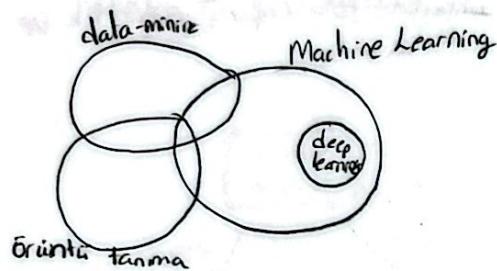


Hafta-1 Makina Öğrenmesi Slayt

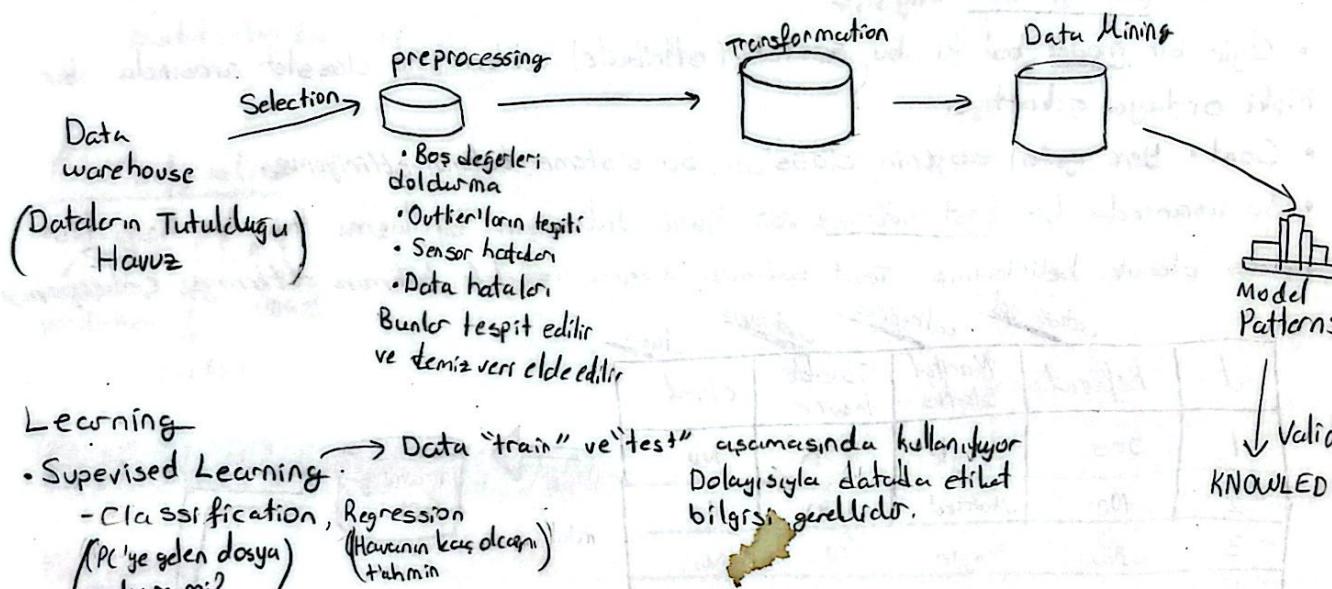
(data)
Makina öğrenmesi deneyimden öğrenir, Öğrenmesi için bazı göreler ve performans metrikleri olacak. Performans metrikleri ile ne kadar başarılı karar verdigine baktırız.

Makina öğrenmesi açıkça programlanmamıştır, If else gibi kurallar yok. Makina öğrenmesinde data çok önemlidir.



Makina öğrenmesindeki birçok problem

- NP-hard, öyle problemler ki bunlara if-else gibi tüm şartları deneyerek çözmeye çalışırsa problemin çözümü yılları alacaktır. Brute-force çözümün çalışma zamanı eksponansiyeldir
- Örneğin 3D reconstruction problemi, missing data

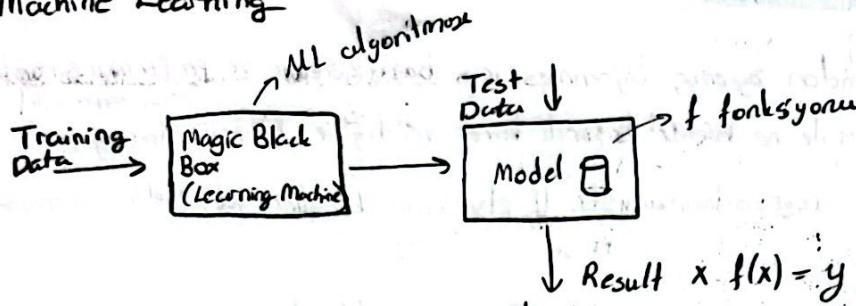


Learning

- Supervised Learning → Data "train" ve "test" açısından kullanıyor. Dolayısıyla datada etiket bilgisi gereklidir.
 - Classification, Regression
(PC'ye gelen dosya) (Havuzun kaç deşen)
(malware mi?) (tahmin)
- Unsupervised Learning → Etiket yok.
 - Clustering
 - K-Means / Dirichlet / Gaussian Processes
- Semi-Supervised Learning → Bazi verilerin etiketi var, bazi verilerin etiketi yok.
 - Constrained Clustering / Distance Metric / Min.

Supervised ve unsupervised açısından bir yaklaşım.

Machine Learning



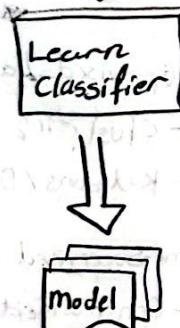
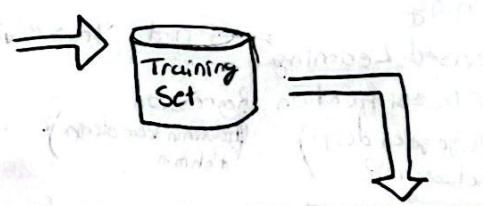
1- Farklı Farklı Fotoğrafları Sisteme verend sisteme koala fotoğrafını öğrettiğ ve model elde ettik

Classification

training set: verilen nesneler, öğrenme kümesi

- Her bir nesne onu temsil eden öznitelik ^(attribute) hâmesini içerip, bu öznitelik ^(attribute) lardan birisi de (class) tür bilgisidir.
- Öyle bir model bul ki bu öznitelikler ^(attribute) vektörler ile classlar arasında bir ilişki ortaya çıkartıgın.
- Goal: Yeni gelen objenin class'ını bu sisteme tahmin ettiğimiz.
- Bu aşamada bir test setimiz var. Yani datamızın bir kısmı train bir kısmı test olarak belirlenmiş. Test setimizi doğru şekilde tahmin ettiğimize çalışıyoruz.

id	Refund	Marital Status	Taxable Income	class	
				continuous	categorical
1	Yes	Single	125K	No	
2	No	Married	100	No	
3	No	Single	70	No	
4	Yes	Married	120	No	
5	No	Divorced	95	Yes	
6	No	Married	60	No	
7	Yes	Divorced	220K	No	
8	No	Single	85	Yes	
9	No	Married	75	No	
10	No	Single	90	Yes	

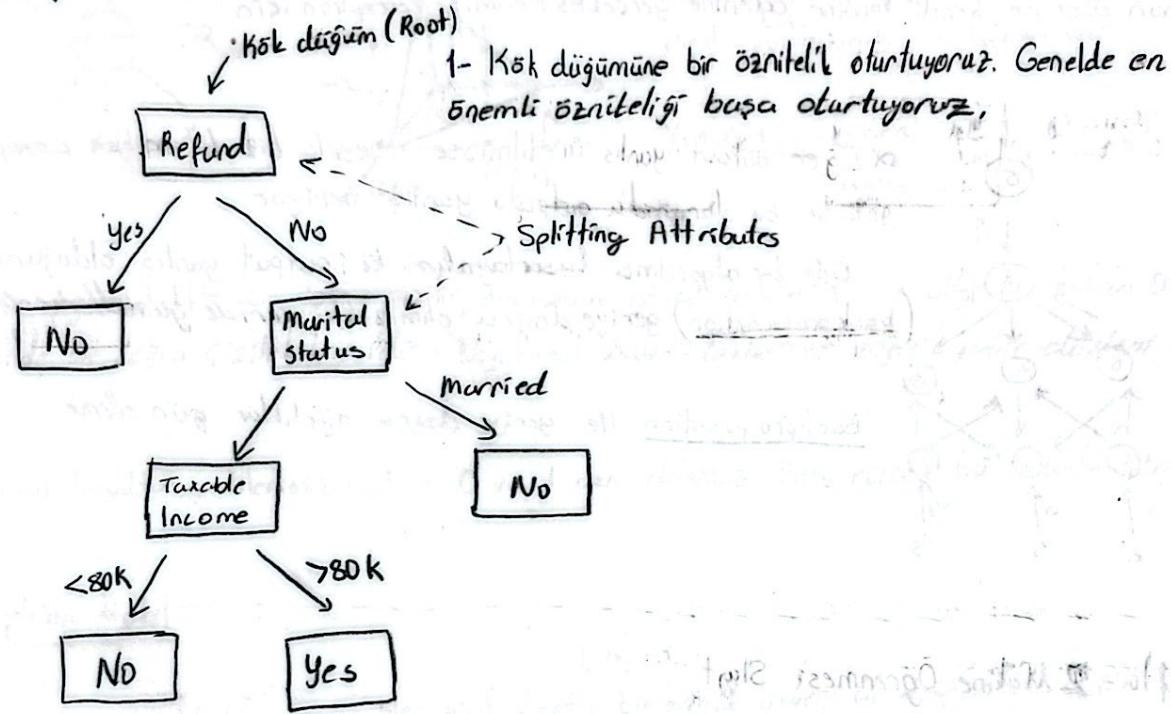


Refund	Marital Status	Taxable Income	Cheat
No	Single	75	?
Yes	Married	50	?
No	Married	150	?
Yes	Divorced	90	?
No	Single	40	?
No	Married	80K	?



Karar Ağacıları

Training Data Tablosunu kullanarak Cheat konusunda hangi karar vereceğiz?



Model: Decision Tree

Classification

$$y = f(x)$$

output prediction

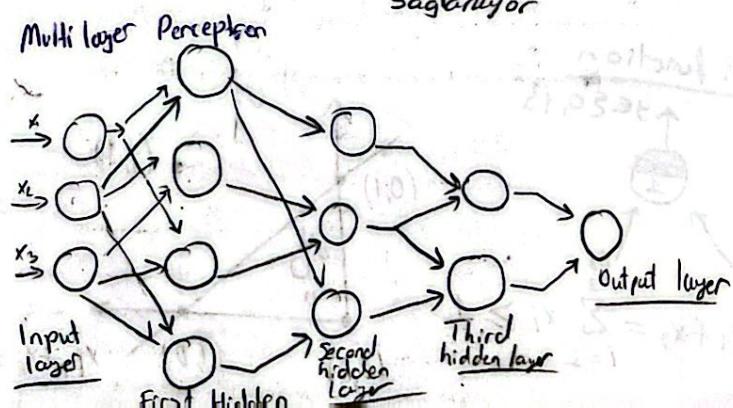
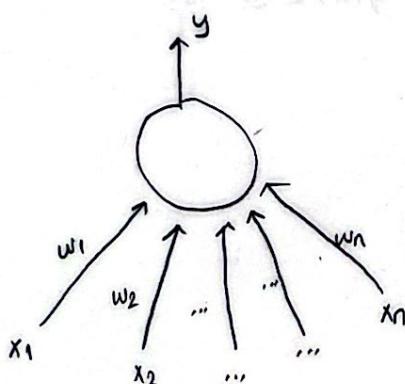
input

function

Hafta-2 Makine Öğrenmesi Slayt

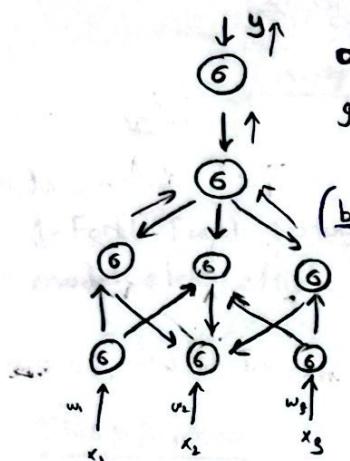
Perceptron

Öğrenen, karar verme yeteneğine sahip bir matematiksel model. Perceptronun piti nöronuna farklı x_1, x_2, \dots, x_n inputları nörona gelirken her bir bağlantı üzerindeki ağırlık olan $w_1, w_2, w_3, \dots, w_n$ ağırlıkları ile çarpılırak nörona gelmektedir. Output dardı 0 ya da 1 gibi bir değer gelmemesi sağlanıyor



Problemin zorluğu arttıkça perceptron mimarileri yeterli olmuştu

Bunun üzerine kendi kendine öğrenme gerçekleştirilmiş Perceptron için

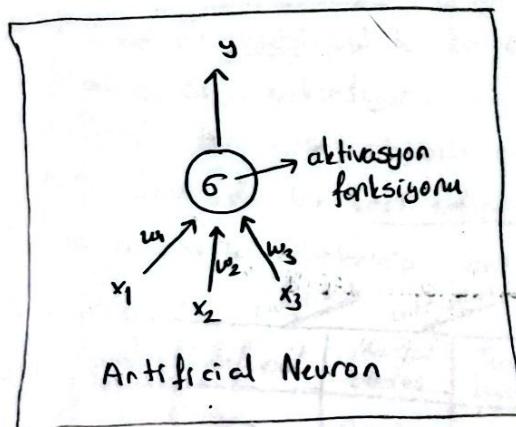


• Eğer output yanlış üretilmişse (mesela biz 1 istekten cevap 0 gelirse) bu durumda outputu yanlış veriyor.

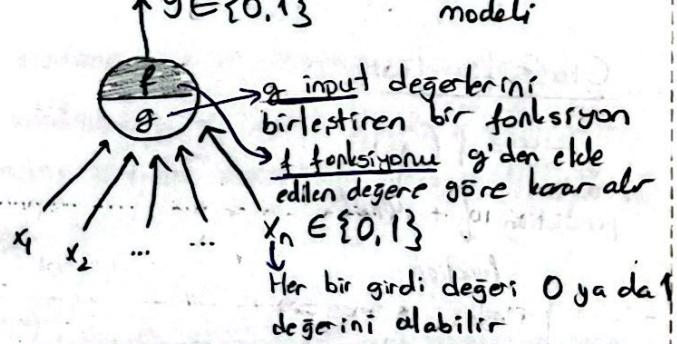
Öyle bir algoritma tasarlayalım ki output yanlış olduğunda (backpropagation) geriye doğru elimizdeki veride güncelleme olsun.

Backpropagation ile geriye doğru ağırlıklar güncellenir

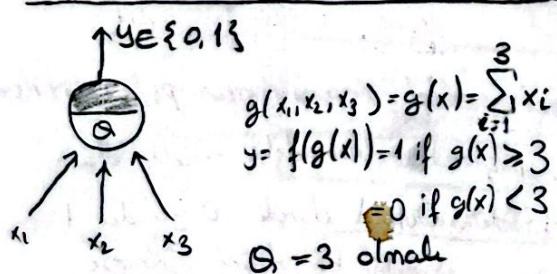
Hafıza - 2 Makine Öğrenmesi Slayt



McCulloch Pitts Neuron Nöronun en basit matematiksel modeli



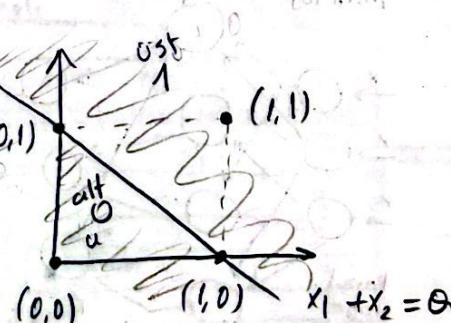
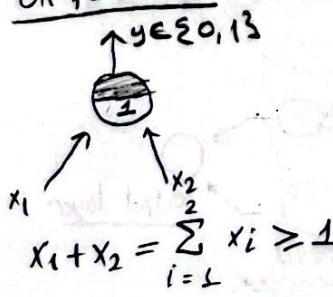
AND function with McCulloch Pitts Neuron

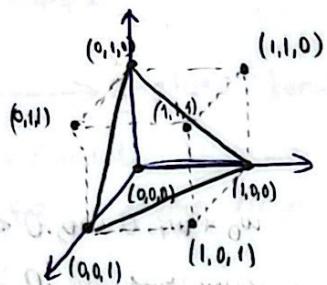
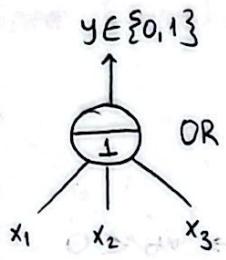


$$g(x_1, x_2, \dots, x_n) = g(x) = \sum_{i=1}^n x_i$$

> $y = f(g(x)) = 1$ if $g(x) \geq Q$
• Eğer $g(x)$ değeri belli bir threshold değerini geçiyorsa ise output 1, geçmiyorsa ise output 0 olacak.
> $y = f(g(x)) = 0$ if $g(x) < Q$

OR function





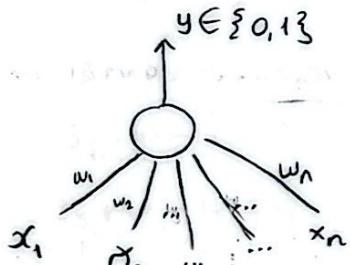
- Bir önceki örnekler gibi tek bir doğru çizerek çözmemeyiz bu problemi çünkü elimde dimension 3 bir düzlem var

$$x_1 + x_2 + x_3 = \Theta_1 = 1$$

Linear Ayrışabilirlikte bir doğru çizilir, doğrunun üstünde kalan 1 ise altında kalan 0 olur şekilde bir doğru çizilebilmesidir. Boyutumuz 2'den fazla ise doğru yerine düzlem çizilir.

! Gerçek hayatta girdilerimiz sadece 0 ve 1 den oluşmaz. Pitts nöronu bu konuda yetersizdir.

Perceptron Model



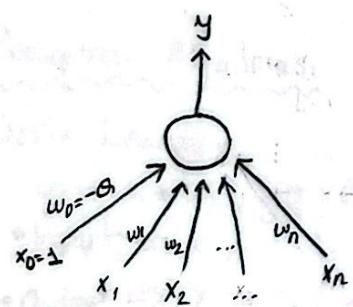
- Her girdi değeri bir ağırlık değeri ile ilişkilendirilmiştir.
- Girdi 0 ya da 1'dır.

$$\begin{aligned} y &= 1 \text{ if } \sum_{i=1}^n w_i \cdot x_i \geq \Theta_1 \\ &= 0 \text{ if } \sum_{i=1}^n w_i \cdot x_i < \Theta_1 \end{aligned}$$

$x_1 \cdot w_1 + x_2 \cdot w_2 + \dots + x_n \cdot w_n$ Pitts nöronundaki gidi bir toplama yapıyor

↓ Dahı kabul edilir hali

$$w_i \cdot x_i - \Theta_1 \geq 0$$



$$y = 1 \text{ if } \sum_{i=0}^n w_i \cdot x_i \geq 0$$

$$= 0 \text{ if } \sum_{i=0}^n w_i \cdot x_i < 0$$

where $x_0 = 1$ and $w_0 = -\Theta_1$

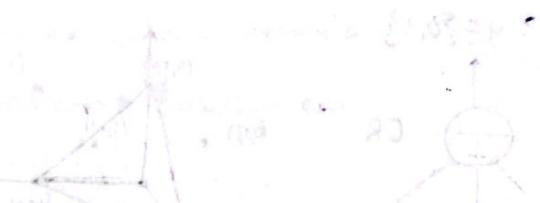
eski değeri bir girdi değeri olarak eklenir
- On ağırlık olarak verilir
x0 girdisi 1 olarak kabul edildi

* Good ⇒ will not work for non-linearly separable data

linearly separable data → can be solved with perceptron

Hafta-3 Derin Öğrenme Slayt

$x_1 - x_2$	OR
0 0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1 0	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
0 1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
1 1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$



$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \rightarrow w_0 < 0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \rightarrow w_1 \geq -w_0$$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \rightarrow w_2 \geq -w_0$$

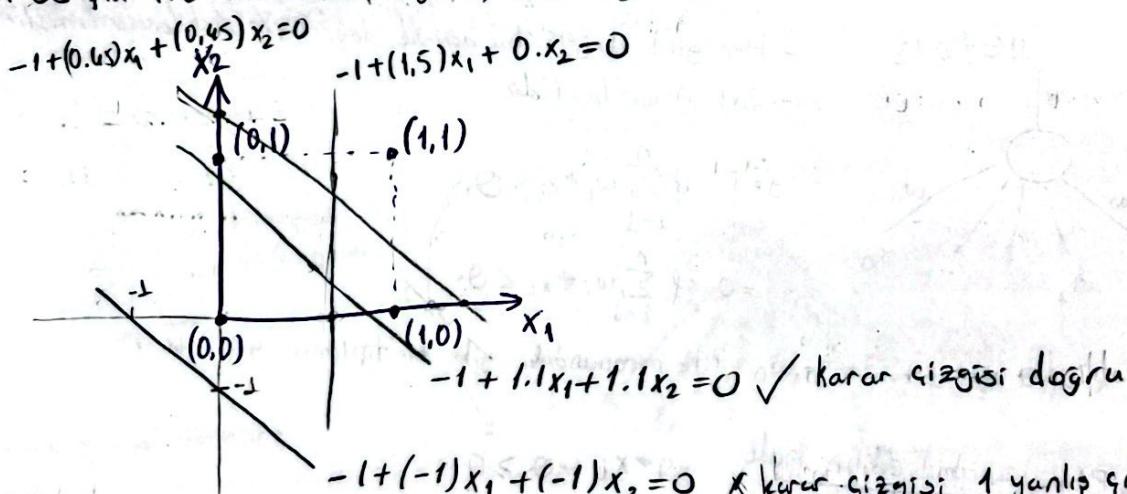
$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 \geq 0 \rightarrow w_1 + w_2 \geq -w_0$$

• Olası bir çözüm $w_0 = -1$, $w_1 = 1.1$, $w_2 = 1.1$

bu durumda bu eşitsizlikleri sağlarsın,

• Bunun gibi birçok çözüm sahibiz

Let us fix the threshold ($-w_0 = 1$) and try different values of w_1 , w_2

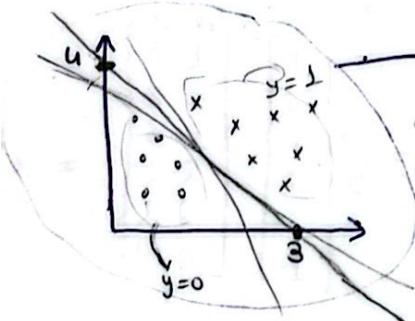


$-1 + (-1)x_1 + (-1)x_2 = 0$ ✗ karar çizgisi 1 yanlış çıktı ve sistemde hata var

w_1	w_2	doğru	yanlış?
-1	-1	3	1
1.5	0	3	1 $\rightarrow -1 + (1.5)x_1 + 0.x_2 = 0$
0.45	0.45	1	3 $\rightarrow -1 + (0.45)x_1 + (0.45)x_2 = 0$

Ağırlık değerleri bir yapay sinir uğı için oldukça önemlidir.

Lineer Classifier Example



2-class binary classification supervised learning

- Soldaki iki grubu birbirinden ayıracak bir karar çizgisi gerekecektir.

gözetimdeki sınıfları bilgilerini biliyoruz

$$y = mx + b$$

intercept değeri (doğrunun y eksenini kestiği kısım)

$$y = -\frac{4}{3}x + b$$

Classifier, geçmiş bilgilerden yararlanarak gelecekte gelen veriyi doğru bir şekilde etiketlemeye çalışıyoruz. Bu durumda aslında bir karar çizgisi çiziyoruz (bir doğru, bir düzleme...)

5. denklem: $y = -\frac{4}{3}x + 4$, (3,3) noktasının sınıfı nedir?

$$3y + 4x - 12 = 0$$

$3y + 4x - 12 \geq 0$, doğrunun üstünde

$3y + 4x - 12 < 0$, doğrunun altını temsil eder

(3,3)

$$9 + 12 - 12 = 9$$

$9 > 0$ noka doğrunun
üstünde
class = 1

1. (1,1) noktası için class nedir?

$$\text{doğa}^u \text{ denklem: } 3y + 4x - 12 = 0$$

$$3 + 4 - 12 = -5 < 0$$

class = 0

Perceptron Algoritması

Delta Learning

$$w_0 + w_1 x_1 + w_2 x_2 \geq 0 \quad \text{Perceptron}$$

• Input $\Rightarrow x_1, \dots, x_d$

• Output $\Rightarrow Y$ (y_i den y_d ye kadar ama Y vektörel gösterim)

• Goal \Rightarrow ağırlık değerlerini $w_1, w_2, w_3, \dots, w_d$ öğrenmek

Error fonksiyonunu minimize eden ağırlık değerleri seçilmeli

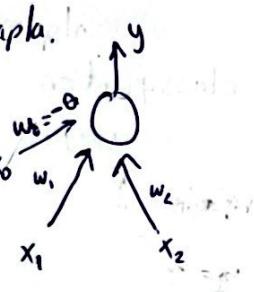
$$E = \frac{1}{2} \sum_{i=1}^N (O_i - y_i)^2$$

False değer

True değer

Örn: Hata değerini hesapla.

x_1	x_2	y	0
0	0	0	1
0	1	1	1
1	0	1	0
1	1	1	0



$$E = \frac{1}{2} \sum_{i=1}^N (o_i - y_i)^2$$

$$E = \frac{1}{2} [1^2 + 1^2 + 1^2] = \frac{3}{2}$$

3 tane Error var

Amacımız hata fonksiyonunun verdiği değeri minimize etmek.

Hata fonksiyonunun verdiği değerin minimum olduğu noktası $\min\{E = \frac{1}{2} \sum_{i=1}^N (o_i - y_i)^2\}$ ağırlık değerleri (w_1, w_2, \dots, w_d) bizim en iyi değerlerimiz

Delta Learning

- Başlangıçta ben w_1, w_2, \dots, w_d değerlerini bilmiyorum.

- Başlangıç ağırlık değerler rastgele atansın. Nöron hataya göre ağırlıkları güncellesin

$$w_{\text{new}} = w_{\text{old}} + \alpha (d_{\text{hata}} \cdot x) \cdot \text{real output}$$

bulunan değer
real output
 d_{hata} → girdi
learning rate, similitil göründür edip 't' dileyim

$$w_1, \dots, w_d = 0 \quad (\text{Ağırlıklar})$$

$$x_1, \dots, x_d \quad (\text{inputlar}) = D \quad (\text{corpus})$$

$$w_0 / b = 0 \quad (\text{bias})$$

for iter = 1 ... Max iter do

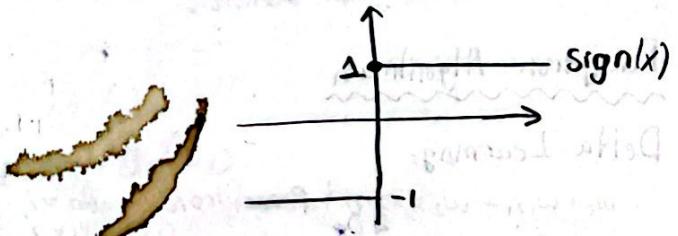
for all $(x, y) \in D$

$$\hat{o} \leftarrow \sum_{i=1}^D w_i x_i + b$$

$$o \leftarrow \text{sign}(\hat{o})$$

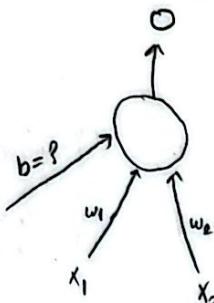
if $y \cdot o \leq 0$: for all $d = 1 \dots D$

$$w_d = w_d + y \cdot x_d \quad (\text{Tüm } w_d \text{ lleri güncelliyoruz})$$



Örnek: Bu x_1 ve x_2 den bir şey öğrenememem
için class bilgileri gerekli

$$X = \begin{bmatrix} x_1 & x_2 \\ 6 & 1 \\ 8 & 3 \\ 3 & 8 \\ 5 & 5 \\ 2 & 6 \\ 4 & 6 \end{bmatrix} \quad y = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \\ -1 \\ -1 \\ -1 \end{bmatrix}$$



Sample 1

$$S_1 = [6 \ 1] \cdot \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix} = w_1 \cdot x_1 + w_2 \cdot x_2 + b$$

$$= 0 + 1 = 1$$

$$\Rightarrow 0 + b = 1$$

$$= 0 + 0 = 0$$

$$O_1 = 0 \sim y_1 = 1 \text{ olmaz}$$

$$\operatorname{sgn}(O_1) = 0$$

$$O_1 \neq y_1 \text{ Hata var}$$

Sample 2 $w = [6 \ 1]$
 $b = 1$

$$S_2 = [8 \ 3] \cdot \begin{bmatrix} 6 \\ 1 \end{bmatrix} = 48 + 3 = 51$$

$$51 + 1 = 52$$

$$O_2 = 52$$

$$\operatorname{sign}(O_2) = 1 = y_2 \checkmark$$

Dogrular

Weight update 1

$$w_1 = w_{1(\text{old})} + y_1 * x_1 = 0 + 1 * 6 = 6$$

$$w_2 = w_{2(\text{old})} + y_1 * x_2 = 0 + 1 * 1 = 1$$

$$b = b(\text{old}) + y_1 = 0 + 1 = 1$$

Sample 3

$$S_3 = [3 \ 8] \cdot \begin{bmatrix} 6 \\ 1 \end{bmatrix} + 1 = 18 + 8 + 1 = 27$$

$$O_3 = 27 \quad y_3 = -1$$

$$\operatorname{sign}(O_3) = 1 \neq y_3$$

Hata var

Weight update 2

$$w_1 = 6 + -1 * 3 = 3$$

$$w_2 = 1 + -1 * 8 = -7$$

$$b = 1 - 1 = 0$$

Sample 4 $w = [3 \ -7]$

$$b = 0$$

$$S_4 = [5 \ 5] \cdot \begin{bmatrix} 3 \\ -7 \end{bmatrix} + 0 = 15 - 35 = -20$$

$$O_4 = -20 \quad y_4 = -1$$

$$\operatorname{sign}(O_4) = -1 = y_4$$

Dogrular

Yukarıdaki ornekte biz $y_i * x$ kullanmak isterseniz $(O - y) * x$ de kullanabilirsiniz.

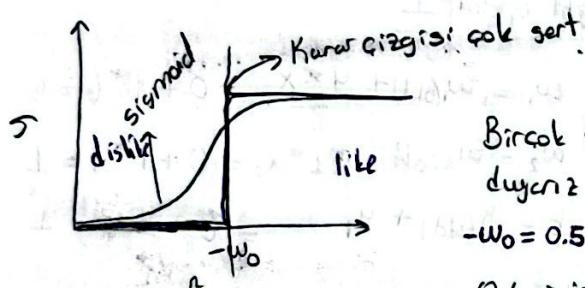
Hafta - 4 Derin Öğrenme

Sigmoid Neuron

Perceptron \rightarrow giriş değerleri $x_n \in \mathbb{R}$
 \rightarrow çıkış değeri $y \in \{0, 1\}$

Sigmoid Neuron \rightarrow giriş değerleri $x_n \in \mathbb{R}$
 \rightarrow çıkış değeri $y \in \mathbb{R}$

threshold: $-w_0$
Threshold'a göre sign() fonksiyonuna
 $x > 0 \rightarrow +1$
 $x \leq 0 \rightarrow -1$
Threshold belirkenek zor iş



Birçok gerçek uygulamada daha yumuşak çizgiye ihtiyacımız

$$-w_0 = 0.5 \text{ olsun}$$

$0.6 \rightarrow$ input için step $\rightarrow 1 \rightarrow y = 1$ like

$0.4 \rightarrow$ input için step $\rightarrow 0 \rightarrow y = 0$ dislike

$$x = 0.6 \rightsquigarrow \text{Like } y = 0.75$$

$$x = 0.4 \rightsquigarrow \text{dislike } y = 0.6$$

$$x = 0.1 \rightsquigarrow \text{dislike } y = 0.2$$

$\rightarrow 0.80$: hatalı sevilmeyecek

$\rightarrow 0.40$: hatalı sevilmeyecek

Aşağıda lojistikin fonksiyonun sigmoid fonksiyon formu
verilmiştir Sigmoid

$$y = \frac{1}{1 + e^{-(w_0 + \sum_{i=1}^n w_i x_i)}}$$

Gözetimli Makine Öğrenmesi Mekanizması (A typical Supervised Machine Learning Setup)

Data: $\{(x_i, y_i)\}_{i=1}^n$
 \rightarrow Atez, kalp atışı, kan basıncı gibi degerler
 \rightarrow Class: Sağlıklı mı? Hastalık mı?

Model: Model x ve y arasındaki bağlantı ile ilişkiliyor. Örneğin:

$$\hat{y} = \frac{1}{1 + e^{-(w^T x)}} \text{ sigmoid aktivasyon fonksiyonu kullanılır}$$

$$\text{or } \hat{y} = w^T x$$

$$\text{or } \hat{y} = x^T w \text{ gibi fonksiyonlar oluşturulabilir}$$

Parametres: Parametrel ağırlıklar (w) ve bias değeri (b)

Learning Algorithm: Ağırlık değerlerini bulmak için gereklidir. Objective function'a atama için

Objective/Loss/Error function: Learning algoritması ağırlık değerlerini Error function'a göre优化.

Amaçımız Loss fonksiyonunu minimize etmek

Örnek:

$$L(w) = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2$$

Mean Squared Error - MSE

Veri setinde çok ~~veri~~ veri varsa $\frac{1}{n}$ ile çarpılır.

Az veri varsa hibrit şekilde bile çarplabılır.

$$\begin{bmatrix} x_1 & x_2 \\ 2 & 3 \\ 4 & 5 \\ 6 & 7 \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} y \\ 1 \\ 0 \\ 1 \\ 3 \end{bmatrix} \begin{bmatrix} \hat{y} \\ 0 \\ 1 \\ 3 \end{bmatrix}$$

$$x_1 = 2 \text{ için } y = 1$$

$$x_2 = 3 \text{ için } y = 0$$

$$E_1 = (0-1)^2 = 1$$

$$E_2 = (1-0)^2 = 1$$

$$E_3 = (3-1)^2 = 4$$

$$L(w) = 1 + 1 + 4$$

$$= 6$$

$$f(x) = \frac{1}{1 + e^{-(w \cdot x + b)}}$$

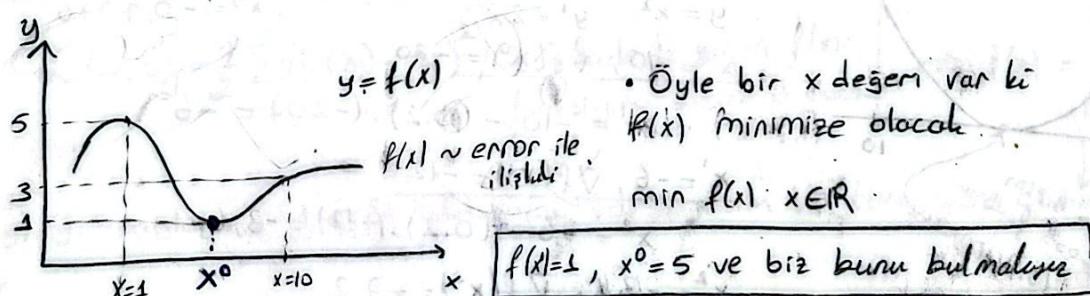
Bu Loss'u minimize etmek amacımız

Training Objective

$$L(w, b) = \sum_{i=1}^N (y_i - f(x_i))^2$$
 Öyle w ve b değerleri bul ki $L(w, b)$ min olsun.

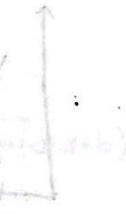
Optimization Basics

Amaç: Verilen bir fonksiyona minimize eden değeri bulmak

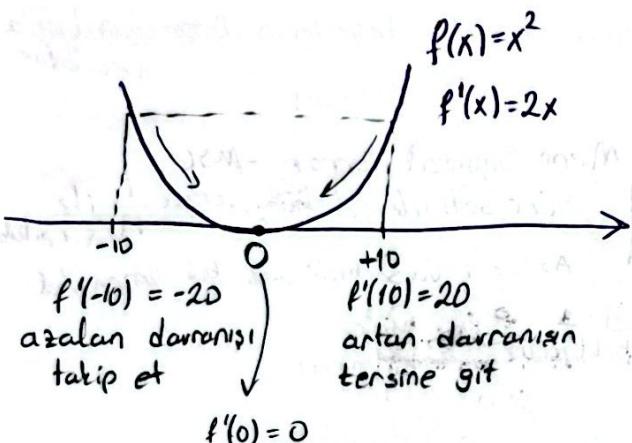


Öyle bir x değeri var ki $f(x)$ minimize olacak.

$$\min f(x) : x \in \mathbb{R}$$



Gradient



if gradient \rightarrow positive

$f(x)$ fonksiyonu artan bir fonksiyon

else if gradient \rightarrow negative

$-f(x)$ fonksiyonu azalan bir fonksiyon

Gradient Descent Algorithm

- t_0 belirle
- x^0 tahmin et
- k 'ya 0 ata

while $\| \nabla f(x^k) \| \geq \epsilon$

$$x^{k+1} = x^k - t_k \nabla f(x^k)$$

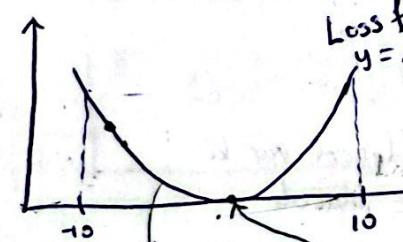
$$k = k + 1$$

end while

return x^k

k : iterasyon sayısı

t_k : adım sayısı (default girilmeli)



$$t_0 = 0.2 \text{ belirlendi}$$

Loss function $x^0 = -10$ diye tahmin ettiğimde $k=0$ altındadır

$$y = x^2 \quad y' = 2x$$

$$x^0 = -10, \nabla f(x^0) = -20.$$

$$x^1 = -10 - (0.2) \cdot (-20) = -6$$

$$x^1 = -6, \nabla f(x^1) = -12$$

$$x^2 = -6 - (0.2) \cdot (-12) = -3,6$$

$$x^2 = -3,6, \nabla f(x^2) = -7,2$$

$$x^3 = -3,6 - 0,2 \cdot -7,2 = -2,16$$

$$x^3 = -2,16, \nabla f(x^3) = -4,32$$

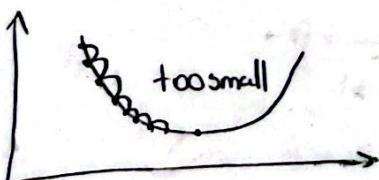
$$x^4 = -2,16 - 0,2 \cdot -4,32 = -1,792$$

$$x^4 = -1,296, \nabla f(x^4) = -2,592$$

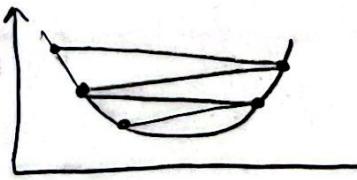
$$x^5 = -0,7776$$

t_k : step size

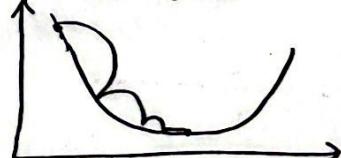
t_k çok büyükse



t_k çok büyük ise



Olaması istenen t_k değişken



$$\theta_{\text{new}} = \theta_t + \eta \cdot \Delta \theta \quad \begin{matrix} t_k : \text{step size} \\ \rightarrow \text{değişim miktari türerle alakolu} \\ \Delta \theta \text{ Taylor Serilerinden geliyor} \end{matrix}$$

Parameter Update Equations

$$w_{t+1} = w_t - \eta \cdot \nabla w_t \quad \begin{matrix} \text{step size / learning rate} \\ \rightarrow \text{change } w_t \end{matrix}$$

$$b_{t+1} = b_t - \eta \cdot \nabla b_t$$

$$\nabla w_t = \frac{\partial \mathcal{L}(w, b)}{\partial w}, \quad \nabla b_t = \frac{\partial \mathcal{L}(w, b)}{\partial b}$$

$$t \leftarrow 0$$

$$\text{max_iterations} \leftarrow 1000;$$

while $t < \text{max_iterations}$ do

$$w_{t+1} \leftarrow w_t - \eta \nabla w_t;$$

$$b_{t+1} \leftarrow b_t - \eta \nabla b_t;$$

$$t \leftarrow t+1$$

end

α Kullandığımız Loss fonksiyonunun $\mathcal{L}(w, b)$ türerini almalıyız ki ∇w_t 'yi bulalım

$$\mathcal{L}(w, b) = \frac{1}{2} \sum_{i=1}^N (f(x_i) - y)^2 \quad \begin{matrix} \text{Burfadede geçen } f(x) \\ \text{fonksiyonu sigmoid olardır} \end{matrix} \Rightarrow f(x) = \frac{1}{1 + e^{-(w \cdot x + b)}}$$

(x, y) yi sağlayan 1 çift olduğunu varsayıyalım. Ve buna göre çözüm

↓

$$\mathcal{L}(w, b) = \frac{1}{2} \cdot (f(x) - y)^2$$

$$\nabla w_t = \frac{\partial \frac{1}{2} \cdot (f(x) - y)^2}{\partial w} = \frac{1}{2} \cdot 2 \cdot (f(x) - y) \cdot \frac{\partial (f(x))}{\partial w} = (f(x) - y) \cdot \frac{\partial}{\partial w} \left(\frac{1}{1 + e^{-(w \cdot x + b)}} \right)$$

$$\frac{\partial}{\partial w} \left(\frac{1}{1 + e^{-(w \cdot x + b)}} \right) = \frac{-1}{(1 + e^{-(w \cdot x + b)})^2} \cdot \frac{\partial}{\partial w} (e^{-(w \cdot x + b)}) = \frac{-x \cdot e^{-(w \cdot x + b)}}{(1 + e^{-(w \cdot x + b)})^2}$$

$$\frac{f(x)}{g(x)} = \frac{f'(x) \cdot g(x) - f(x) \cdot g'(x)}{g(x)^2}$$

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

$$f(x) \star (1 - f(x)) \star x$$

$$f(x) = e^u \quad f'(x) = u' \cdot e^u$$

Böylede:

1 point (x, y) içində

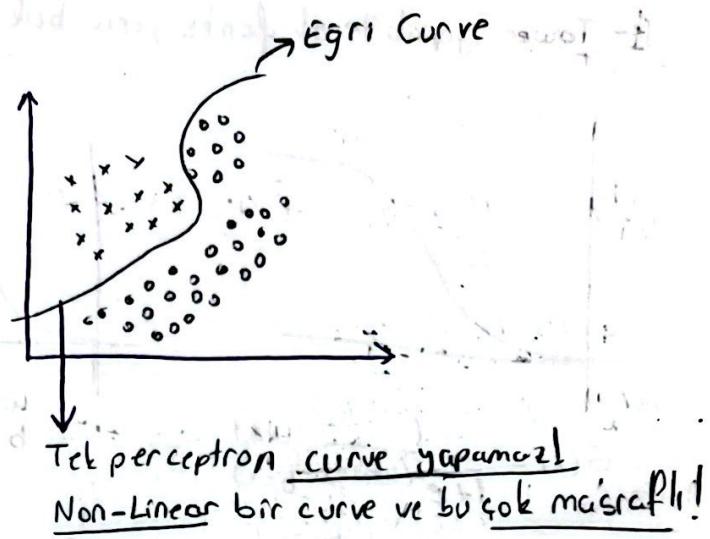
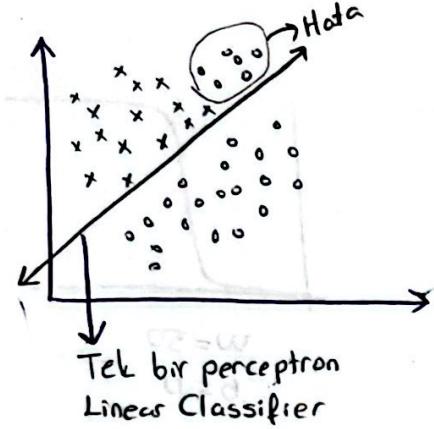
$$\nabla w = (f(x) - y) \cdot f'(x) \cdot (1 - f(x)) \cdot x$$

2 points (x, y) içində

$$\nabla w = \sum_{i=1}^2 (f(x_i) - y_i) \cdot f'(x_i) \cdot (1 - f(x_i)) \cdot x_i$$

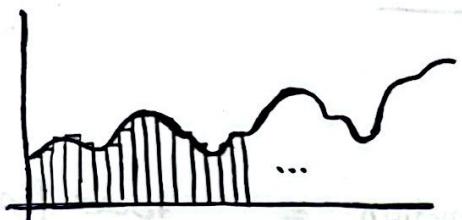
$$\nabla b = \sum_{i=1}^2 (f(x_i) - y_i) \cdot f'(x_i) \cdot (1 - f(x_i))$$

Hafta-5 Derin Öğrenme

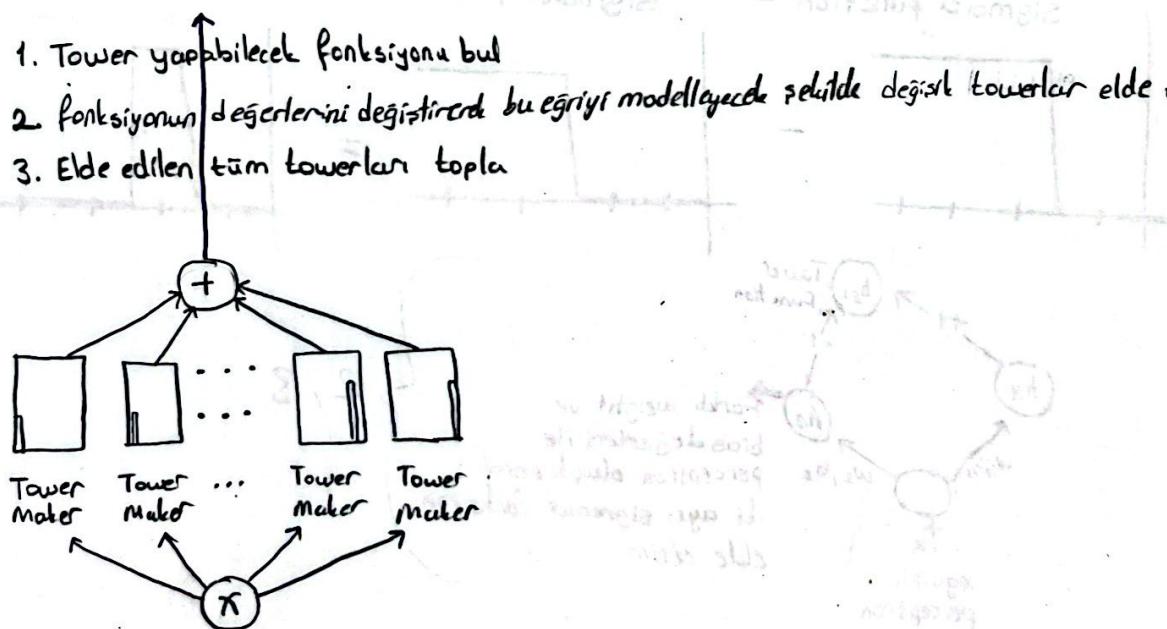


Doğrudan bir non-Linear curve çizmek çok zor! O zaman...

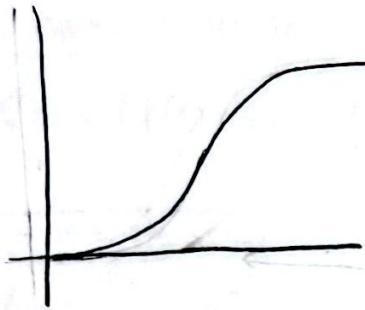
- Soldaki gibi bir eğriyi perceptron ile elde etmek çok zor ve malipli.
- Bunun için tower fonksiyonu kullanabiliriz



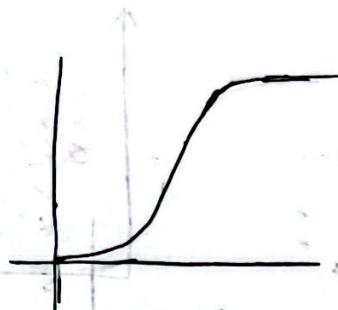
1. Tower yapabilecek fonksiyonu bul
2. Fonksiyonun değerlerini değiştirdik bu eğriyi modellaycede şekilde değişik towerlar elde ediyim.
3. Elde edilen tüm towerları topla



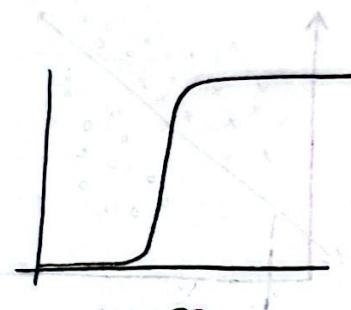
1- Tower yapabilecek fonksiyonu bul



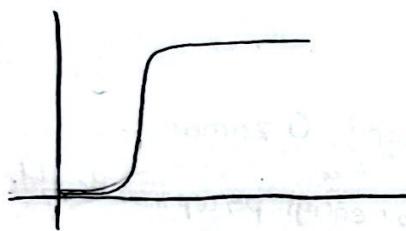
$$G(x) = \frac{1}{1+e^{-(wx+b)}} \quad w=1 \quad b=0$$



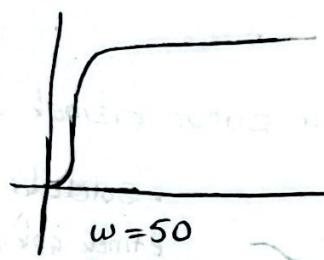
$$w=22 \quad b=0$$



$$w=50 \quad b=0$$

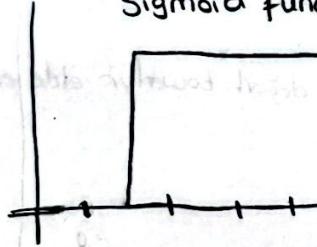


$$w=50 \quad b=10$$

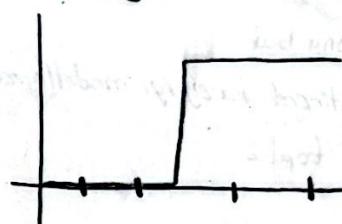


$$w=50 \quad b=40$$

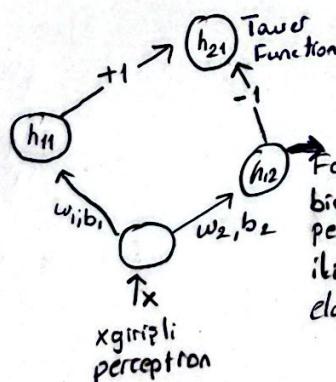
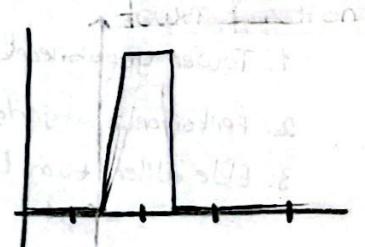
Sigmoid function -



Sigmoid function =



tower function

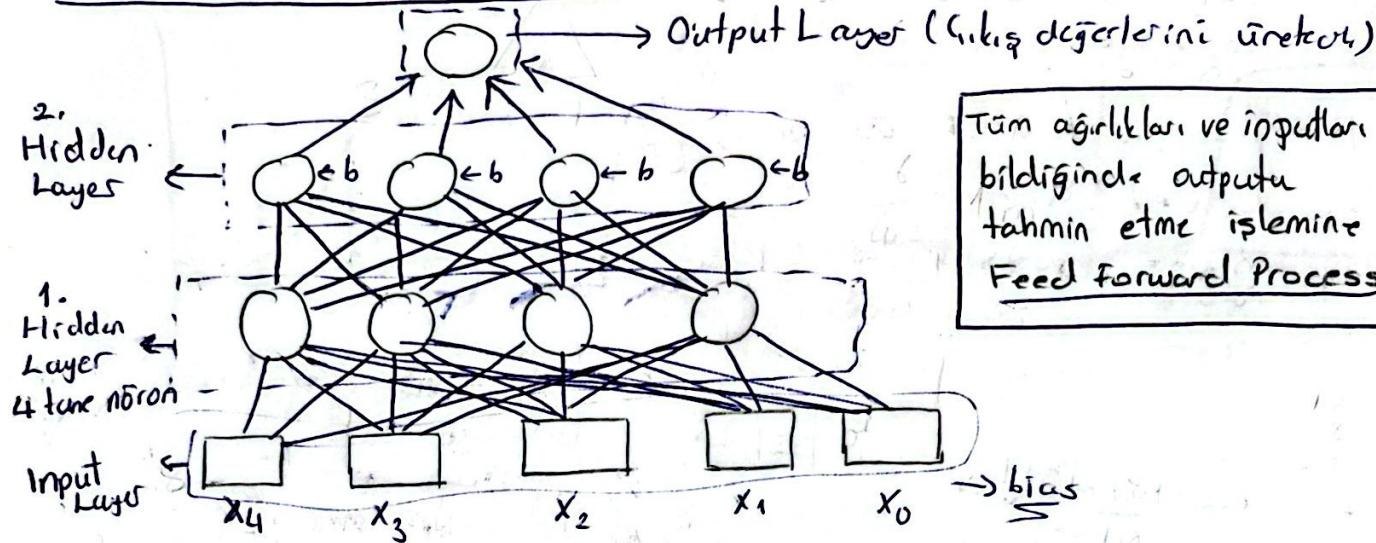


Farklı weight ve
bias değerleri ile
perceptron oluştururken
iki ayrı sigmoid fonksiyon
elde ettim

2, 3

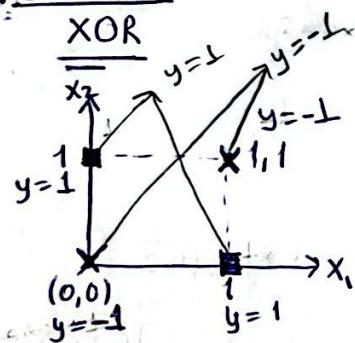
☒ Artık tek bir perceptron yok

Feed Forward Neural Network (Full Connected)

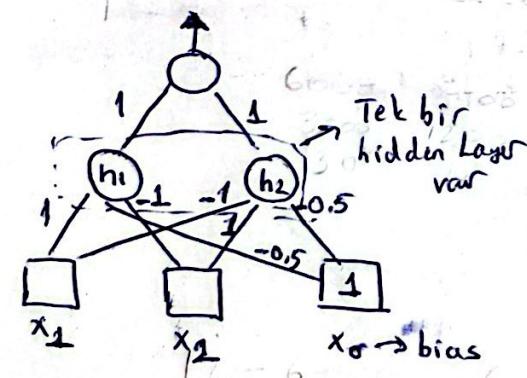


2 katmanlı bir neural network

! ÖRNEK:



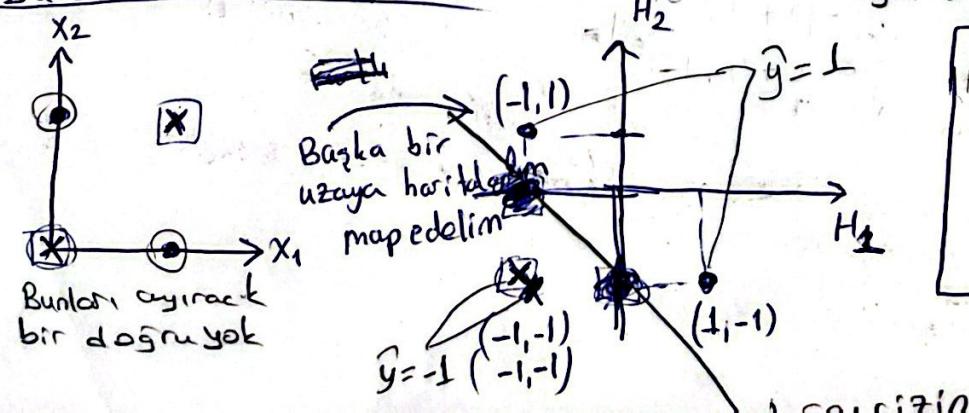
- XOR'u tek bir perceptron ile gerçekleştiremiyoruz. Dolayısıyla iki perceptron ile gerçekleştireceğiz.
- Mümkünse nöron (perceptron) sayısını azaltmadan problemleri gözleme çalşalım. Katman sayısı arttıkça yapılan hesaplama miktarı artar ve model ağırlasır



h_1, h_2 hesapla. h_1 ve h_2 için Toplam negatif ise -1
Oyada pozitifse 1

x_1	x_2	Class	h_1	h_2	g
0	0	-1	-1	-1	-1
0	1	1	-1	1	1
1	0	1	1	-1	1
1	1	-1	1	-1	-1

Bu örnekte biz ne yaptık?



Linear uzayda yapamadığımızı başka bir uzayda yapmaya çalışıyoruz

Backpropagation (Geri Yayılm)

Amaç: $\min \mathcal{L}(\Theta)$ Loss fonksiyonunu çok perceptronlu, çok katmanlı yapilar için minimize etmeye çalışiyor.

1. Network'e rastgele ağırlık ataması yap

2. Outputları sırasıyla üretiyoruz

3. Eğer istediğimiz output elde edilmediyse veya Loss varsa Gradient - Descent ile $\min \mathcal{L}(\Theta)$ için ağırlıkları güncelliyoruz.

4. Çok katmanlıda gradientler geriye doğru hesaplanarak ağırlıklar güncellenir.

Gradient Descent Hatırlayalım

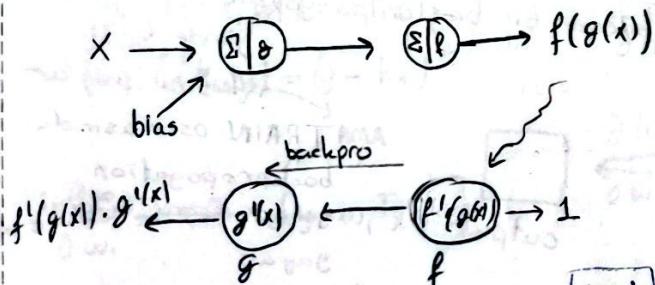
$$w_{i+1} = w_i - \eta \cdot \Delta w_i \quad \Theta = (w, b)$$

$$\frac{\partial \mathcal{L}(\Theta)}{\partial w} = \frac{\partial}{\partial w} \left(\frac{1}{2} \cdot (f(x) - y)^2 \right)$$

$$\Delta w = (f(x) - y) \cdot f'(x) \cdot (1 - f(x)) \cdot x$$

$$f(x) = \frac{1}{1 + e^{-(wx+b)}} : \text{Sigmoid fonksiyon}$$

Backpropagation is an algorithm which is created to test errors which will travel back from input nodes to output nodes

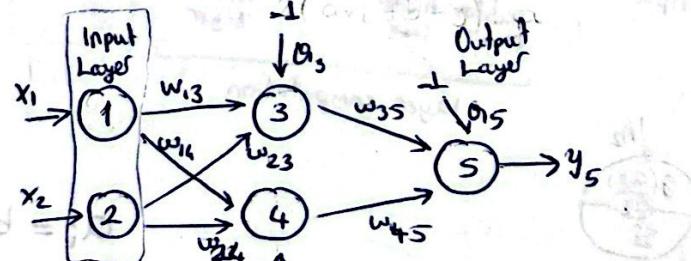


Backpropagation yaklaşımında $f(g(x))$ fonksiyonun türerini alırız.

Örnek

$$X = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \quad y = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Activation Function



$$G(w^T x) = \frac{1}{1 + e^{-(w^T x)}}$$

$$x = [1 \ 1] \text{ için } y_3 = G(1 \cdot 0,5 + 1 \cdot 0,4 - 1 \cdot 0,8) = G(0,1) = \frac{1}{1 + e^{-0,1}} = 0,525$$

Init Random Variables

$$w_{13} = 0,5$$

$$w_{14} = 0,9$$

$$w_{23} = 0,4$$

$$w_{24} = 1,0$$

$$w_{35} = -1,2$$

$$w_{45} = 1,1$$

$$\theta_3 = 0,8 \quad \theta_4 = -0,1$$

$$\theta_5 = 0,3$$

$$y_5 = G(0,525 \cdot -1,2 + 0,8808 \cdot 1,1 - 0,3) = G(0,03888) = 0,5097$$

$$e = (0 - 0,5097) = -0,5097$$

$$\text{new } w_{35} = \text{old } w_{35} - \eta \cdot \Delta w_{35}$$

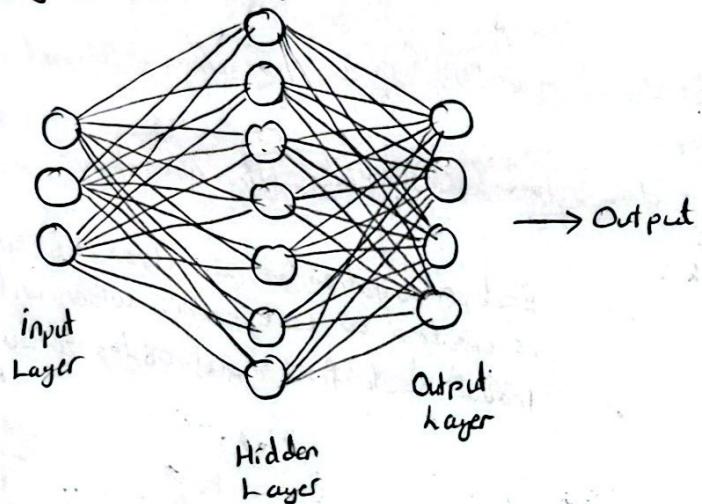
Hata

Derin Öğrenme Hafta - 6

(Gelişmiş nöral ağları) next page ongland

Shallow Neural Network (Sig Neural Network)

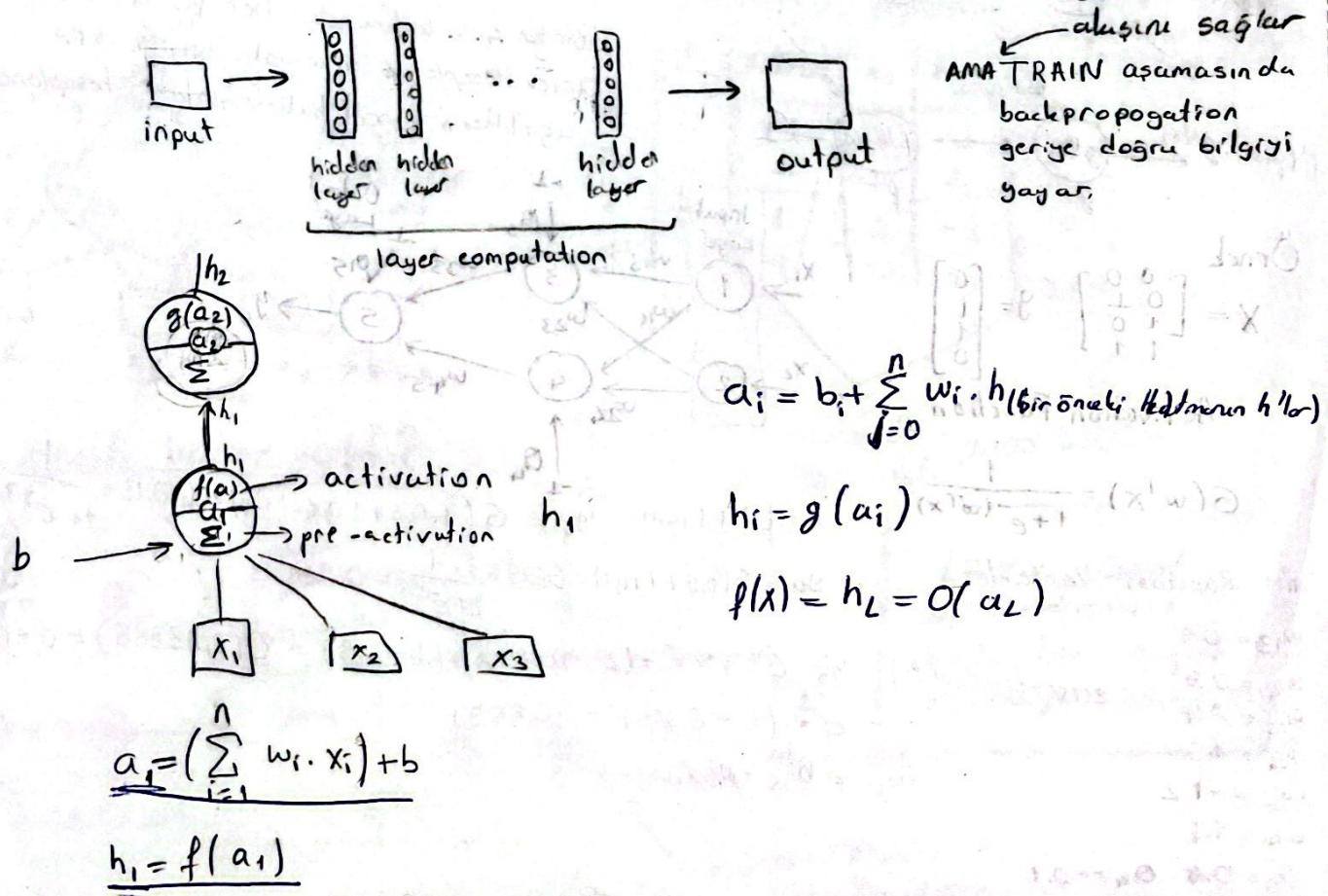
- Bir input layer, bir output layer ve bir hidden layerden oluşur. Katman sayısı artınıldığı zaman deep neural network deniyor.



! Burada input layerların hepsi hidden layer'a ve hidden layerların hepsi her output layer'a bağlı olduğu için full connected bir yapıdadır.

Feedforward Neural Network (Multilayered network of Neurons) Çok katmanlı nöral network

- Deep neural network içeriir bunu
- 1 input layer, 1 output layer, birden çok full connected hidden layer var
- Artificial yapay sinir ağları. Cycle oluşturabilecek bir bağlantı yoktur. İlerle tek bir yönde bilgi akışını sağlar



Example

$k=2$

assume ($y=1$)

$$e = \ell(f_2(f_1(x, w_1), w_2)) = \ell(h_2, y)$$

$$e = -\log(G(w_1^T x)) = -\log(h_2)$$

$$\frac{\partial e}{\partial w_1} = ? = \frac{\partial e}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_2} \cdot \frac{\partial w_2}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_1}$$

$$\frac{\partial h_1}{\partial w_1} = x$$

$$\frac{\partial e}{\partial h_2} = \frac{\partial}{\partial h_2} \cdot -\log(h_2) = -\frac{\partial}{\partial h_2} \cdot \frac{1}{\ln 10} \cdot \ln h_2 = \frac{-1}{\ln 10} \cdot \frac{1}{h_2}$$

$$\frac{\partial h_2}{\partial h_1} = \frac{\partial}{\partial h_1} G(h_1) = G(h_1) \cdot (1 - G(h_1))$$

$$\frac{\partial e}{\partial w_1} = \frac{-1}{\ln 10} \cdot \frac{1}{h_2} \cdot G(h_1) \cdot (1 - G(h_1)) \cdot x$$

$$G'(h_1) = G(h_1) \cdot (1 - G(h_1))$$

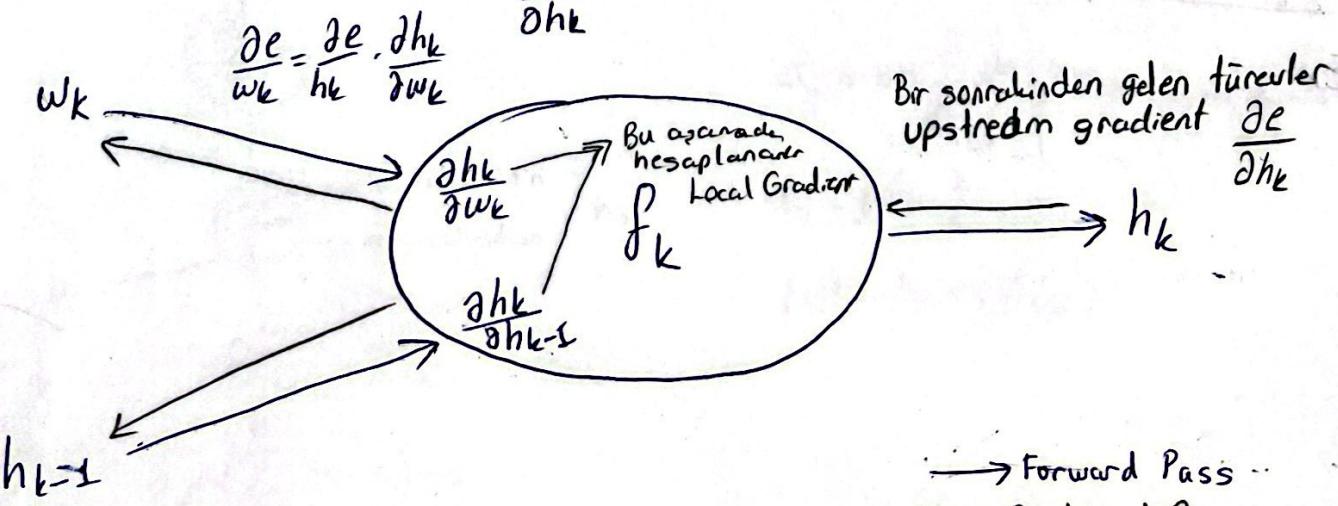
$$= \frac{-1}{\ln 10} \cdot \frac{1}{G(h_1)} \cdot G(h_1) \cdot (1 - G(h_1)) \cdot x$$

$$(1 - G(w_1^T x)) = G(-w_1^T x)$$

$$= \frac{-x}{\ln} \cdot -G(-w_1^T x) = -G(-w_1^T x) \cdot \frac{x}{\ln}$$

$$\frac{\partial e}{\partial w_k} = \left[\frac{\partial e}{\partial h_{k+2}} \cdot \frac{\partial h_{k+2}}{\partial h_{k+1}} \cdots \frac{\partial h_{k+1}}{\partial h_k} \right] \cdot \frac{\partial h_k}{\partial w_k}$$

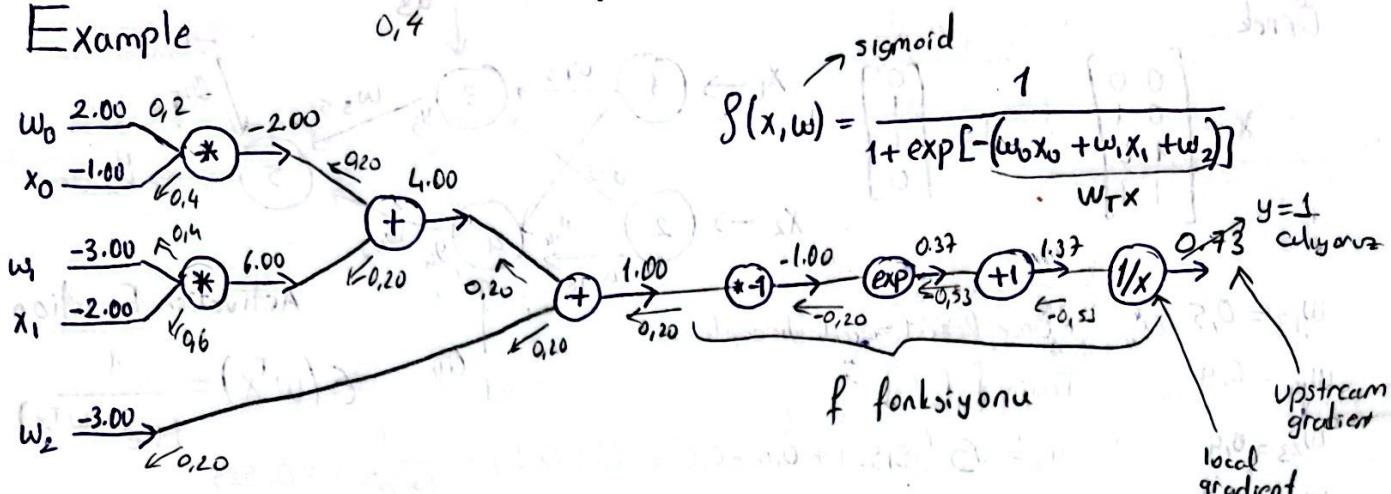
Upstream Gradient Local gradient



→ Forward Pass

← Backward Pass

Example



Forward yapalum öncelikle

$$1. \text{ adim } \frac{1}{x} \text{ deyim } e = \frac{1}{x} \quad h_n = 1,37$$

$$\left(\frac{1}{x}\right)' = -\frac{1}{x^2}$$

$$-\frac{1}{(1,37)^2} \rightarrow 1 = -0,53$$

$$2. \text{ adim } +1 \text{ deyim } h_{n+1} = -0,53 \quad h_n = 0,37$$

+ Add gate → gradientleri eşit şekilde dağıtır

* Multiply gate → gradientleri switch eder

-0,53 ü geçirir

$$3. \text{ adim } \exp \quad h_{n+1} = -0,53 \quad h_n = -1,00$$

$$\exp(x) = e^x$$

$$(e^x)' = e^x$$

$$\exp(-1) \cdot (-0,53) = -0,20$$

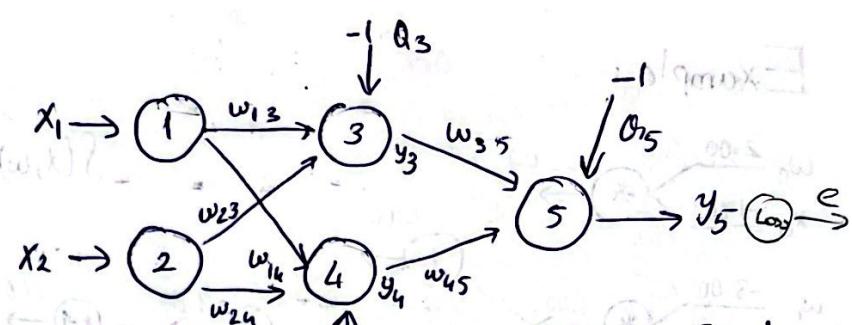
$$4. \text{ adim } *(-1)$$

switch eder ama tek yol var
o yuzden sorgulam

$$(-0,20) * (-1) = 0,20$$

Örnek

$$X = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \quad y = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$



$w_{13} = 0,5$

$w_{14} = 0,9$

$w_{23} = 0,4$

$w_{24} = 1,0$

$w_{35} = -1,2$

$w_{45} = 1,1$

$\theta_3 = 0,8$

$\theta_4 = -0,1$

$\theta_5 = 0,3$

$\eta = 0,1$

* Once Feed forward hesaplama

Train [1 1]:

$$y_3 = G(10,5 \cdot 1 + 0,4 - 0,8) = G(0,1) = \frac{1}{1+e^{-0,1}} = 0,525$$

$$y_4 = G(1,0 + 0,9 + 0,1) = G(2,0) = \frac{1}{1+e^{-2,0}} = 0,8808$$

$$y_5 = G(-1,2 \cdot 0,525 + 1,1 \cdot 0,8808 - 0,3) = 0,5097 \quad y = 0 \text{iken} \\ y = 0,5097 \text{ çıktı,}$$

$e = \frac{1}{2} \cdot (G(w^T x_i) - t_i)^2$

$e = \frac{1}{2} \cdot (0,5097 - 0)^2$

SS notasyonu

$$\frac{\partial e}{\partial y_5} = \frac{\partial \cdot}{\partial (G(w^T x))} \cdot \frac{1}{2} (G(w^T x_i) - t_i)^2 = \overbrace{(G(w^T x_i) - t_i)}^e \cdot G(w^T x_i) \cdot (1 - G(w^T x_i)) \\ = e \cdot G(h_5) \cdot (1 - G(h_5))$$

$\Delta w_{35} = \frac{\partial e}{\partial y_5} \cdot \frac{\partial y_5}{\partial w_{35}}$

$\Delta w_{35} = S_5 \rightarrow y_3$

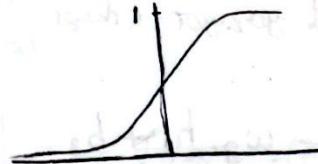
$w_{35} = w_{35} - \eta \cdot \Delta w_{35}$

$\uparrow S_5 \text{ upstream gradient} = -0,5097 \cdot 0,5097 \cdot (1 - 0,5097) = -0,1274$

$\frac{\partial f}{\partial w_{35}} = (f(x) - y) \cdot f'(x) \cdot (1 - f(x)) \cdot x$

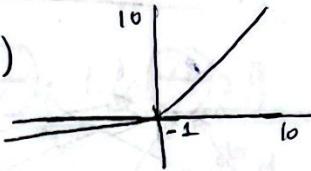
Sigmoid

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



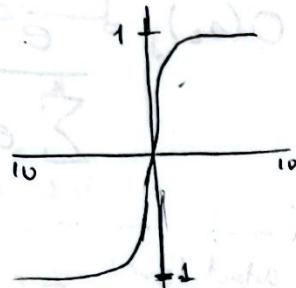
Leaky ReLU

$$\max(0.1x, x)$$



tanh

$$\tanh(x)$$

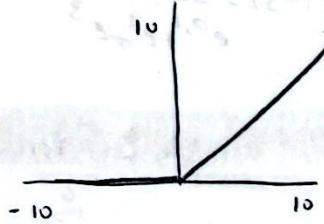


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

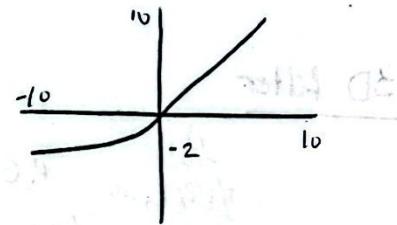
ReLU

$$\max(0, x)$$



ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Sigmoid

$$g(x) = \frac{1}{1+e^x} \xrightarrow{\text{Türevi}} g(x) \cdot (1-g(x))$$

TAN

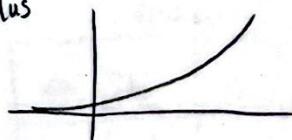
$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \xrightarrow{\text{Türevi}} 1 - g(x)^2$$

ReLU

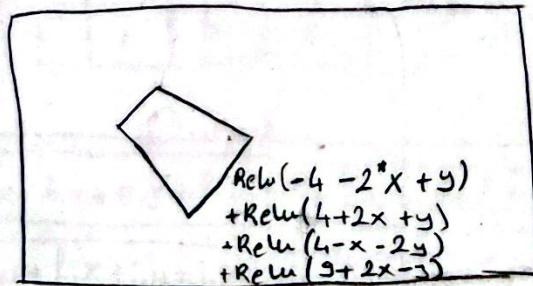
$$(x) = \begin{cases} 0, & x \leq 0 \\ x, & x > 0 \end{cases}$$

$$\xrightarrow{\text{Türevi}} \begin{cases} 0, & x \leq 0 \\ 1, & x > 0 \end{cases}$$

Softplus



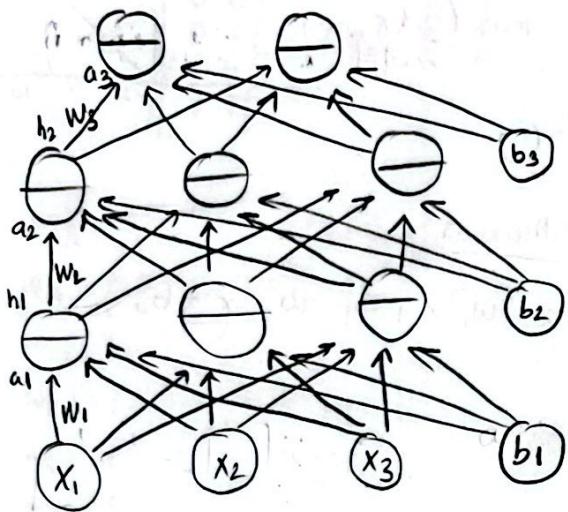
$$\ln(1+e^x) \xrightarrow{\text{Türevi}} \frac{e^x}{1+e^x} = \frac{1}{1+e^{-x}}$$



ReLU kullanarak birde fazla perceptron ile ReLU fonksiyonlarını toplayıp bir çokgen elde edebiliyoruz.

Non-Linear bir dağılım varsa ReLU kullanabiliriz.

Softmax function Olasılıksal olarak yazıyor ekrana



$$a_3 = w_3 \cdot h_2 + b_3$$

$$a_L = w_L \cdot h_{L-1} + b_L$$

$$\hat{y}_j = O(a_L)_j = \frac{e^{a_L, j}}{\sum_{i=1}^k e^{a_L, i}}$$

$$\sum_{i=1}^k e^{a_L, i}$$

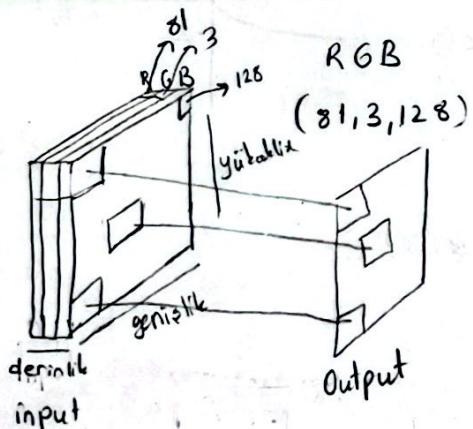
Hidden Layer → Output Layer

$$O(a_1) = \frac{e^{a_1}}{e^{a_1} + e^1 + e^2} = 0,1$$

$$O(a_2) = \frac{e^1}{e^{a_1} + e^1 + e^2} = 0,2$$

$$O(a_3) = \frac{e^2}{e^{a_1} + e^1 + e^2} = 0,7$$

3D Filter



Pratikte 3D görüntüler için de 2D konvolusyon işlemleri yapıyoruz

Loss function: Cross-entropy (Gaproz Entropi)

iki tane olasılık dağılımının arasındaki farklı ortaya çıkarıyor

$y = [1, 0, 0, 0]$ betlediğim değerler

\hat{y} : predicted value

$$\mathcal{L}(\theta) = - \sum_{c=1}^k y_c \log \hat{y}_c$$

class sayısı, yani output sayısı
actual value
predicted value

Example: Cross-Entropy
 \hat{y} : predicted y : actual

0,1	0
0,2	0
0,7	1

Softmax ile bulunmuş

$$\begin{aligned} \mathcal{L}(0_1) &= -(0 \cdot \log(0,1) + 0 \cdot \log(0,2) + 1 \cdot \log(0,7)) \\ &= -\log(0,7) \end{aligned}$$

Example: Softmax

$$\begin{aligned} \hat{y} &= \begin{bmatrix} 0,1 \\ 1 \\ 2 \end{bmatrix} \text{ Softmax layer} \\ \hat{y}_1 &= \frac{e^{0,1}}{e^{0,1} + e^1 + e^2} \approx 0,1 \\ \hat{y}_2 &= \frac{e^1}{e^{0,1} + e^1 + e^2} \approx 0,2 \\ \hat{y}_3 &= \frac{e^2}{e^{0,1} + e^1 + e^2} \approx 0,7 \end{aligned}$$

Hafta - 7 Derin Öğrenme

$$s_t = \sum_{a=0}^6 x_{t-a} w_a$$

$$\begin{array}{ccccccccc} w_{-6} & w_5 & w_4 & w_3 & w_2 & w_1 & w_0 \\ \hline w & 0,01 & 0,01 & 0,02 & 0,02 & 0,04 & 0,4 & 0,5 \end{array}$$

$$\begin{array}{cccccccccc} -6 & -5 & -4 & -3 & -2 & -1 & 0 \\ \hline x & 1,00 & 1,10 & 1,20 & 1,40 & 1,70 & 1,80 & 1,90 & 2,10 & 2,20 & 2,40 & 2,50 & 2,70 \end{array}$$

$$\begin{array}{cccccc} s & 1,80 & 1,96 & 2,11 & 2,16 & 2,28 & 2,42 \end{array}$$

$$s_6 = 0,01 \cdot 1,00 + 0,01 \cdot 1,10 + 0,02 \cdot 1,20 + 0,02 \cdot 1,40 + 0,04 \cdot 1,70 + 0,4 \cdot 1,80 + 0,5 \cdot 1,90 = 1,80$$

Input			
a	b	c	d
e	f	g	h
i	j	k	l

w	x
y	z

roldukta mola (2) sbitc2

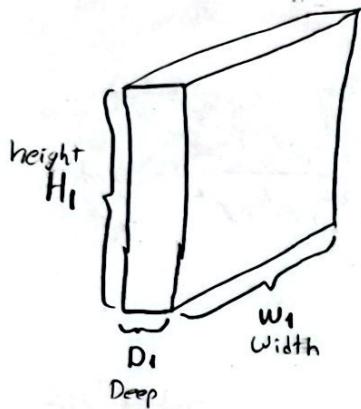
1	0	1	1	0
1	1	0	0	1

Output

$$\begin{array}{ccc} a \cdot w + b \cdot x + e \cdot y + f \cdot z & b \cdot w + c \cdot x + f \cdot y + g \cdot z & c \cdot w + d \cdot x + g \cdot y + h \cdot z \\ e \cdot w + f \cdot x + i \cdot y + j \cdot z & f \cdot w + g \cdot x + j \cdot y + k \cdot z & g \cdot w + h \cdot x + k \cdot y + l \cdot z \end{array}$$

Kernel işleminden sonra oluşan çıktıının boyutları orjinal girdiye göre farklı olabilir. Buna sebep olan faktörler:

- inputs
 - filters
 - outputs.



Kaç tane filtre kullanıyorsunuz?

Filtrelerimizin uzaysal boyutu (spatial extent) (F) ne?

(Stride) Kayma. Miktar, ne?

Output: $W_2 \times H_2 \times D_2$ yukarıdaki değerlere bağlı olarak hesaplanacak.

Kenar bilgileri önemliyse padding yaparım. Örneğin zero-padding yapabilirsiniz

Input						
10	0	0	0	0	0	0
10						0
10						0
0						0
10						0
10						0
0	0	0	0	0	0	0

$$* \quad \begin{matrix} \text{Kernel} \\ 3 \times 3 \\ \text{sin} \end{matrix} \Rightarrow \begin{matrix} \text{Output} \\ 5 \times 5 \\ \text{placeholder} \end{matrix}$$

Kenarları sıfırla
doldurdum

$$\underline{5 \times 5} \rightarrow 7 \times 7$$

$$\text{Örnek 11} \quad \begin{array}{ccccc} \text{Input} & \text{Kernel} & \text{Output} & W_2 = W_1 - F + 2P + 1 \\ 7 \times 7 * 5 \times 5 & \rightarrow & 3 \times 3 & H_2 = W_1 - F + 2P + 1 \end{array}$$

kernel boyutu
↑ padding

Stride (S) Adam Milton

Input

$$7 \times 7 + 29$$

$$P = 1$$

Keweenaw

Output

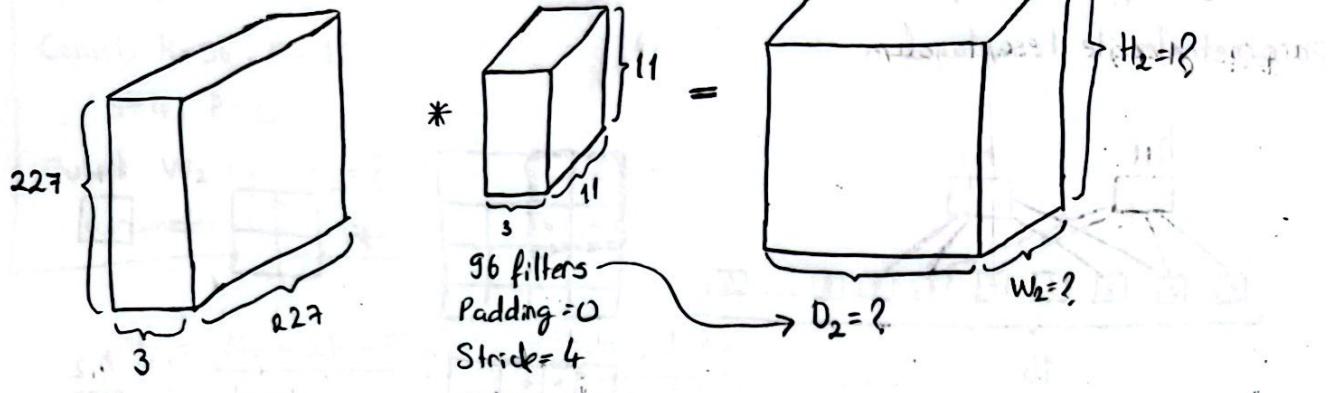
$S = 1$ also
Output 7×7
ohne

$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$

$$H_2 = \frac{H_1 - F + 2P}{S} + 1$$

$$\frac{7-3+2}{2} + 1 = 4$$

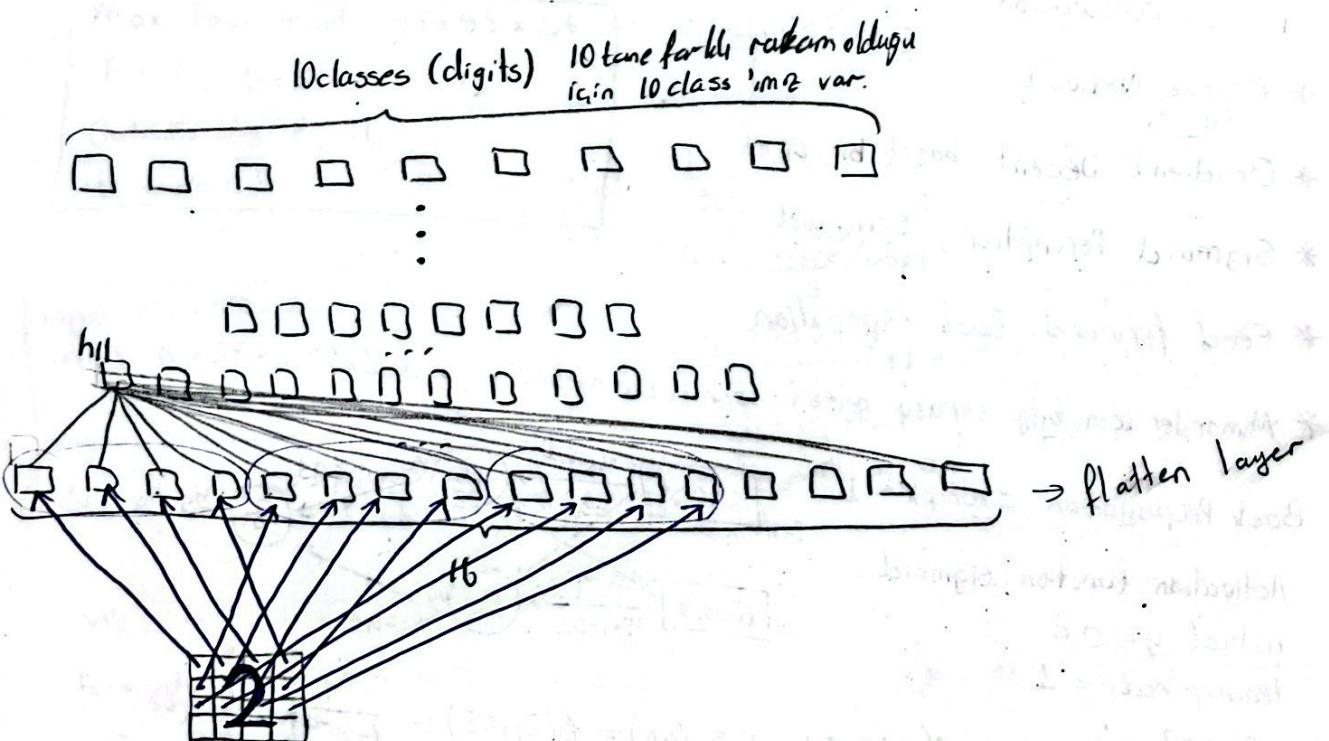
Example:



$D_2 = 96$ // 96 tane filtre kullanmak 96 feature map oluşturmaktadır.

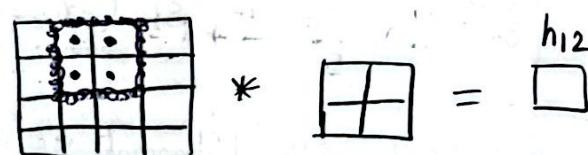
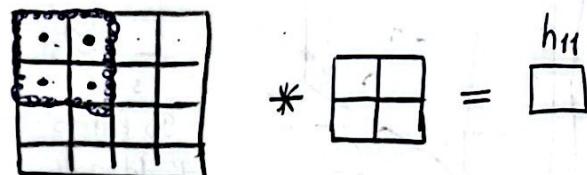
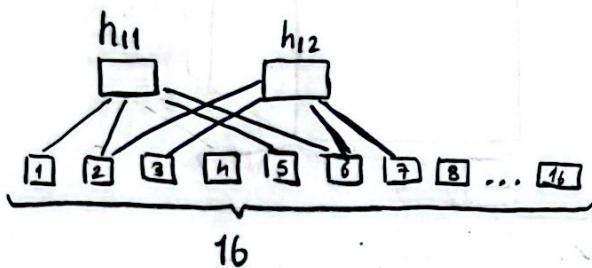
$$W_2 = \frac{W_1 + 2P - F}{S} + 1 = \frac{227 + 0 - 11}{4} + 1 = 55$$

$$H_2 = \frac{H_1 + 2P - F}{S} + 1 = \frac{227 + 0 - 11}{4} + 1 = 55$$



Örneğin yukarıdaki gibi feed-forward nöral network yapılsayda h_{11} in hesaplanması için 16 inputu da kullanmamız gerekecekti. Yani görüntüün tüm pixel değerleri kullanılocaktı = çok fazla çarpma ve parametre işlemi anlamına gelir.

Biz h_{11} 'in hesaplanmasıında tüm pixelleri kullanmak yerine kernelimizdeki parametreler ile hesaplayalım



- * Konvolüsyon işlemi bize daha ayrik (sparse) (dense olmayan) bir analiz sağlıyor

CNN feedforward neural network gibi bir hesaplama sürecine sahip olmasına rağmen her bir kısımındaki local pixel değerleri aslında output oluşturduğu için bağlantı sparse

Vize için konular (4-5 soru olacak)

- * CNN, layer calculation, Örnek yaptık

- * Neural Network

- * Gradient Decent basit bir örnek

- * Sigmoid Perceptron, Sigmoid iterasyon

- * Feed forward, Back Propogation

- * Mimariler için bilgi sorusu gelebilir parametre nedir? layer nedir?

Back Propogation Example - 1

Activation function: sigmoid

Actual $y = 0.5$

learning rate = 1

$$X = \begin{bmatrix} 0,35 \\ 0,9 \end{bmatrix}$$

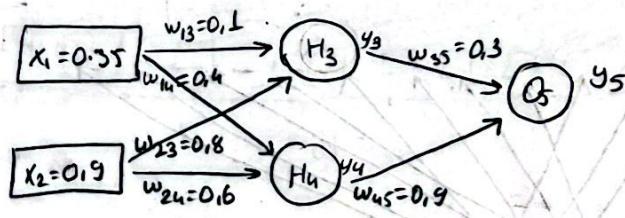
$$y_3 = G(0,35 \cdot 0,1 + 0,9 \cdot 0,8) = G(0,755) = \frac{1}{1+e^{-0,755}} = 0,68$$

$$y_4 = G(0,35 \cdot 0,4 + 0,9 \cdot 0,6) = G(0,68) = \frac{1}{1+e^{-0,68}} = 0,6637$$

$$y_5 = G(0,68 \cdot 0,3 + 0,6637 \cdot 0,9) = G(0,801) = 0,69 \text{ (Network output)}$$

$$\text{Error} = y_{\text{target}} - \hat{y}_5 = -0,19 \quad w_{\text{new}} = w_{\text{old}} -$$

$$0,5 - 0,69 = -0,19$$



Convolution

Input: $227 \times 227 \times 3$

Conv1: $K=96, F=11$

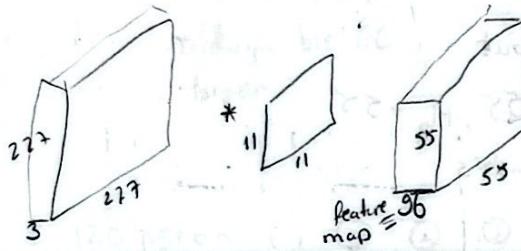
$S=4, P=0$

Output: $W_2 = ?$ $H_2 = ?$

Parameters: ?

$$W_2 = \frac{W_1 + 2P - F}{S} + 1 = \frac{227 + 0 - 11}{4} + 1 = 55$$

$$H_2 = 55 \quad \text{Parameters} = 96 * (11 * 11 * 3)$$



Max Pool Input: $55 \times 55 \times 96$

$F=3, S=2$

Output: $W_2 = ?, H_2 = ?$

Parameters: ?

Input: $27 \times 27 \times 96$

Conv1: $K=256, F=5$

$S=1, P=0$

$$W_2 = \frac{W_1 + 2P - F}{S} + 1 = \frac{27 + 0 - 5}{1} + 1 = 23$$

$$W_2 = H_2 = 23$$

$$\text{Parameters} = (5 * 5 * 96) * 256 = 0.6M$$

$$\text{Parameters} = (3 * 3 * 384) * 384$$

$$\text{Parameters} = (3 * 3 * 384) * 256$$

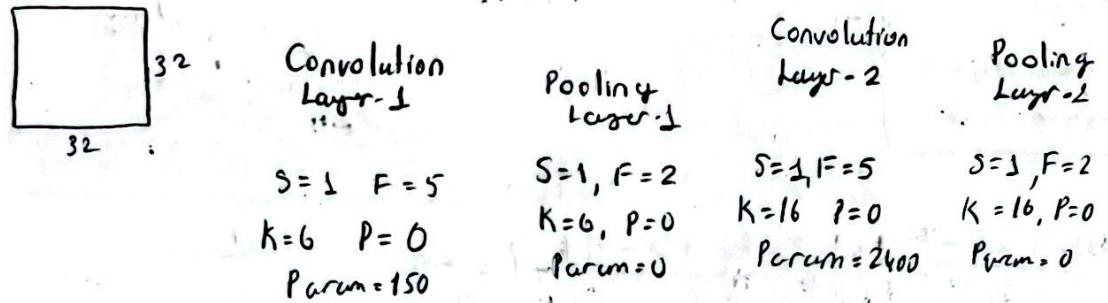
$$W_2 = \frac{11 - 3}{1} + 1 = 7$$

$$\text{Parameters} = (3 * 3 * 256) * 384 = 0.8M$$

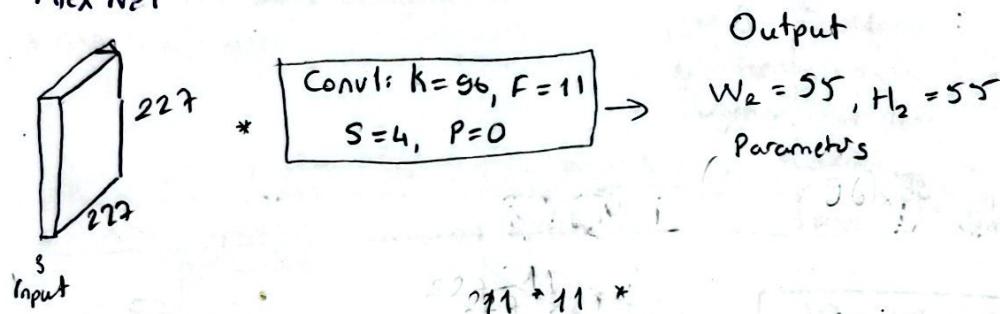
$$\text{Parameters} = 0$$

Hafta - 9 Ders

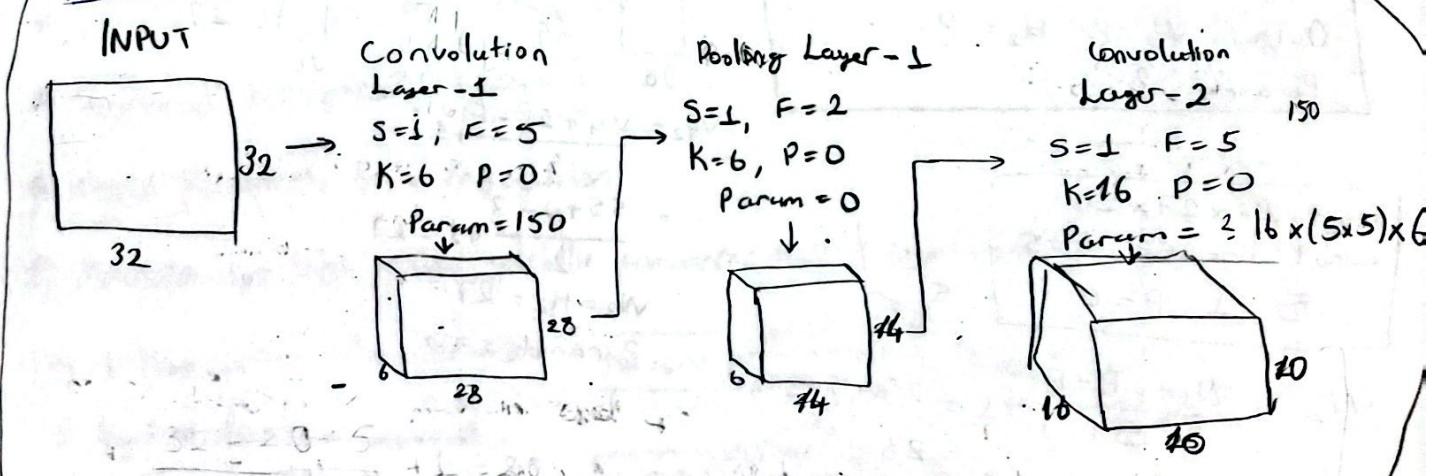
LeNet-5



Alex Net

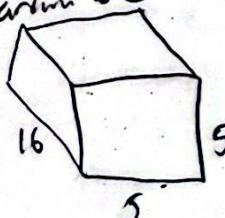


ÖRNEK:



Pooling Layer-2

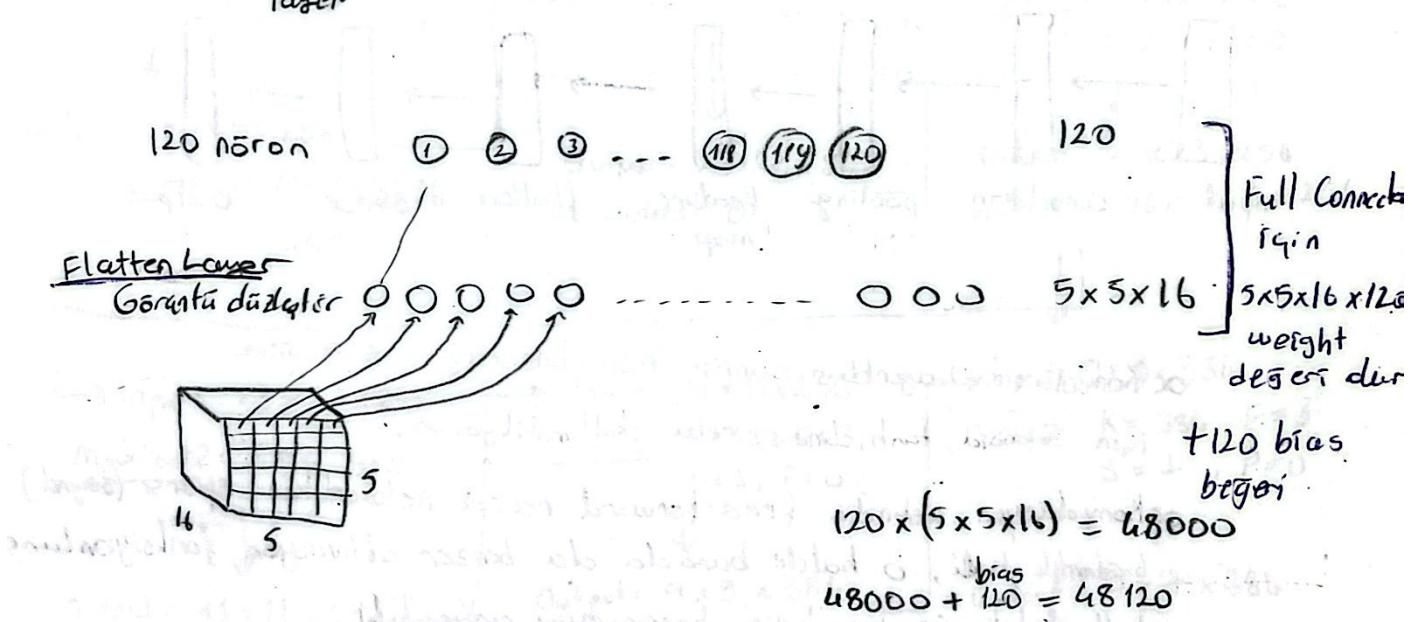
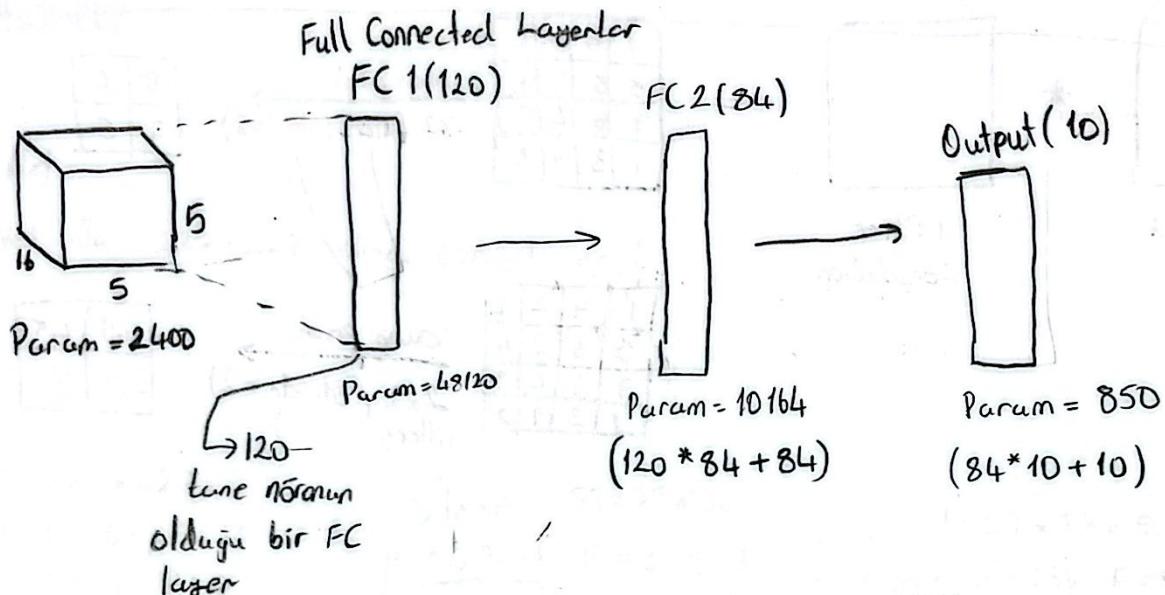
$S=1, F=2$
 $K=16, P=0$
 $\text{Param} = 0$



FEATURE ENGINEERING (Feature Extraction)

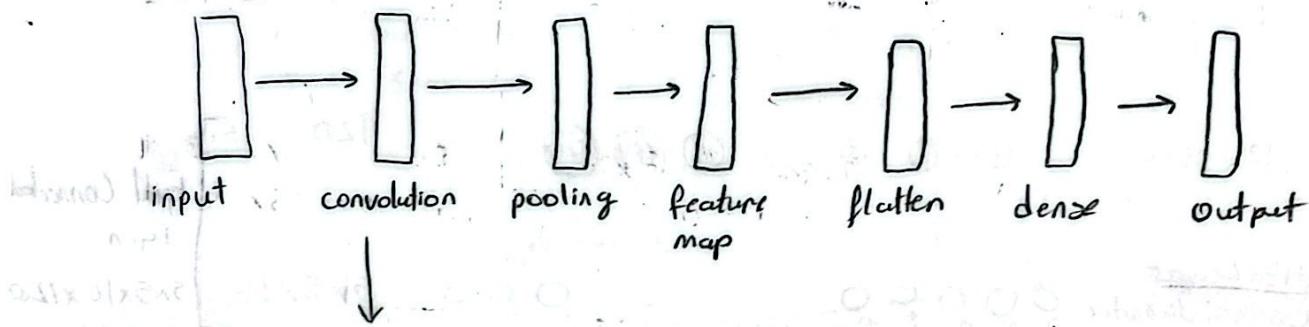
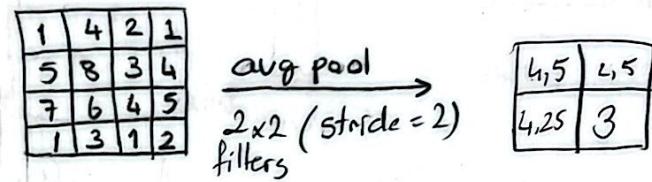
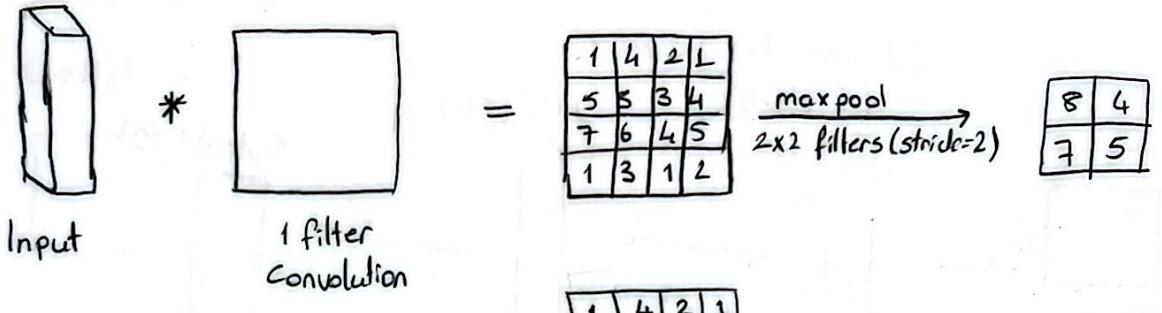
Birim boyutlu görüntülerin
matrikslerine dönüştürülerek, ondan featurelar
elde edilir.

Feature Çıkartma işlenmesinden sonra



Pooling Layer

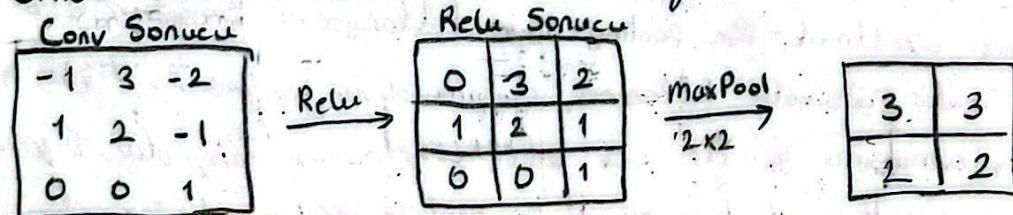
Parametre sayımızı azaltmak için pooling layer kullanıyoruz. Boyutumuz ne kadar fazla ise o kadar parametre kullanırız, çarpma işlemi yaparız. Bu bizim sisteminizin yavaş olmasına yarar da çok belirli türlerde sebepler olur. O yüzden arada bir pooling layer kullanarak hem gürültülü genetsiz verileri elime ediyoruz hem de boyut azaltıyoruz.



α Konvolusyon Layer'dan sonra non-linearity'ı saglamak
çünktü out için sigmoid, tanh, Linear, relu kullanabiliyoruz.

α Konvolusyon aslinda feed-forward neural network'in sparse (seyrek) baglantili hali, o halde burada da benzer aktivasyon fonksiyonlari kullanabilir ve bu bizi basarimizi artıracabilir

Örnek Conv - Activation ReLU - Max Pooling

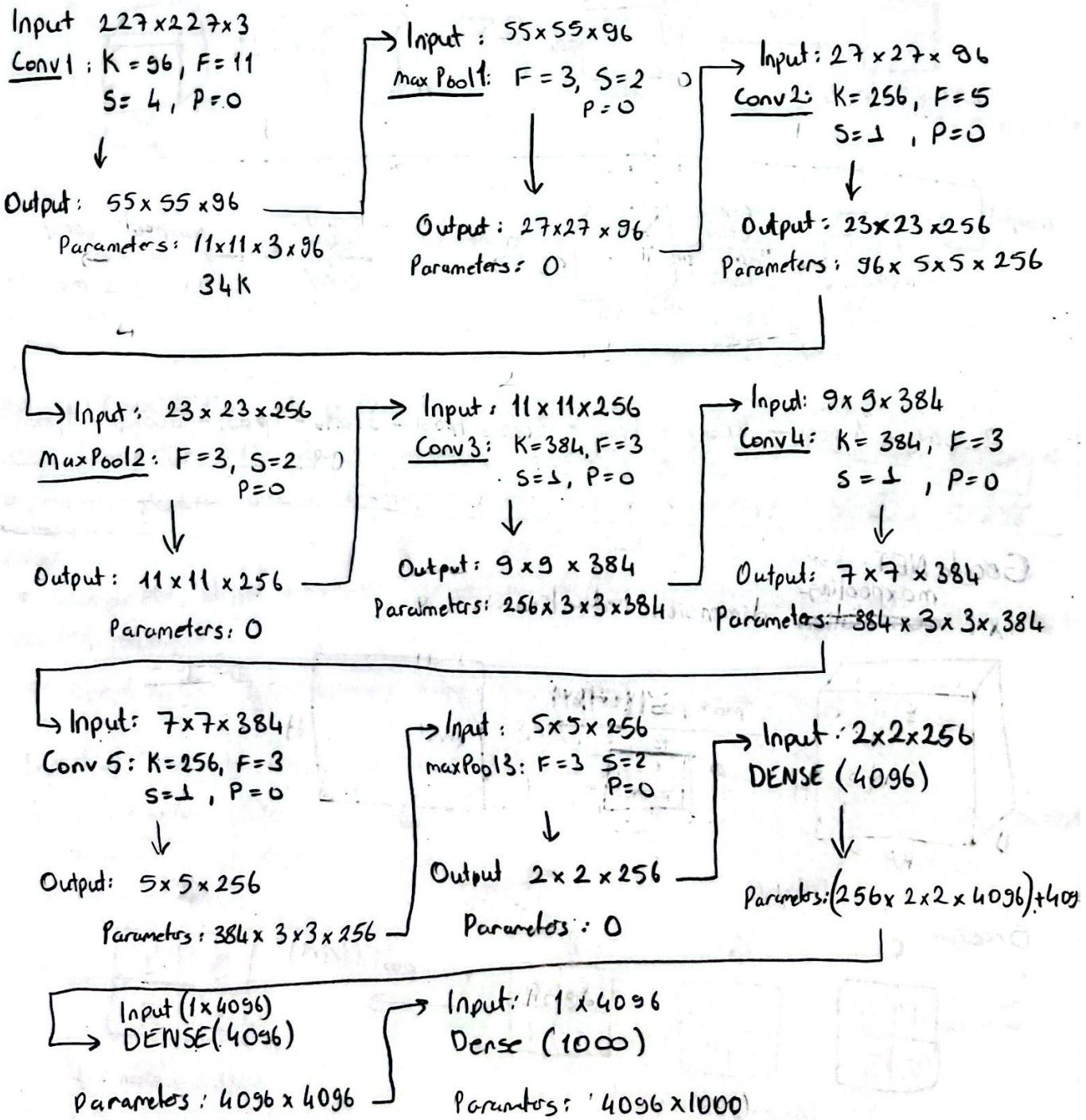


~~EX-1~~

Hafıza - 11 /

AlexNet

o AlexNet illa 227×227 olmak zorunda değil



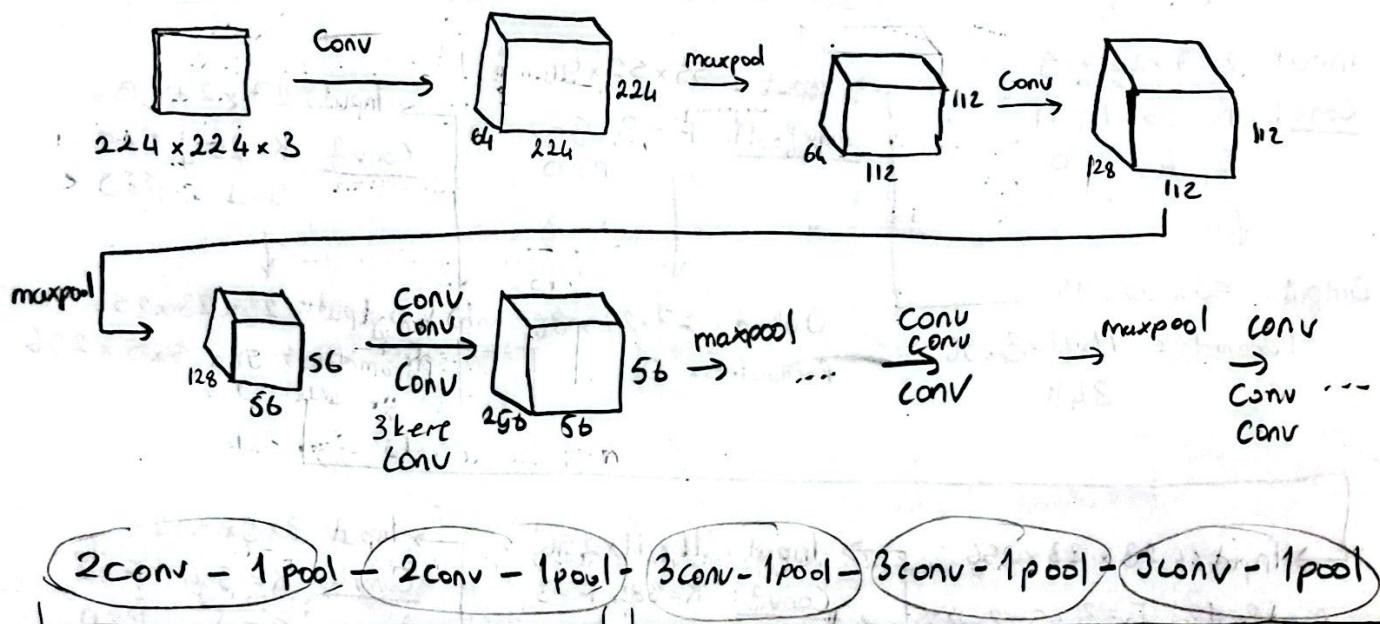
ZFNET

ZFNET, ALEXNET'in modifiye versiyonu diyebiliriz

Temel fark kullandığı filtrelerin size boyutu daha küçük, stride daha büyük

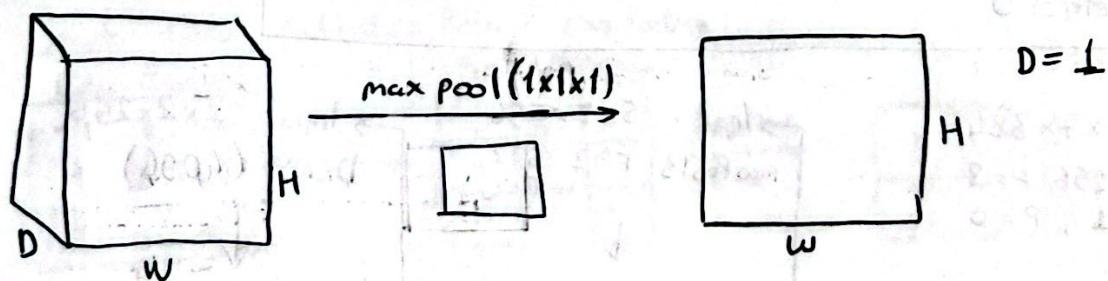
VGG NET "Very Deep Convolutional Network"

= Visual Geometry Group



Google NET

maxpooling
 1×1 ~~rotation~~ dimension reduction



Örneğin R

1	0
2	3

2	1
1	0

3	0
1	1

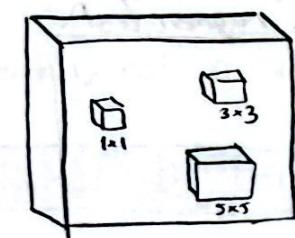
maxpool $(1 \times 1 \times 1)$

3	1
2	3

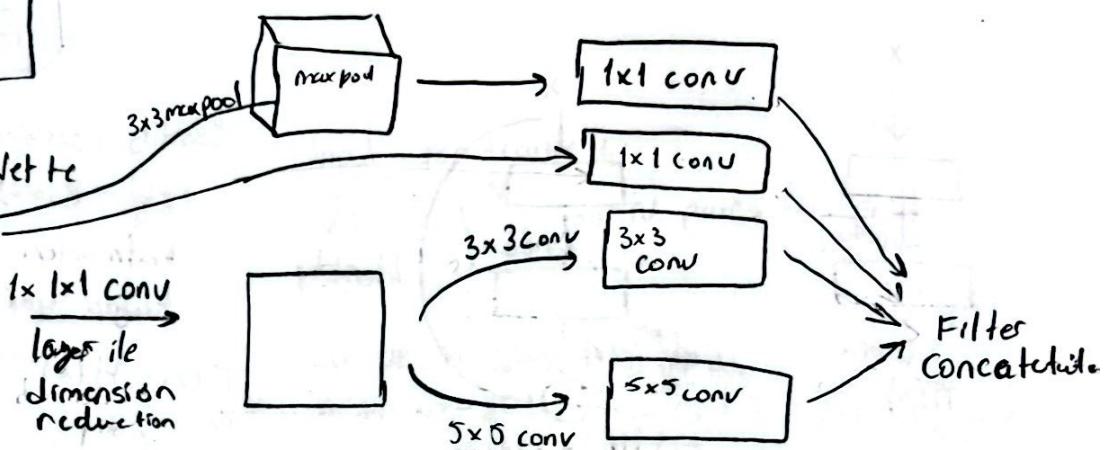
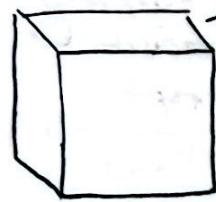
Dimension = 3

Dimension = 1

Birden fazla filtre boyutunu degüttirerek bir avantaj yakalayabilir miyim?



Mesela GoogleNet'te



Belli 3x3 te

yakaladığı feature

5x5'te yakalanmıyor.

Belki 5x5'te yakaladığı feature

3x3'te yakalanmaz

Filter Concatenation ile üretilen tüm feature map'ler birleştirilmiştir.

Bu dimensionu 1'e düşürerek sonra üzerinde farklı filtre boyutları kullanarak yaptığınız işlemlerde oluşan feature map'leri birleştirme okuyuncı Inception Module denir.

GoogleNet, diğer AlexNet, ZFNET ve LNET'ten farklı olarak filtre boyutlarında esneklik sağlıyor.

GoogleNet'in bize sunmuş olduğu bir modüldür Inception Modülü.

Important Trick Dimensionu küçülterek full-connected layer'a girmek parametre

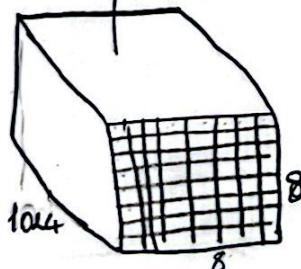
sayısı açısından daha avantajlı

$8 \times 8 \times 1024 \times 1000$

\uparrow Dense (1000)

$8 \times 8 \times 1024$

\uparrow Flatten



Avantajlı Olan

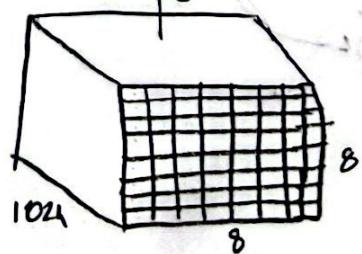
1024×1000

\uparrow Dense (1000)

1024

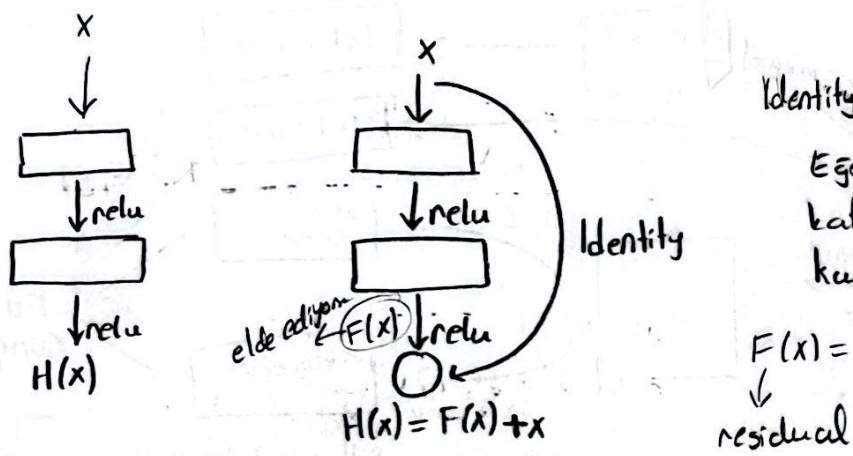
\uparrow Platten

1024
(pick average) dimension reduction



Res Net

Teoride daha katmanlı, derin bir mimari ile daha başarılı bir sonuç elde etmeyi umuyoruz ama практике train öğrenme oranı iyileşmesiz rağmen test hatalasının arttığını görüyoruz. B



$$\text{Identity fonksiyonu: } H(x) = x$$

Eğer bu identity fonksiyonun katmanları arasında bir kuyruk yapılıyorsa

$$F(x) = H(x) - x$$

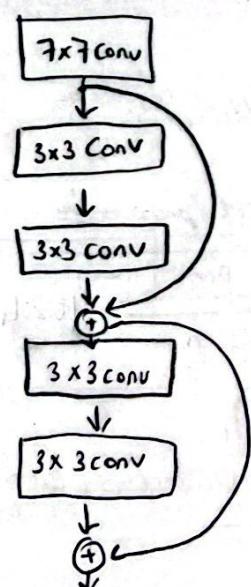
residual

! Identity ile orjinal uzaydaki nitilikleri yeniden hatırlatıyoruz

• ResNet 152 layers!

• Katman sayısı arttıkça her zaman basarını elde etmem o yüzden ResNet ara ara bir önceki katmanın girişini bir sonraki katmana birlleştiriyor

Resnet



Özet: Model derinleştirilince orjinal uzaydan gittikçe uzaklaşıyoruz ve daha az dahi sıkık geçen bir örüntüyü göz ardı etmeye başlıyoruz bu durumda ResNet Identity fonksiyonu ekliyor Residual fonksiyon ekliyor diyor

! Bu illaki orjinal veriyi eklemek demet değil çünkü orjinal veri daha maliyetli dolayısı ile birkaç katman önceki haliini entegre ediyor

- Batch Normalization, ResNet kullanıyor. Google → Dropout kullanıyor
- ResNet Stochastic Gradient Descent kullanıyor.
(SGD)

-Learning rate 0,1 den başlarsa 10'a bölerde orasıyor Bu dinamik değişim anlaşıma geliyor

Hafta-11, DEVAM

$$x \rightarrow \textcircled{6} \rightarrow y = f(x)$$

1

$$f(x) = \frac{1}{1 + e^{-(wx+b)}}$$

Input for training

$$\{x_i, y_i\}_{i=1}^N \rightarrow N \text{ pairs for } (x, y)$$

$$w \text{ ve } b \text{ yi bul} \\ \text{minimum } \mathcal{L}(w, b) = \sum_{i=1}^N (y_i - f(x_i))^2 \text{ için}$$

Loss fonksiyonunu minimize ederek x ve y 'leri bul

$\min \mathcal{L}(w, b)$ için gradient-descent yapalım

$$w_{t+1} = w_t - n \cdot \nabla w_t$$

$$b_{t+1} = b_t - n \cdot \nabla b_t$$

$$\text{where } \Delta w_t = \frac{\partial \mathcal{L}(w, b)}{\partial w}, \quad \Delta b_t = \frac{\partial \mathcal{L}(w, b)}{\partial b}$$

$$t \leftarrow 0$$

$$\text{max-iteration} \leftarrow 1000;$$

while $t < \text{max-iteration}$:

$$w_{t+1} = w_t - n \cdot \nabla w_t$$

$$b_{t+1} = b_t - n \cdot \nabla b_t$$

end

$$\nabla w = \frac{\partial}{\partial w} \left[\frac{1}{2} \cdot (f(x) - y)^2 \right]$$

$$= \frac{1}{2} \left[2 \cdot (f(x) - y) \cdot \frac{\partial}{\partial w} (f(x) - y) \right]$$

$$= (f(x) - y) \cdot \boxed{\frac{\partial}{\partial w} \left(\frac{1}{1 + e^{-(wx+b)}} \right)}$$

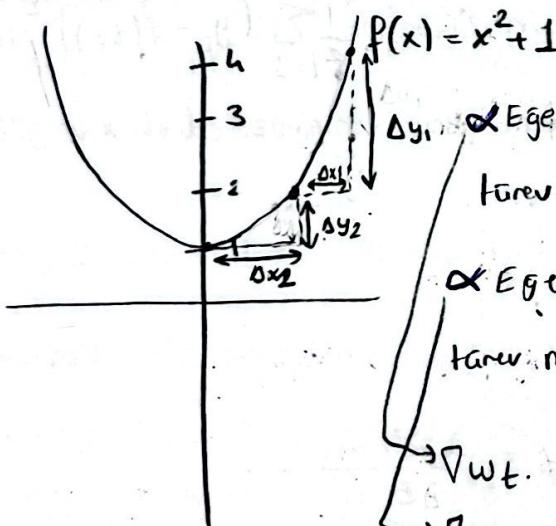
$$\Rightarrow \frac{\partial}{\partial w} \left(\frac{1}{1 + e^{-wx+b}} \right)$$

$$= \frac{1}{1 + e^{-(wx+b)}} \cdot \frac{e^{-(wx+b)}}{1 + e^{-(wx+b)}} \cdot x$$

$$= \boxed{f(x) \cdot (1 - f(x)) \cdot x}$$

$$\nabla w = (f(x) - y) \cdot f(x) \cdot (1 - f(x)) \cdot x$$

$$\nabla b = (f(x) - y) \cdot f(x) \cdot (1 - f(x))$$



• Eğer eğim çok dikse $\frac{\Delta y_1}{\Delta x_1}$ büyükse
tarım miktarı yani gradientler daha büyük olur.

• Eğer eğim dik değilse $\frac{\Delta y_2}{\Delta x_2}$ küçükse
tarım miktarı yani gradientler daha küçük olur.

∇w_t büyükse sent. geris

∇w_t küçükse yunuslu geris

MOMENTUM BASED GRADIENT DESCENT

- Eğer dik bir yokuş değil, yumuşak bir yokuş varsa gradient descent bu bölgelerde çok küçük ilerliyor. Ve bu çok fazla zaman alabiliyor.
- Bu sorunu gözmet için momentum özelliğini kullanalım
- Momentum bize belli bir aşamadan sonra bölge ile ılgılı bir filtre sahipsen hızlanın diyor
- Top örneği verebiliriz. Top çok minik bir yokuş ile karsılıktığında ırmelenebilir. Birakılıgını hızda devam etmez.

$$\text{Gradient - Descent} \quad w_{t+1} = w_t - n \cdot \nabla w_t$$

$$\begin{array}{l} \text{Momentum Gradient Descent} \\ \text{update}_t = \gamma \cdot \text{update}_{t-1} + n \cdot \nabla w_t \\ w_{t+1} = w_t - \text{update}_t \end{array} \quad \gamma: \text{ayartma parametresi}$$

$$\text{update}_0 = 0$$

$$\text{update}_1 = \gamma \cdot \text{update}_0 + n \cdot \nabla w_1 = n \cdot \nabla w_1$$

$$\text{update}_2 = \gamma \cdot \text{update}_1 + n \cdot \nabla w_2 = \gamma \cdot n \cdot \nabla w_1 + n \cdot \nabla w_2$$

NESTEROV ACCELERATED GRADIENT DESCENT

Hızlandırmış Versiyon

- Momentum based GD, vanilla GD'e göre hızlı ama çok fazla zikzag yapar ve optimum noktası kaçınıyor. Bu salınımın Nesterov GD ile ortadan kaldırılabilir.

NAG

$$w_{\text{look-ahead}} = w_t - \gamma \cdot \text{update}_{t-1}$$

$$\text{update}_t = \gamma \cdot \text{update}_{t-1} + n \cdot \nabla w_{\text{look-ahead}}$$

$$w_{t+1} = w_t - \text{update}_t$$

Çok fazla geçmiş bilgiyi kullanıldığında çok hızlanıyoruz ve çok zikzag yapıyor. $w_{\text{look-ahead}}$ ile geçmiş bilgisi yemisatılıyor.

* Gradient - Descent; bütün eğitim verisini kullanarak her iterasyonda parametreleri günceller.

* Stochastic GD: her iterasyonda sadece bir eğitim örneği kullanarak parametreler günceller.

Stochastic GD

- > Çok hızlı iterasyonlar, özellikle büyük veri setlerinde
- > Daha az bellek kullanımı
- > Daha genitelli çünkü sadece bir örnek üzerinden gradyan hesaplar

GD

- > Daha stabil ve tutarlı
- > Daha doğru bir yolculuk因为 tüm veri seti kullanır.
- > Bellek kullanımı fazla ve maliyetli因为 her iterasyonda tüm veri seti kullanılır.

Mini-Batch GD

α Ne çok yavaş update olsun ne de çok hızlı update olsun = Mini Batch Gradient Descent

Bu iki yöntemin arasında bir yaklaşım da mini-batch GD dir. (Toplu İşleme)

Bu yöntem, her iterasyonda tüm veri seti yerine küçük bir alt kümeye (mini-batch) kullanarak gradient-descent hesaplar. Bu yöntem hem hız hem稳定性 açısından denge sunar.

1 epoch → tüm verinin bir kez sisteme geçmesi

1 step → parametrelerin bir kez update olmasi

$N \rightarrow$ Veri sayımız

$B \rightarrow$ Mini batch boyutu

Algorithms

of steps in 1 epoch

Vanilla GD

1

Stochastic GD

N

Mini Batch GD

$\frac{N}{B}$

Learning Rate

- Eğer gentle stoplar (nozuk geçiş) var ise yüksek learning rate ile büyük adımlar atarak optimuma ulaşabilirsiniz.
- Ama aynı momentum GD'de olduğu gibi yüksek adımlar bir süre sizin zikzak yapmanıza neden olabilir.

Adagrad

$$v_t = v_{t-1} + (\nabla w_t)^2$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t + \epsilon}} * \nabla w_t$$

! Adagrad'ın eğrime oranı çok agresif bir şekilde azalma gerçekleştirilebiliyor

∇w_t degeri çok büyükse learning rate çok büyük bir düzleme oladır.

RMSProp

$$v_t = \beta * v_{t-1} + (1-\beta) * (\nabla w_t)^2$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{v_t + \epsilon}} * \nabla w_t$$

Adam

kümülatif history dedigimiz kümelerden gelen toplam gradienti kullanarak şekilde baska bir yaklaşım kullanabiliriz. Adam

$$m_t = \beta_1 * m_{t-1} + (1-\beta_1) * \nabla w_t$$

$$v_t = \beta_2 * v_{t-1} + (1-\beta_2) * (\nabla w_t)^2$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

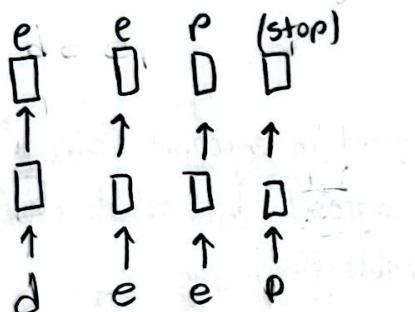
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} * \hat{m}_t$$

Lecture 14 RNN Recurrent Neural Network

Feed Forward ve CNN'de input boyutu sabittir öneçeler verili

Sequence Learning Problem



deep belimeşini önerirken

1. kullanıcı d yazın ve e önerir

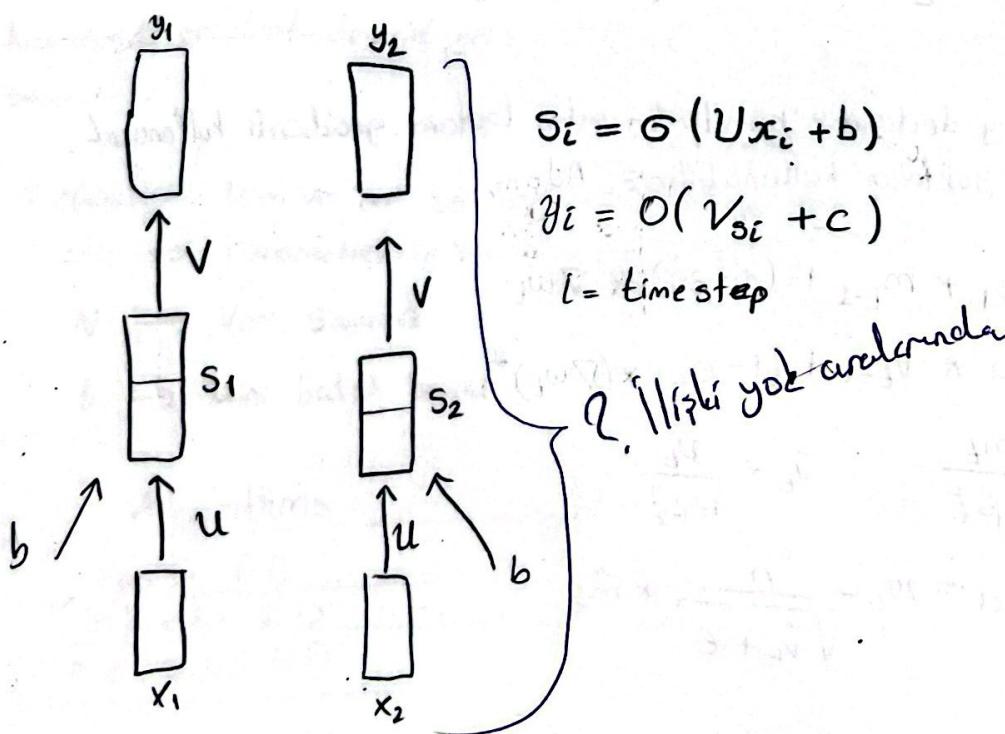
2. de den dolaylı d ve e ye bağlı e öner

3. dee, d, e, e ye bağlı p öner

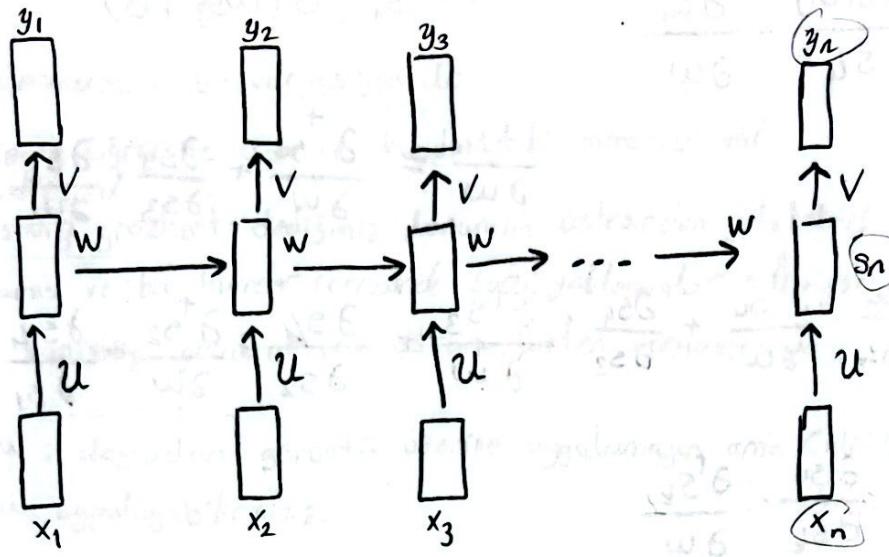
4. deep, hepsine bağlı stop öner

Bu tür dizi problemleri RNN "Yenilenen Sinir Ağları" bir konusu

- 1) inputlar birbiriyle ilişkili
- 2) Sistem input sayısına bağlı değil
- 3) Her bir $t, t+1, t+2$ ardılı inputları için aynı işi yapmaz gerler



Yinelemeli bağlantı ile birbirleriyle ilişkilendirilmek



$$s_i = G(u x_i + w s_{i-1} + b),$$

$$s_3 = G(u x_3 + w s_2 + b)$$

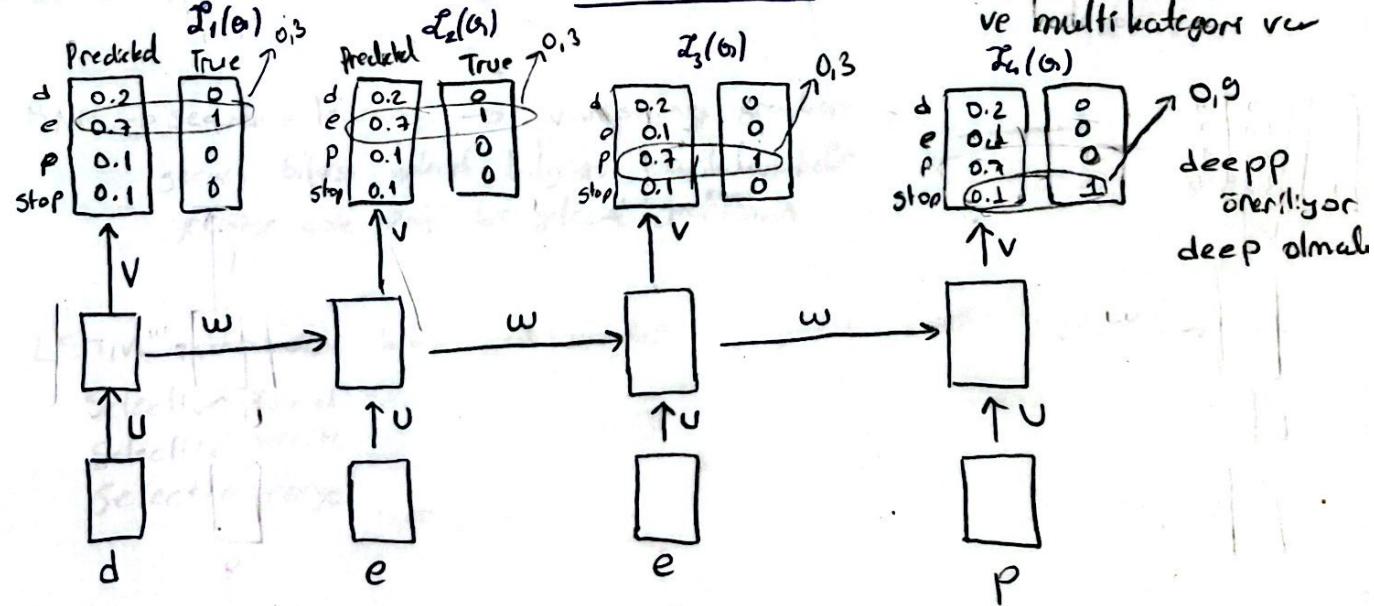
$$y_3 = O(s_3, V + b)$$

$$y_i = f(x_i, s_{i-1}, w, u, V, b, c)$$

s_i : Network'ün i inci zamandaki durum bilgisi

Bu görev için çıkış fonksiyonu (softmax) olmalıdır. Olasılıksal dağılımlar

Bu görev için loss fonksiyonu (cross entropy) olmalıdır. Softmax kullanmadırmıza rağmen ve multi kategori var



Feed-Forward'da tek bir loss hesap ediyorduk. Burada toplam loss tüm output sınıflarının beklenen değerden farklılığı toplamını bulmak

$$\text{Toplam Loss: } \mathcal{L}(O) = \sum_{t=1}^T \mathcal{L}_t(O_t)$$

$$\mathcal{L}_t(O_t) = -\log(y_{tc})$$

Cross entropy
her bir zaman aralığının tahmin edilmesi beklenen karakterin olasılığı

$$\frac{\partial \mathcal{L}_4(\theta)}{\partial w} = \frac{\partial \mathcal{L}_4(\theta)}{\partial s_4} \cdot \frac{\partial s_4}{\partial w}$$

Hidden layerdaki aktivasyon

$$s_4 = \sigma(w_{s_3} + b) \rightarrow \text{X4.U bilgisi}$$

$$\frac{\partial s_4}{\partial w} = \frac{\partial^+ s_4}{\partial w} + \frac{\partial s_4}{\partial s_3} \cdot \frac{\partial s_3}{\partial w}$$

Kısaca

$$\frac{\partial s_4}{\partial w} = \frac{\partial s_4}{\partial s_4} \cdot \frac{\partial^+ s_4}{\partial w} + \frac{\partial s_4}{\partial s_3} \cdot \frac{\partial^+ s_3}{\partial w} + \frac{\partial s_4}{\partial s_2} \cdot \frac{\partial^+ s_2}{\partial w} + \frac{\partial s_4}{\partial s_1} \cdot \frac{\partial^+ s_1}{\partial w}$$

$$= \sum_{k=1}^4 \frac{\partial s_4}{\partial s_k} \cdot \frac{\partial^+ s_k}{\partial w}$$

t anindaki bir los için genel formülizasyon

$$\frac{\partial \mathcal{L}_4(\theta)}{\partial w} = \frac{\partial \mathcal{L}_t(\theta)}{\partial s_t} \cdot \sum_{k=1}^t \frac{\partial s_k}{\partial s_k} \cdot \frac{\partial^+ s_k}{\partial w}$$

Yukarıda $\frac{\partial s_k}{\partial s_k} = 1$ olduğu için bu ifadeyi $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ile yazabiliriz.

Özetle $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesi $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesine eşdeğerdir.

Yani $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesi $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesine eşdeğerdir.

Yani $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesi $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesine eşdeğerdir.

Yani $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesi $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesine eşdeğerdir.

Yani $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesi $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesine eşdeğerdir.

Yani $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesi $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesine eşdeğerdir.

Yani $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesi $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesine eşdeğerdir.

Yani $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesi $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesine eşdeğerdir.

Yani $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesi $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesine eşdeğerdir.

Yani $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesi $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesine eşdeğerdir.

Yani $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesi $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesine eşdeğerdir.

Yani $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesi $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesine eşdeğerdir.

Yani $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesi $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesine eşdeğerdir.

Yani $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesi $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesine eşdeğerdir.

Yani $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesi $\frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}$ ifadesine eşdeğerdir.

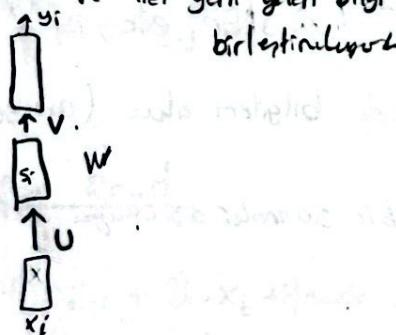
LSTM Mimarisi (Long Short Term Memory Cells)

Uzun Kısa Dönem Nitaza Hücreleri

- ✓ RNN mimarisinin bir varyasyonudur
- ✓ Sequence öğrenme üzerine kurulmuş bir mimarisi var
Exploding and Vanishing gradient
- ✓ Vanishing gradient dediğimiz durumun üstesinden gelebilmek için bozı hücreler tanımlanmış ve bu hücreler üzerinde bozı yaklaştırıcılar kullanıyor. Ancak, yine RNN faktat vanishing gradient için çözüm öneren varyasyonu *gradient exploding*

LSTM i doğrudan görüntü üzerine uygulamayın ama CNN'den sonraki bir katmandada LSTM uygulayabilirsiniz.

RNN'de bir önceki zanneden gelen bilgi (s_i) durum bilgisinde tutuluyor da ve her yeni gelen bilgi ile birlleştiriliyor.



RNN \rightarrow sequence learning \rightarrow vanishing gradient
geçmiş bilgsi gelecek bilgisi bastırabilmek
geçmişe çok bağlı bir gelecek üretirsin

LSTM içerisindeki bozı yaklaştırıcılar

Selective Read
Selective Write
Selective Forget

Bu algoritma nedir

Bu algoritmanın bundan farkı nedir

CNN örnek

Feed forward örnek

Slayt teont konu hakkında

E.

LSTM White Board Analogy

Beyaz tahta kapasitor hali

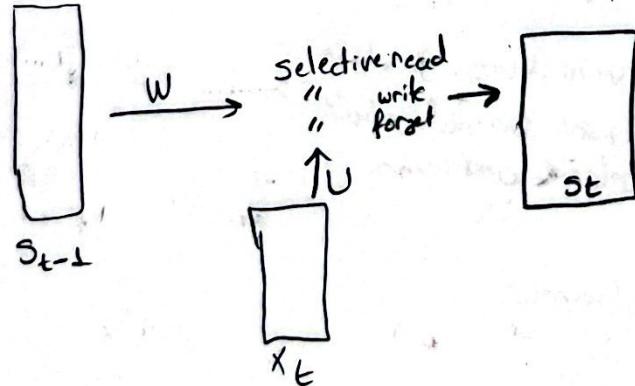
Bundan sonra bunu önden sonra bunu yapacağım diye her şeyi gizleriz

Ara adımlar yaz her kere adımı yazma! Sekerle yaz

forget: Gerekli olmayan bilgileri unut (a, the, etc, of) pozitiflik ve negatiflik ilişkisi

okuma: Negatifliği pozitifliği etki edecek bilgileri oku (awesome, amazing)

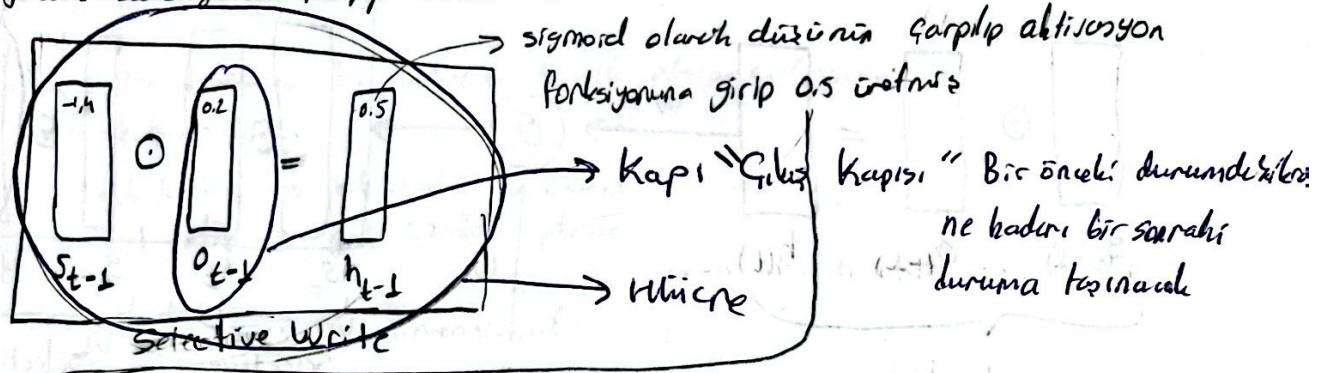
write: awesome, amazing gibi bilgiler bir sonraki aşamaya aktarılır



$$s_t \rightarrow f(s_{t-1} * w + U \cdot x_t)$$

Selective Write

$s_t = \sigma(w_s s_{t-1} + u_s x_t)$ doğrudan s_t 'ye bu şekilde tüm geçmiş bilgilerini aktaranak yerine "awesome", "beautiful" gibi etkileyen değerleri yüksek katsay ile çarpıp aktardan.



o_{t-1} i bız hesap edip bir önceki durumdan gelen bozı bilgilerin yazılmasını sağlayıp bozı bilgikrin değerini azaltırırs

$$o_{t-1} = \sigma(w_o h_{t-1} + u_o x_{t-1} + b_o)$$

$$\rightarrow h_{t-1} = o_{t-1} \odot \sigma(s_{t-1})$$

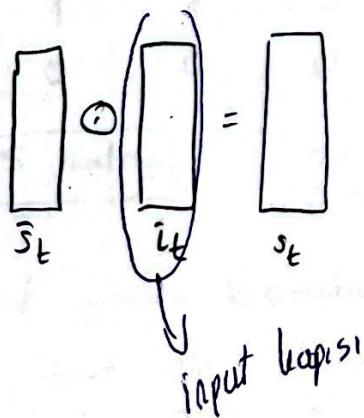
Selective Read

$s_t = \sigma(w_s h_{t-1} + u_s x_t + b)$ sadece write varken böyle idi. s_t 'yi böyle hesap ederlik Bu aşamada yeni bir geçici \hat{s}_t ortaya çıkarırı

$$\hat{s}_t = \sigma(w_i h_{t-1} + u_i x_t + b) \quad \text{geçici durum bilgisi elde edioz.}$$

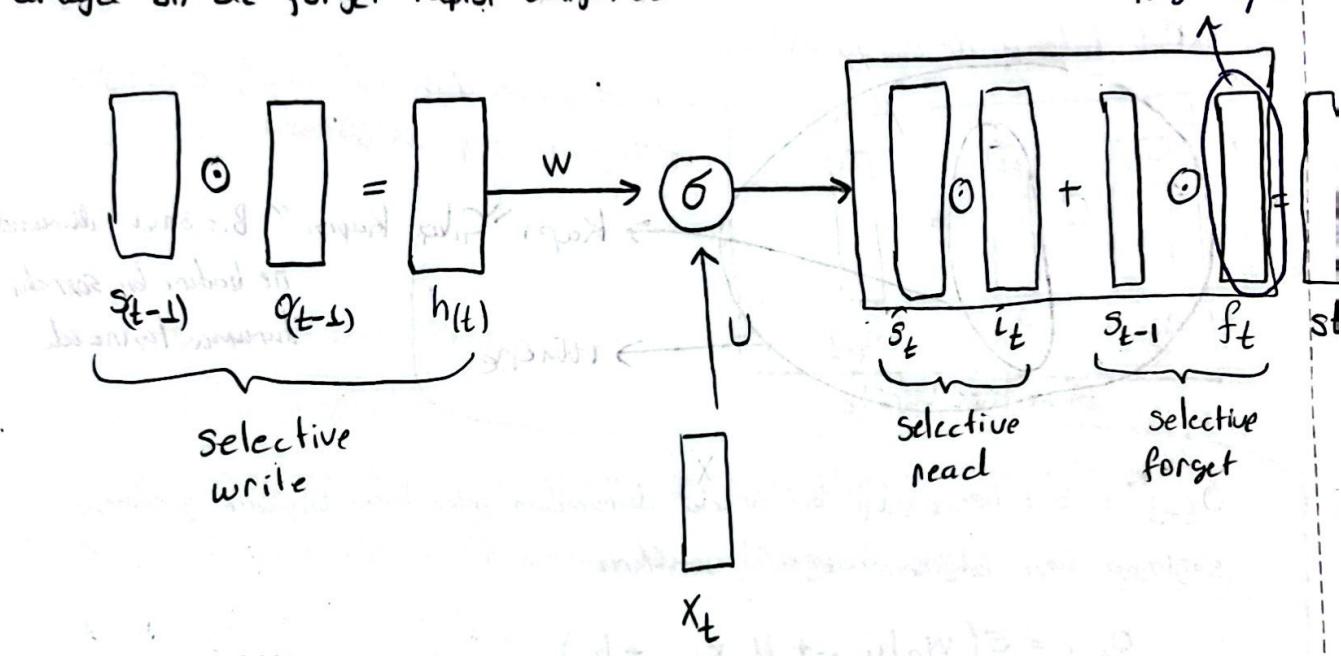
$$\sigma(\text{geçici durum bilgis}) \times \text{ağırlık değerleri} = (\text{Sigmoid durum bilgisi})_{st}$$

$$i_t = \sigma(w_i h_{t-1} + u_i x_t + b_i)$$



Selective Forget

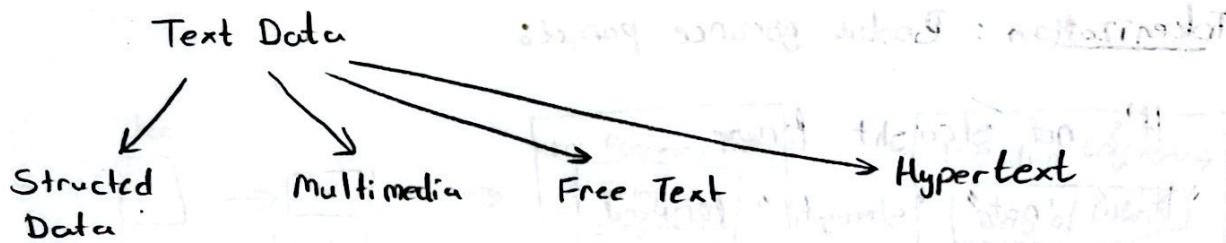
Burada s_t 'yi doğrudan selective read'den sonra hesaplamak yerine
anaya bir de forget kapısı ekliyoruz



geçmişteki bazı bilgileri unut demek istiyoruz

$$f_t = \sigma(w_f \cdot h_{t-1} + u_f \cdot x_t + b_f)$$

$$s_t = f_t \odot s_{t-1} + \hat{s}_t \cdot i_t$$



Amaç: Bir dokümanın bir veritabanı tensile etmeye çalışması

term = kelime = ifade

tek kelime → tek öznitelik word

tek kelimeler → çok öznitelik phrase

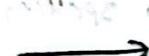
"kelimeler" → N term → N dimensional

Bag of Tokens

Doküman

w w w
w w w
w w w
w w w

sentence



Token Sets

notion - 5 + nation 5 koz (özellikler)

civil - 1

war - 2

(unique)

Dokümandaki kelimelerin listeleri

Bag of Words Representation

Binary Temsil

	text	info	identit	mining	is	
Doc1	1	1	1	0	1	Bilgisayar anlayıcı sayısallaştırılmış
Doc2	1	1	0	0	1	dokümanın içeriğinden okunurken, okunma anlamına gelir.
Doc3	0	0	0	0	1	sayısal değerlerin sıralaması

[3x5 matris]

[dimension: 5]

Bag of words Representation Frekans

	text	info	identit	mining	is
Doc1	1	2	2	0	1
Doc2	1	1	0	0	3
Doc3	0	0	0	0	2

Sequential verinin, anlamsız! ilişkisi tutmuyor

Gramer olacak böyle çok bir şey vermey

Tokenization: Bozuk görünce parçalá

It's not straight forward

"It's" "not" "straight" forward

Stop word: the, a, an, we, do, to

Tail words: Dokimanda nadir görülen birip kelimeler sesquipedalianism

Normalization

U.S.A. → USA

All in lower cases SPELLING → spelling

Delete hyphens pre-process → prep

Fix typos

Dictionary-based

car → "vehicle", "automobile"

Stemming kökünü bul - köke git

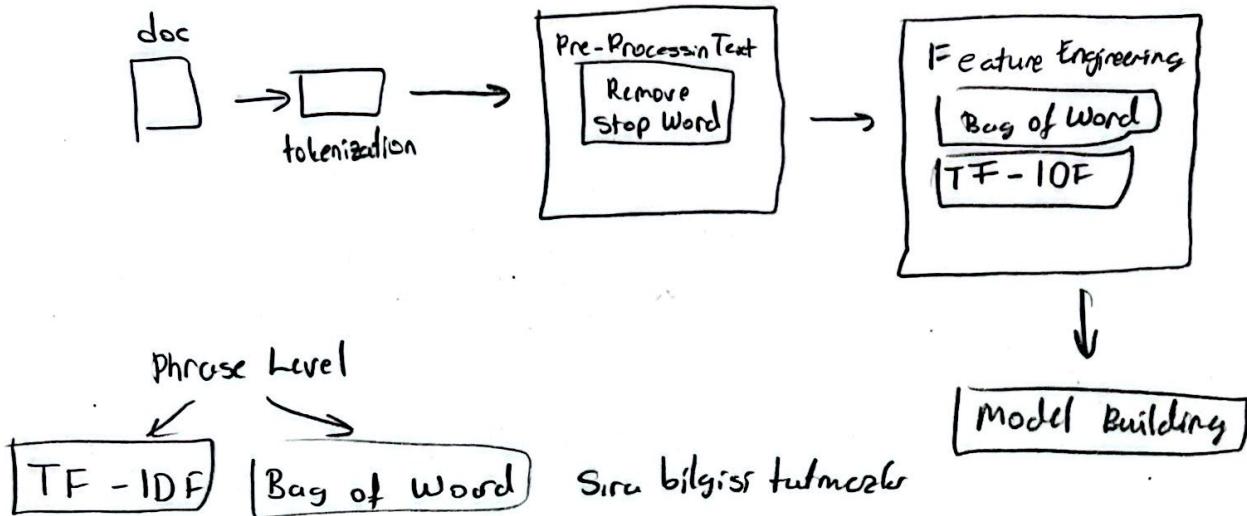
TF sıklıkla görülen kelimeler anlamca ayırmalarla, terim frekansı

IDF "is" kelimesinin TF i fazla dokümanlar arasında çok görülen anlamda ayırmak

$$IDF = 1 + \log \left(\frac{\text{toplam dokümen sayısı}}{\text{kırmızı dokümden}} \right)$$

kırmızı dokümden
gözlenenler

$$w(t,d) = TF \times IDF$$



N-Gram

$N=1$: This is a sentence

this
is
a
sentence

$N=2$: This is a sentence

this is
is a
a sentence

$N=3$: This is a sentence

this is a
is a sentence

Word Properties

Homonymy bank, river bank same form, dif meaning

Polysemy same form, related meaning blood bank, financial institution

Synonym dif form, same meaning big, large

Hyponym Meal Pork fish