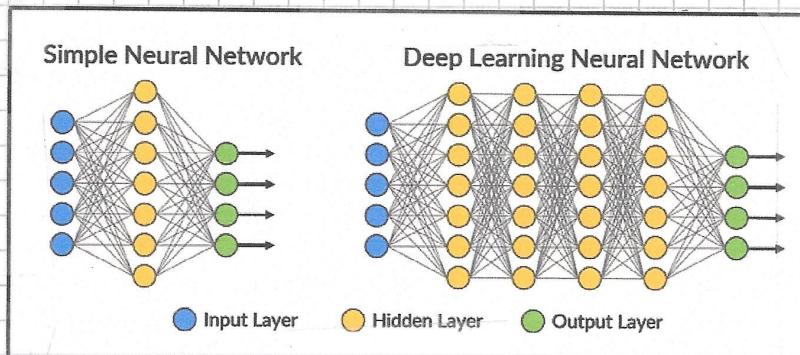
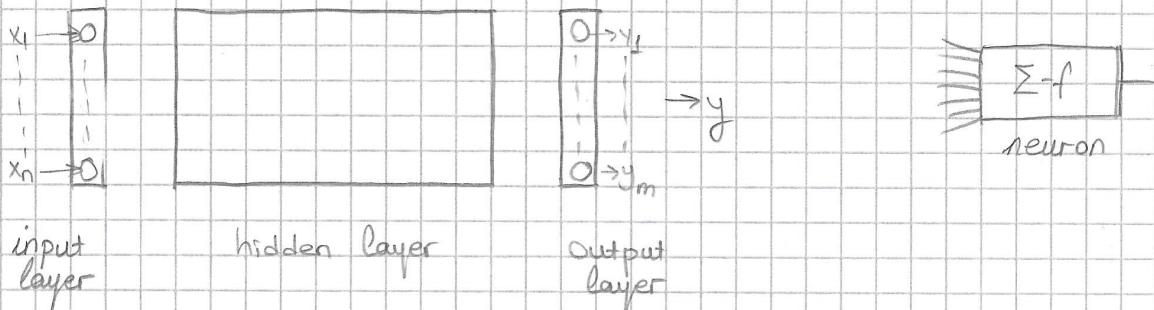


ANN = Artificial Neural Network



→ Teknik anlamda bir değişiklik yok. Farkı yöntemleri kullanıyorlar

Chart of Neural Networks:

Perception: Tek katmanlı algılayıcı



AND - OR tek katmanda
çözmüştür fakat XOR çözememiştir.

→ DFF'in (Deep Feed Forward) FF'den (Feed Forward) tek farklı daha çok process elemanına sahip olmasıydı.

Recurrent Neural Network (RNN): Bir önceki adıma dayanarak sıradakini işler.
Öz: Bir metin işlerken önceki cümleşin anmasını ile sıradakını bağlaştırırız.

*- Dil işleme de önemli - *

Generative Adversarial Network (GAN): Görüntüyü işleyip başka yerde kullanır.
Öz: Birisinin görüntüsünü isledi başka yerde bir yüz oluşturabilir.

→ Auto Encoders (AE) ve GAN'lar "deep fake" de öğrenip yüzü giydirmeye yapar.

A mostly complete chart of Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org

○ Backfed Input Cell

○ Input Cell

△ Noisy Input Cell

○ Hidden Cell

○ Probabilistic Hidden Cell

△ Spiking Hidden Cell

○ Output Cell

○ Match Input Output Cell

○ Recurrent Cell

○ Memory Cell

△ Different Memory Cell

● Kernel

○ Convolution or Pool

Perceptron (P)

Feed Forward (FF)

Radial Basis Network (RBF)

Recurrent Neural Network (RNN)

Long / Short Term Memory (LSTM)

Gated Recurrent Unit (GRU)

Auto Encoder (AE)

Variational AE (VAE)

Denoising AE (DAE)

Sparse AE (SAE)

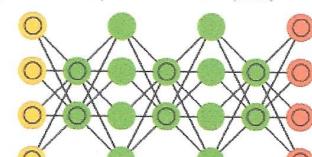
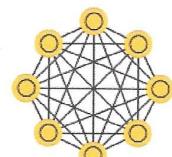
Markov Chain (MC)

Hopfield Network (HN)

Boltzmann Machine (BM)

Restricted BM (RBM)

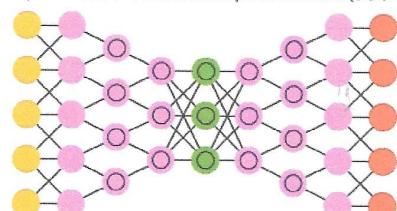
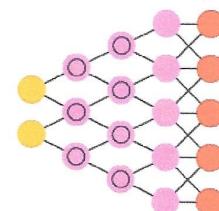
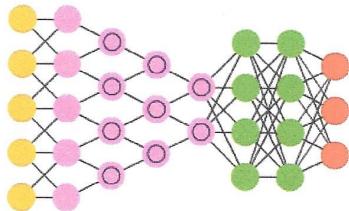
Deep Belief Network (DBN)



Deep Convolutional Network (DCN)

Deconvolutional Network (DN)

Deep Convolutional Inverse Graphics Network (DCIGN)

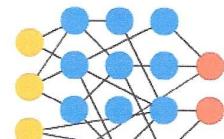
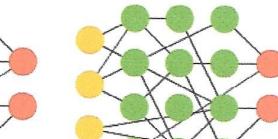
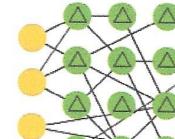
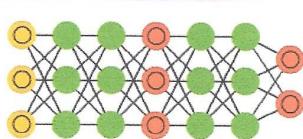


Generative Adversarial Network (GAN)

Liquid State Machine (LSM)

Extreme Learning Machine (ELM)

Echo State Network (ESN)

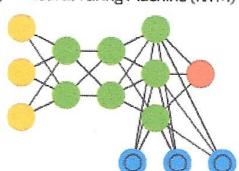
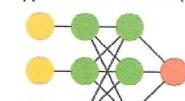
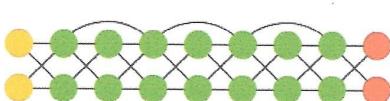


Deep Residual Network (DRN)

Kohonen Network (KN)

Support Vector Machine (SVM)

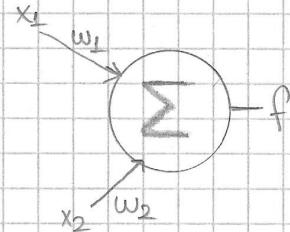
Neural Turing Machine (NTM)



Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = \max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^z)$	Multi-layer Neural Networks	

Copyright © Sebastian Raschka 2016

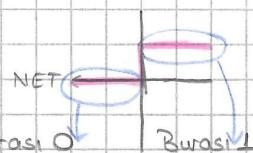
[http://sebastianraschka.com]



$$NET = x_1 w_1 + x_2 w_2$$

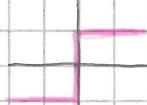
$$f(NET) = \text{output}$$

Unit Step:



Activation function
buna yada 1 verecek
ya da 0.

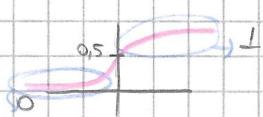
Sign:



Act. funct.
1 yada -1 verir

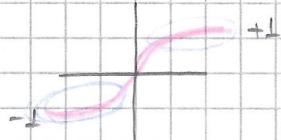
* Aradakileri çok kullanıyor.*

* Logistic (Sigmoid):
* (0,5 hala iyi bir sonuc değil)



→ Difer act-func. gradient ile
aktive olmadığı için genelde bunu
kullanıyor.

Hyperbolic:
(tangenthyperbolic tanh)



→ Çok katmanlılar için

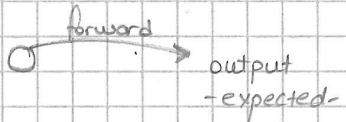
ReLU:



→ DLN için en çok kullanılmıştır

4

"Vanishing Gradient Problem" vs "Gradient Problem"

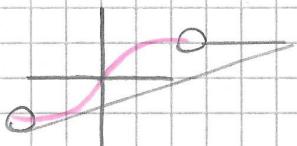


$$E = |Exp - Calc|$$

∇ = Loss function
 \downarrow
 Error

* Forward'da hatayı hesapla,
backward'da w'leri değiştir

→ Logistic & Hyperbolic'de elindeki network birçok process elemanından olustugundan backward'da (geri dönenken) w'ler çok değişmez. w'ler değişimse ögrenme sağlanamaz.



Bu uclara gittikce türer azalacağından
roller değişmez hale gelir, böyle olunca ögrenemezsin.

Vanishing Gradient Problem:

ReLU



→ ReLU works very well in the hidden layer

* Biper networkte çok iyi calır. *

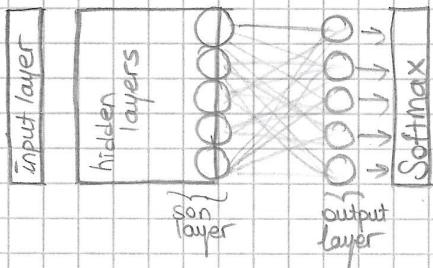
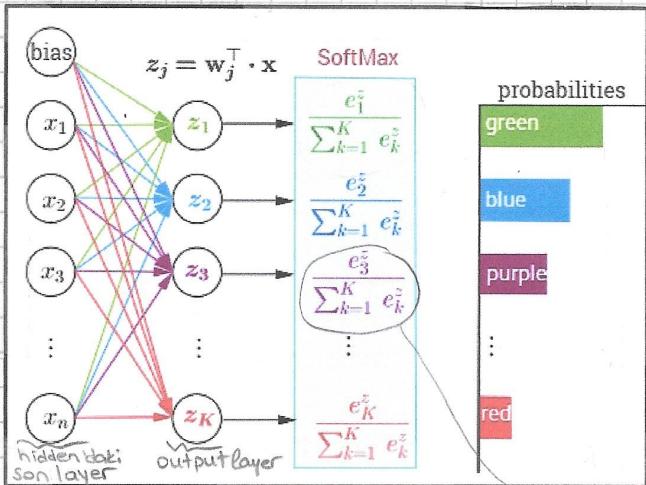
Yapay Sinir Ağları Kullanımı DL'ler

Output layer \rightarrow Relu olamaz, onun yerine softmax (en büyük olan koçanın) kullanılır.

* ReLU output layer'ea getirir. Softmax ile konu oluşturur.

→ Vanishing' de sigmoid ve hyperbolic tullanımının nedeni \rightarrow network büyütükle bunlarla kontrol edemeyebilirsin. O yüzden Relu'yu kullanırsın.

SOFTMAX:



Softmax: Choose the big probability for the class
(En yüksek olasılığa sahip olanı seçiyor.)

Let's say: $e^{f(\text{NET})}$

$$\begin{aligned} 0 &\rightarrow -1 \rightarrow e^{-1} = 0,36 \\ 0 &\rightarrow 0 \rightarrow e^0 = 1 \\ 0 &\rightarrow 3 \rightarrow e^3 = 20 \\ 0 &\rightarrow 5 \rightarrow e^5 = 148 \end{aligned}$$

Probability

$$\frac{0,36}{(0,36+1+20+148)}$$

$\rightarrow \frac{e^{\text{output}}}{\sum_{k=1}^K e^{\text{output}}} \rightarrow e^{\text{output}}$ burada hata depil.
1'den K'ya kadar hepsini al,
döz, olasılığı bul.

Bazen sadece sınıf söylemek yeterir olmuyor. O sınıfı ne kadar ait olduğunu söylemen gerekiyor.

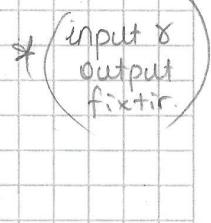
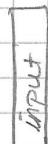
Örneğin, bu draba SUV, kaç oranda SUV?

* Bir ANN için neye ihtiyacım var?

$X = \{x, r\}$ → gereksiz verileri atılmış,
temizlenmiş data verildi.

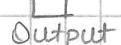
1) Topoloji oluşturman lazım.

- Input layer sabit, kaç process elementi olacağını belirle. →



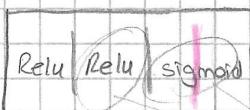
2) Outputu belirle

- Eğer regresyon problemi ise bir tane outputun olacak
classification ise $\xrightarrow{\text{0}} \xrightarrow{\text{1}} \text{kac class olacak belirle?}$



3) Hidden layer'i belirle

- Hidden layer birden çok ve farklı cıgıt layerlarından olabilir.



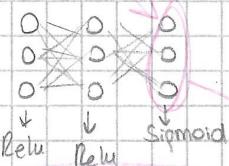
olabilir fakat



burada hepsi
tek cıgıt olmalı

hidden layer

farklı farklı
kullanabilirsin.

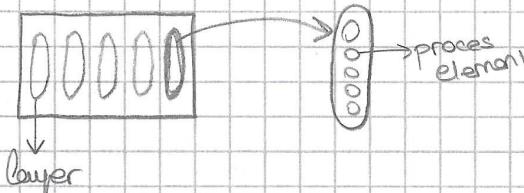


* Sırtında hep aynı şey
kullanmalısın.

- Hidden layerda kaç tane layer olacak belirle.

- Her layerda kaç tane process elementi olacak belirle.

* NOT



* Ne kadar çok layer kullanırsam,
o kadar iyi sonuç alırım diye bir
sey yok - Yık söyle bir dünya "

λ = learning rate | w'lerin ne kadar değiştiri.

Küçük olmalı → çok da küçük olmasın ama yavaş oluyor.

Epoch = weightlerin değiştip başa dönmesi - Python'da epoch sayısını vermem gerekiyor
Bu data üzerinde kaç tur dönüp w'leri değiştiresin ↪

→ w çok küçükse, bu o x'in per bir önemi olmadığını anlamına geliyor.