FİGES

Deep Learning A-Z™
Python ile Derin Öğrenme
Dr. Merve Ayyüce KIZRAK

Deep Learning A-Z™
Python ile Derin Öğrenme

Dr. Merve Ayyüce KIZRAK

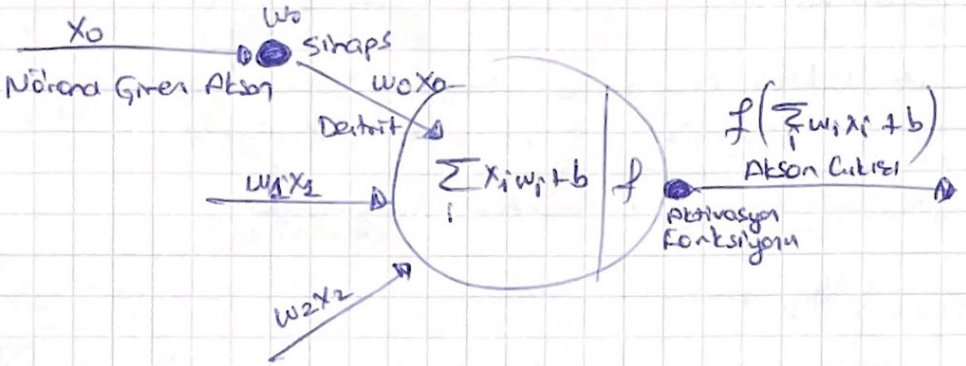
UDEMY

Doğukan Mehmet KILINÇ

FİGES

...../...../20.....

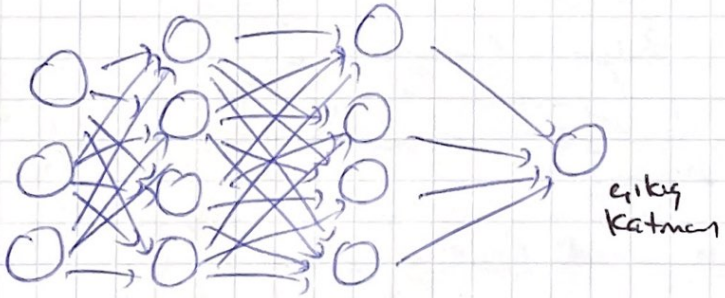
İnsanlardaki sinir ağlarının matematiksel modellemesi aşağıdadır.



Buradaki fonksiyon çıkışı nihai çıkış veya bir başka fonksiyonun girişi olabilir.

$$\text{çıkış} \leftarrow (y) = (w) \times (x) + (b)$$

w: ağırlıklar, x: girişler, b: bias



FİGES

...../...../20.....

EĞİTİCİLİ (GÖZETİMLİ, SUPERVISED) ÖĞRENME

Sınıflandırma (Classification) Yöntemleri

→ Destek Vektör Makineleri (Support Vector machines)
→ Ayrıştırma Analizi (Discriminant Analysis)
→ Naif Bayes (Naive Bayes)
→ En Yakın Komşu (Nearest Neighbor)

Bağlanım (Regression) Yöntemleri

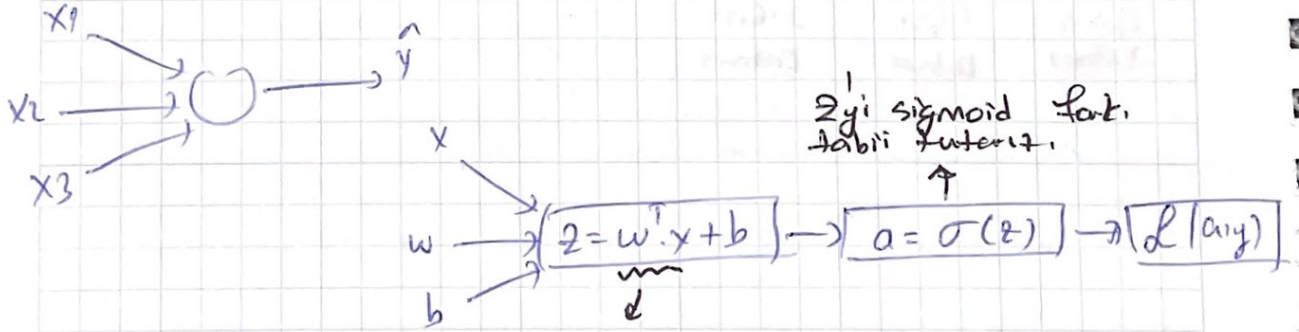
→ Lineer Bağlanım
→ Karar Ağaçları (Decision Trees)
→ Yapay Sinir Ağları

EĞİTİCİSİZ (GÖZETİMSİZ, ~~UNSUPERVISED~~ REINFORCEMENT) ÖĞRENME

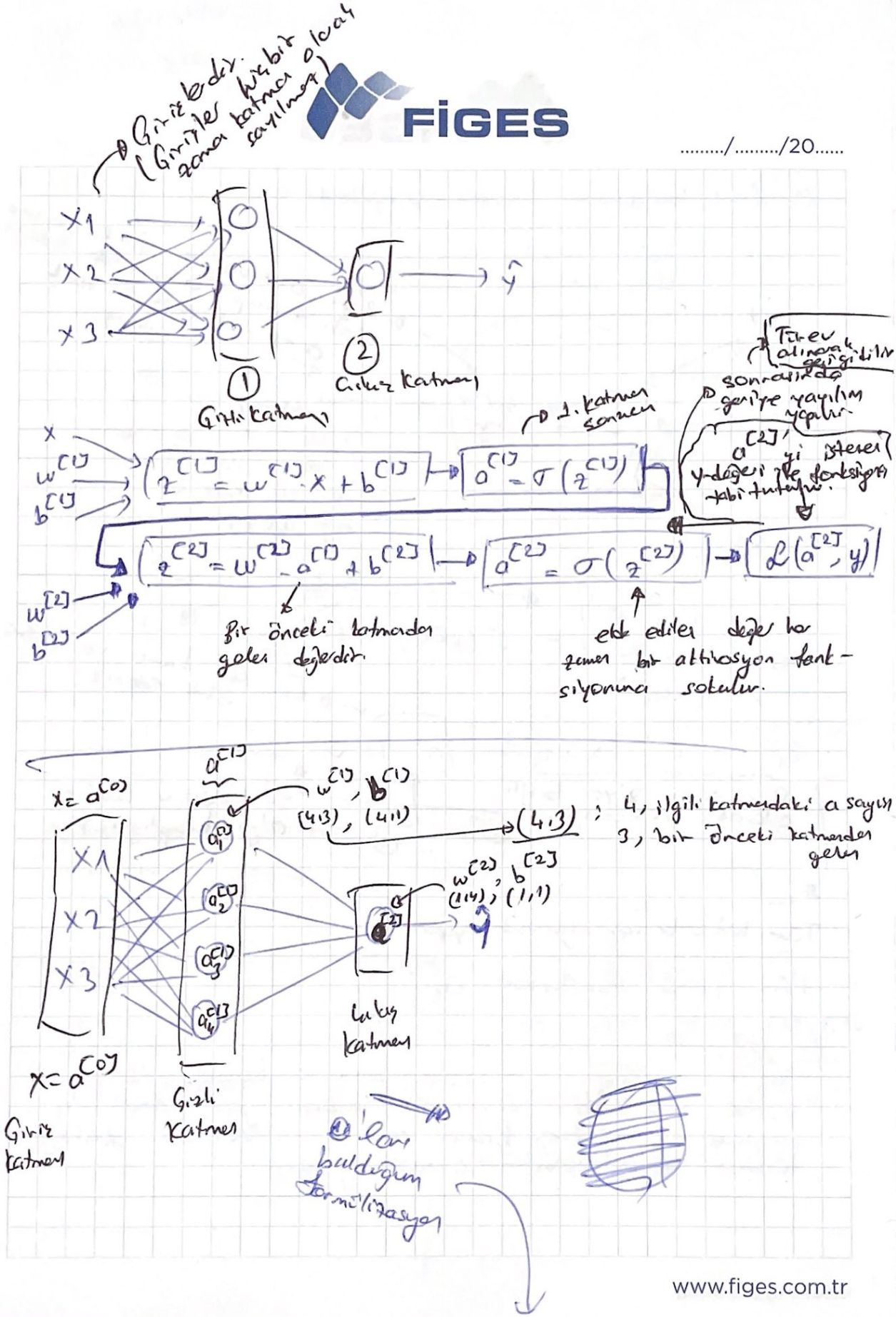
Kümeleme Yöntemleri

→ K-Ortalama, Bulanık C-Ortalama
→ Gauss Karışımı (Gaussian Mixture)
→ Saklı Markov Modeli (Hidden markov model)
→ Yapay Sinir Ağları

PEKİ ÖĞRENİM NASIL GERÇEKLEŞİYOR?



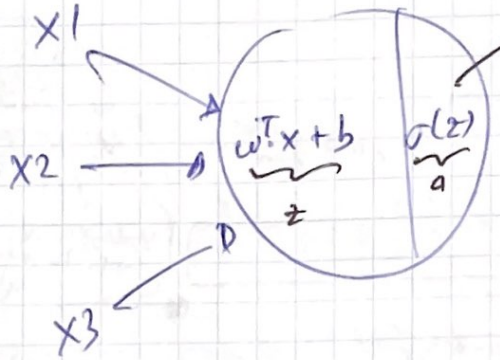
Girişlerle ağırlıklar çarpılır.
Bunlar bir vektör olduğu için birinin satır sayısı ile diğerinin sütun sayısı eşit olmalı. Bu yüzden ağırlığın transpozunu alıyoruz.


Girişlerdir. (Girişler kendi başına katman olarak sayılmaz)
x1
x2
x3
1
Gizli katman
2
Çıkış katmanı
ŷ
1. katmanın sonucu
Türev alınarak geriye doğru gidilirken sonrasında geriye yayılım yapılır.
a[2]'yi istenen y değeri ile fonksiyona tabi tutuyor.
x
w[1]
b[1]
z[1] = w[1]·x + b[1]
a[1] = σ(z[1])
z[2] = w[2]·a[1] + b[2]
a[2] = σ(z[2])
ℒ(a[2], y)
w[2]
b[2]
Bir önceki katmandan gelen değerdir.
elde edilen değer her zaman bir aktivasyon fonksiyonuna sokulur.
x = a[0]
a[1]
w[1], b[1]
(4,3), (4,1)
(4,3)
4, ilgili katmandaki a sayısı
3, bir önceki katmandan gelen
w[2], b[2]
(1,4), (1,1)
ŷ
Çıkış katmanı
x = a[0]
Giriş katmanı
Gizli katmanı
Öğrendiğim formülizasyon

FİGES

...../...../20.....

a'ları bulduğum model aşağıdadır.



$z = w^T \cdot x + b$

$a = \sigma(z)$

Budur!

$$a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ a_2^{[1]} \\ a_3^{[1]} \\ a_4^{[1]} \end{bmatrix} = \sigma(z^{[1]})$$

En son bunu ve 2,3,4.'ü de elde ederiz.

$z_1^{[1]} = w_1^{[1].T} \cdot x + b_1^{[1]}$, $a_1^{[1]} = \sigma(z_1^{[1]})$

Buradaki z'ye bağlı olacak aktivasyon fonksiyonu sonucu a'yı hesaplıyoruz

① ②

...

Tüm katmanlar için aynısını yaparız.

İleri yönde beslerken $\begin{bmatrix} a_1^{[1]}, \\ a_2^{[1]}, \\ a_3^{[1]}, \\ a_4^{[1]} \end{bmatrix}$ buluruz. Fakat ileri yönde beslebildikten sonra asıl mesele geri yönde yayılım yapmaktır. Bunun için de aktivasyon fonksiyonlarının türevlenebilir olması gerekir.

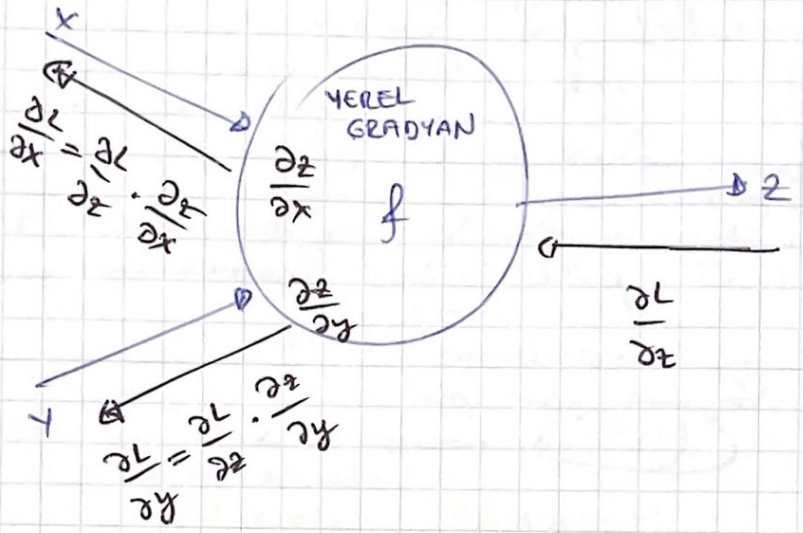
W: Ağırlıklar
X: Girişler

FİGES

...../...../20.....

GERİYE YAYILIM ALGORİTMASI

Giriş bilgisine ulaşıldığında sistem sürekli ileri ve geri yönde beslemeye devam eder.



Her geri yayılımdan sonra parametreler güncellenir. Fonksiyon minimize edilene kadar bu devam eder. Bu işlemin tamamına öğrenme işlemi denir.

İleriye Yayılım Algoritması (Feed forward Propagation)

$$Z^{[1]} = W^{[1]} \cdot X + b^{[1]}$$

$$A^{[1]} = g^{[1]}(Z^{[1]})$$

$$Z^{[2]} = W^{[2]} \cdot A^{[1]} + b^{[2]}$$

$$A^{[2]} = g^{[2]}(Z^{[2]}) = \sigma(Z^{[2]})$$

Geriye Yayılım Algoritması (Back Propagation)

$$dZ^{[2]} = A^{[2]} - Y$$

$$dW^{[2]} = \frac{1}{m} dZ^{[2]} \cdot A^{[1]T}$$

$$dZ^{[1]} = \underbrace{W^{[2]T} \cdot dZ^{[2]}}_{(n^{[1]} \cdot m)} * \underbrace{g^{[1]} \cdot Z^{[1]}}_{(n^{[1]} \cdot m)}$$

$$dW^{[1]} = \frac{1}{m} dZ^{[1]} \cdot X^{T}$$

FİGES

...../...../20.....

ARA HATIRLATMA

Bu şekilde çarpılır

$$\begin{bmatrix} 2 & 3 & 1 \\ 2 & -7 & 4 \end{bmatrix} \times \begin{bmatrix} 3 & 4 & 5 \\ 1 & 1 & 4 \\ 2 & 1 & 4 \end{bmatrix} = \begin{bmatrix} (6+3+2) & (8+3+1) & (10+12+4) \\ (6-7+8) & (8-7+4) & (10-28+16) \end{bmatrix}$$

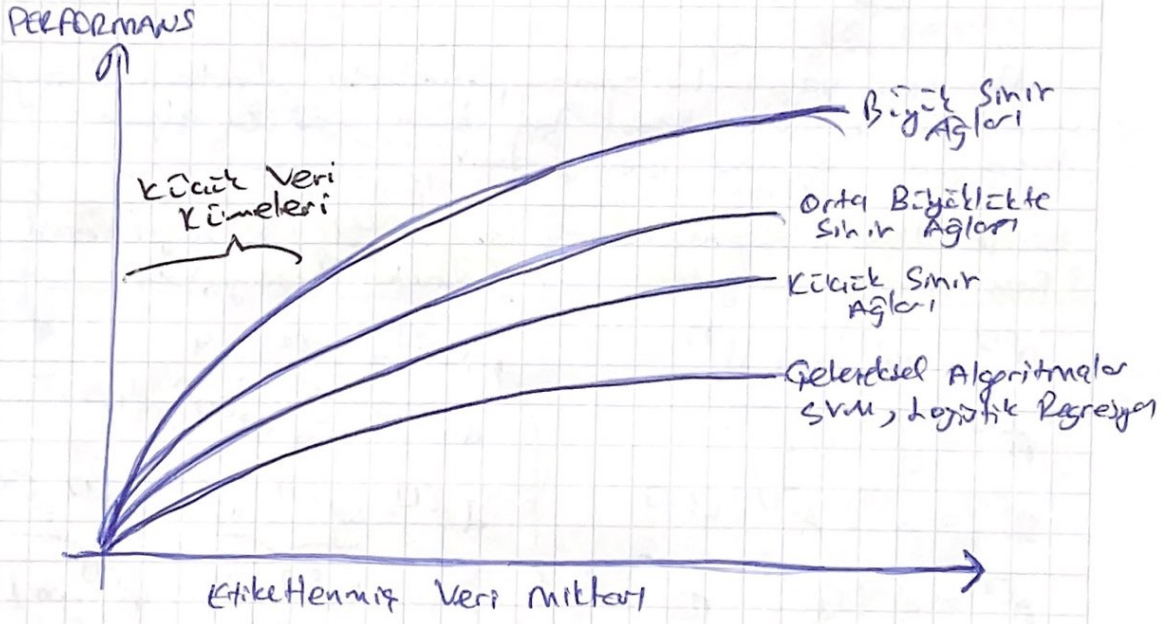
2×3 ve 3×3

Bu iki matrisi çarpabilmek için bu değerlerin eşit olması gerekir.

$$= \begin{bmatrix} 11 & 12 & 26 \\ 7 & 5 & -2 \end{bmatrix}_{2 \times 3}$$

Yeni elde edilen matrisimiz bu olur.

ve yeni matrisimiz bu boyuta eşit olur. (2×3)



Veri Az ise Derin Sinir Ağı kullanmaya gerek yok çünkü işlem yükü fazla. Geleneksel Algoritmalar kullanılabilir. Ancak etiketlenmiş veri miktarı arttığında performans ciddi bir şekilde değişeceğinden Büyük Sinir Ağları tercih edilebilir.

FİGES

03./01./2024

HİPERPARAMETRE NEDİR?

Veriden öğrenen makine öğrenmesi ya da derin öğrenme modelleri tasarlarken ne olması gerektiğine çeşitli koşullar göz önüne alınarak modeli tasarlayan kişiye bırakılmış; probleme, veri setlerine ve donanıma da bağlı olarak değişkenlik gösteren parametrelere hiperparametre denir. Tamamen değişken ve hiçbir zaman net bir şey söyleyemeyeceğimiz konudur.

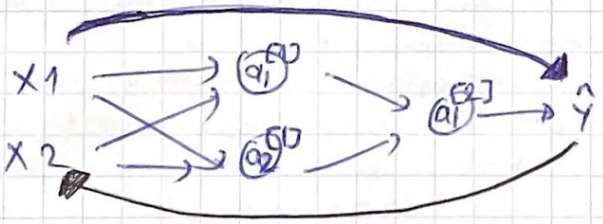
REGULARIZATION: Modele uygun parametrelerin hiperparametre optimizasyonu ile seçilmesidir.

NEDEN BASİT MODEL SEÇMELİYİZ?

1) Hesaplama karmaşıklığının düşük olması gerekiyor.
2) Eğitimin kolay olması, yer kaplamaması için.
3) Kolay açıklanabilir olması için.
4) Genelleştirilebilir olmalı.

BAŞLANGIÇ AĞIRLIKLARININ BELİRLENMESİ

- Ağırlıklar sıfır (0) ile başlatılmamalı.
- Rastgele küçük sayılarla başlatmakta fayda var.



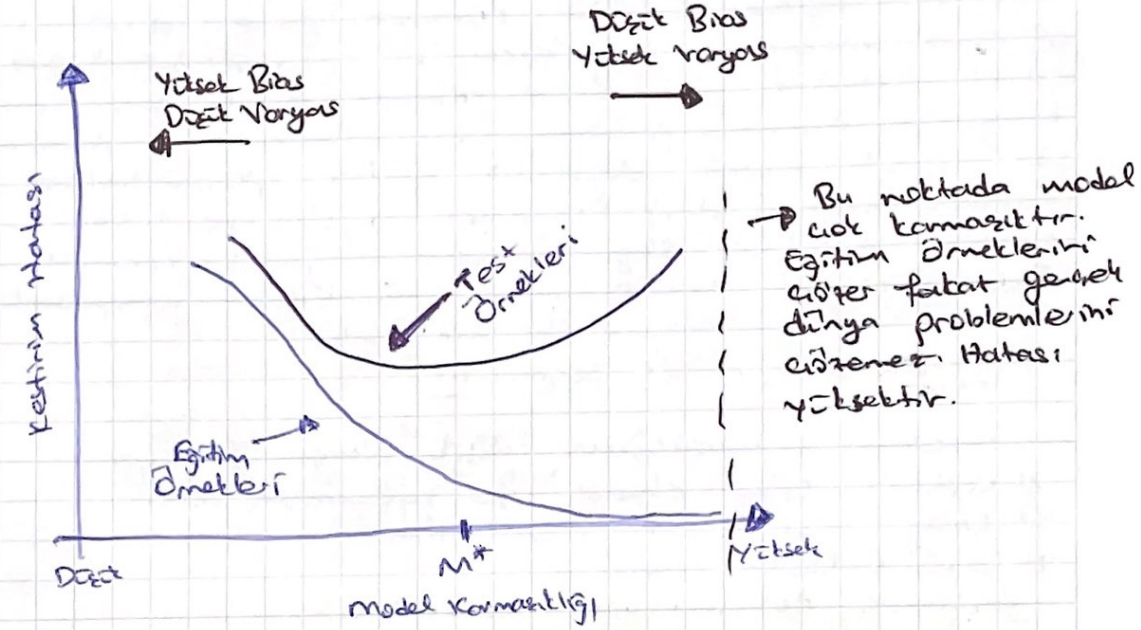
Her bir nöronda geriye yayılımda yeni değerle öğrenmesi için girişlerin sıfır değil küçük bir sayı ile başlatılmalıdır.

Sıfırı eğer bir regresyon yapıyorsak, tek bir nöronumuz varsa sıfırla başlatmakta problem yoktur.

FİGES

...../...../20.....

BİAS - VARİANCE İLİŞKİSİ



Bias yüksek olduğunda, model çok ezberlenebilir ve genellenebilir durumdadır - fakat başarısı eğitim için de test için de çok yüksek değildir. Kestirim hatası yüksektir. Başarımı düşüktür.

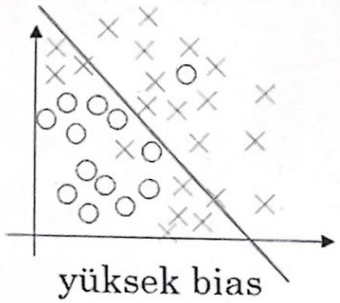
model karmaşıklaşmaya başladıkça hatalar azalır test ve eğitim birlikte. Bir süre sonra testin hatası ile eğitimin hatası birbirinden uzaklaşmaya başlar. Eğitim hatası düşmeye devam eder, burada varyas yüksek olur çünkü test ve eğitim hatası arası mesafe açılır. Fakat bias düşük olur çünkü eğitimin hatası düşüktür.

Öyle bir model karmaşıklığı seçmeliyiz ki mümkün olan bias - variance ilişkisi optimum noktada olsun.

FİGES
www.figes.com.tr

Bias vs. Variance

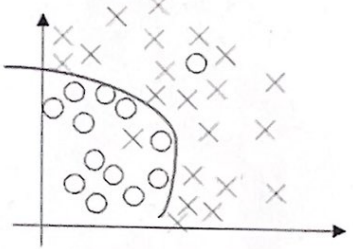
model çok genel
az karmaşık



yüksek bias

Az Uydurma

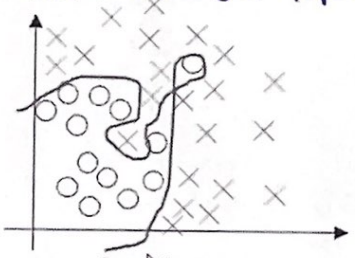
Underfitting



"tam kararında"

✓ Ne yüksek bias
Ne yüksek varyans

model mükemmel şekilde ayırır. Ancak başka bir sınıf verilseydi yapamayacaktı

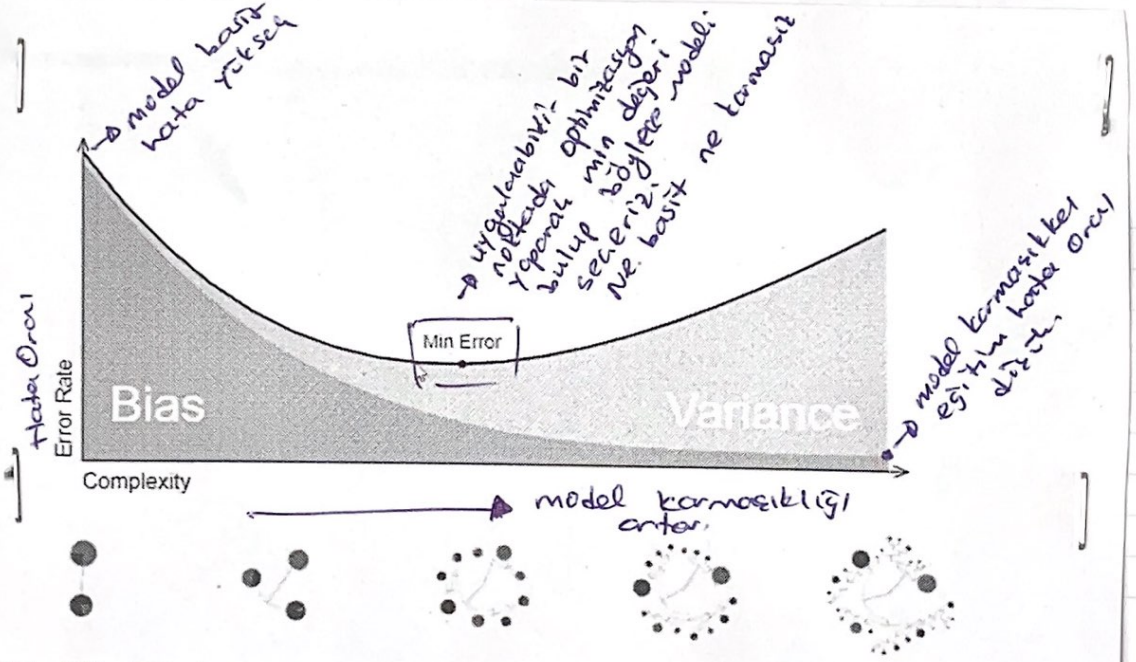


yüksek varyans

Aşırı Uydurma

Overfitting

Bias = modelin tahminlerindeki sistematik hata.
Variance = eğitim setinde örnekleme gürültüsü tahminlerdeki gürültüye ne neden olur.



FİGES

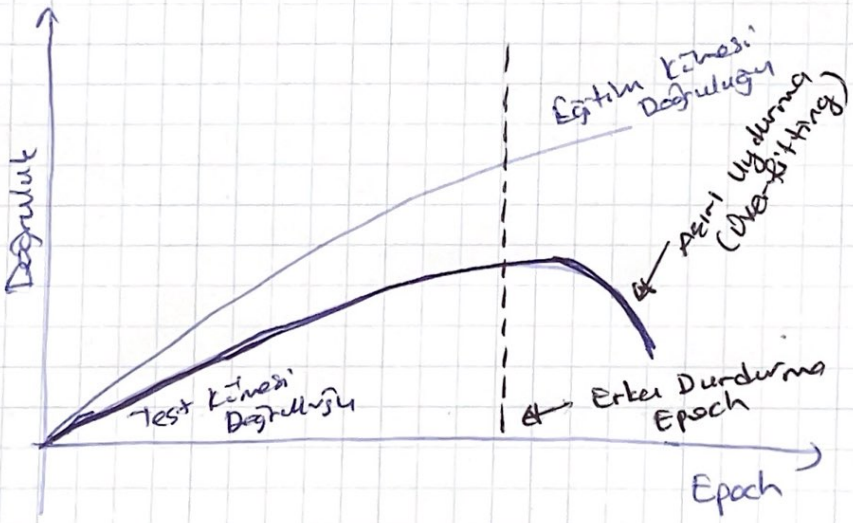
...../...../20.....

DÜĞÜM SEYRELTME (DROPOUT)

Bu da regularizasyon yani dönüşüm yöntemlerinden biridir. Tam bağlantılı katmanlarda eşik değeri altında kalan nöronların iptal edilmesi başarımı arttırabilir. Ağın yükünü azaltır.
HEDEF: Modeli genelleştirmeye yardımcı olur.

Genellikle en sona eklenen Dropout katmanında %50 kullanılır. (%50'sini unut der.)

AŞIRI UYDURMA – AZ UYDURMA
(Overfitting vs. UNDERFITTING)

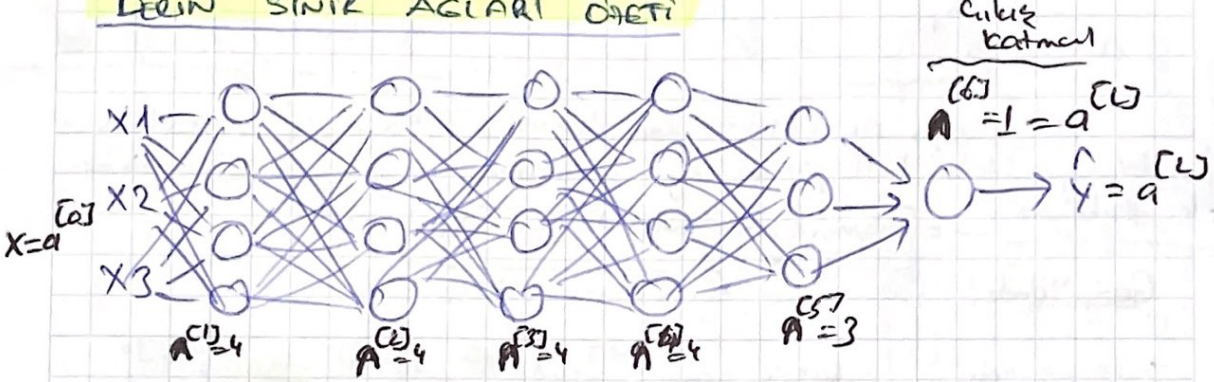


Aradaki mesafe artmaya başlayınca Early Stopping (Erken Durdurma) yapmalı ve devam eden epochlar eğitilmemelidir. Böylelikle bunun önüne geçmiş oluruz.

TRANSFER ÖĞRENME (TRANSFER LEARNING)

Özellikle çok katmanlı modellerde performansı ciddi derecede etkileyen bir diğerleme yöntemidir.

DERİN SİNİR AĞLARI ÖZETİ

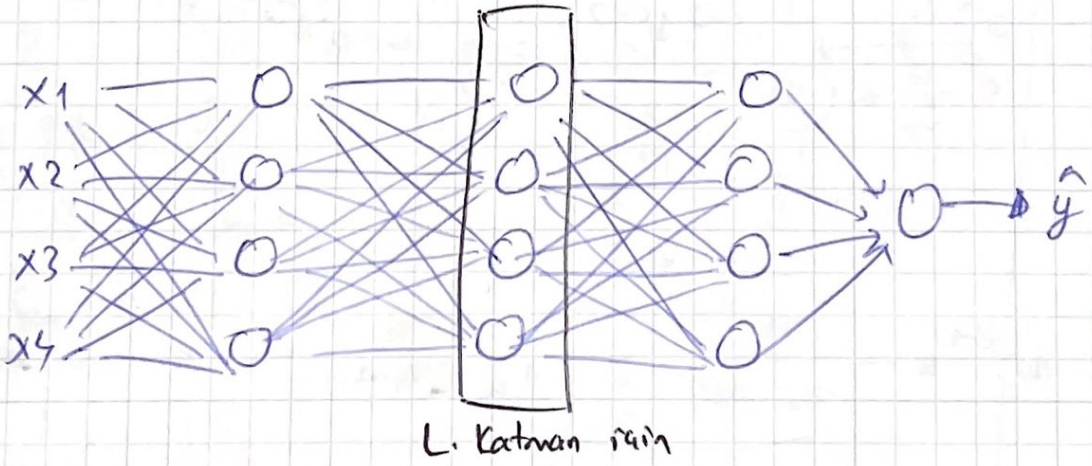


L: katman sayısı $= 6$

$n^{[\ell]}$: ℓ. katmandaki nöron sayısı

$a^{[\ell]} = g^{[\ell]}(z^{[\ell]})$, $w^{[\ell]}$: ℓ. katmandaki ağırlıklar.

Bu tip bir sinir ağının ileri ve geri yayılımı için aşağıdaki yolu kullanırız



L. katman için ileri yönde yayılım yaptığımızda bir önceki $(\ell-1)$ katmanın çıktıları benim girişim olacak.

FİGES

...../...../20.....

İleri Yönde: → Bir önceki katmanda gelen bilgiler.

Giriş: $a^{[l-1]}$, Çıkış: $a^{[l]}$, ARA: $z^{[l]}$

$z^{[l]} = w^{[l]} \cdot a^{[l-1]} + b^{[l]}$ → (X girişleri) · (kollardaki ağırlıklar) + (bias değerleri)

$a^{[l]} = g^{[l]}(z^{[l]})$

g: aktivasyon fonksiyonu → z'yi Aktivasyon fonksiyonuna tabi tuttuğumda a'yı hesaplarım.

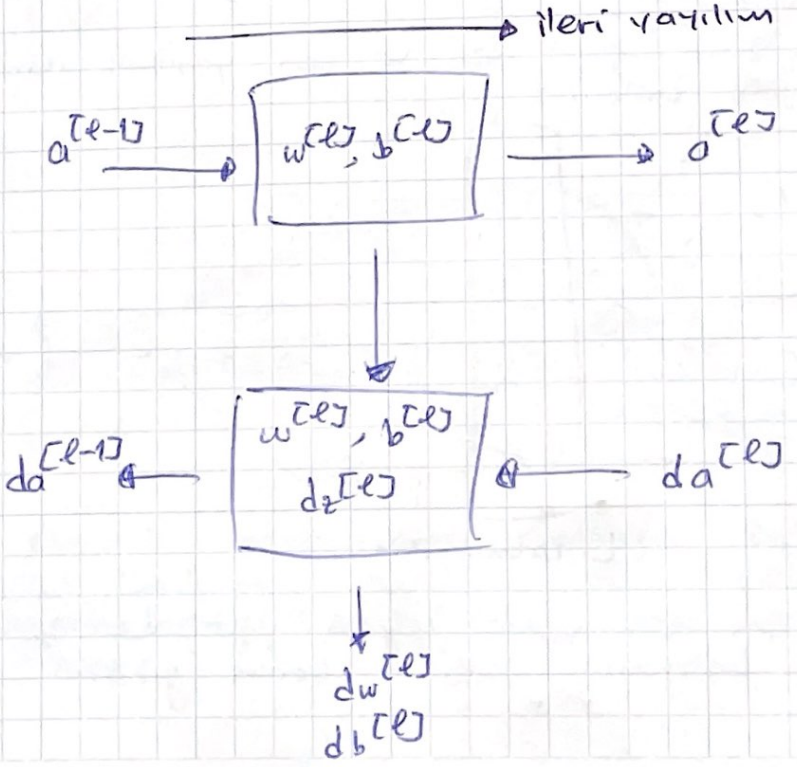
Bir önceki katmandan gelen girişleri ifade eder.

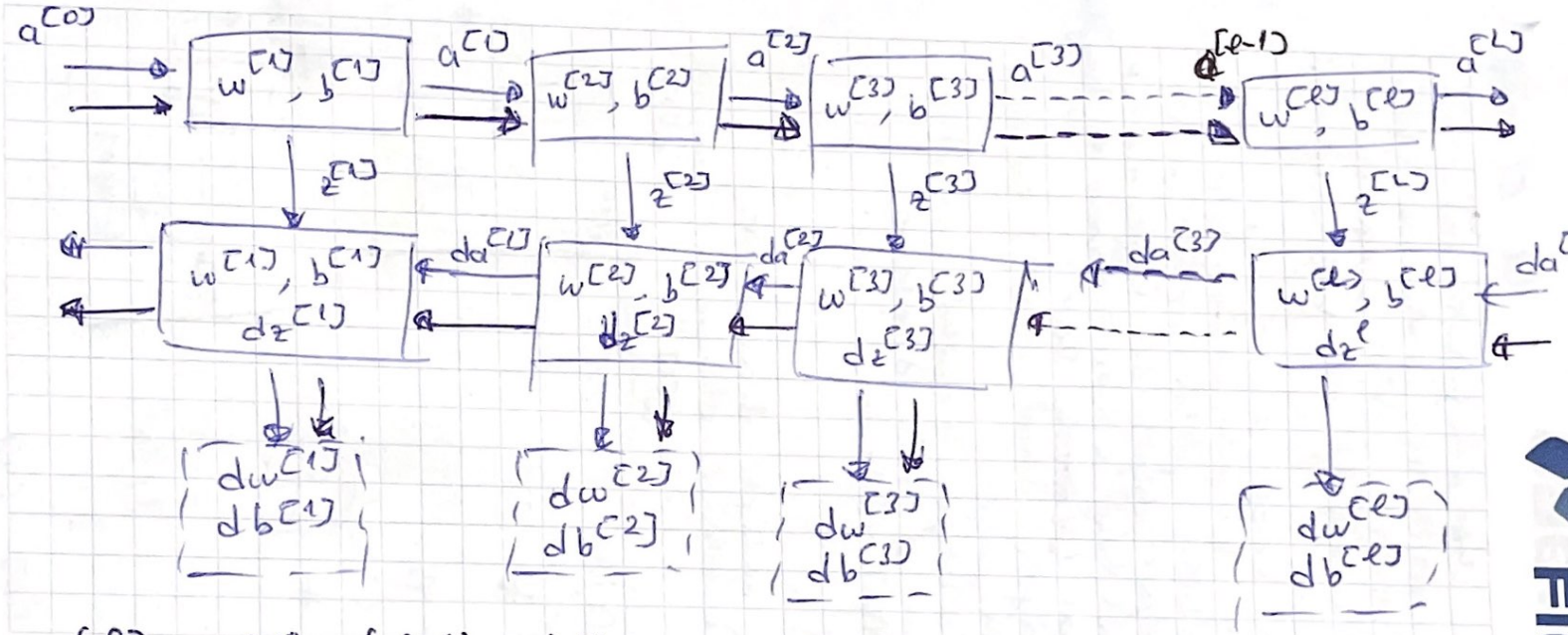
Sigmoid, Hiperbolik Tanjant, Relu olabilir.

Geri Yönde: Bu sefer türevlerini alıyız.

Giriş: $da^{[l]}$, Çıkış: $da^{[l-1]}, dw^{[l]}, db^{[l]}$, ARA: $z^{[l]}$

L. katman için:





Elde edilen Loss fonksiyonu minimize etmek için geriye yayılım algoritması kullanırız.

$$z^{[l]} = w^{[l]} \cdot a^{[l-1]} + b^{[l]}$$

$$w^{[l]} = w^{[l]} - \alpha \cdot dw^{[l]}$$

$$b^{[l]} = w^{[l]} - \alpha \cdot db^{[l]}$$

α → Öğrenme Oranı

İLERİ YAYILIMDA

Girdi: $a^{[l-1]}$ (Bir önceki katman)

Çıktı: $a^{[l]}$, Ara: $z^{[l]}$

$z^{[l]} = w^{[l]} \cdot a^{[l-1]} + b^{[l]}$

$a^{[l]} = g^{[l]}(z^{[l]})$

GERİ YAYILIMDA:

Girdi: $da^{[l]}$ Çıktı: $da^{[l-1]}$, $dw^{[l]}$, $db^{[l]}$

$dz^{[l]} = da^{[l]} * g'^{[l]}(z^{[l]})$

$dw^{[l]} = dz^{[l]} * a^{[l-1]}$

$db^{[l]} = dz^{[l]}$

$da^{[l-1]} = w^{[l]T} * dz^{[l]}$

$$dz^{[l]} = w^{[l+1]T} \cdot dz^{[l+1]} * g'^{[l]}(z^{[l]})$$

FİGES

www.figes.com.tr

(*) → Çarpma değil evrişim sembolüdür.

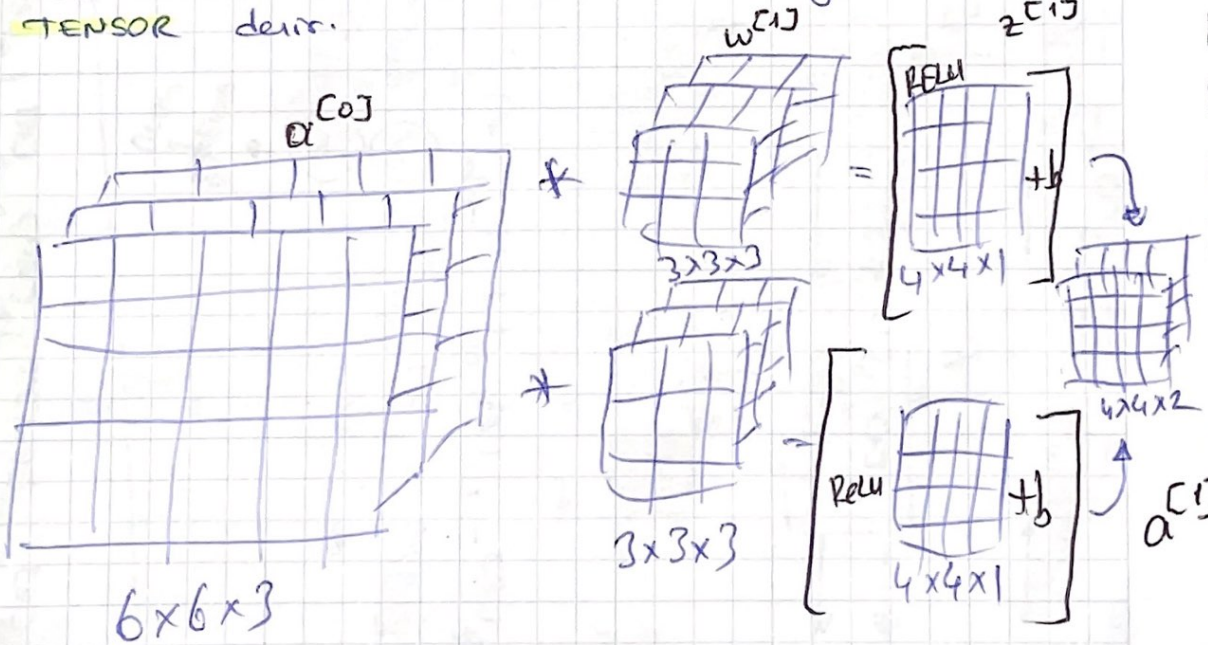
FİGES

04.../.01.../20.24

PARAMETRELER: Modellerde her zaman bulunması gereken, hesaplanması gereken ağırlıklar ve bias değerleridir.
- $(W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]})$ W: Ağırlıklar, b: Bias Değerleri

HİPERPARAMETRELER: Bizim karar vereceğimiz ve parametrelerin hesaplanmasına katkı sağlayan, modelin performansını direkt olarak etkileyen değerlerdir. Değişkendir ve her modelde her zaman yüksek performans sağlayamayabilir.
- Öğrenme Oranı (Learning Rate)
- İterasyon Sayısı
- Gizli Katman Sayısı
- Aktivasyon Fonksiyonu Seçimi
- Küme Boyutu ve diğer düzenleme (regularization) yöntemleri
- ...

Amaç iterasyon sayısında maliyeti düşürmektir.
Bir matris bir çok kanaldan oluşuyorsa buna TENSOR denir.



$$z^{[1]} = w^{[1]} \cdot a^{[0]} + b^{[1]}$$

$$a^{[1]} = g^{[1]}(z^{[1]})$$

Giriş boyutu büyüdüğünde filtrelerin ne kadar kullanıldığına bağlı olarak parametreler değişir. Giriş boyutuna göre değil. Ancak sadece filtreleri girişe uygularken daha fazla işlem yapmamız gerekiyor.

Bu filtreler birler için öznitelikleri hesaplanmasıdır.

Filtre Boyutu: $f^{[l]}$

Pixel Doldurma: $p^{[l]}$ → Padding

Adım Kaydırma: $s^{[l]}$ → stride

filtre (kanal) Sayısı: $n_c^{[l]}$ → N-Chanel

Her bir filtrenin boyutu ; $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$ — Giriş görüntüsün kanal sayısı ile aynı olmalı

Aktivasyon fonksiyonu ; $a^{[l]} \rightarrow n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}$ — yükseklik (height), genişlik (weight), kanal (Chanel)

Ağırlıklar ; $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$ — çıkış boyutu ile aynı olmalı formülde

Bias ; $n_c^{[l]} \rightarrow (1,1,1, n_c^{[l]})$

Giriş: $n_H^{[l-1]} \times n_w^{[l-1]} \times n_c^{[l-1]}$

Çıkış: $n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}$

$$n_{H,w}^{[l]} ; \left[\frac{n_H^{[l-1]} + 2p - f^{[l]}}{s^{[l]}} + 1 \right]$$

FİGES

...../...../20.....

Bir Evrişimli Sinir Ağı için Gerekli Katmanlar

- Evrişim Katmanı (Aktivasyon Fonksiyonu, Bias değeri)
- Ortaklama Katmanı (maksimum ya da Ortalama Ortaklama)
- Tam / Tüm Bağlantı Katmanı (Klasik Yapay Sinir Ağı Bağlantıları)

Örnek

Giriş Görüntüsü $\Rightarrow$ $29 \times 29 \times 3$ olsun $\rightarrow a^{[0]}$

1. Adım Boyutu

$\rightarrow n_H^{[0]} = n_w^{[0]} = 29$

$n_c^{[0]} = 3$

$f^{[1]} = 3$

$p^{[1]} = 0$

$s^{[1]} = 1$ olan bir filtre uygulandığında ve 10 filtre uygulanınca $\Rightarrow$ YENİ ÇIKTI:

$$\frac{n + 2p - f}{s} + 1$$

$$= \frac{29}{1} + 1 = 25$$

$25 \times 25 \times 10$ $\rightarrow$ filtre sayısı yeni matris elde edeceğiz.

2. Adım) $a^{[1]}$ $25 \times 25 \times 10$ matrisine

$n_H^{[1]} = n_w^{[1]} = 25$

$n_c^{[1]} = 10$

$f^{[2]} = 5$

$p^{[2]} = 0$

$s^{[2]} = 2$ değerli filtre uygulayalım 20 tane filtre uygulansın.

$$\frac{n + 2p - f}{s} + 1 \text{'den}$$

$$\frac{25 + 0 - 5}{2} + 1 = 11$$

$\rightarrow$ $11 \times 11 \times 20$ matris halinde ikinci katmanda bulurum.

20 filtre uygulandı

$a^{[2]}$

$a^{[3]}$

FİGES

...../...../20.....

3. Adım: $11 \times 11 \times 20$'lik boyuttaki matrise

$n_H^{[2]} = n_W^{[2]} = 11$

$n_C^{[2]} = 20$

$f^{[3]} = 5$

$p^{[3]} = 0$

$s^{[3]} = 2$ ve 40 filtre uygulayalım.

$$\frac{n + 2p - f}{s} + 1 \text{'den} \rightarrow \frac{11 + 2 \cdot 0 - 5}{2} = 3 + 1 = 4$$

Yeni matris boyutum $\rightarrow$ $a^{[3]}$ $4 \times 4 \times 40$

4. Adım: $4 \times 4 \times 40$, benim son katmanım ise artık bunu vektör haline getiririm ve $4 \times 4 \times 40 = 640$ satırlık bir vektör olur. Bu da çıkış nöronuna ulaştırılır. Burası da benim elde ettiğim kestirim değeri olur.

Ortaklama katmanında kanal sayısı değişmez. Sadece yükseklik ve genişlik bakımından boyut azaltma gerçekleştirilir.

Yapılan işlemler bu adımla sonrasında yükseklik ve genişlik azalırken kanal sayım artacaktır.

$n_H \downarrow$ $n_W \downarrow$ $n_C \uparrow$

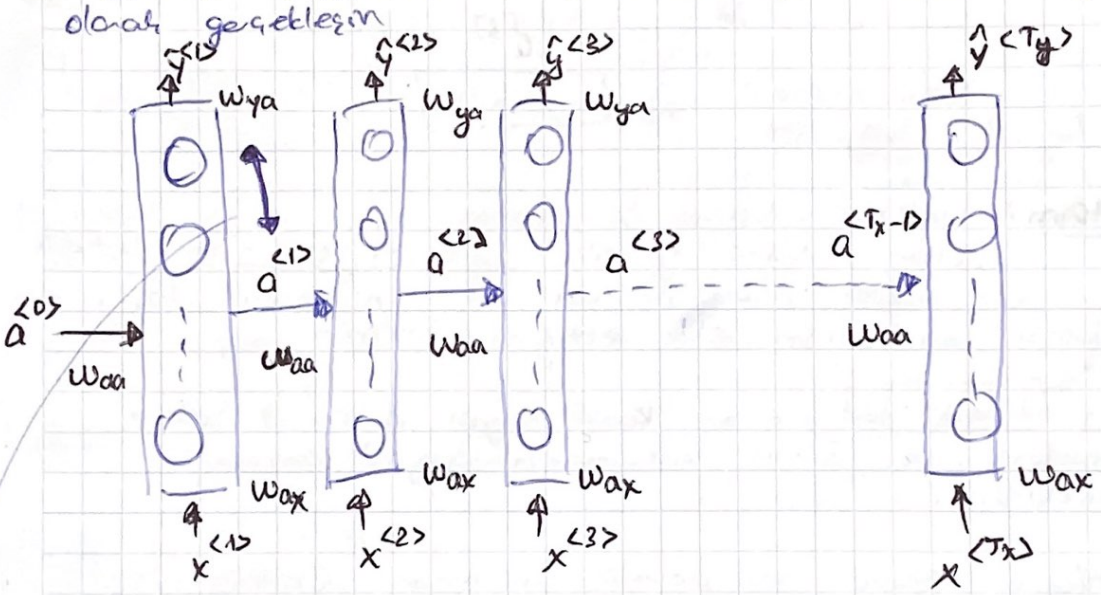
04/01/2023

ÖZYİNELEMELİ SİNİR AĞLARI
(Recurrent Neural Networks – RNN)

Zaman serisi şeklinde ifade edilen problemler için çözüm diyebiliriz.

Özellikle ses işleme, ses ile ilgili problemler. Çünkü ses zaman boyunca süregelen bir veridir ve bu veri özyinelemeli sinir ağları ile çözülebilir bir veridir.

RNN'de kestirimler her zaman bir önceki koşula bağlı olarak gerçekleşir.



$a^{<0>} = \vec{0}$ (bir önceki katman, girişteki değerler, a için bias)

$$a^{<1>} = g_1\left(W_{aa} \cdot a^{<0>} + W_{ax} \cdot x^{<1>} + b_a\right)$$

Bu bir aktivasyon fonksiyonudur. (ReLU, RNN'de pek tercih edilmez.) (Genellikle hiperbolik tanjant tercih edilir.)

$$\hat{y}^{<1>} = g_2\left(W_{ya} \cdot a^{<1>} + b_y\right)$$

Aynı aktivasyon fonksiyonu olmak zorunda değiller. Aynı da olabilir.

NOTASYON

W_{ya} → y: Hesapladığım, a: Çarptığım şeklinde yazılır.

Önce hesaplanır sonra çarpılır yazılır.

www.figes.com.tr

Genelleme Yapacak Olursak:

$$a^{<t>} = g(W_{aa} \cdot a^{<t-1>} + W_{ax} \cdot x^{<t>} + b_a)$$
$$\hat{y}^{<t>} = g(W_{ya} \cdot a^{<t>} + b_y)$$

t : Ait olduğumuz an.
$t-1$: Bir önceki an.

Genellikle a'lar hesaplarken ⇒ Hiperbolik Tanjant veya ReLU tercih edilir. Ancak ReLU çok nadir tercih edilir.

Genellikle $\hat{y}$ (çıktı) hesaplarken ⇒ Sigmoid tercih edilir.

W_{aa}, W_{ax}'ler → MATRİS
a ve t'ler → VEKTÖR'dür.

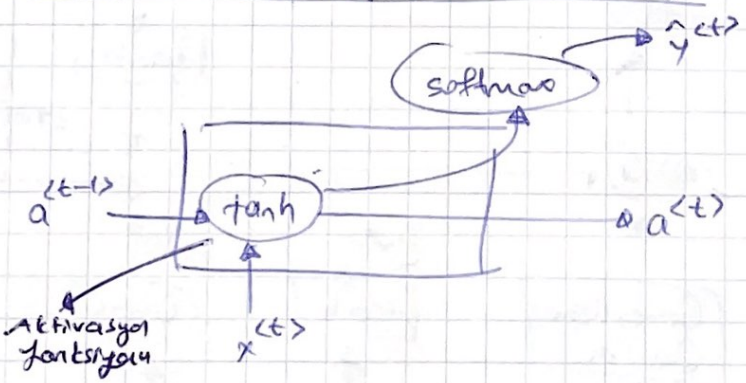
Notasyonu daha da sıkıştırmak istersek:

$$a^{<t>} = g(W_a[a^{<t-1>}, x^{<t>}] + b_a) \quad \text{ve}$$

$$y^{<t>} = g(W_y[a^{<t>}], b_y) \quad \text{şeklinde de ifade edebiliriz.}$$

! YUKARIDAKİ İLERİ YAYILIMDIR !

GRU (Gated Recurrent Units)
(Geçitlenmiş Özyinelemeli Birimler)



$$a^{<t>} = g(W_a[a^{<t-1>}, x^{<t>}] + b_a)$$

BELLEK

$$c^{<t>} = a^{<t>}$$

Γ: unutma değeri

FİGES

...../...../20.....

LSTM (Long Short Term Memory)
(Uzun Kısa Vadeli Bellek)

GRU ram Denklemler

İlgililik

$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c) \rightarrow \hat{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$

Güncelleme Geçidi

$\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u) \rightarrow \Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$

İlgililik Geçidi

$\Gamma_r = \sigma(W_r[c^{<t-1>}, x^{<t>}] + b_r)$

Unutma Geçidi

$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$

Çıkış

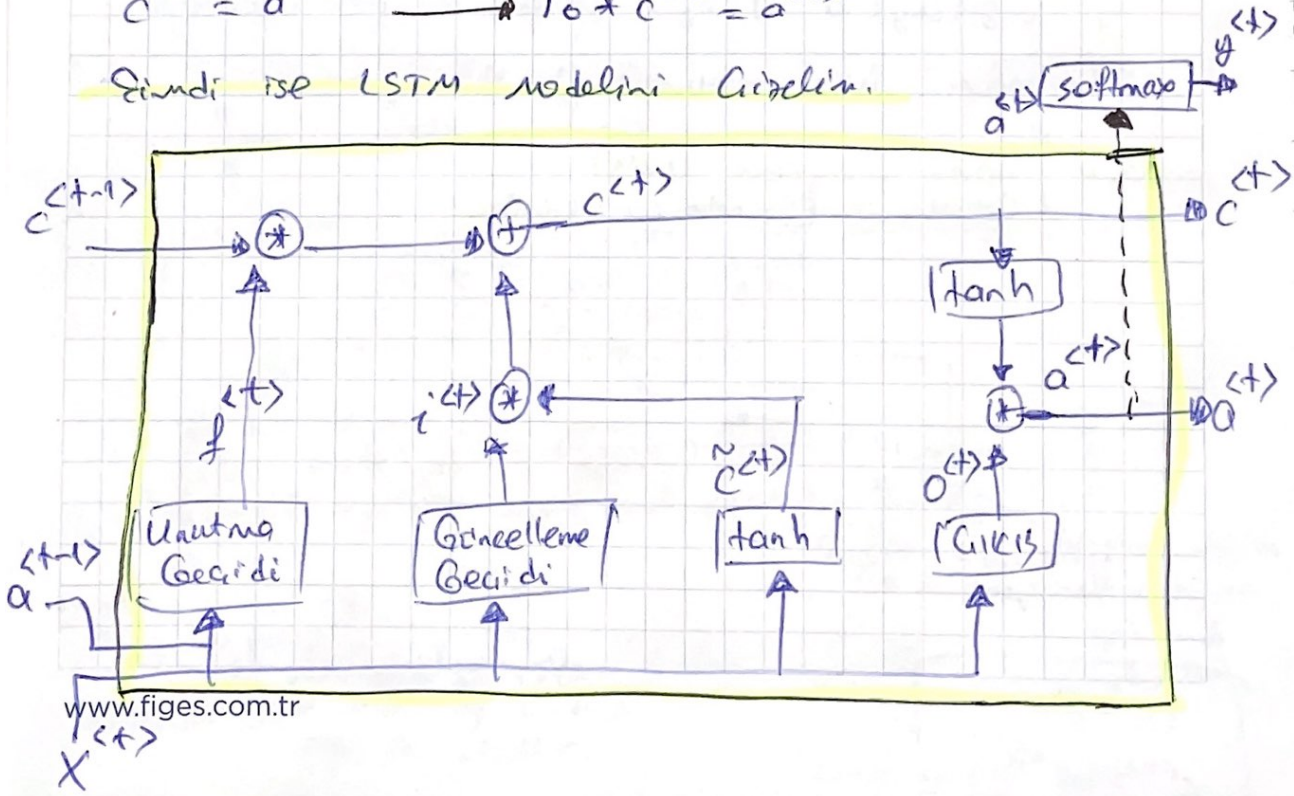
$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$

$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 + \Gamma_u) * c^{<t-1>}$

$\rightarrow c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (\Gamma_f) * c^{<t-1>}$

$c^{<t>} = a^{<t>} \longrightarrow \Gamma_o * c^{<t>} = a^{<t>}$

Şimdi ise LSTM modelini çizelim.



FİGES

05/01/2024

Bu haritayı formülleştirirsek;

$\tilde{C}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$

$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$ ← Güncelleme geçidi olduğu yer

$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$ ← Unutma geçidi olduğu yer.

$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$ ← Çıkış geçidi olduğu yer.

$C^{<t>} = \Gamma_u * \tilde{C}^{<t>} + (\Gamma_f) * C^{<t-1>}$

$\Gamma_u * \tilde{C}^{<t>}$: Güncelleme değeri ve c'ye aday olan t anındaki değer geçiyor.

(+)

$(\Gamma_f) * C^{<t-1>}$: Unutma geçidinden çıkan değerle de bir önceki t anından çıkan C değeri geçiyor.

$\Gamma_o * C^{<t>} = a^{<t>}$

Bunlar birbirine ard arda eklenebilir. Birinin çıkışı diğerinin girişi olur.