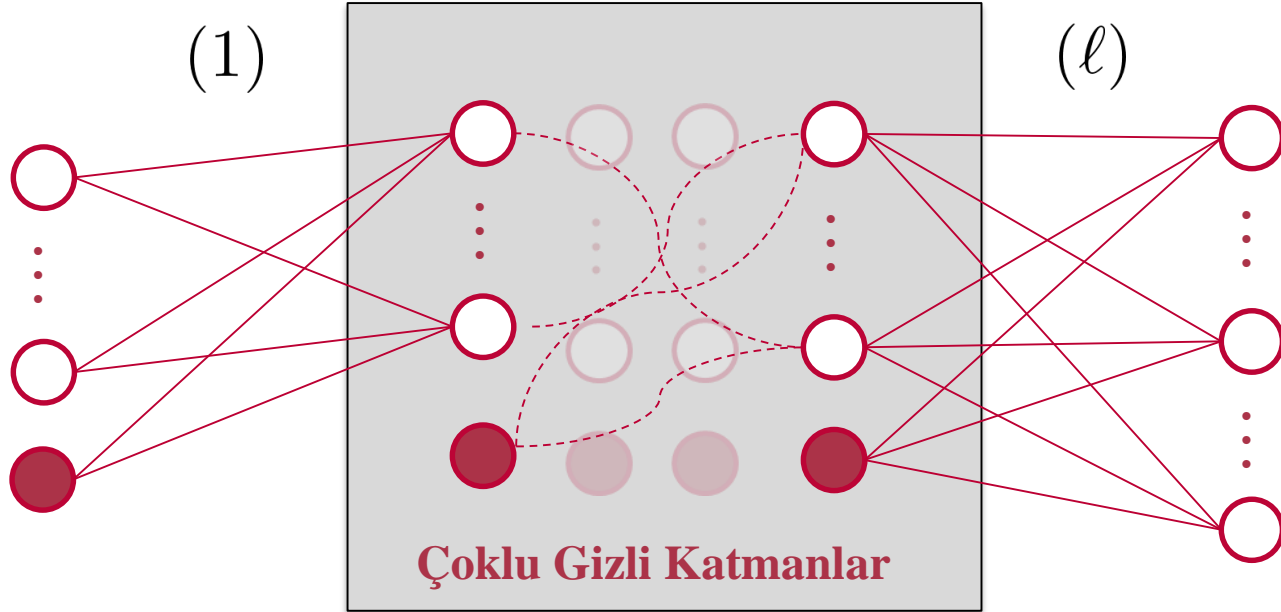
$$z_j = h(a_j), \quad j = 1, \dots, M \quad \hat{y}_k = \sigma(a_k), \quad k = 1, \dots, K$$

$$\hat{y}_k(X, \beta) = \sigma \left(\sum_{j=1}^M \beta_{kj}^{(2)} h \left(\sum_{i=1}^p \beta_{ji}^{(1)} X_i + \beta_{j0}^{(1)} \right) + \beta_{k0}^{(2)} \right)$$

$$X_0 = 1, \quad z_0 = 1$$

$$\hat{y}_k(X, \beta) = \sigma \left(\sum_{j=0}^M \beta_{kj}^{(2)} h \left(\sum_{i=0}^p \beta_{ji}^{(1)} X_i \right) \right)$$

Ağ Fonksiyonu



$$\hat{y}_k(X, \beta) = \sigma \left(\sum_j \beta_{kj}^{(\ell)} h \left(\sum_s \beta_{js}^{(\ell-1)} h \left(\dots h \left(\sum_i \beta_{ji}^{(1)} X_i \right) \dots \right) \right) \right)$$

Ağın Eğitilmesi

Hata Fonksiyonları

Bağlanım

$$(x_i, y_i), i = 1, 2, \dots, n$$

$$x_i \in \mathbb{R}^p \quad y_i \in \mathbb{R}^K$$

$$E(\beta) = \sum_{i=1}^n E_i(\beta)$$

Kareler Toplamı

$$K = 1$$

$$E(\beta) = \frac{1}{2} \sum_{i=1}^n (\hat{y}(x_i, \beta) - y_i)^2$$

$$K > 1$$

$$E(\beta) = \frac{1}{2} \sum_{i=1}^n \|\hat{y}(x_i, \beta) - y_i\|^2 = \frac{1}{2} \sum_{i=1}^n \sum_{k=1}^K (\hat{y}_k(x_i, \beta) - y_{ik})^2$$

$$\hat{y}_k = \sigma(a_k) = a_k$$

$$\frac{\partial E_i}{\partial a_k} = \hat{y}_k(x_i, \beta) - y_{ik}$$

Ağın Eğitilmesi

Hata Fonksiyonları

K -Farklı İkili Sınıflandırma

$$(x_i, y_i), i = 1, 2, \dots, n$$
$$x_i \in \mathbb{R}^p \quad y_i \in \{0, 1\}^K$$

$$E(\beta) = \sum_{i=1}^n E_i(\beta)$$

Çapraz (Cross) Entropi

$$K = 1$$

$$E(\beta) = - \sum_{i=1}^n (y_i \ln(\hat{y}(x_i, \beta)) + (1 - y_i) \ln(1 - \hat{y}(x_i, \beta)))$$

$$K > 1$$

$$E(\beta) = - \sum_{i=1}^n \sum_{k=1}^K (y_{ik} \ln(\hat{y}_k(x_i, \beta)) + (1 - y_{ik}) \ln(1 - \hat{y}_k(x_i, \beta)))$$

$$\hat{y}_k = \sigma(a_k) = \frac{1}{1 + e^{-a_k}}$$

$$\frac{\partial E_i}{\partial a_k} = \hat{y}_k(x_i, \beta) - y_{ik}$$

Ağın Eğitilmesi

Hata Fonksiyonları

Çoklu Sınıflandırma

$(x_i, y_i), i = 1, 2, \dots, n$ $x_i \in \mathbb{R}^p$
 $y_i \in \mathbb{R}^K$, birim vektör

$$E(\beta) = \sum_{i=1}^n E_i(\beta)$$

Çapraz Entropi

$$E(\beta) = - \sum_{i=1}^n \sum_{k=1}^K y_{ik} \ln(\hat{y}_k(x_i, \beta))$$

$$\hat{y}_k = \sigma(a_k) = \frac{e^{a_k}}{\sum_{j=1}^K e^{a_j}}$$

$$\frac{\partial E_i}{\partial a_k} = \hat{y}_k(x_i, \beta) - y_{ik}$$

Ağın Eğitilmesi

Optimizasyon

$$\min_{\beta} E(\beta) = \min_{\beta} \sum_{i=1}^n E_i(\beta)$$

$$\nabla E(\beta) = \sum_{i=1}^n \nabla E_i(\beta) = 0$$

Yığın Gradyan İnişi (Batch Gradient Descent)

$$\beta^{\tau+1} = \beta^{\tau} - \eta \sum_{i=1}^n \nabla E_i(\beta^{\tau})$$

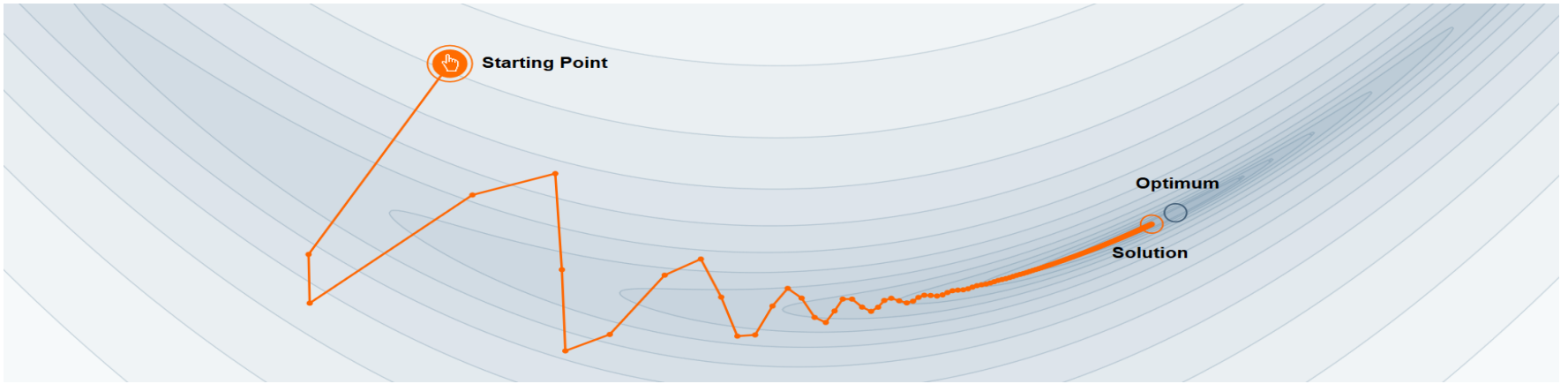
τ : iterasyon
 η : öğrenme hızı
(learning rate)

Rassal (Stochastic) Gradyan İnişi (RGI)

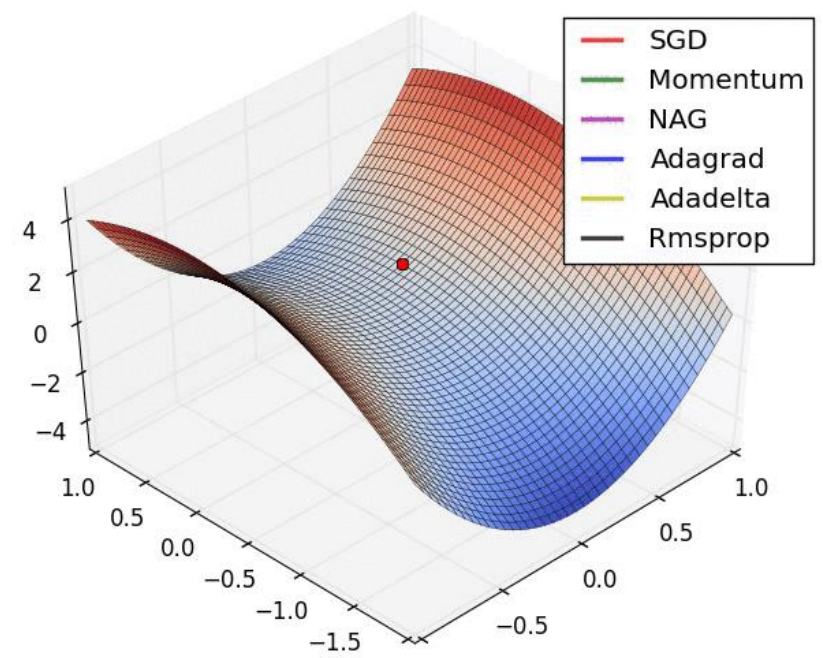
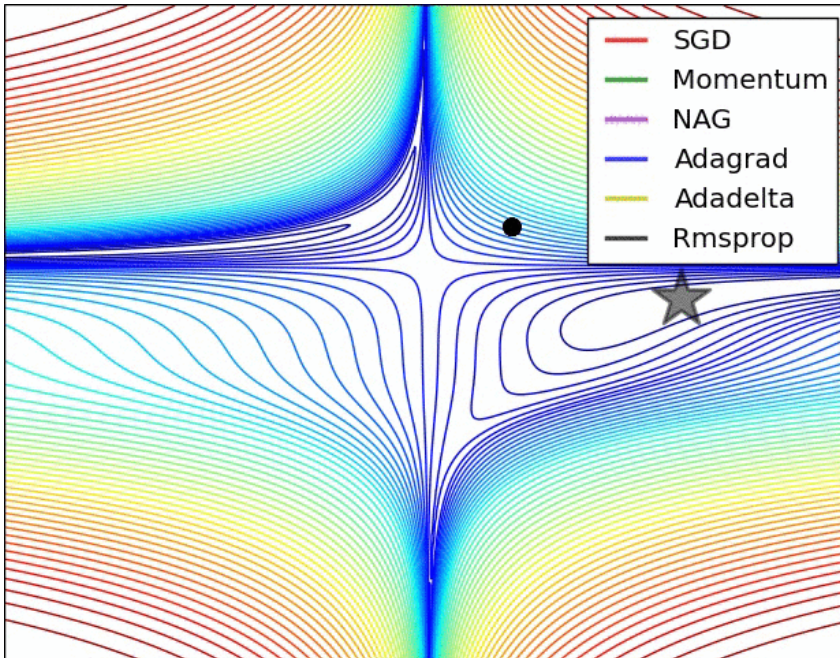
$$\beta^{\tau+1} = \beta^{\tau} - \eta \nabla E_j(\beta^{\tau}), \quad j \in \{1, \dots, n\}$$

Mini-yığın RGI

$$\beta^{\tau+1} = \beta^{\tau} - \eta \sum_{j \in \mathcal{J}} \nabla E_j(\beta^{\tau}), \quad \mathcal{J} \subseteq \{1, \dots, n\}$$



(kaynak)



(kaynak)

Ağın Eğitilmesi

Geri Yayılım (Backpropagation)

$$\nabla E(\beta) = \sum_{i=1}^n \nabla E_i(\beta) = 0 \qquad \frac{\partial E}{\partial \beta_{ts}^{(l)}} = \sum_{i=1}^n \frac{\partial E_i}{\partial \beta_{ts}^{(l)}} \quad ?$$

Hatırlatma

$$E(\beta) = - \sum_{i=1}^n \sum_{k=1}^K y_{ik} \ln(\hat{y}_k(x_i, \beta)) \qquad E(\beta) = \frac{1}{2} \sum_{i=1}^n \sum_{k=1}^K (\hat{y}_k(x_i, \beta) - y_{ik})^2 \qquad \frac{\partial E_i}{\partial a_k} = \hat{y}_k(x_i, \beta) - y_{ik}$$

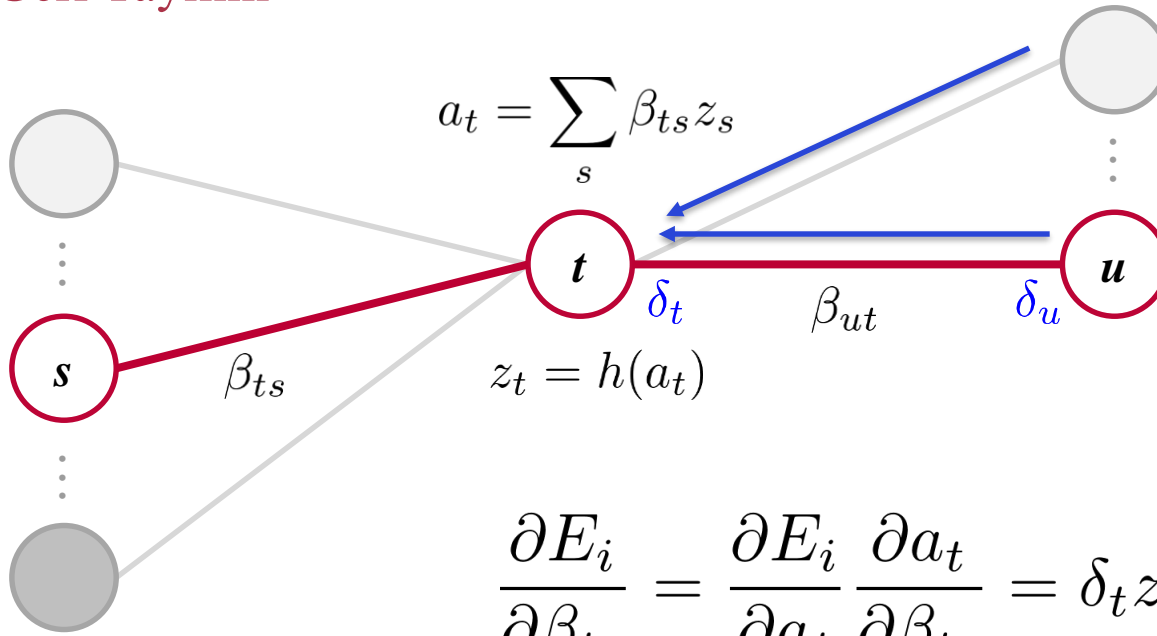
$$\hat{y}_k(X, \beta) = \sigma \left(\sum_j \beta_{kj}^{(\ell)} h \left(\sum_s \beta_{js}^{(\ell-1)} h \left(\dots h \left(\sum_i \beta_{ji}^{(1)} X_i \right) \dots \right) \right) \right)$$

Zincir kuralını arka arkaya uygulasak?

Evet ama bunu doğrudan ağın üzerinde de yapabiliriz!

Ağın Eğitilmesi

Geri Yayılım



$$\hat{y}_{ik} \triangleq \hat{y}_k(x_i, \beta)$$

$$\frac{\partial E_i}{\partial a_k} = \hat{y}_{ik} - y_{ik}$$

$$\frac{\partial E_i}{\partial \beta_{ts}} = \frac{\partial E_i}{\partial a_t} \frac{\partial a_t}{\partial \beta_{ts}} = \delta_t z_s$$

$$\delta_t \triangleq \frac{\partial E_i}{\partial a_t}$$

$$\delta_t = \frac{\partial E_i}{\partial a_t} = \sum_u \frac{\partial E_i}{\partial a_u} \frac{\partial a_u}{\partial a_t} = h'(a_t) \sum_u \delta_u \beta_{ut}$$

$$\delta_k = \frac{\partial E_i}{\partial a_k} = \hat{y}_{ik} - y_{ik}, \quad k = 1, \dots, K$$

Ağın Eğitilmesi

Geri Yayılım

x_i girdi vektörünü uygula ve aşağıdaki şekilde ağ üzerinde yayılımını sağla:

$$a_t = \sum_s \beta_{ts} z_s, \quad z_t = h(a_t) \quad \text{ve} \quad \hat{y}_k = \sigma(a_k)$$

Çıktı birimleri için şu hesapları yap:

$$\delta_k = \frac{\partial E_i}{\partial a_k} = \hat{y}_{ik} - y_{ik}, \quad k = 1, \dots, K$$

Gizli katmanlar için aşağıdaki ifadeleri uygula:

$$\delta_t = \frac{\partial E_i}{\partial a_t} = \sum_u \frac{\partial E_i}{\partial a_u} \frac{\partial a_u}{\partial a_t} = h'(a_t) \sum_u \delta_u \beta_{ut}$$

Gradyanın bileşenlerini hesapla:

$$\frac{\partial E_i}{\partial \beta_{ts}} = \frac{\partial E_i}{\partial a_t} \frac{\partial a_t}{\partial \beta_{ts}} = \delta_t z_s$$

Ağın Eğitilmesi

Aşırı Modelleme

- Çapraz Geçerlilik Sınaması
- **Düğüm Silme Yöntemi (Dropout Method):** Rastgele bazı düğümleri ve onların bağlantılarını ağdan çıkarma (www)

Ağın Eğitilmesi

Düzenlileştirme (Regularization)

ℓ_2 -norm

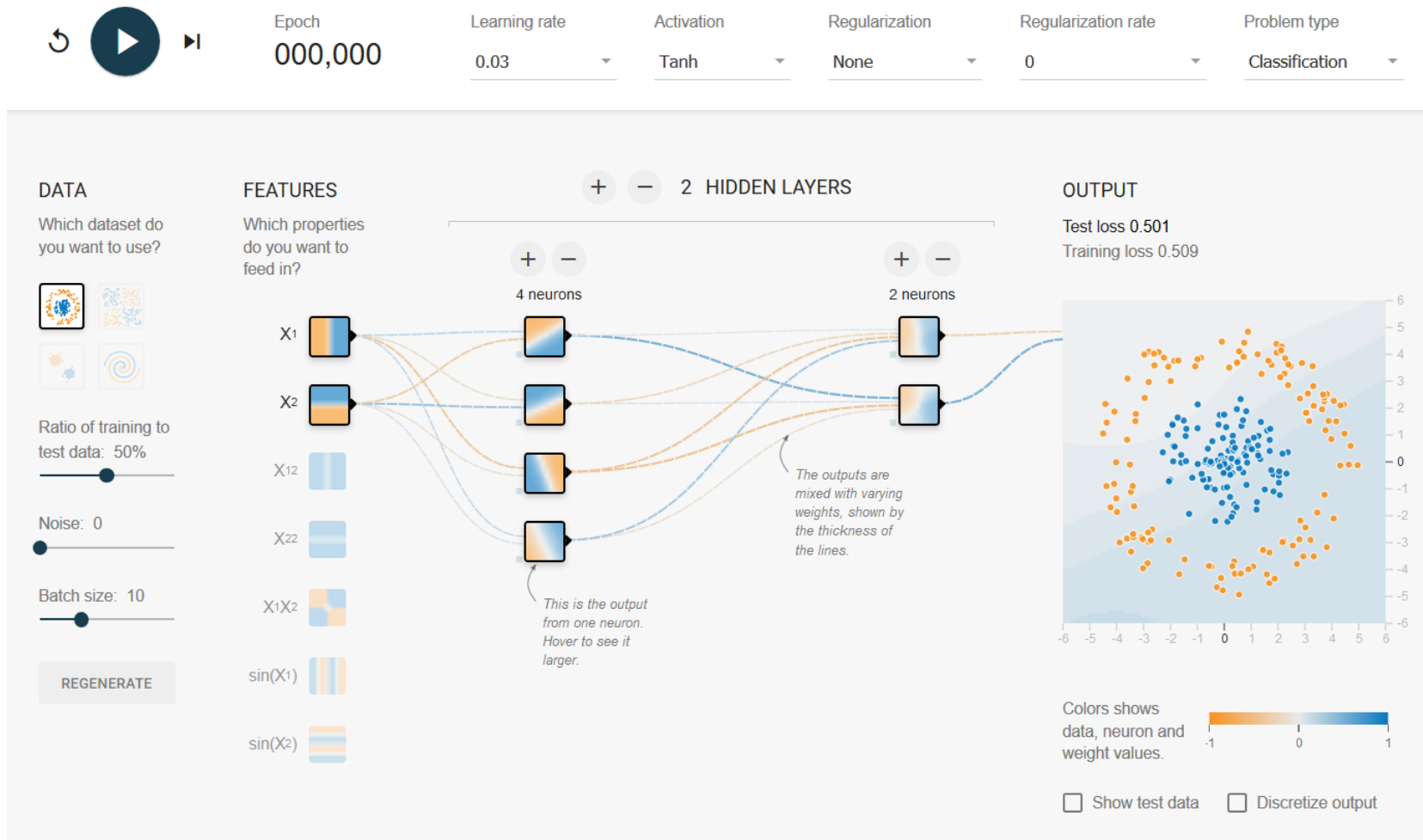
$$\min_{\beta} E(\beta) + \lambda \sum_{s,t} \beta_{st}^2$$

ℓ_1 -norm

$$\min_{\beta} E(\beta) + \lambda \sum_{s,t} |\beta_{st}|$$

türevlenebilir değil!

Tensorflow



Ağın Eğitilmesi

Sıkça Sorulan Sorular

- Kaç gizli katman? Her katmanda kaç düğüm? Hangi aktivasyon fonksiyonları?
 - *Neural Network Design*, M.T. Hagan vd. ([PDF](#))
 - Sıkça Sorulan Sorular ([www](#))
 - Ve daha pek çoğu... (StackExchange!)
- TensorFlow oyun alanı ([www](#))

Derin Öğrenme

Çok azalan/artan Gradyan

- Son katmanlardan ilk katmanlara doğru giderken gradyan azalır (artar)
- Eğitim ya çok uzun sürer ya da başarısız olur

$$\frac{\partial E_i}{\partial \beta_{ts}} = \delta_t z_s$$

$$= \left(\sigma'(a_t) \sum_u \delta_u \beta_{ut} \right) z_s$$

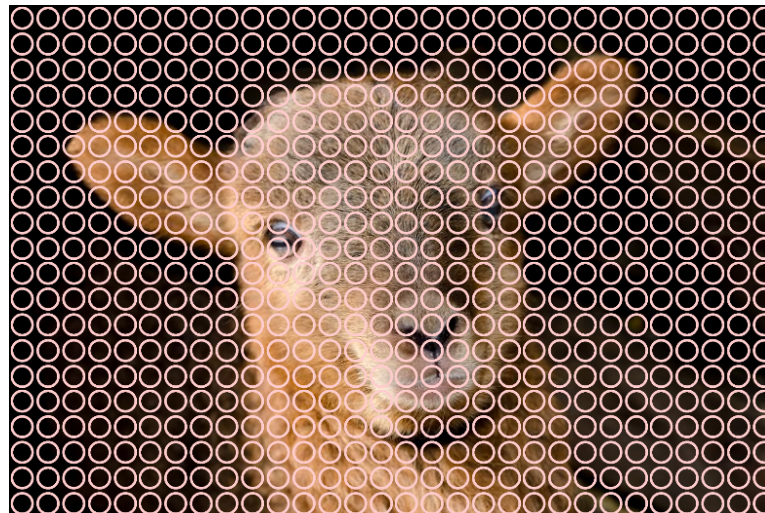
$$= \left(\underbrace{\sigma'(a_t)}_{< 1} \sum_u \left(\underbrace{\sigma'(a_u)}_{< 1} \sum_v \delta_v \beta_{vu} \right) \underbrace{\beta_{ut}}_{\leq 1} \right) z_s$$

$$\sigma(a) = \frac{1}{1 + e^{-a}} \implies \max_{a \in \mathbb{R}} \sigma'(a) = \frac{1}{4}$$

- Küçük başlangıç ağırlıkları çok azalan gradyana sebep olur
- Büyük başlangıç ağırlıkları gradyanın çok büyümesine sebep olur

Evrişimsel Sinir Ağı - ESA (Convolutional Neural Network)

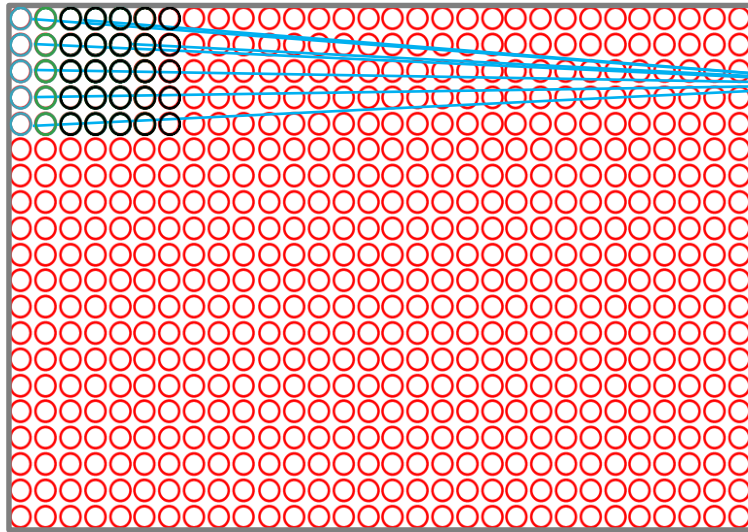
- Özenle kurgulanmış çok katmanlı sinir ağı
- Verinin uzamsal yapısını (spatial structure) kullanır (nesne tanıma)
- İlk katmanlar kolay örüntüleri öğrenir
- Sonraki katmanlar da ilk katmanları yeniden şekillendirir
- Gizli katmanlar lokal yapıyı ortaya çıkarır
- Aktivasyon fonksiyonu olarak ReLU kullanılır
- Toplam parametre sayısı ciddi oranda düşer



ESA - Kurgulama

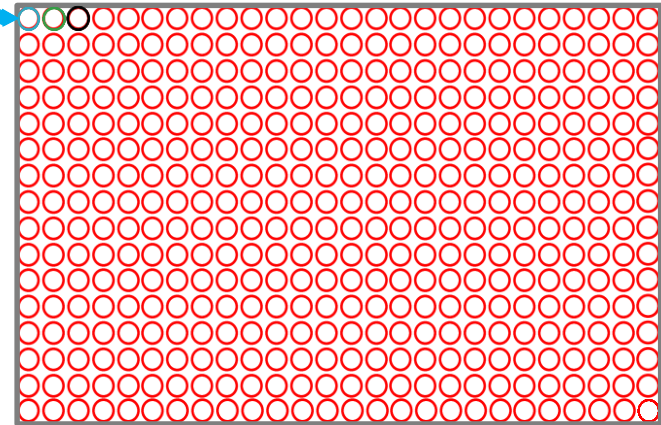
lokal alıcı alanları (5×5)

öznitelik eşlemi (feature map)
(ReLU kullanır)



adım uzunluğu(one)

girdi: 20×30 nöron

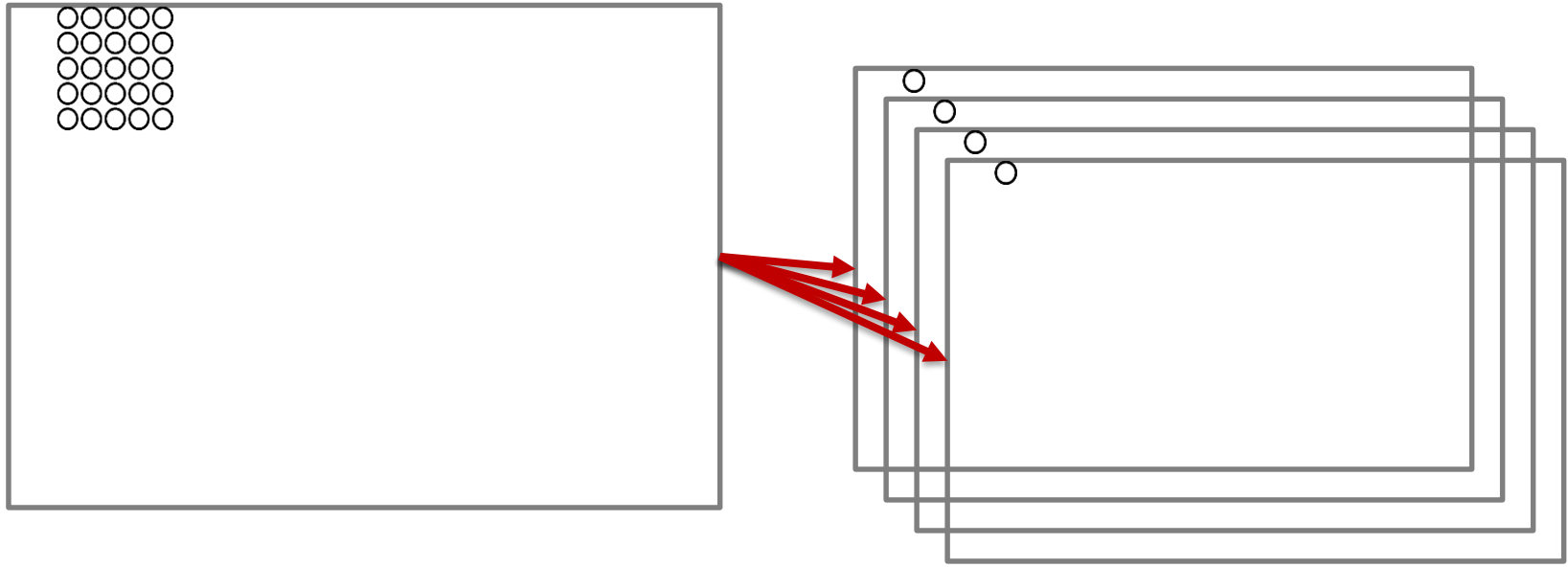


16×26 nöron

Ağırlıklar ve yanlılık her bir nöron tarafından paylaşılır
(paylaşılan ağırlıklar ve yanlılık bir **filtre** veya **çekirdek** tanımlar)

ESA – Gizli Katman

birçok öznitelik eşlemi

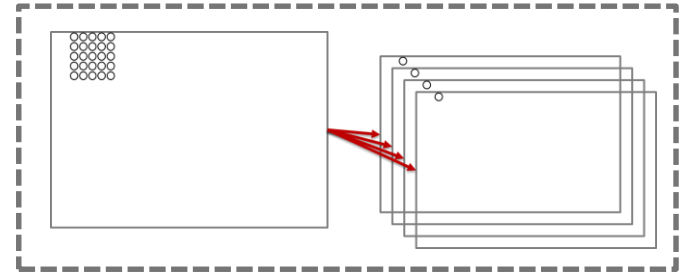


girdi: 20×30 nöron

ilk gizli katman: $4 \times 16 \times 26$ nöron
(evrimsel katman)

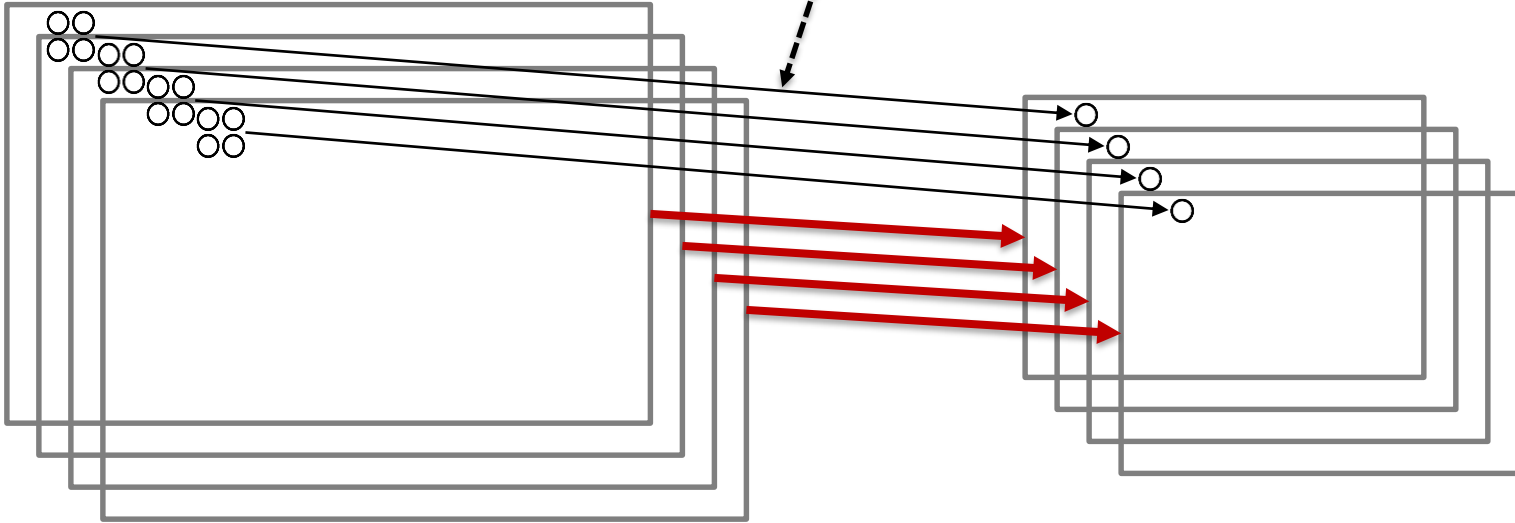
ağırlıkların ve yanlılıkların sayısı = $(25 + 1) \times 4 = 104$

ESA – Ortaklama Katmanı (Pooling Layer)



max ortaklama

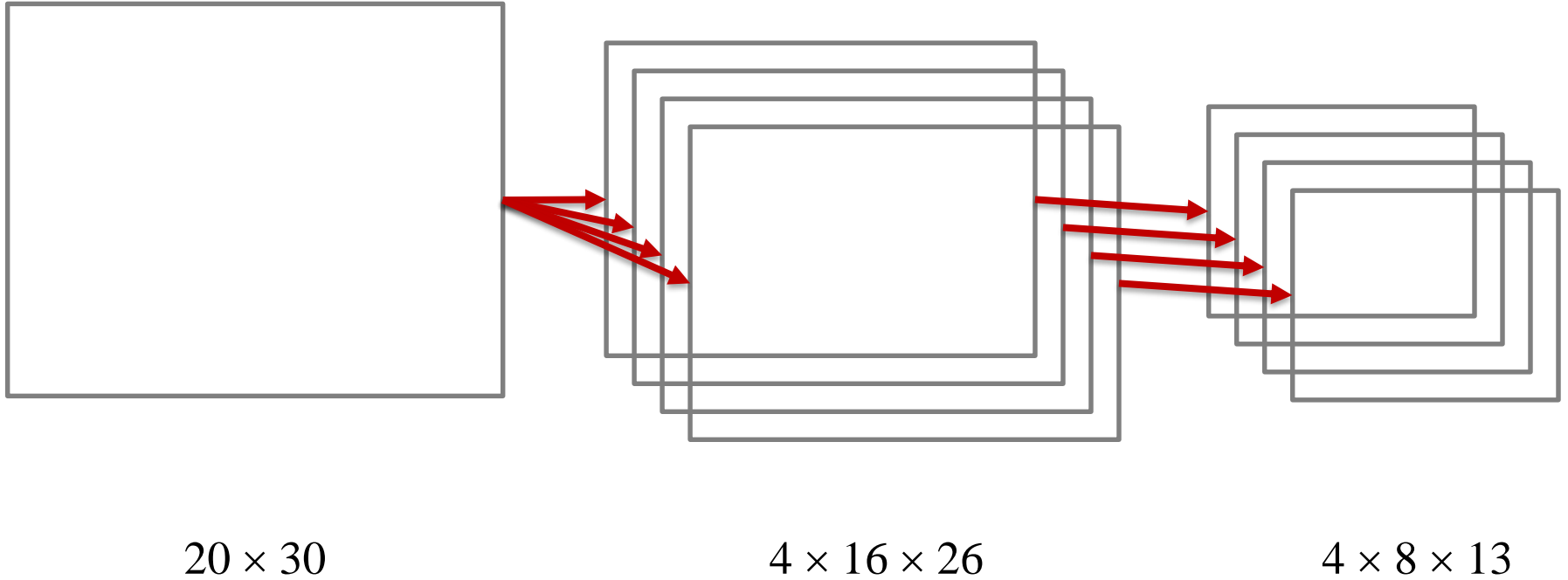
(2 × 2 alanda maksimum aktivasyon)



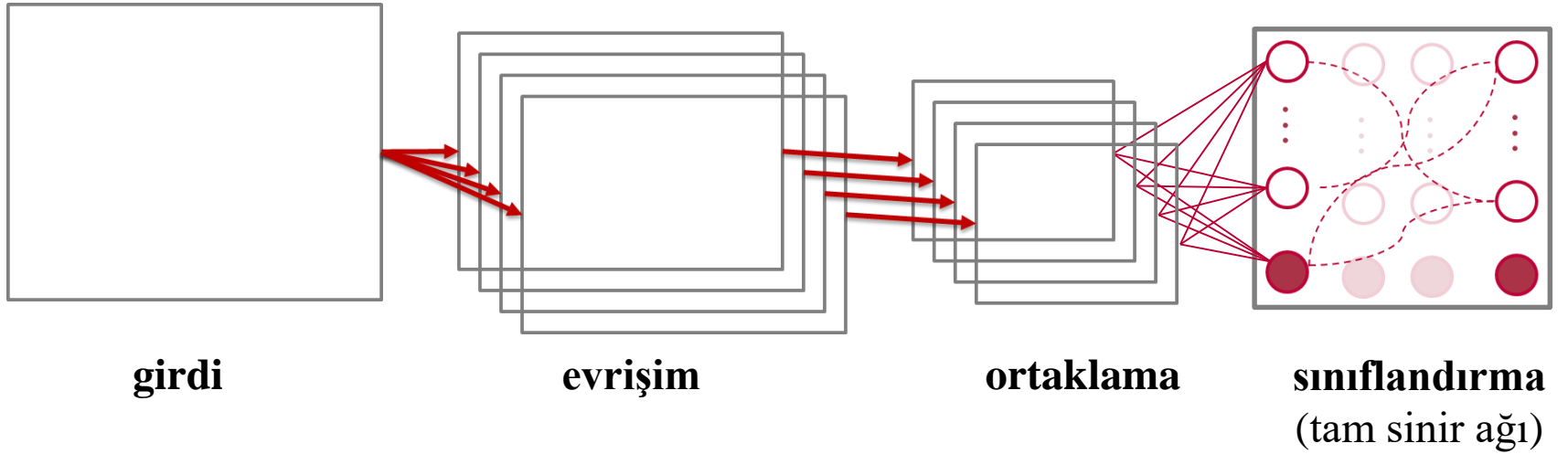
ilk gizli katman: $4 \times 16 \times 26$ nöron

ikinci gizli katman: $4 \times 8 \times 13$ nöron
(ortaklama katmanı)

ESA – Birleştirme Katmanı (Merging Layers)

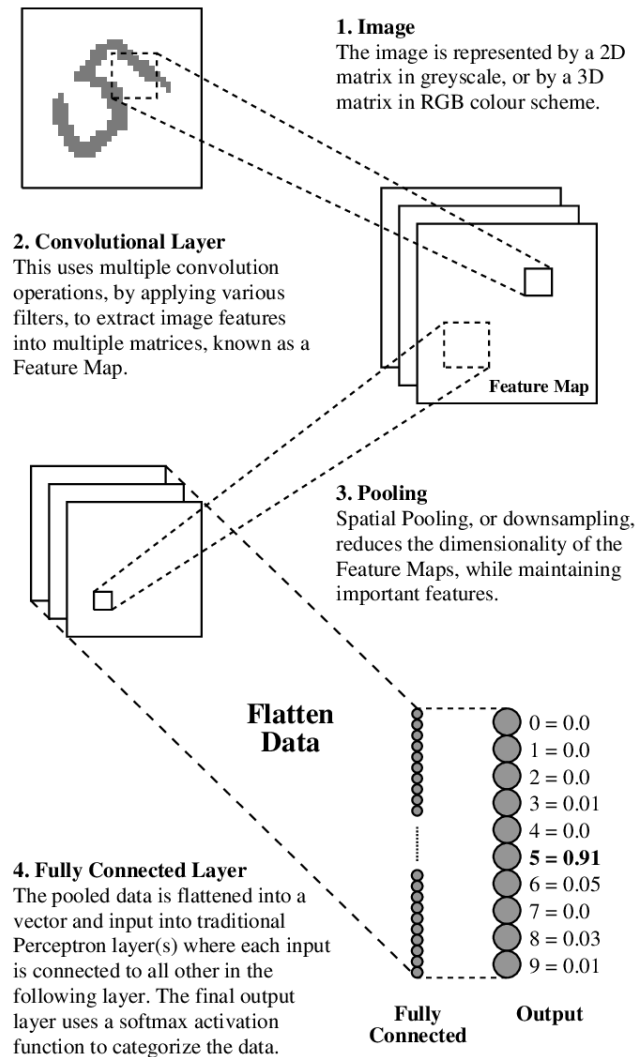


ESA – Bütün Ağ

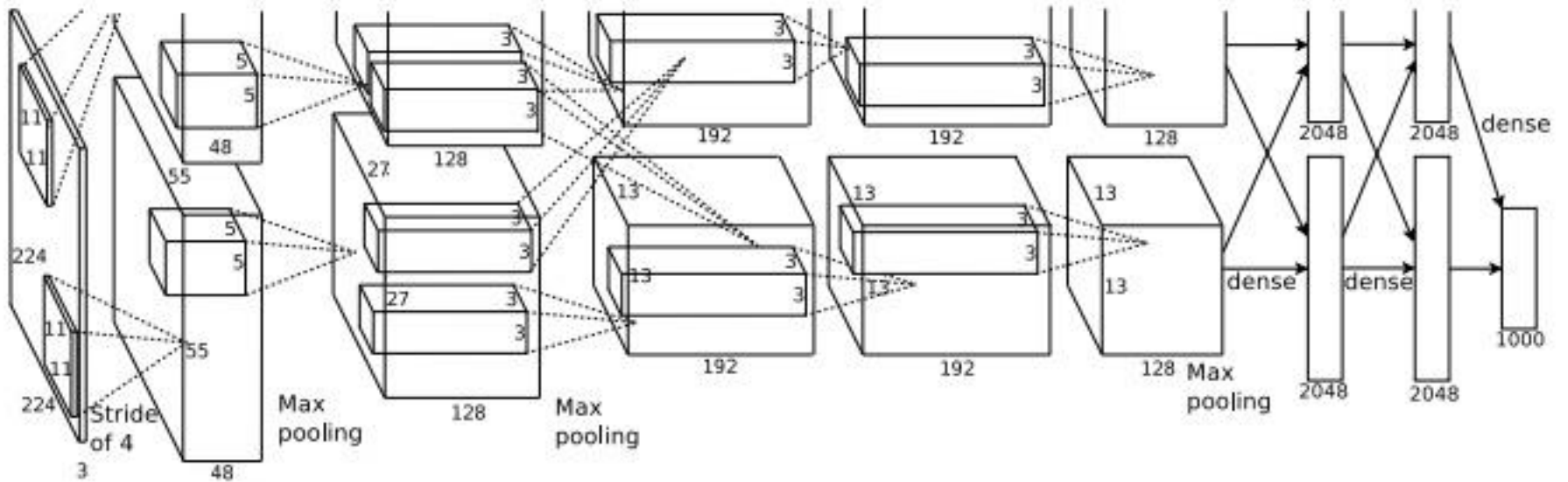


- Sınıflandırma için son aşamada genellikle tek katman kullanılır
- Birden çok evrişim katmanı olabilir
- Çok fazla matris veya vektör çarpımı yapılır
- Ekran kartları (GPU) ile vektörleştirme (vectorization) kullanılır

ESA – Nesne Tanıma

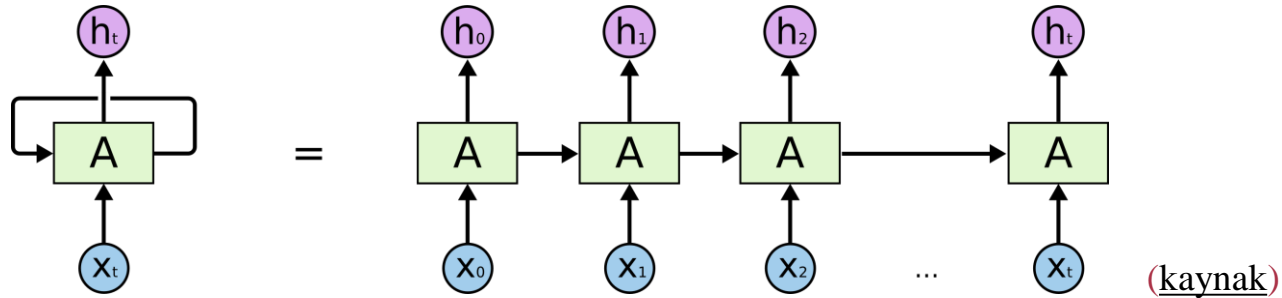


ESA – ImageNet Sınıflandırma



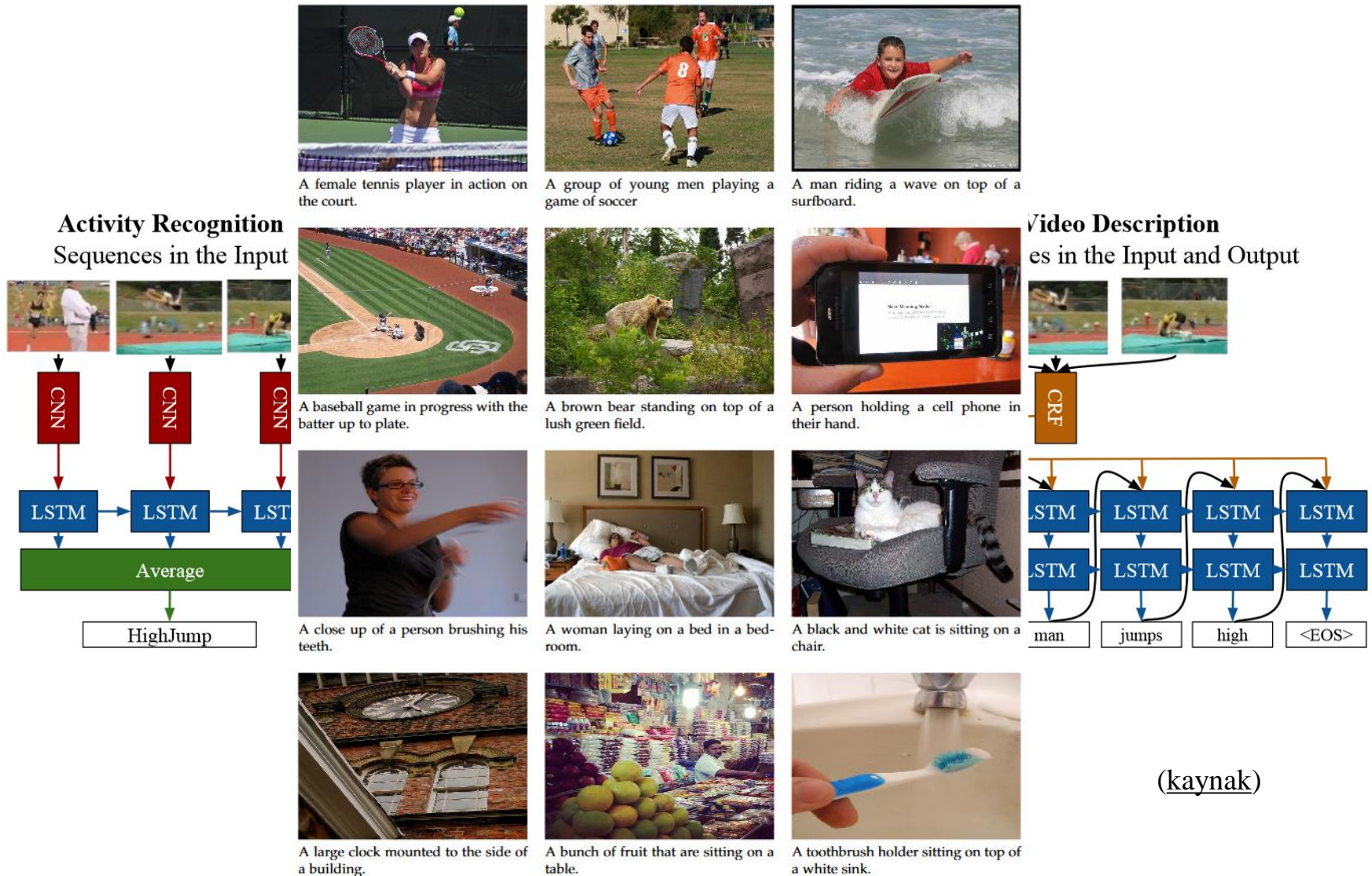
(kaynak)

Yineleyen Sinir Ağları - YSA (Recurrent Neural Network)



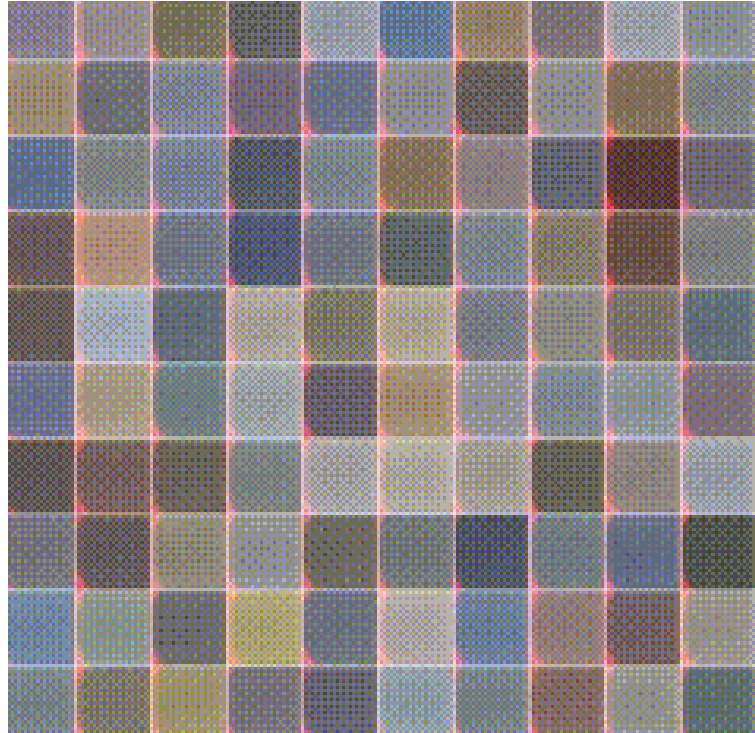
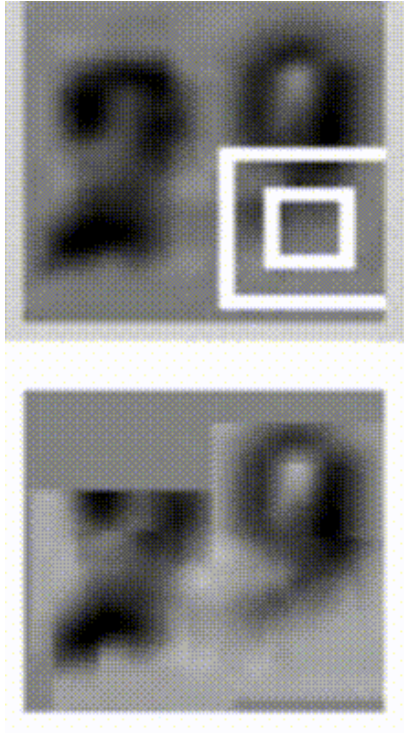
- Geçmiş bilginin kullanımının öğrenilmesi: yinelenen alt ağlar (subnetworks)
- Farklı evrelerde değişkenlik gösteren veri – zaman serileri analizi
- Girdi dizileri – Çıktı dizileri
- Uzun dönemlere zayıf bağlılık (çok azalan gradyan)
 - Uzun Kısa Dönemli Hafıza (Long Short Term Memory)
 - GRU (Gated Feedback Recurrent Neural Networks)
- Yineleyen sinir ağları hakkında iyi bir çalışma ([link](#))

YSA – Görüntülere Altyazı Ekleme (Image Captioning)



(kaynak)

YSA – Karakter Üretimi (Character Generation)

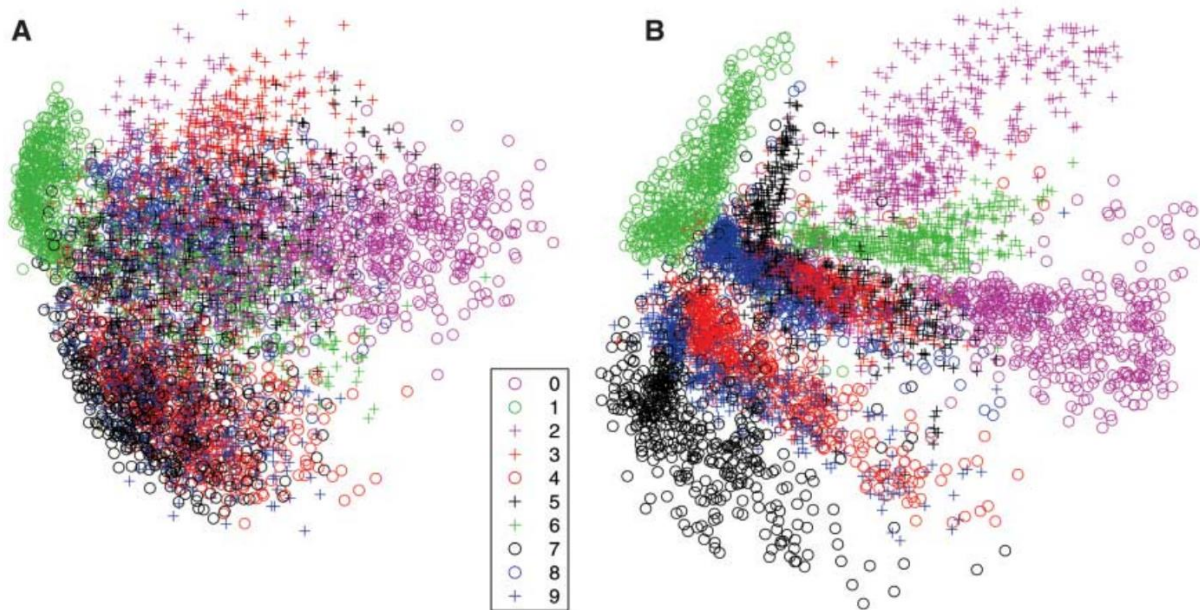


(kaynak)

Özkodlayıcılar (Autoencoders)

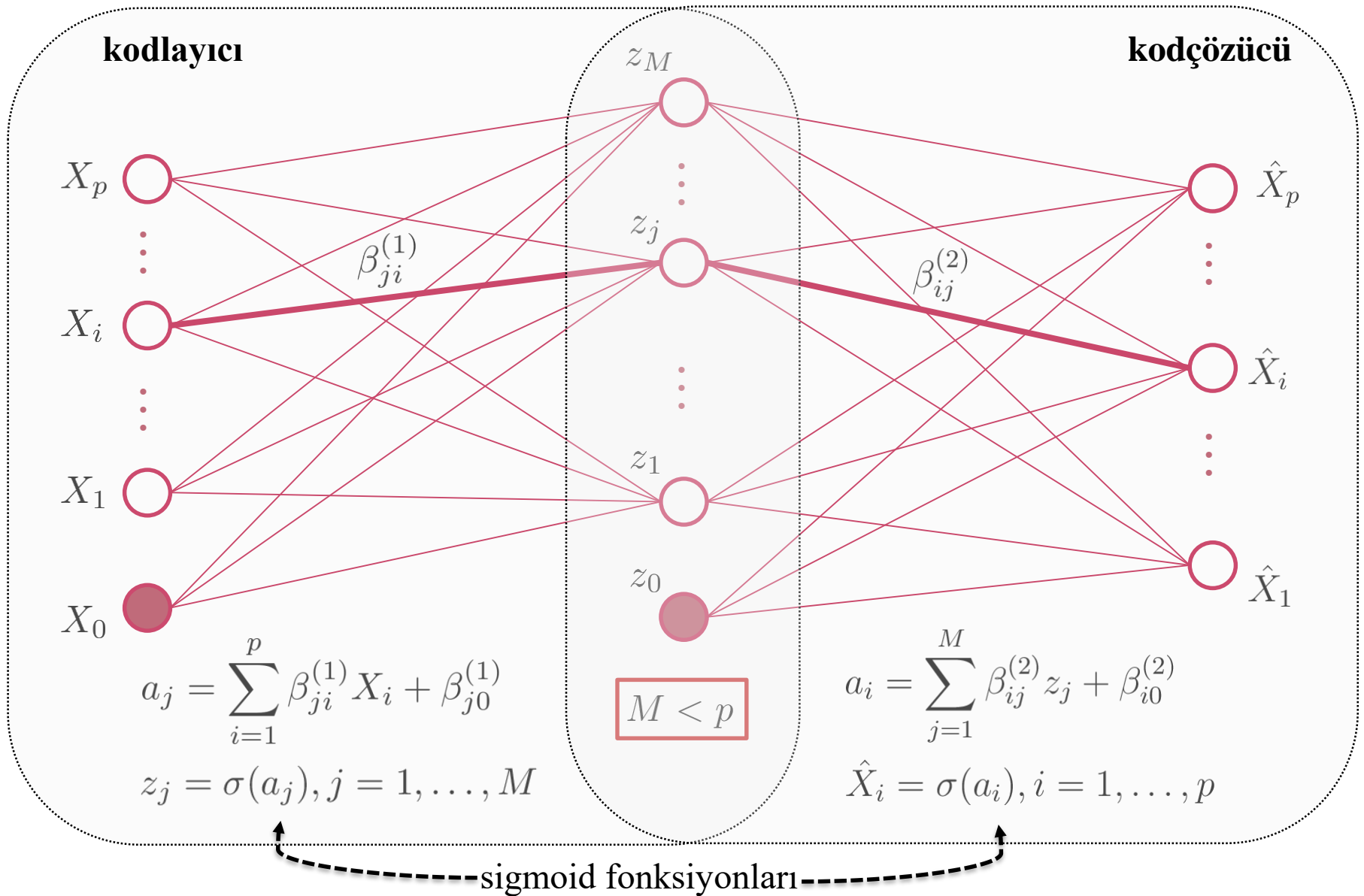
- GÜdümsüz öğrenme
- İlke: Öznitelikleri belirleme, girdiyi tekrar oluşturma
- Boyut küçültme tekniği

Fig. 3. (A) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. (B) The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (8).



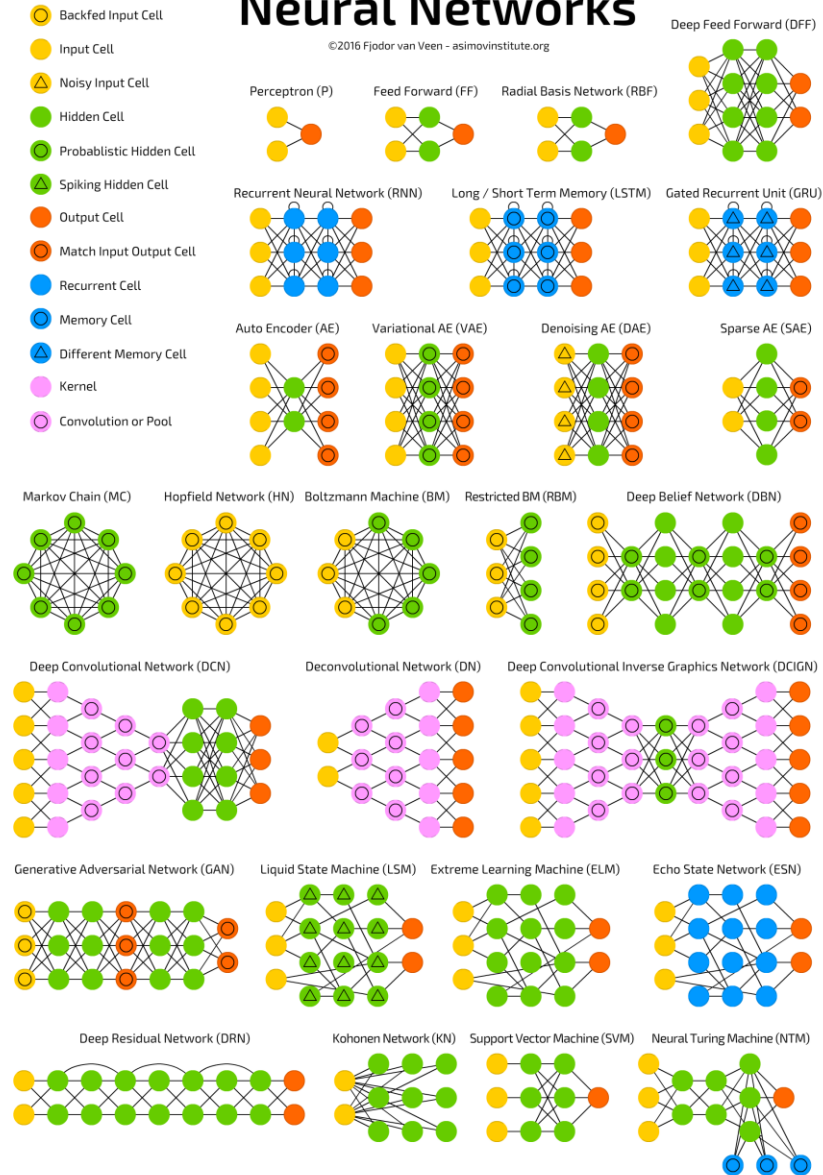
(kaynak)

Özkodlayıcılar (Autoencoders)



A mostly complete chart of
Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org



(kaynak - 2016)

Barn owl or apple



(kaynak)

Özet

- İleribeslemeli yapay sinir ağları
- Hata fonksiyonları
- Geri yayılım
- Aşırı öğrenme
- Çok azalan/artan gradyan
- Derin Öğrenme – GÜDÜMLÜ
- Derin Öğrenme – GÜDÜMSÜZ