Deep Learning for Applications – I

KISTI-고려대 의과대학 데이터사이언스 교육

Dr. Kyong-Ha Lee (kyongha@kisti.re.kr)

KiSTi 한국과학기술정보연구원
Korea Institute of Science and Technology Information
www.kisti.re.kr

강사 소개

- 이경하 (kyongha@kisti.re.kr)
  - 한국과학기술정보연구원
    초거대AI연구단장 /책임연구원
  - UST 응용AI전공 전임 부교수
  - 과기부 미래국방기술전략분과 위원
  - 한국정보과학회 학회지편집위원 및
    이사 역임
  - 한국정보과학회 데이터소사이어티 상
    임이사 등

안 내

- 수업 진행 방식
  - 파워포인트 강의: 개념 및 이론
  - 실습: Google CoLab을 활용한 실습
    - Google 계정 보유 필요
    - Internet Browser(Google Chrome 또는 MS Edge)

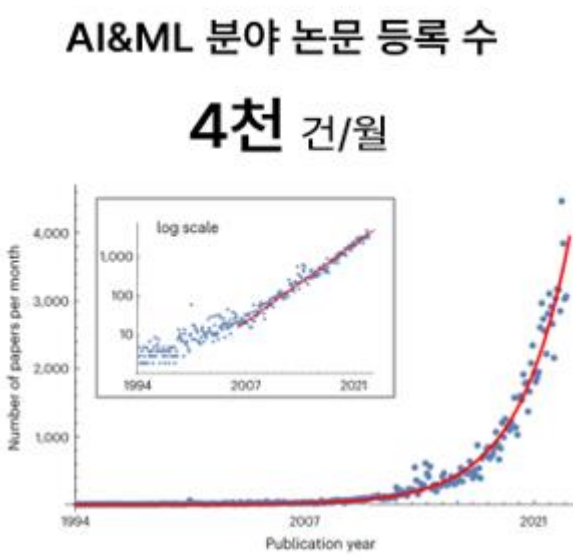
- 강의 자료
  - GitHub을 이용한 자료 공유
  - https://github.com/bart7449/24KoreaMedical

Contents

Class1 : 인공지능 특히 뉴럴 네트워크에 대한 기초 개념과 구성요소를 강의와 실습을 통해 학습합니다.

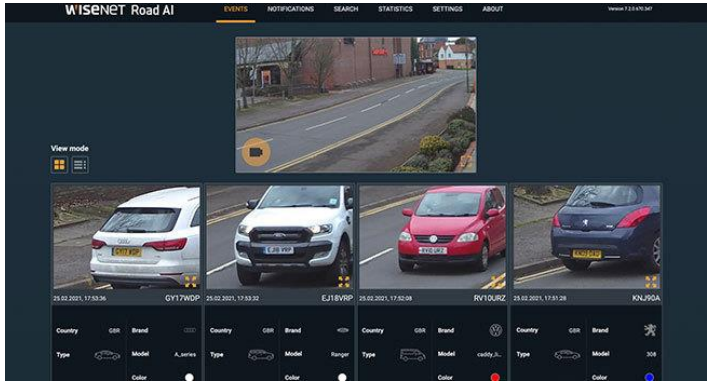
A Few Quates about Machine Learning

- "A breakthrough in machine learning would be worth ten Microsofts" Bill Gates, Chairman, Microsoft)
- "Machine learning is the next Internet"
  (Tony Tether, Director, DARPA)
- Machine learning is the hot new thing"
  (John Hennessy, President, Stanford University)
- "Web rankings today are mostly a matter of machine learning"
  (Prabhakar Raghavan, Dir. Research, Yahoo)

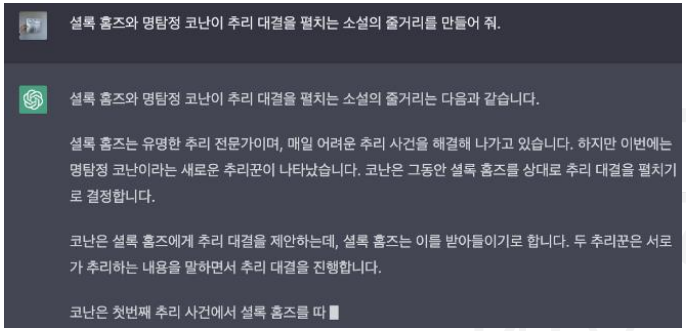


Application example

Vehicle Identification



Large Language Models



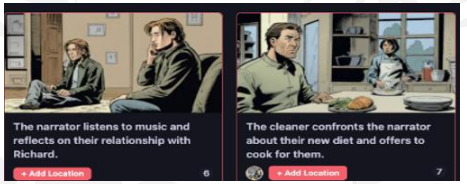
Stable Diffusion

Autonomous Driving

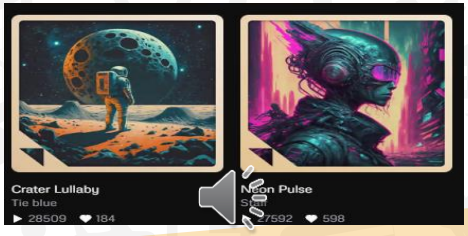
SORA (Text2Video)

Large Language Models

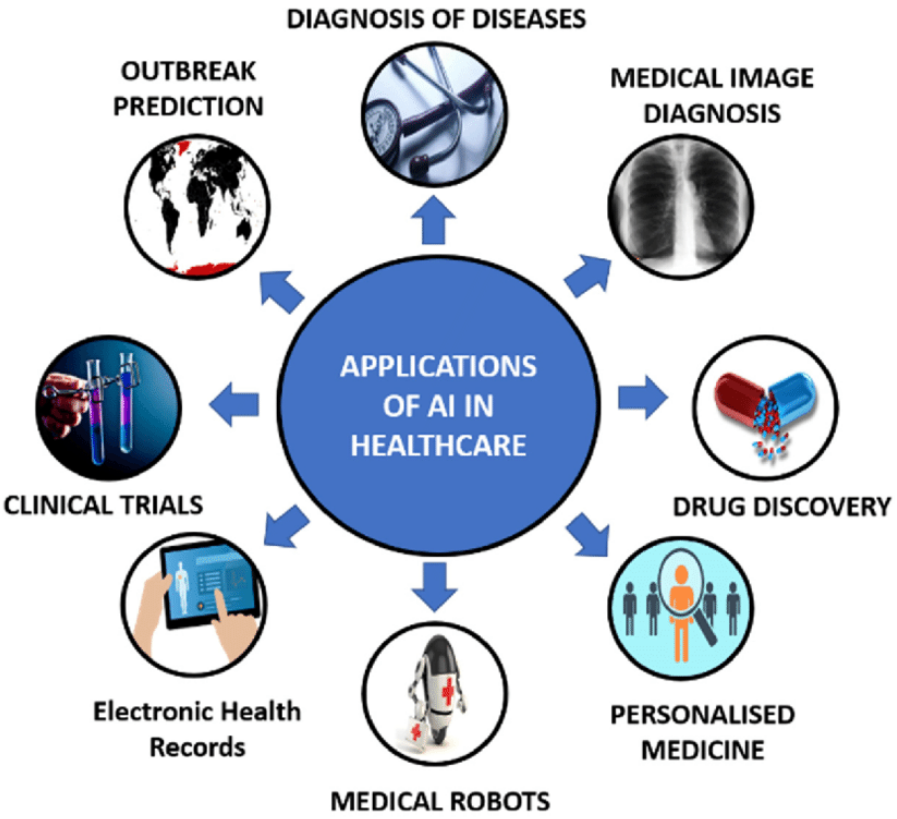
Lore Machine (Story2Comic)



Udio (Text2Music)



Healthcare Applications



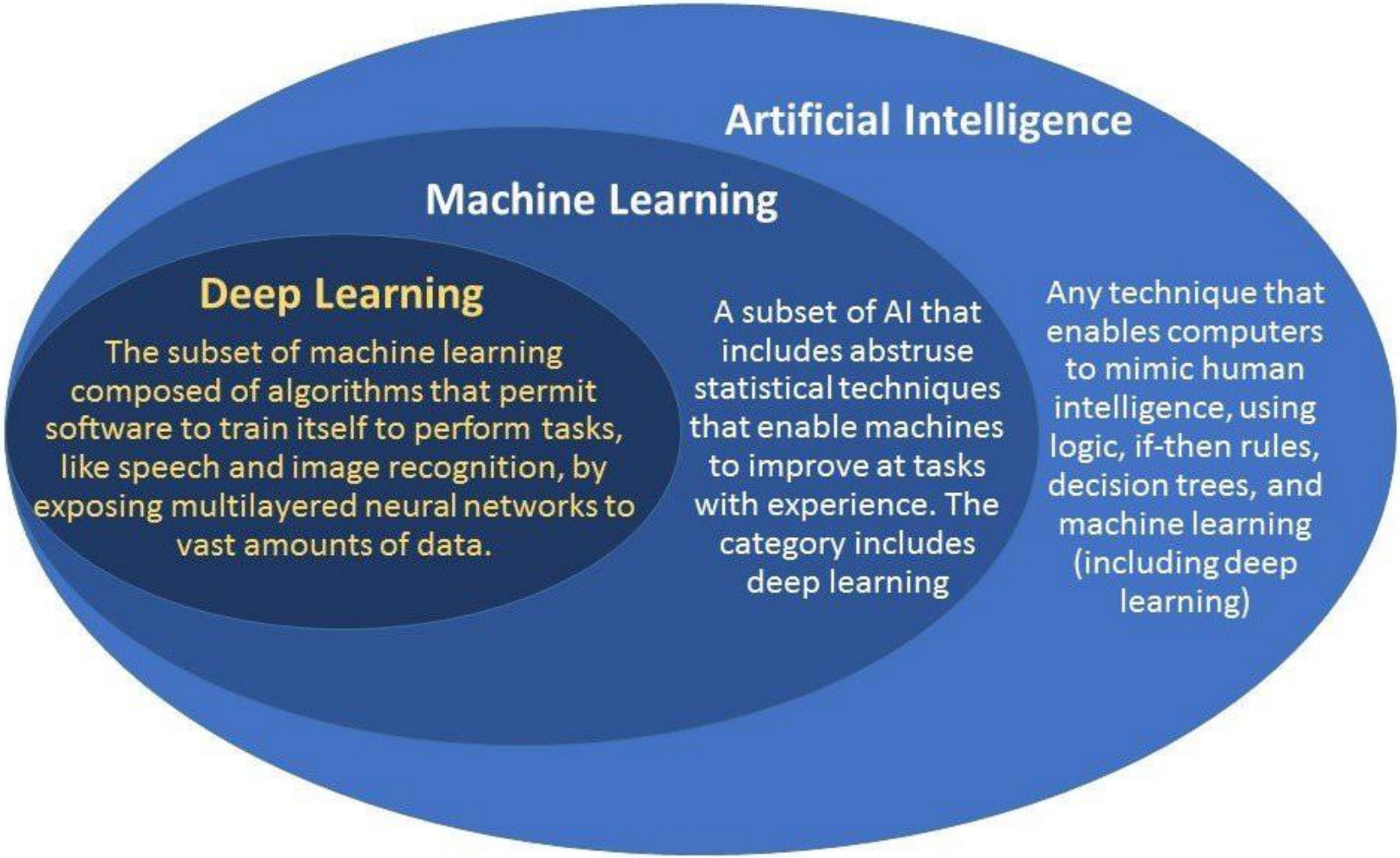
10 AI Applications That Could Change Health Care

APPLICATION	POTENTIAL ANNUAL VALUE BY 2026	KEY DRIVERS FOR ADOPTION
Robot-assisted surgery	$40B	Technological advances in robotic solutions for more types of surgery
Virtual nursing assistants	20	Increasing pressure caused by medical labor shortage
Administrative workflow	18	Easier integration with existing technology infrastructure
Fraud detection	17	Need to address increasingly complex service and payment fraud attempts
Dosage error reduction	16	Prevalence of medical errors, which leads to tangible penalties
Connected machines	14	Proliferation of connected machines/devices
Clinical trial participation	13	Patent cliff; plethora of data; outcomes-driven approach
Preliminary diagnosis	5	Interoperability/data architecture to enhance accuracy
Automated image diagnosis	3	Storage capacity; greater trust in AI technology
Cybersecurity	2	Increase in breaches; pressure to protect health data

SOURCE ACCENTURE

Harvard Business Review

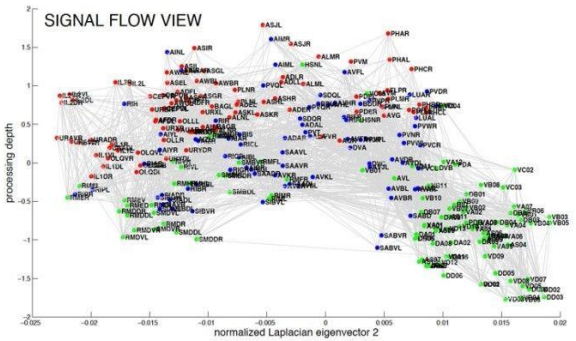
Deep Learning /Machine Learning /AI?

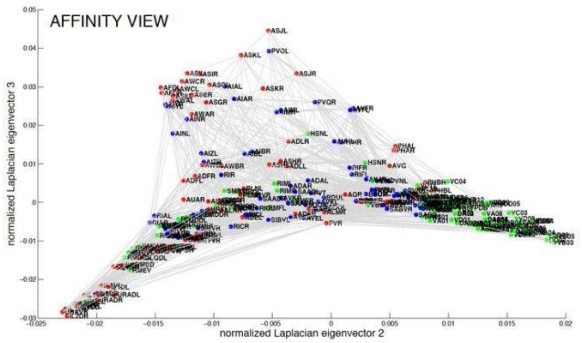


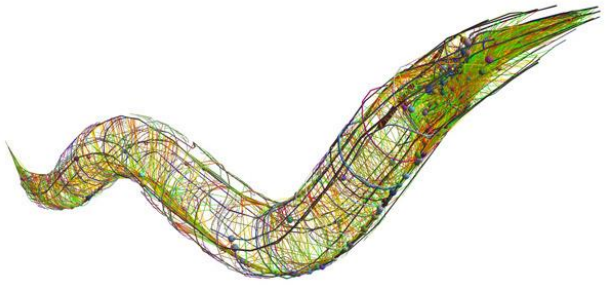
Interdisciplinary Linkage

- 예쁜 꼬마 선충(Caenorhabditis elegans)
  - 대표적인 유전학 모델
  - 노벨상 수상에 기여(4회)
    - 세포자살, RNAi, GFP, miRNA
  - 다세포동물중 최초로 DNA서열이 다 밝혀진 동물
  - 302개 뉴런으로 구성된 신경계

- Connectome
  - 뇌 속에 있는 신경세포들의 연결을 종합적으로 표현한 **뇌지도**(또는 뇌 회로도)







Interdisciplinary Linkage

- Lego MindStorm으로 예쁜 꼬마선충의 뉴런연결정보를 구현하여 로봇 개발
  - 단순히 connectome을 구현한 것만으로 기본적인 움직임을 행함
- 생물학→ 전산학 →로봇, 기계공학 → 계산생물학 → 심리철학으로 까지의 연구 촉발

기계학습의 특성

- 기계학습(Machine Learning)의 정의
  - "환경(Environment, E)과의 상호작용을 통해서 축적되는 경험적인 데이터(Data, D)를 바탕으로 지식 즉 모델(Model, M)을 자동으로 구축하고 스스로 성능(Performance, P)을 향상하는 시스템" (Mitchell, 1997)

$$ML : D \xrightarrow{P} M$$

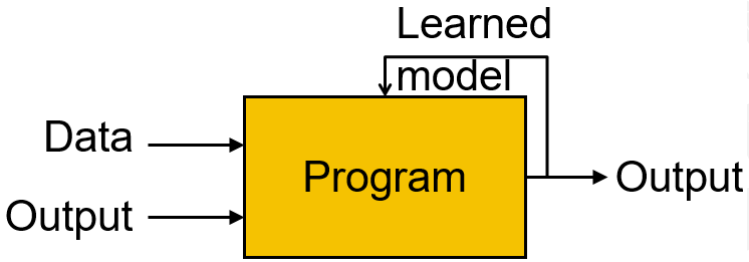
프로그래밍 방식과의 차이점

- 구성요소: 환경 E, 데이터 D, 모델 M, 성능지수 P
  - 환경(E) : 학습 시스템이 상호작용하는 대상, 학습할 문제
  - 데이터(D) : 환경과 상호작용을 통해 축적된 경험
    - 프로그램이 작성될 때 모든 가능한 입력을 고려하여 그 경우만을 다루는 것과 구별됨
  - 모델(M) : 데이터를 모델링하는 학습 시스템의 구조
  - 성능지수(P) : 학습 시스템의 성능 평가 지표
    - 학습 시스템이 목표를 이루기 위하여 최적화 해야 하는 지표

프로그래밍 방식과의 차이점

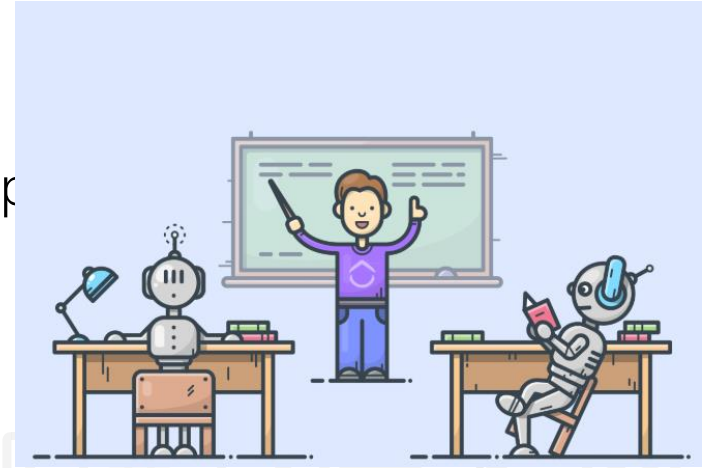
- 일반적인 컴퓨터 프로그램
  - 사람이 알고리즘 설계 및 코딩
  - 주어진 문제(데이터)에 대한 답을 출력

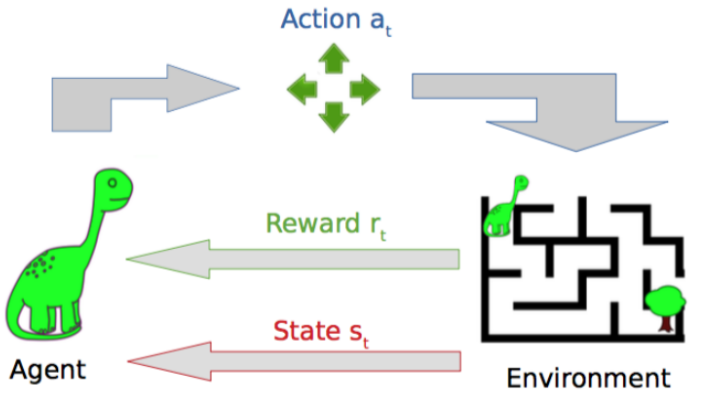
- 머신 러닝 프로그램
  - 사람이 모델을 코딩
  - 기계학습 알고리즘을 통한 모델 학습
  - 데이터에 대한 프로그램을 출력



기계학습의 유형

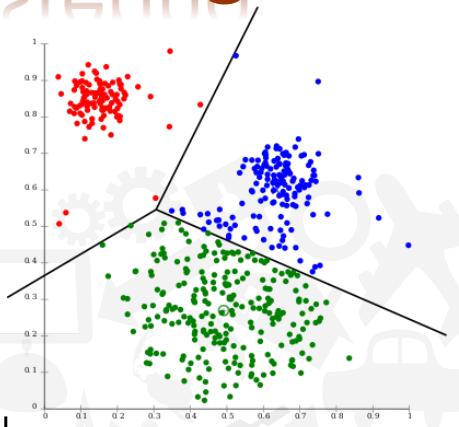
- Unsupervised learning
  - Training data does not include desired outp
  - E.g., clustering
- Supervised (inductive) learning
  - Training data includes desired outputs
  - E.g., Classification, regression/prediction
- Semi-supervised learning
  - Training data includes a few desired outpu
- Reinforcement learning
  - Rewards from sequence of actions



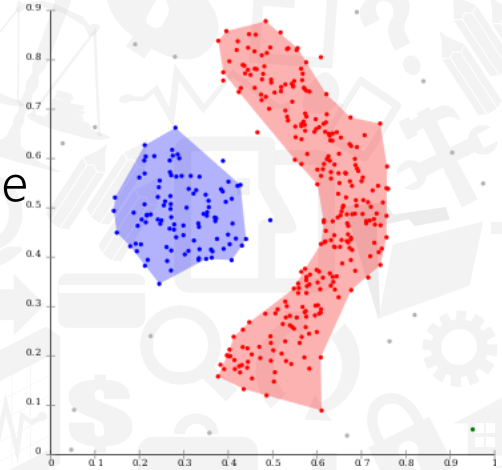


Unsupervised learnig 사례 -Clustering

- Definition
  - A task of grouping a set of objects in such a way that objects in the same cluster
- *Centroid*-based clustering
  - Each cluster is represented by a central vector
  - To find the *k* cluster centers and assign the objects to the nearest cluster center wrt. the squared distances from the cluster are minimized
  - e.g., *k*-means, RAM, CLARA, …
- *Density*-based clustering
  - Clusters are defined as areas of higher density than the remainder of data set.
  - e.g., DBSCAN and OPTICS, …
- Other clustering approaches also exist
  - hierarchical clustering and distribution-based approaches



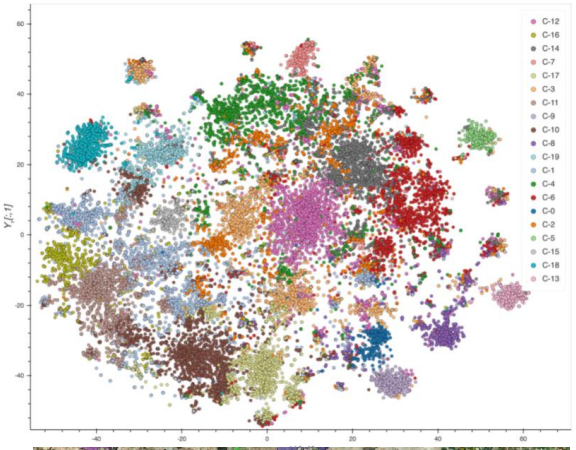
K-means



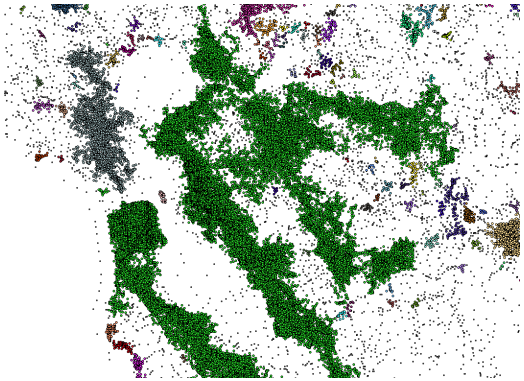
DBSCAN

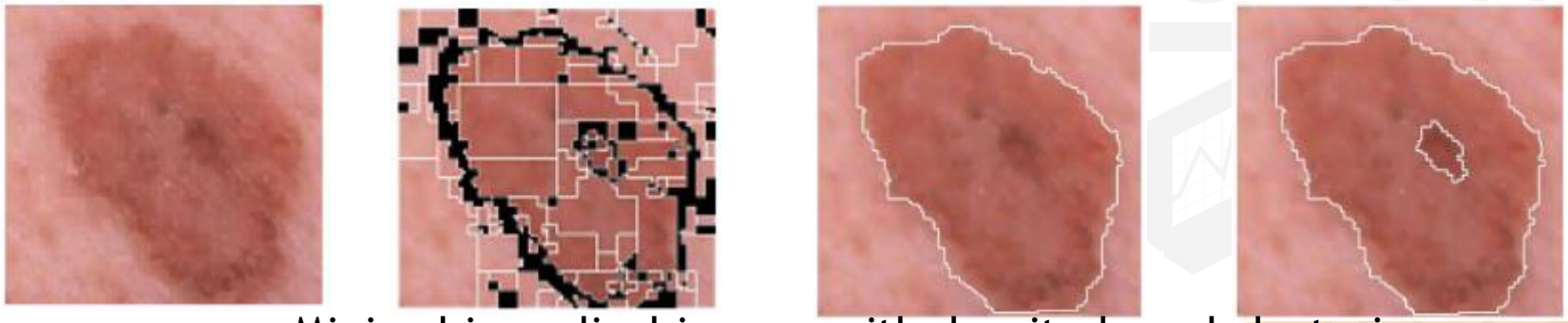
군집화 응용 사례

COVID-19
Literature
Clustering



Census survey
data





Mining biomedical images with density-based clustering

K-Means (Centroid-based clustering)

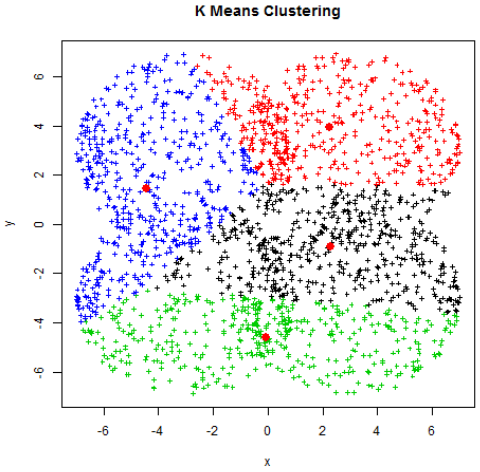
- Given a *k*, find a partition of *k* clusters to optimize the chosen partition criterion (cost function)
  - A heuristic approach : each cluster is represented by the centre of the cluster and the algorithm converges to stable centroids of clusters

0. Initialization : set seed points (randomly)

1. Find closest centroids
   - Assign each item to the cluster of the nearest seed point measured with a specific distance metric

2. Update centroid *s*
   - Compute new seed points as the centroids(mean points) of the clusters of the current partition

3. Go back to Step 1 until no more new assignmen
   - i.e., memberships in each cluster no longer change

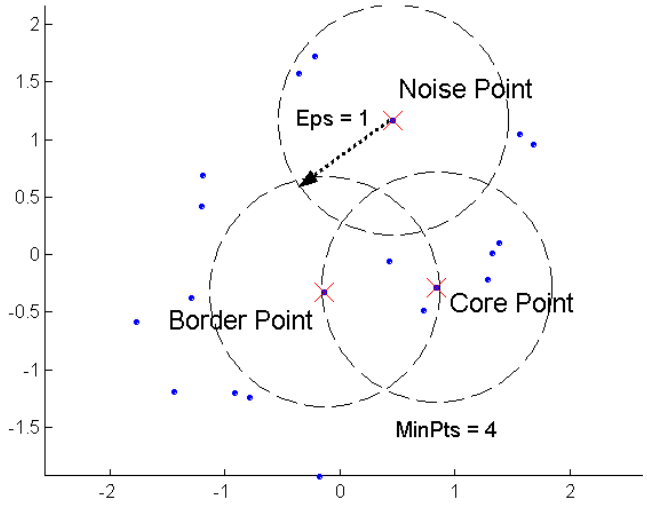


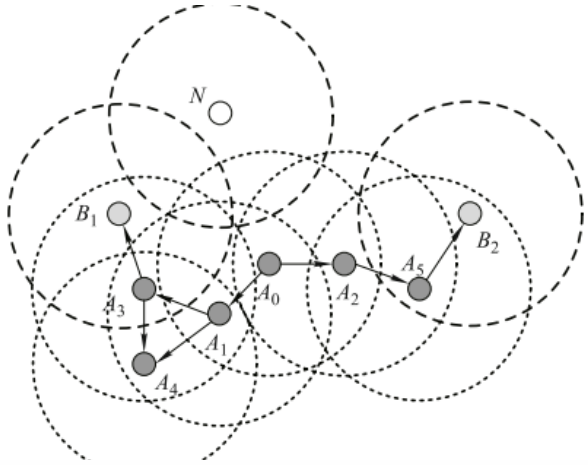
DBSCAN (Density-based clustering)

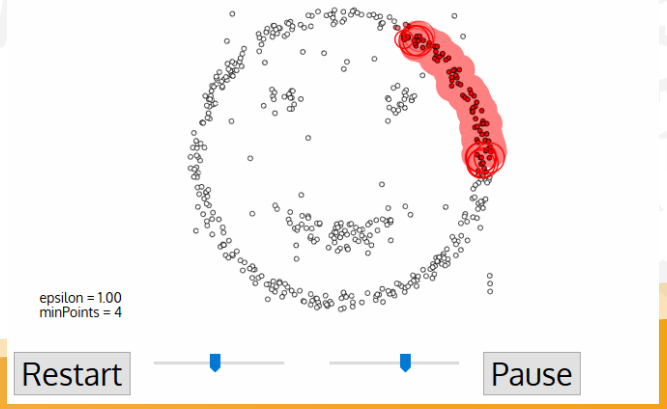
- Density = # of points within a specified radius ε (*Eps*)
- a core point is a point if it has more than a specified number of points (*MinPts*) within *Eps*
  - points at the interior of a cluster
- A border point has fewer than *MinPts* within *Eps*, but is in the neighborhood of a core point
- A noise point is any point that is not a core point or a border point
- Major features
  - Discover clusters of arbitrary shape
  - Handle noise
  - One scan
  - Need density parameters

DBSCAN

1. Create a graph whose nodes are the points to be clustered
2. For each core-point *c* create an edge from *c* to every point *p* in the ε-neighborhood of *c*
3. Set *N* to the nodes of the graph;
4. If *N* does not contain any core points terminate
5. Pick a core point *c* in *N*
6. Let *X* be the set of nodes that can be reached from *c* by going forward;
    1. create a cluster containing $X \cup \{c\}$
    2. $N = N/(X \cup \{c\})$
7. Continue with step 4

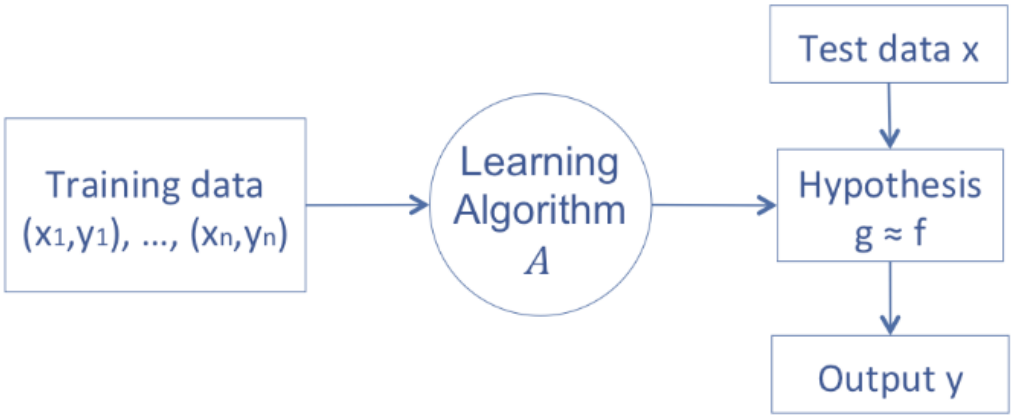






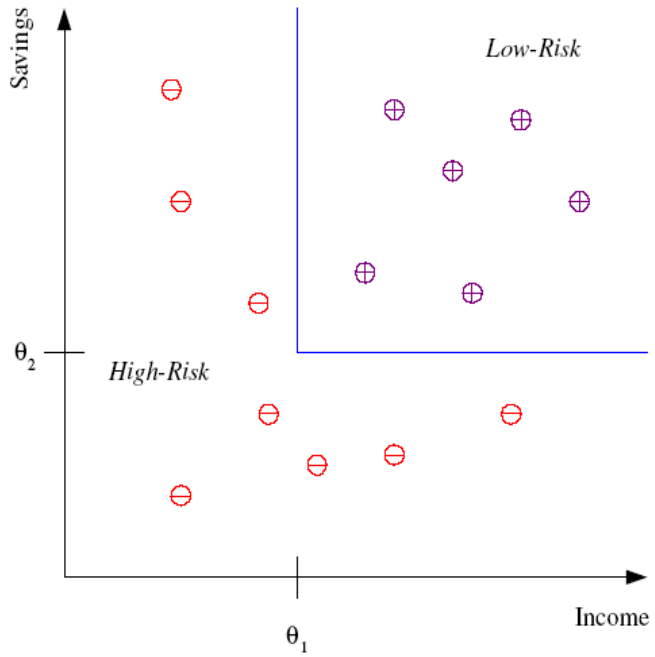
지도 학습(supervised learning)

- Given examples of a function (x, y) where *y = g(x)*
- Predict function *f(x)* for new examples *X*
  - Discrete f(x) : **classification**
  - Continuous f(x) : **regression**
  - f(x) := probability(x) : **Probability estimation**



분류 (classification)

- Example: Credit scoring
- Differentiating between low-risk and high-risk customers from their *income* and *savings*

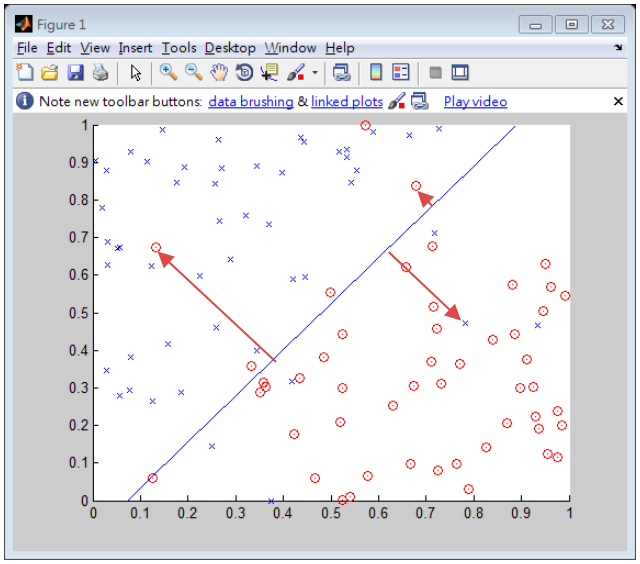


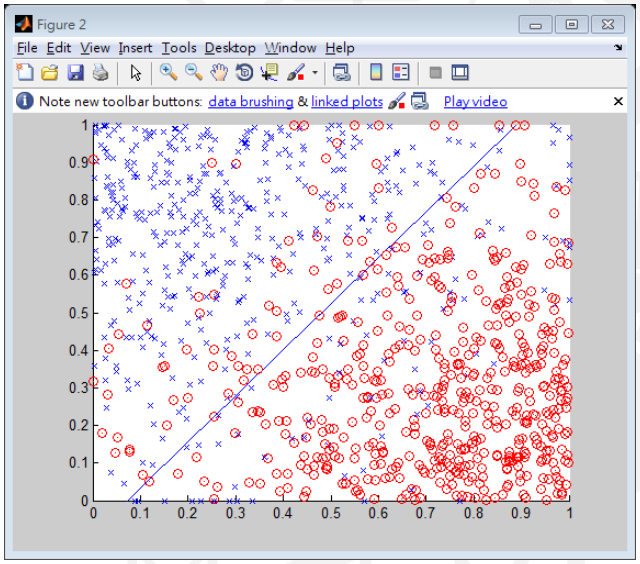
Discriminant: IF *income* > θ_1 AND *savings* > θ_2
THEN low-risk ELSE high-risk

Model

선형 분류기(Linear classifier)

- Classification decision is made based on the value of a linear combinations
  - If function(or model) is simple, error rate goes up.





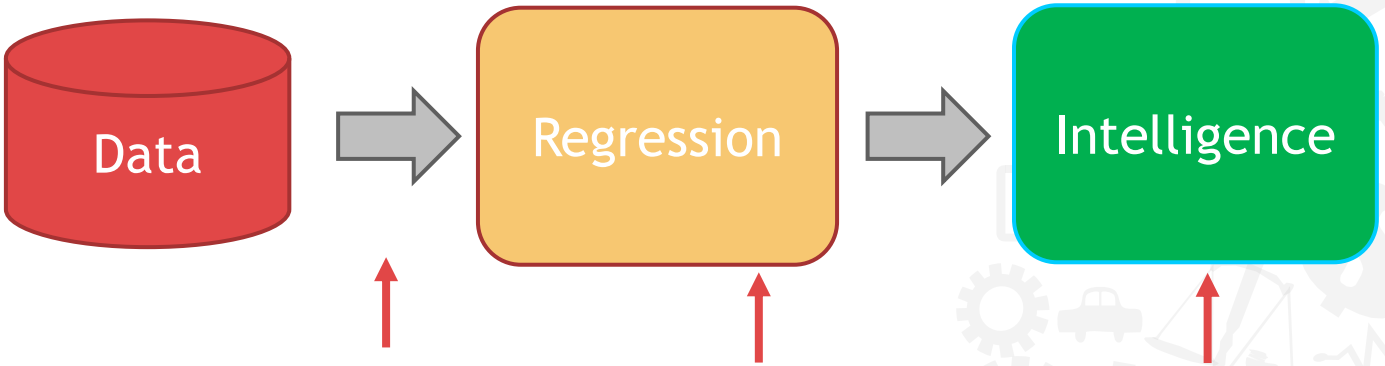
- More complicated function(or model) is required for guaranteeing a minimal error rate

분류의 응용 사례

- Face recognition: pose, lighting, occlusion (glasses, beard), make-up, hair style
- Character recognition: Different handwriting styles.
- Speech recognition: Temporal dependency
  - Use of a dictionary or the syntax of the language.
  - Sensor fusion: Combine multiple modalities; eg, visual (lip image) and acoustic for speech
- Medical diagnosis: From symptoms to illnesses
- Web Advertising: Predict if a user clicks on an ad. on the Internet
- ….

회귀 분석(Regression Analysis)

From features to predictions



Prediction: Regression

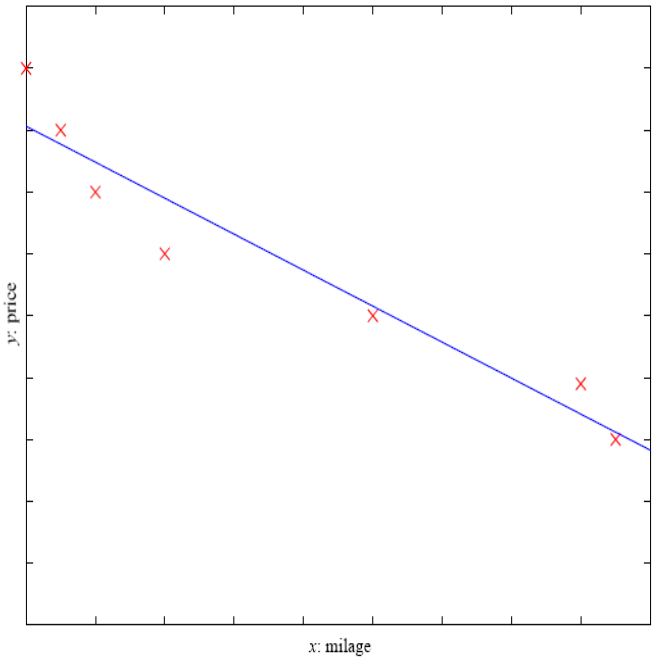
- Example: Price of a used car
- x: car attributes

  y: price

  $$y = g(x \mid \theta)$$

  $g(\ )$ model,

  θ: parameters



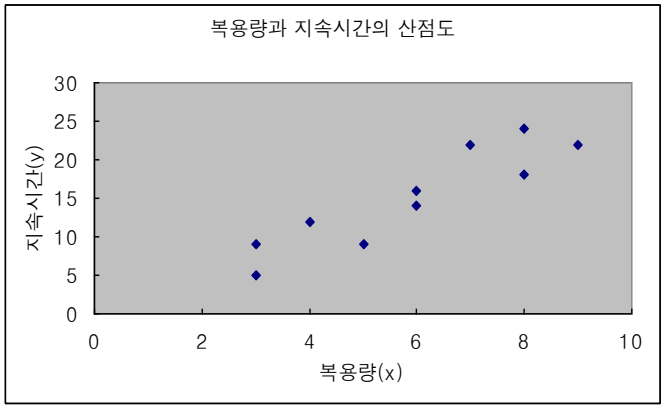
Regression

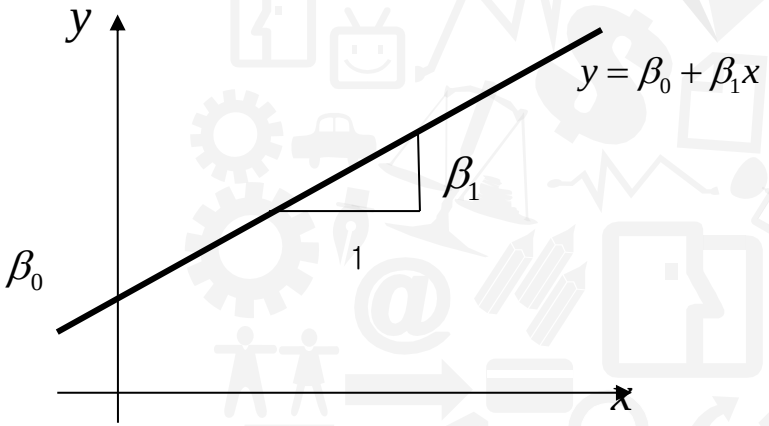
- x : 독립변수 (independent variable)
  - 원인
  - 특징(Feature)
- Y : 종속변수(depedent variable)
  - 결과 : 예측 가능

- 단순회귀분석(simple regression analysis)
  - 독립 변수 1개와 종속 변수 1개의 관계를 분석
- 다중회귀분석(multiple regression analysis)
  - 여러 독립변수와 하나의 종속 변수 사이의 관계를 규명

Regression examples

- 복용량에 따른 효과의 지속시간의 관계
  - x : 약품의 복용량
  - y : 효과가 지속되는 기간





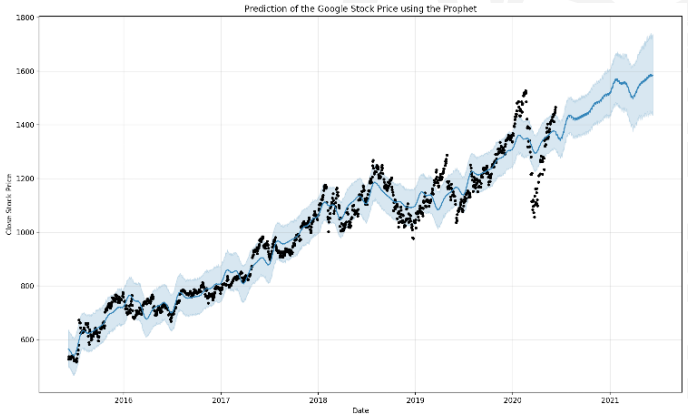
$$y = \beta_0 + \beta_1 x + \varepsilon$$

θ { β_0, β_1 : 회귀모수 (미지의 상수)
ε : 오차항

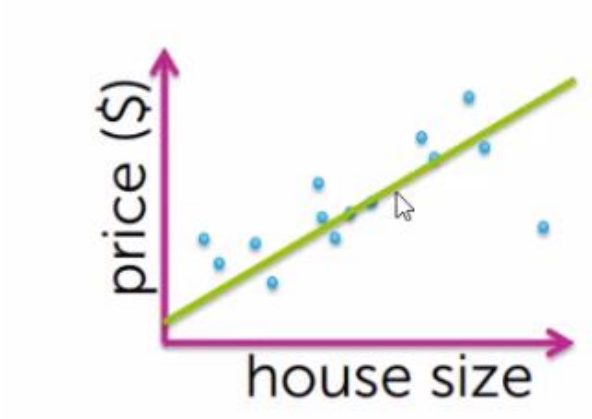
Regression examples

Stock Prediction

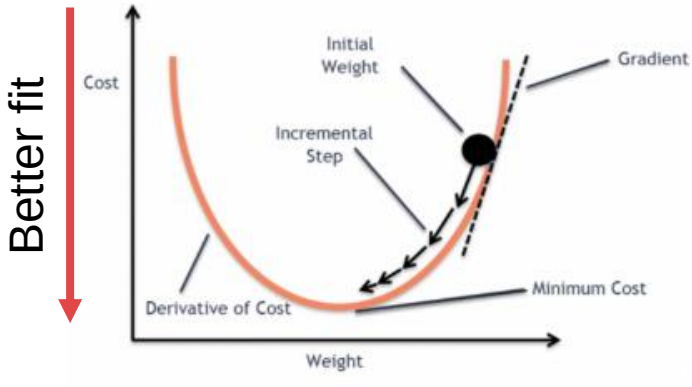
- Predict the price of a stock y
- Depends on x =
  - Recent history of stock price
  - News events
  - Related commodities



Simple Regression



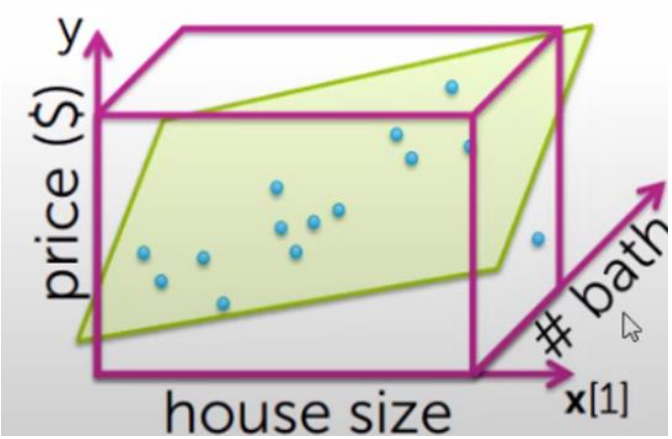
Define goodness-of-fit
Metric for each possible line



Gradient descent algorithm
Get estimated parameters
- interpret
- use to form predictions

Multiple regression

Fit more complex relationships than just a line



Incorporate more inputs

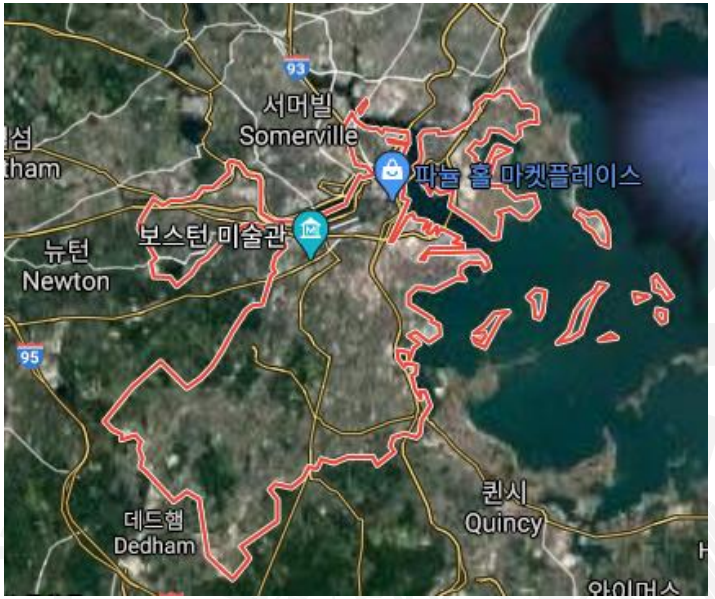
- Square feet
- Num. of bathrooms
- Num. of bedrooms
- Lot size
- Year built
- . . .

Lab 1: Predicting house price

Boston Housing Price Dataset

- Dr. Jason에 의해 작성된 1978년도 보스턴 교외지역부동산 관련 정보
  - 14개 변수(column)로 구성된 506개의 데이터(row)
    - 506 x 14 tabular data
  - 종속 변수 (1개)
    - MEDV : 1978년 보스턴 교외 506개 타운의 주택 가격 중앙값(단위 $1,000)
  - 독립 변수 (13개)
    - CRIM, INDUS, NOX, RM, LSTAT, B, PTRATIO, ZN, CHAS, AGE, RAD, DIS, TAX

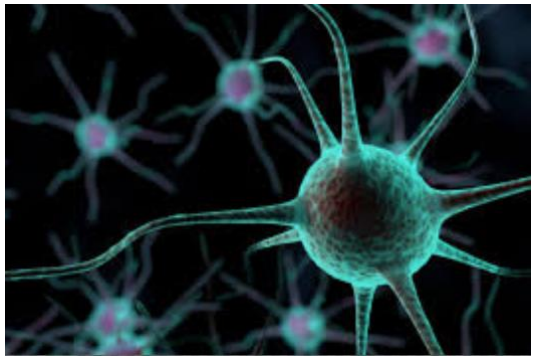
Features

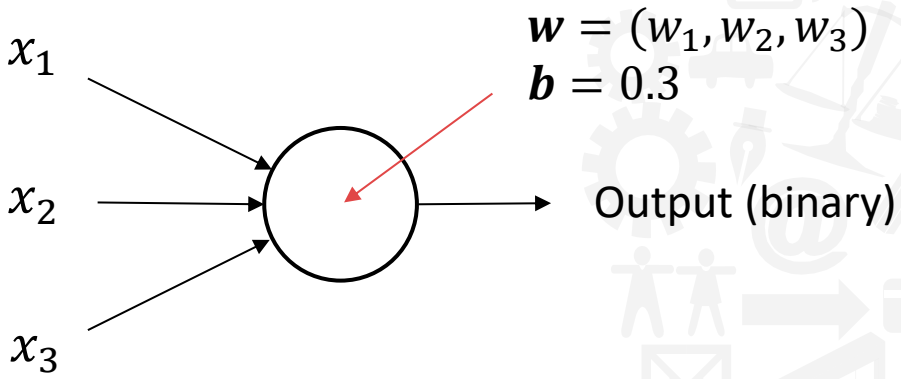


- CRIM : 범죄율
- INDUS: 비소매상업지역 면적 비율
- NOX: 일산화질소 농도
- RM : 주택당 방수
- LSTAT : 인구중 하위 계층 비율
- B : 인구중 흑인 비율
- PTRATIO: 학생/교사 비율
- ZN: 25,000 평방피트를 초과한 거주지역 비율
- CHAS : 찰스강의 경계에 위치한 경우 1 아니면 0
- AGE : 1940 년 이전에 건축된 주택의 비율
- RAD 방사형 고속도로까지의 거리
- DIS : 직업센터의 거리
- TAX : 재산세율

Neuron

- Basic building block for composition is a perceptron (artificial neuron) (Rosenblatt c.1960)
- Linear classifier
  - With a vector of weights *w* and a bias *b*



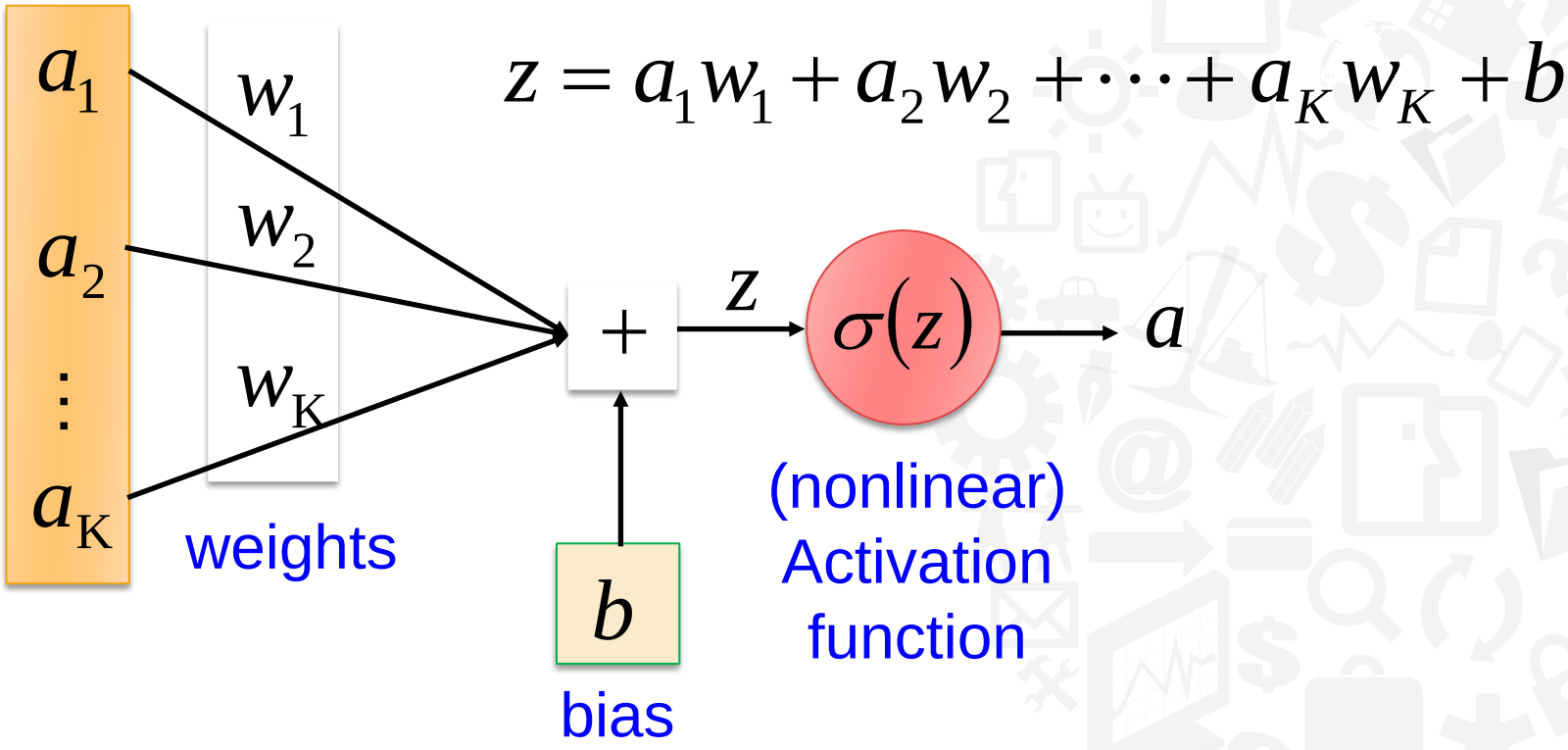


$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

$$w \cdot x \equiv \sum_j w_j x_j$$

Element of Neural Network

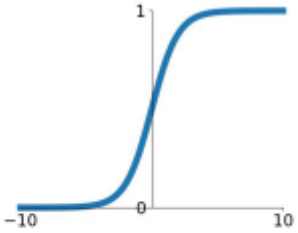
Neuron $f: R^K \rightarrow R$



Samples of Activation Functions

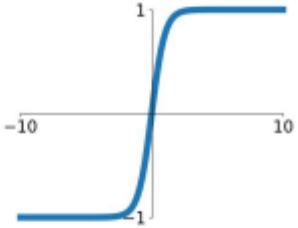
Sigmoid

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



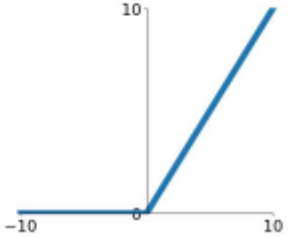
tanh

$$\tanh(x)$$



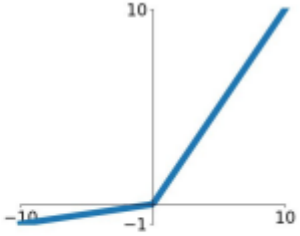
ReLU

$$\max(0, x)$$



Leaky ReLU

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

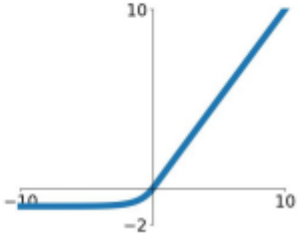


Maxout

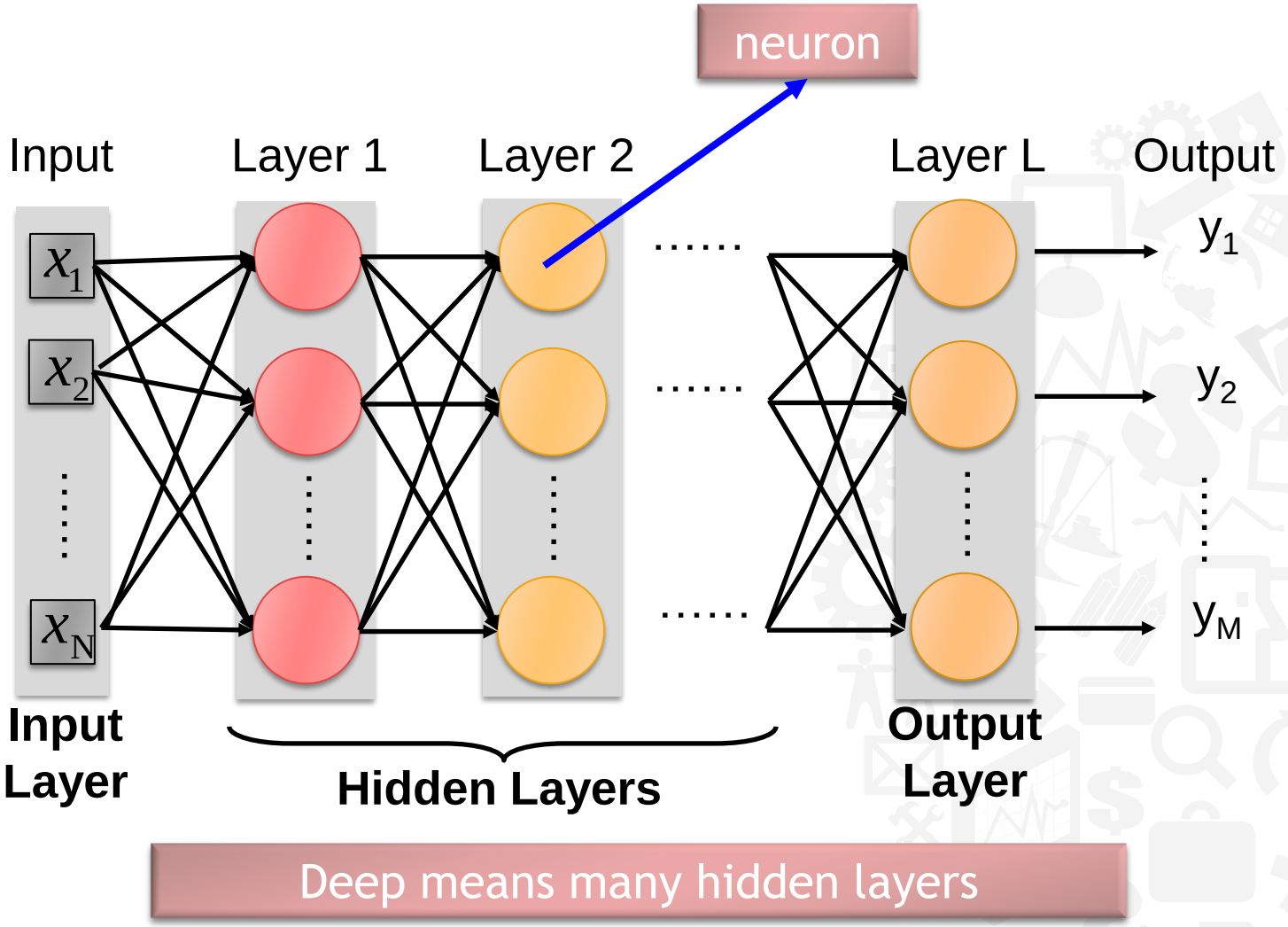
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

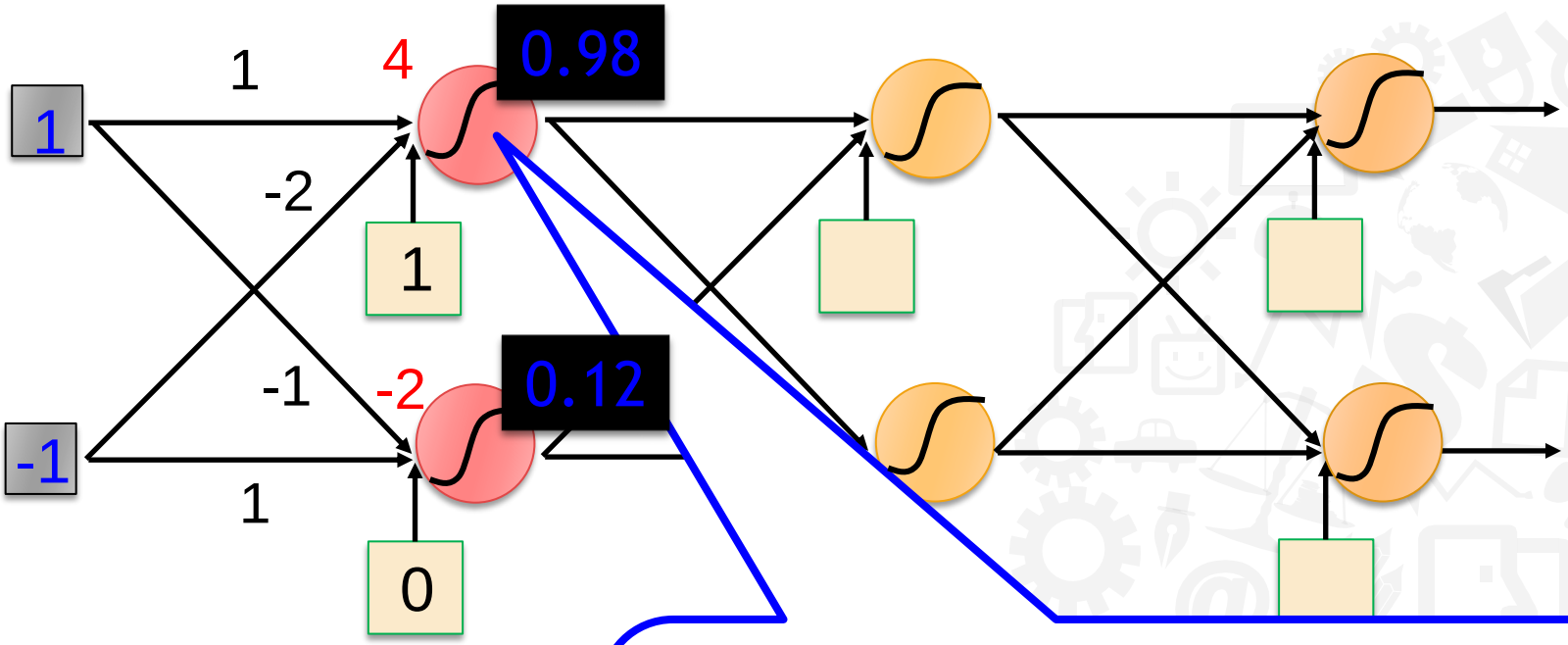
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



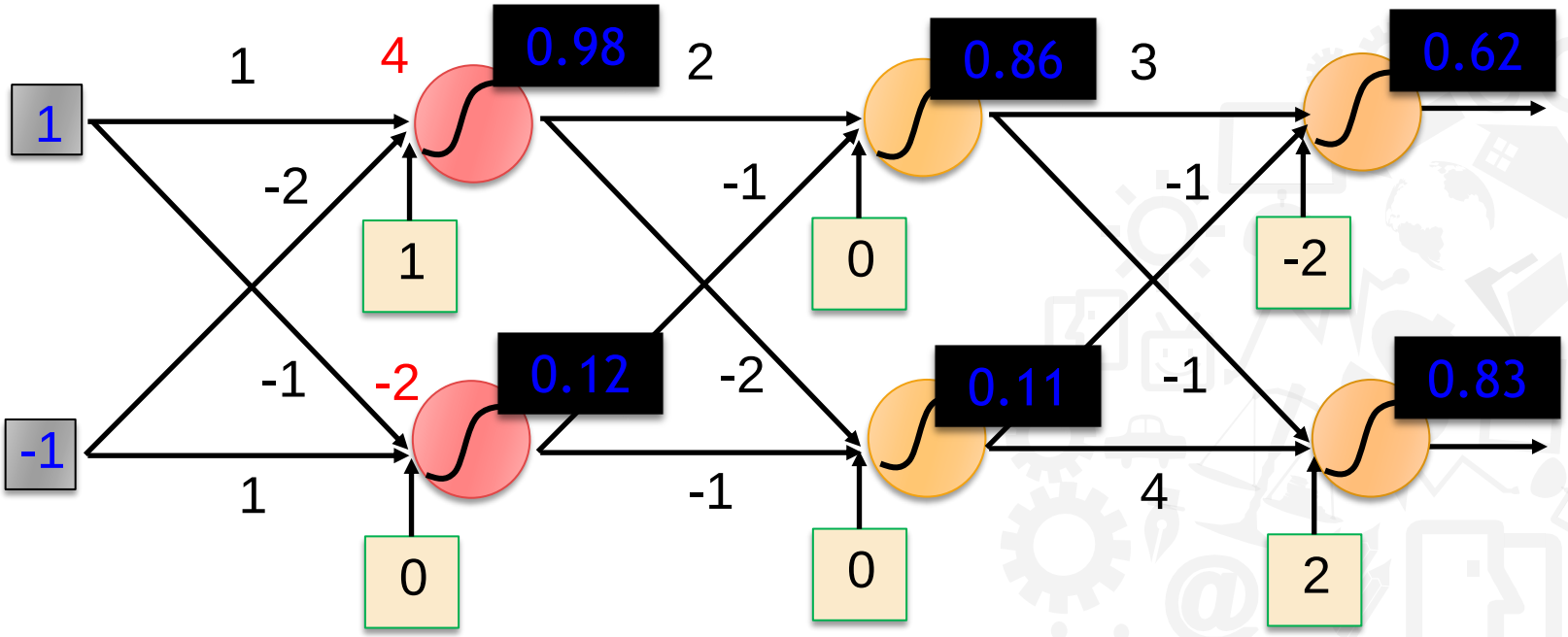
(Deep) Neural Network



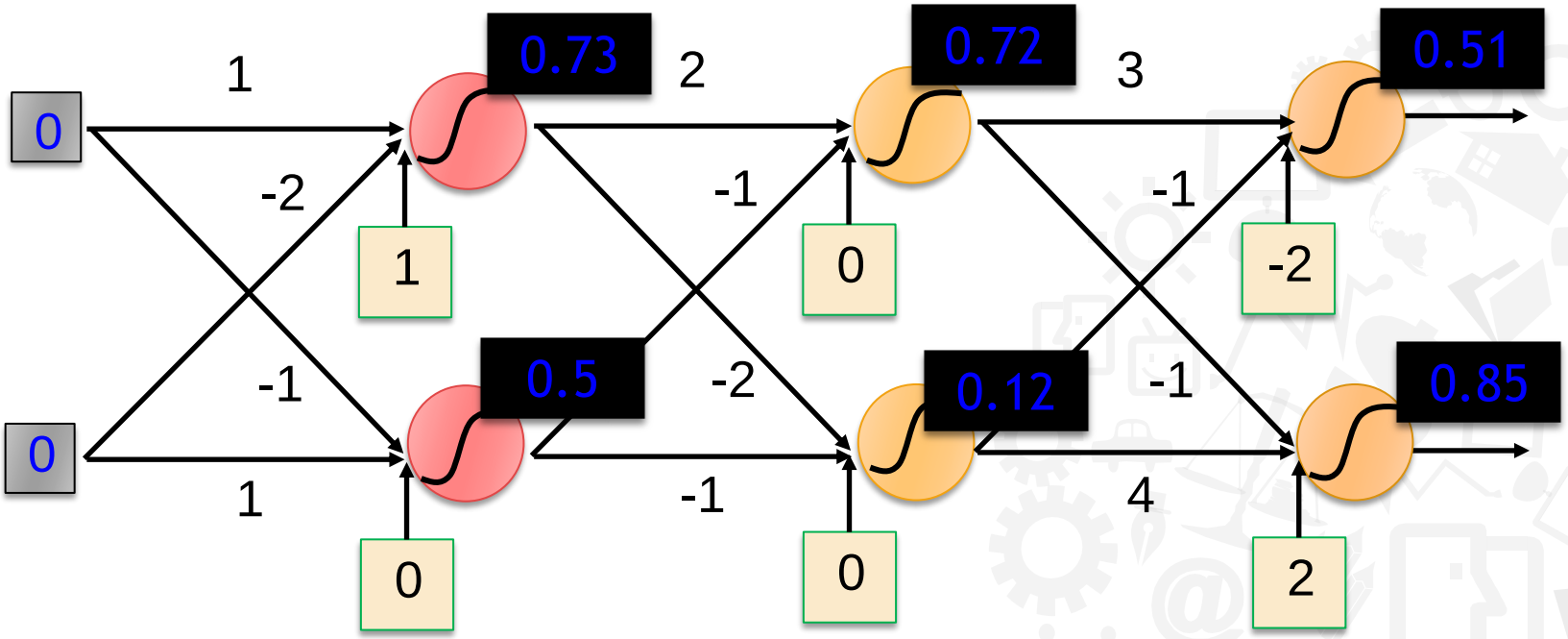
Example of neural network



Example of Neural network



Example of Neural network

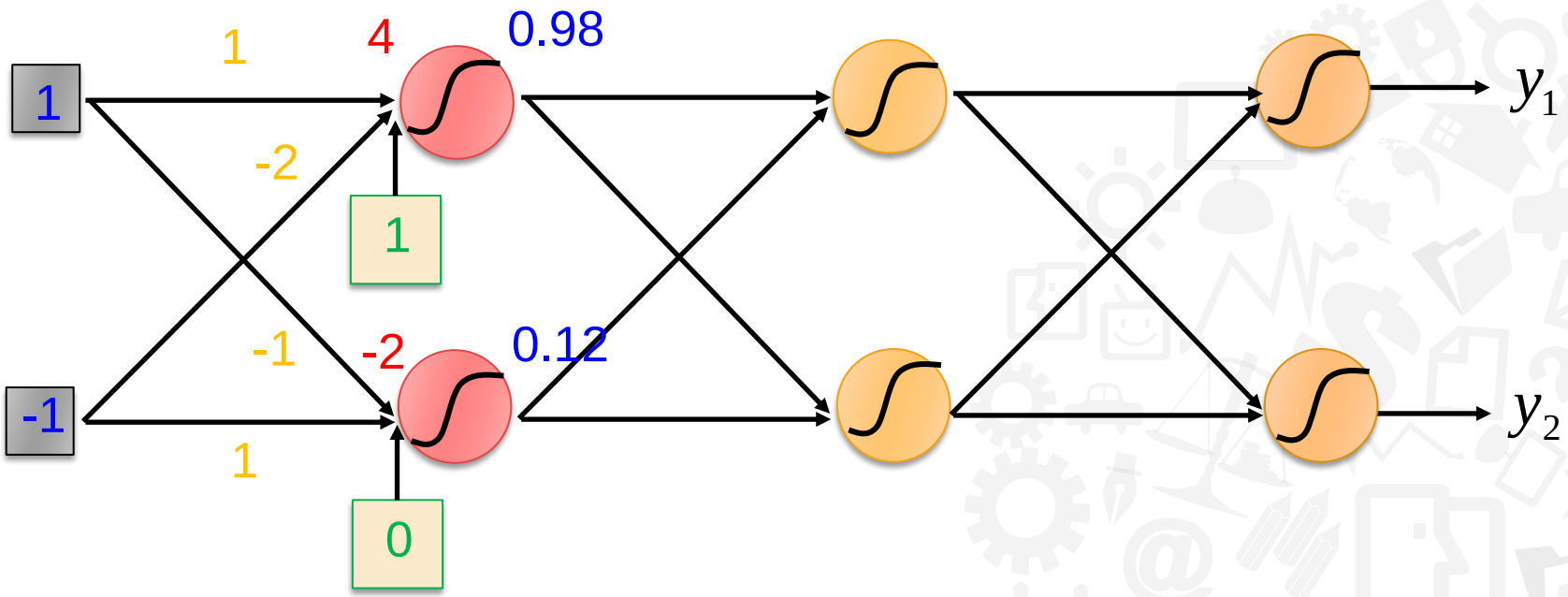


$f: R^2 \rightarrow R^2$

$$f\left(\begin{bmatrix}1\\-1\end{bmatrix}\right) = \begin{bmatrix}0.62\\0.83\end{bmatrix} \quad f\left(\begin{bmatrix}0\\0\end{bmatrix}\right) = \begin{bmatrix}0.51\\0.85\end{bmatrix}$$

Different parameters define different function

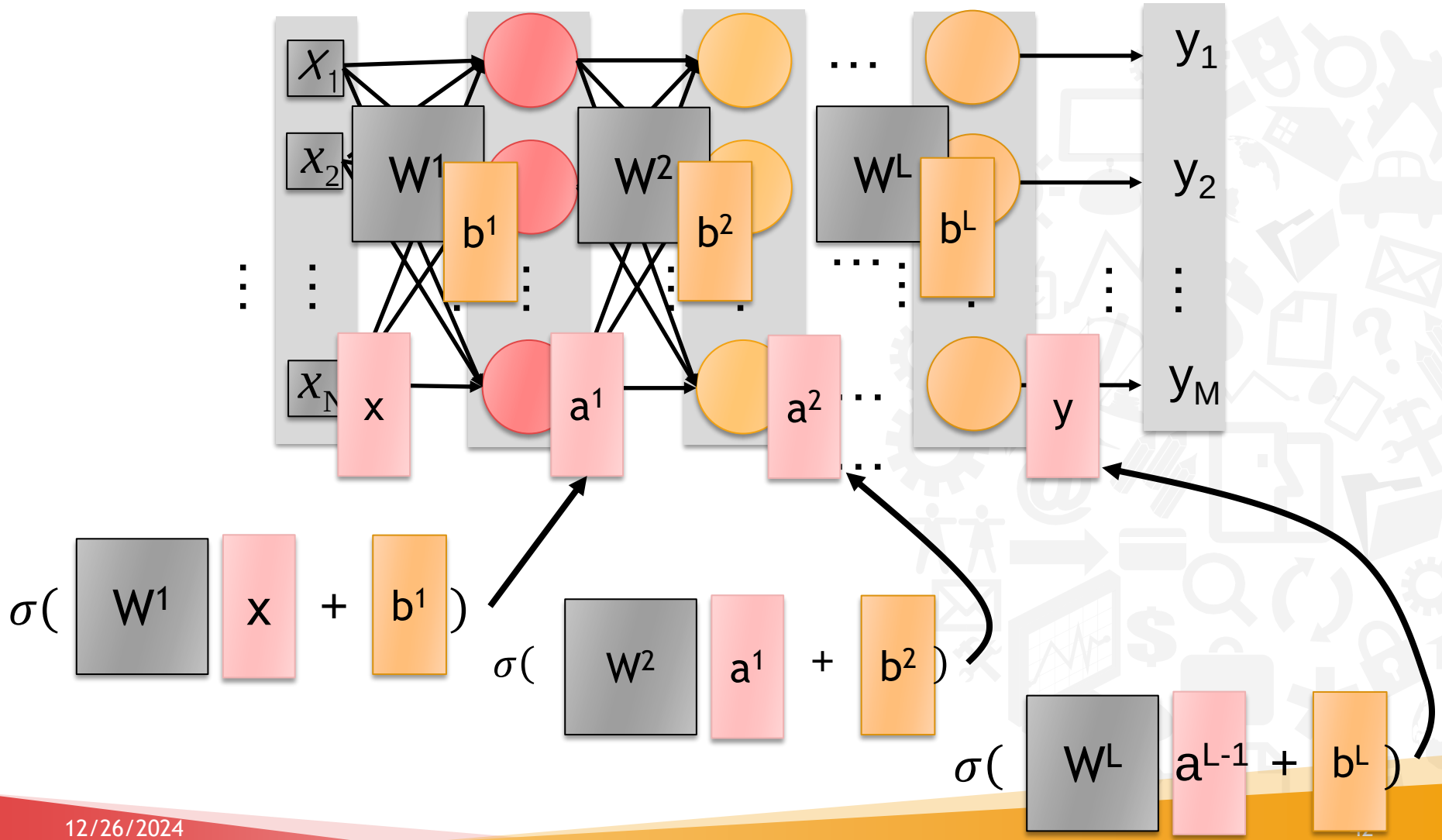
Matrix operation



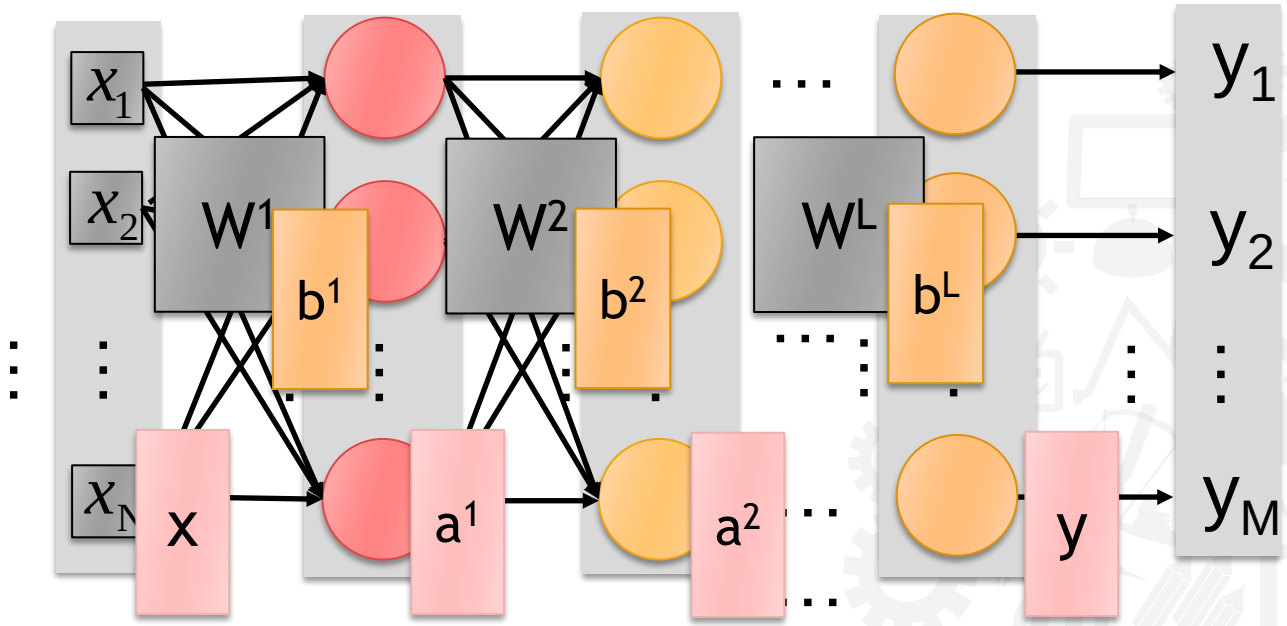
$$\sigma\left(\begin{bmatrix}1 & -2\\ -1 & 1\end{bmatrix}\begin{bmatrix}1\\ -1\end{bmatrix}+\begin{bmatrix}1\\ 0\end{bmatrix}\right)=\begin{bmatrix}0.98\\ 0.12\end{bmatrix}$$

$$\begin{bmatrix}4\\ -2\end{bmatrix}$$

Neural network



Neural Network



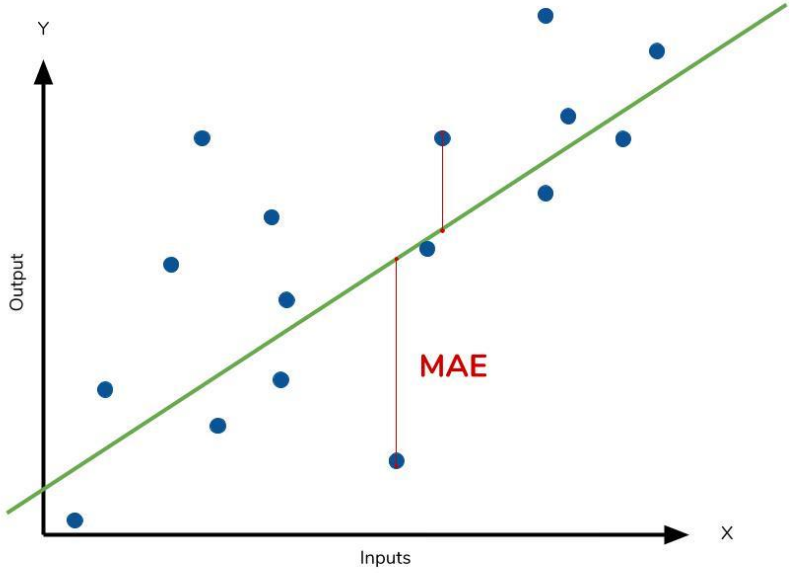
$$y = f(x)$$

Using parallel computing techniques to speed up matrix operation

$$= \sigma(W^L \cdots \sigma(W^2 \sigma(W^1 x + b^1) + b^2) \cdots + b^L)$$

Assessing the performance of the Model

$$MAE(Mean\ Absolute\ Error) = \frac{1}{m}\sum_{i}^{m}|y_i - \hat{y}_i|$$

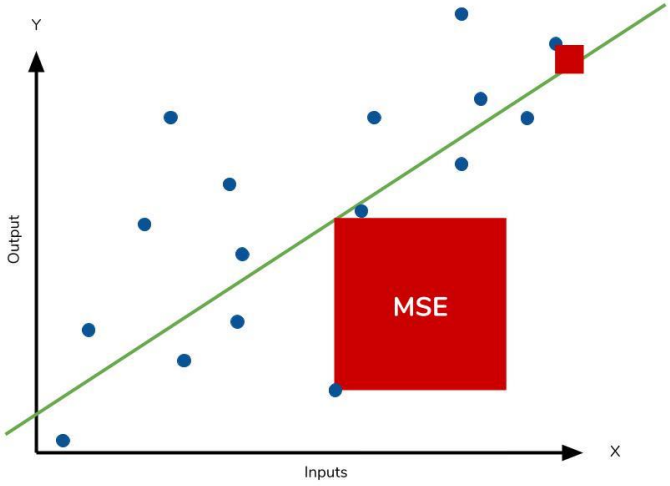


- m : data points
- Y : vector of observed values
- $\hat{Y}$: vector of predicted values

- 절대값을 취하기 때문에 직관적
- MSE보다 특이치에 robust
- Regression 평가에 사용

Assessing the performance of the Model

$$Mean\ Square\ Error = \frac{1}{m}\sum_{i=1}^{n}(\hat{Y} - Y)^2$$

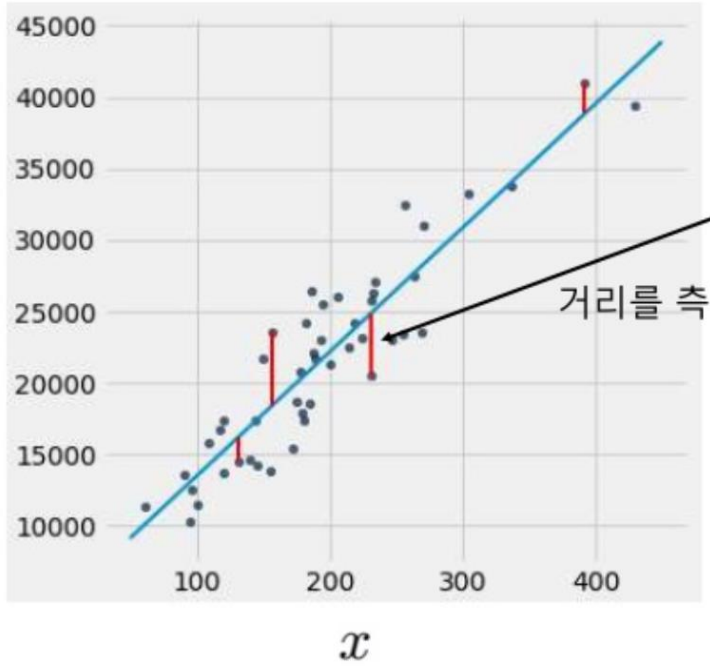


- m : data points
- Y : vector of observed values
- $\hat{Y}$: vector of predicted values

- 모델의 예측값과 실제 값 차이이의 면적의 합
- 특이치에 민감
- Training에 사용

How to minimize the cost?

- $MSE = \frac{1}{m}\sum_{i=1}^{n}(\hat{Y} - Y)^2$



$$H(x) = Wx + b$$

$$cost(W, b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$

(m = data size, y = 실제값)

cost 함수

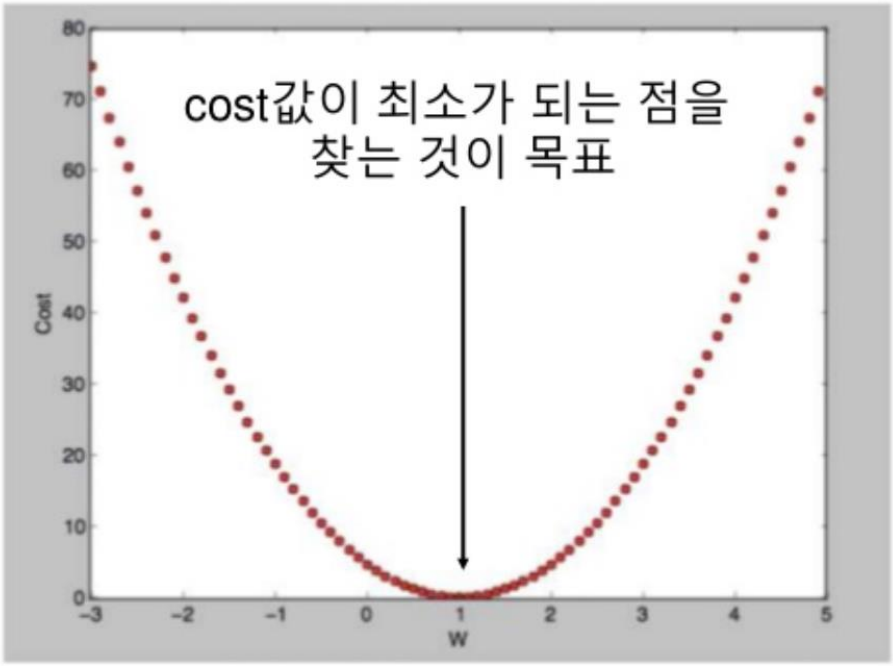
↓

W , b 의 함수

↓

Cost값을 작게 가지는 W , b 를 학습
= linear regression 의 학습

Cost function(or loss function)



(e.g. MSE)

$$H(x) = Wx$$

$$cost(W) = \frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})^2$$

무작위로 $H(x)$을 그어서
$cost(W)$가 최소가 되는 점을 찾는다?

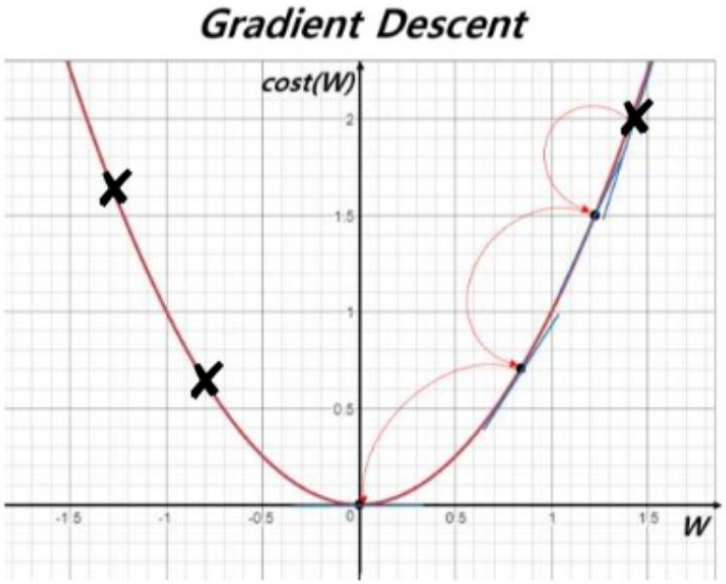
↓

$cost(W)$가 최소가 되는 점을
기계적으로 찾아내야 함

↓ optimization

Gradient descent algorithm

Gradient descent algorithm



✘ = 시작점

learning rate(step size)
= 수렴속도 조절

$$W := W - \eta \frac{1}{m} \sum_{i=1}^{m} \left(W \cdot x^{(i)} - y^{(i)} \right) \cdot x^{(i)}$$

$\frac{\partial cost(W)}{\partial W}$: 가파른 정도(slope)와 방향

*참고: $\frac{\partial (x^n)}{\partial x} = nx^{n-1}$

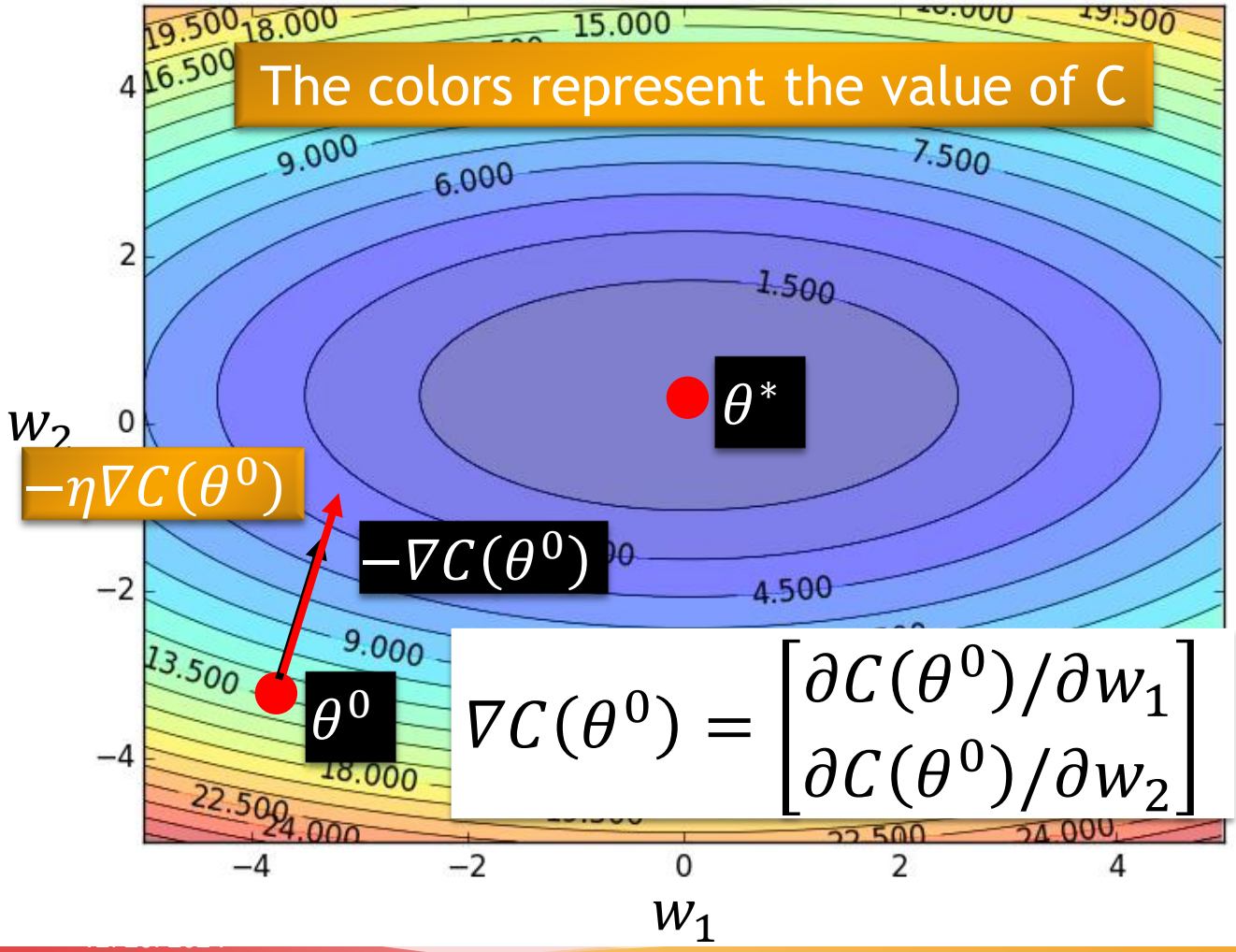
1. 시작점의 경사도에 따라 이동
2. 이동된 위치의 경사도를 따라 다시 이동
3. Cost(W)가 최소 즉 경사도 0인 지점까지 반복

Gradient Descent

Assume there are only two parameters w_1 and w_2 in a network.

$\theta = \{w_1, w_2\}$

Error surface



Randomly pick a starting point θ^0

Compute the negative gradient at θ^0

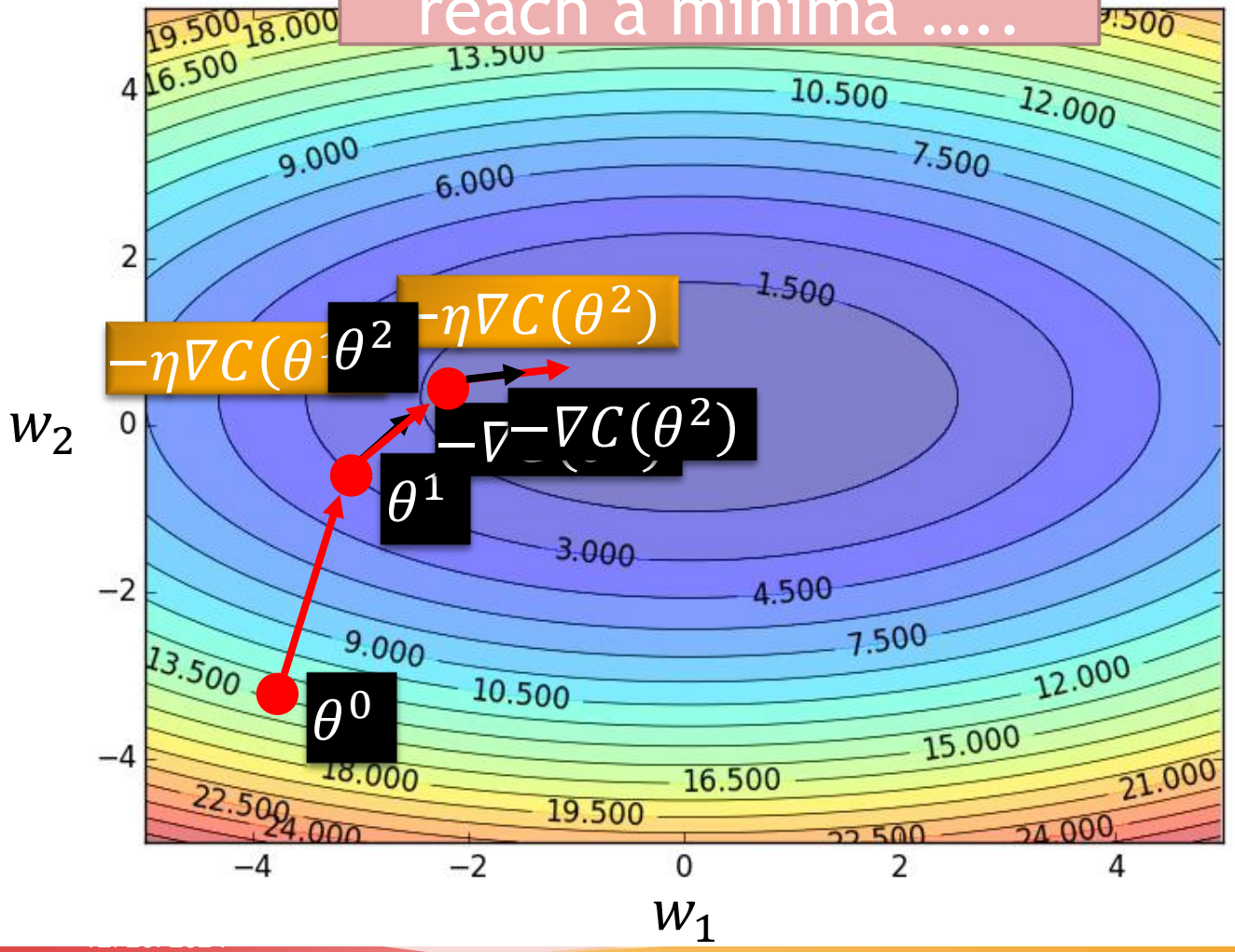
$-\nabla C(\theta^0)$

Times the learning rate η

$-\eta\nabla C(\theta^0)$

Gradient Descent

Eventually, we would reach a minima …..



Randomly pick a starting point θ^0

Compute the negative gradient at θ^0

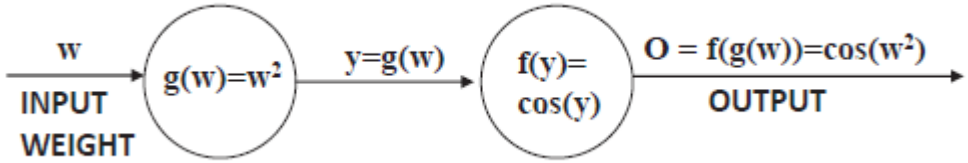
$-\nabla C(\theta^0)$

Times the learning rate η

$-\eta\nabla C(\theta^0)$

Backpropagation

- To compute the partial derivative of the loss function wrt. each intermediate weight
  - Not a simple matter with multi-layer architectures



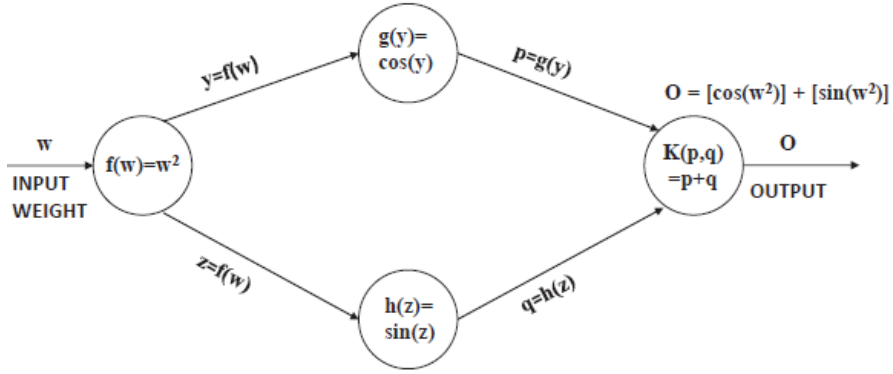
- In the univariate chain rule, we compute product of local derivatives

$$\frac{\partial f(g(w))}{\partial w} = \underbrace{\frac{\partial f(y)}{\partial y}}_{-sin(y)} \cdot \underbrace{\frac{\partial g(w)}{\partial w}}_{2w} = -2w \cdot \sin(y) = -2w \cdot \sin(w^2)$$

- Local derivatives are easy to compute because they care about their own inputs and outputs

Backpropagation

- Multivariate chain rule



$$\frac{\partial o}{\partial w} = \underbrace{\frac{\partial K(p,q)}{\partial p}}_{1} \cdot \underbrace{g'(y)}_{-\sin(y)} \cdot \underbrace{f'(w)}_{2w} + \underbrace{\frac{\partial K(p,q)}{\partial q}}_{1} \cdot \underbrace{h'(z)}_{\cos(z)} \cdot \underbrace{f'(w)}_{2w}$$

$$= -2w \cdot \sin(y) + 2w \cdot \cos(z)$$

$$= -2w \cdot \sin(w^2) + 2w \cdot \cos(w^2)$$

- Product of local derivatives along all paths from w to o

Backpropagation

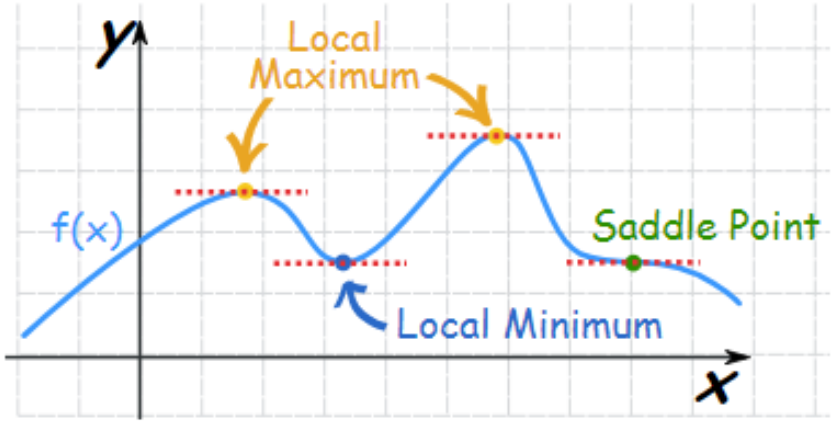
- A neural network can have millions of parameters
  - The size of today's NNs tends to increase
    - E.g., GPT-3 consists of 175B parameters
- Many toolkits can compute the gradients automatically

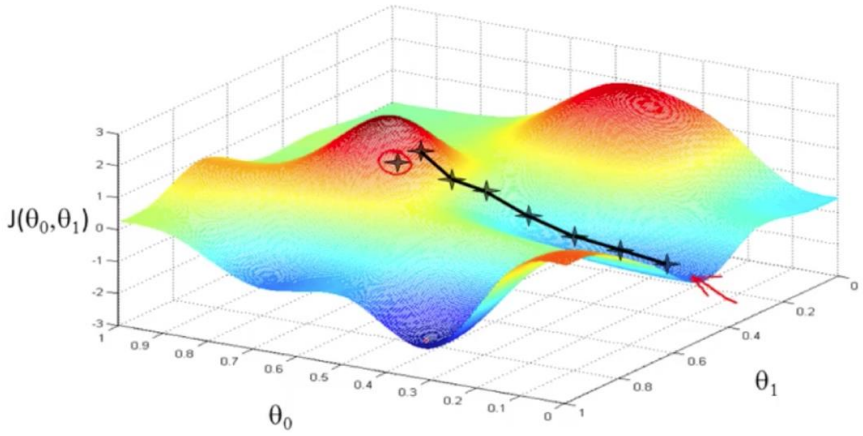
PyTorch TensorFlow Keras theano

- Pytorch: 연구자들이 선호, 많은 연구자 커뮤니티 사이트, HuggingFace에 등록된 모델의 92%가 Pytorch(`23.3)
- TensorFlow : 산업계에서 주로 활용, Model Serving, Debugging 유리

Local Minima

- Gradient descents never guarantee global minima

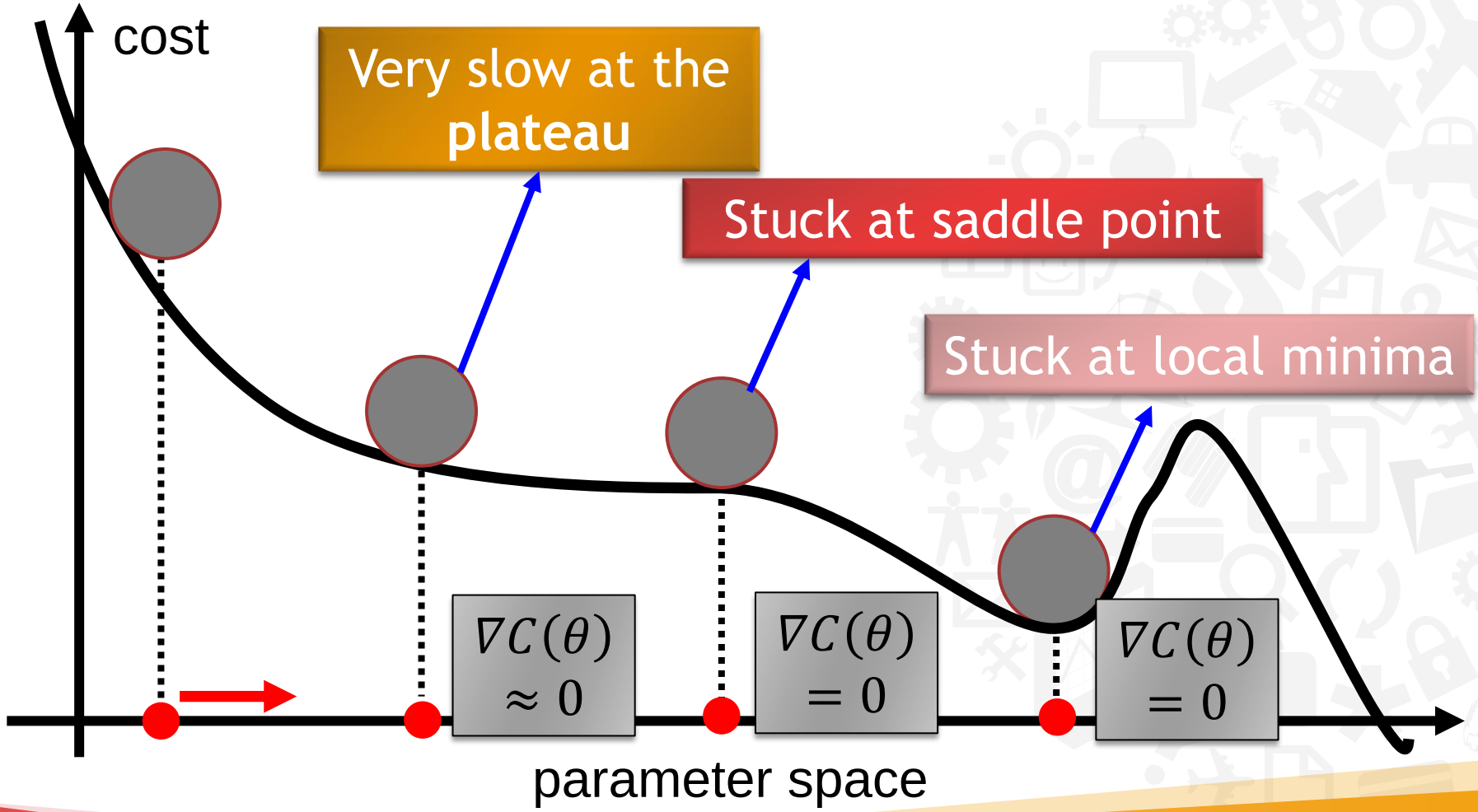




Different initial point θ^0

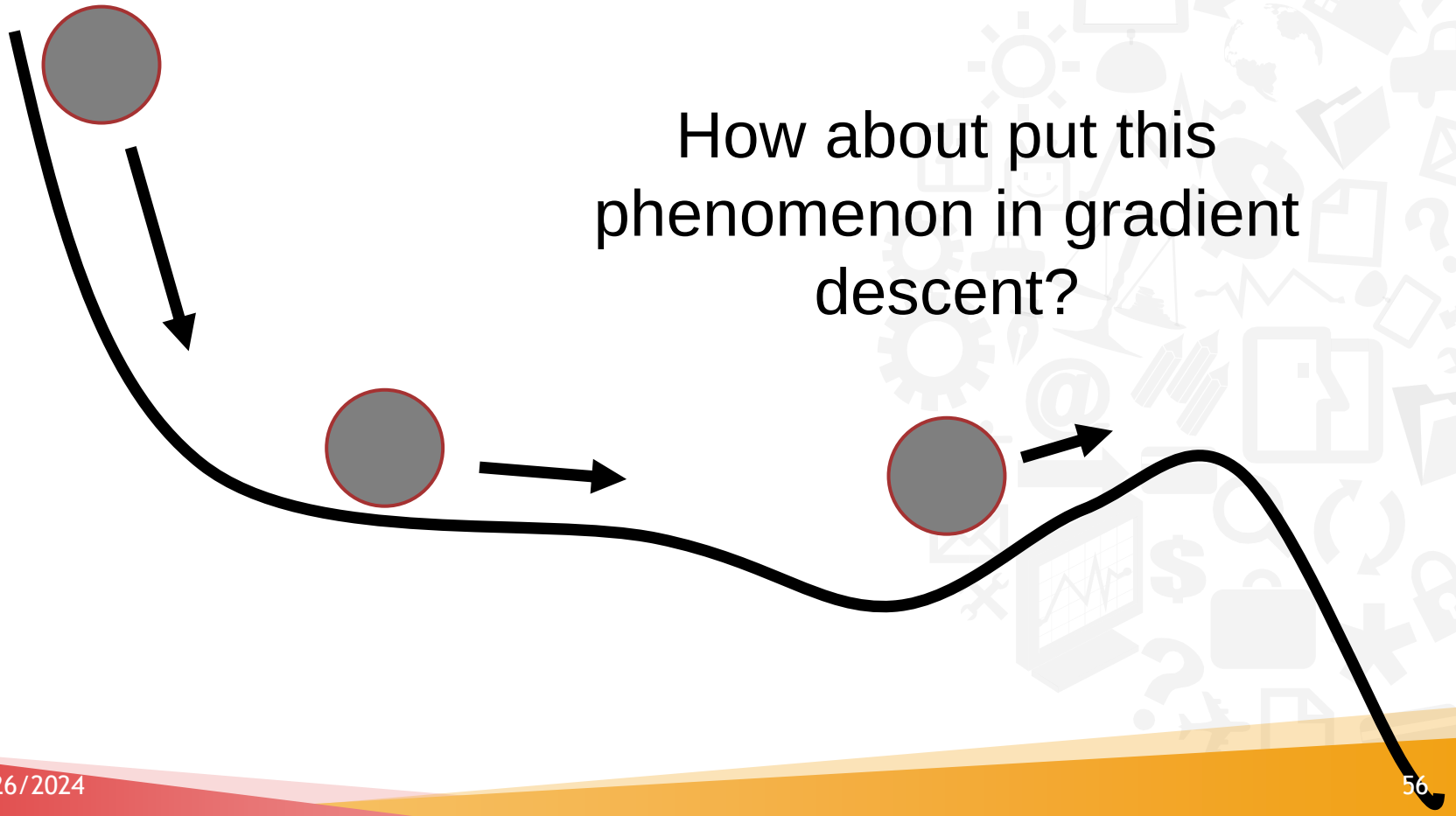
Reach different minima, so different results

Besides local minima .....



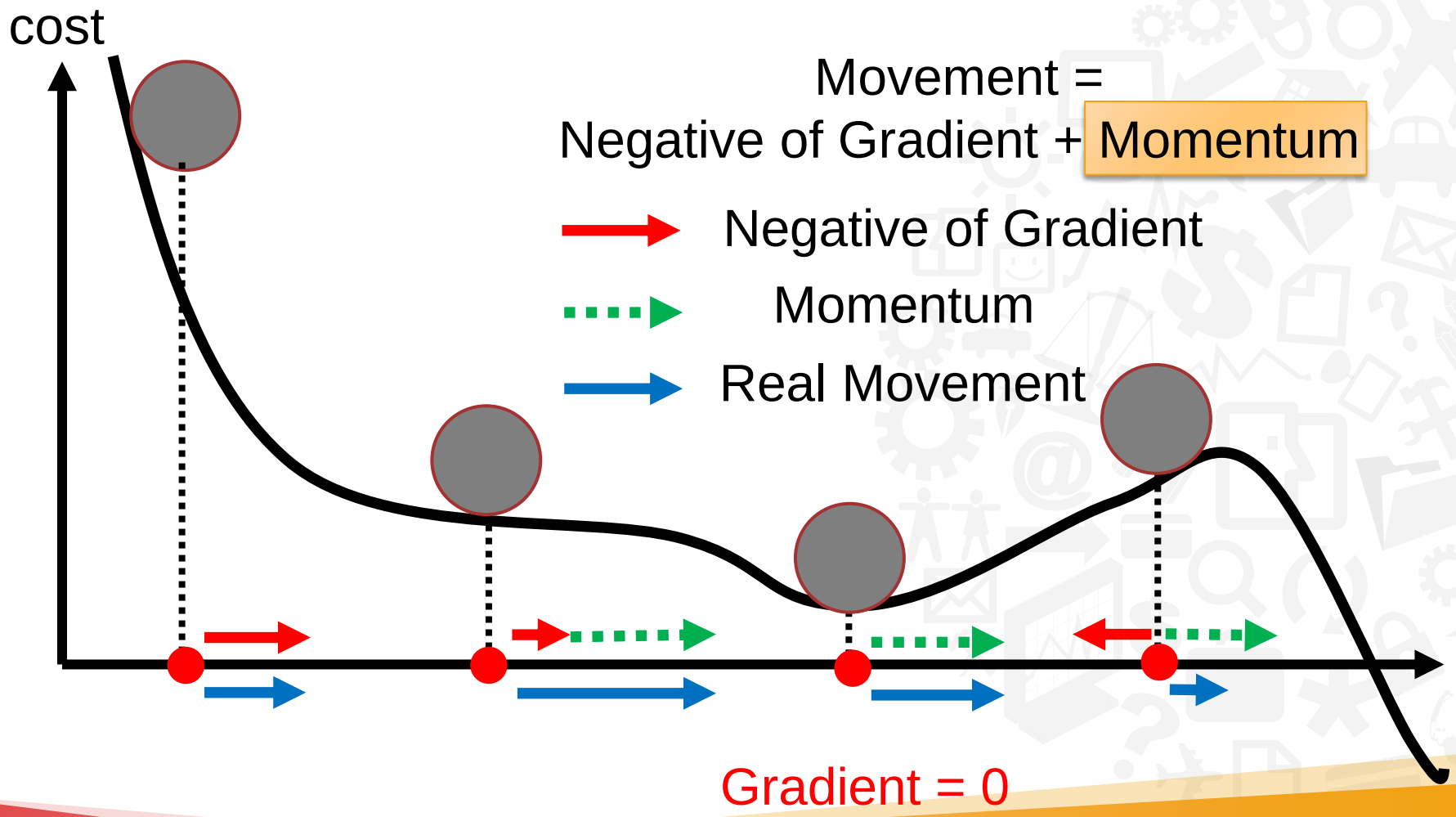
In physical world ……

- Momentum



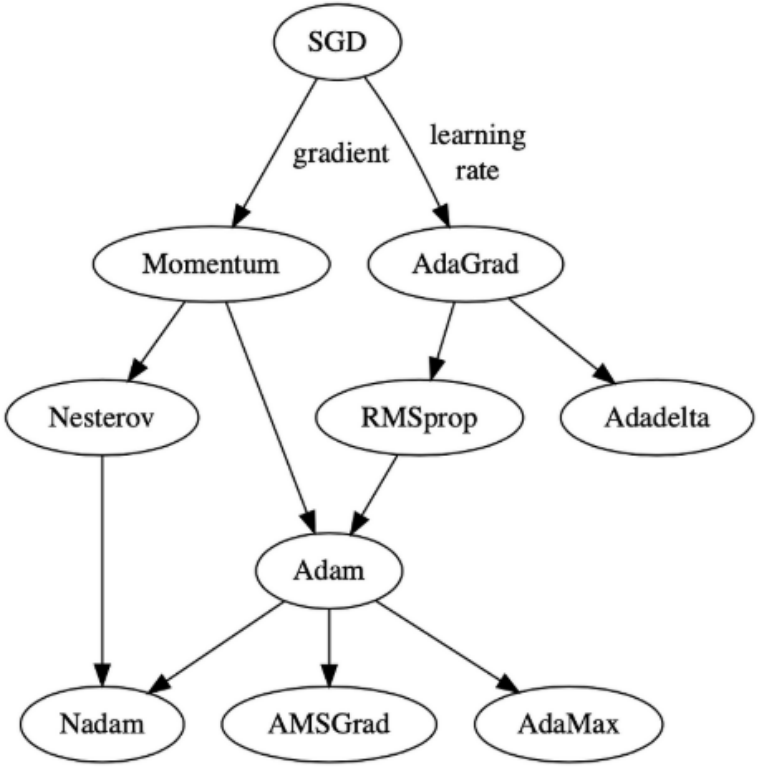
Momentum

Still not guarantee reaching global minima, but give some hope ……

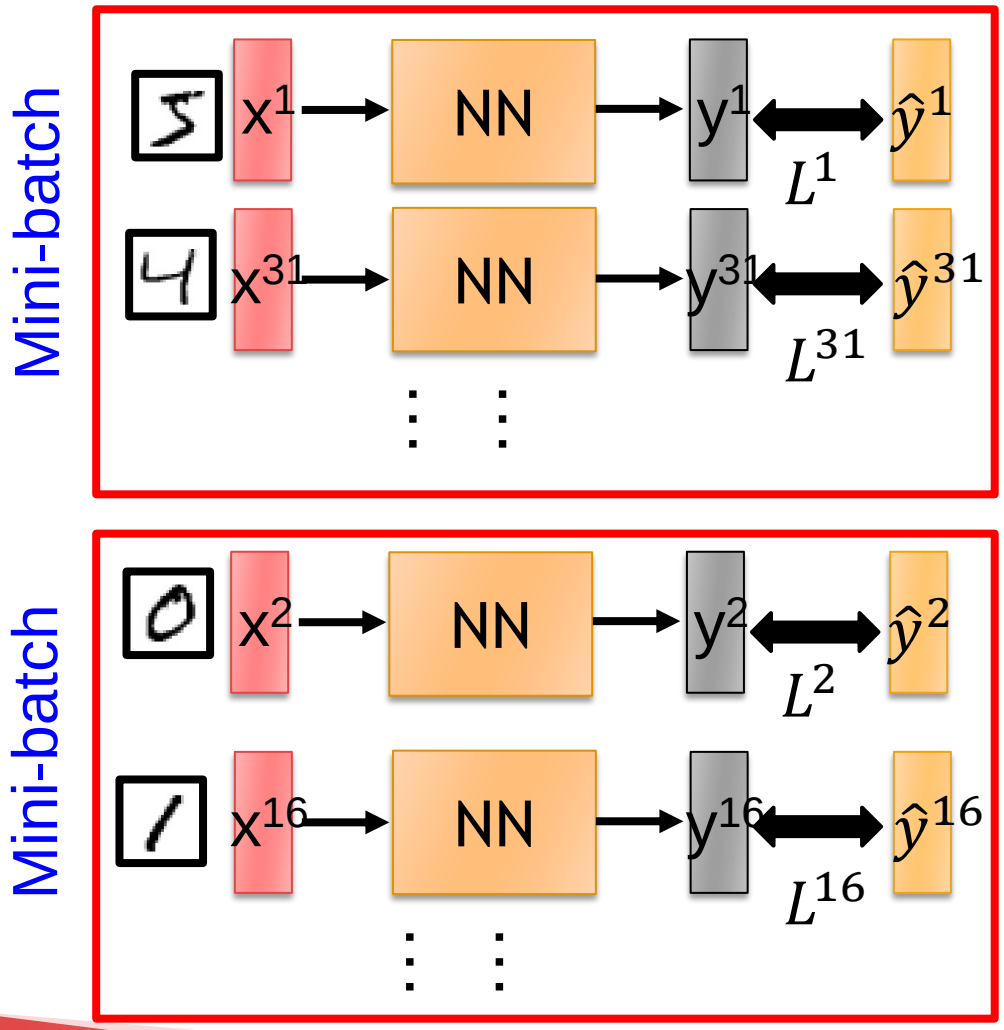


Gradient Descent Optimization

Optimiser	Year	Learning Rate	Gradient
Momentum	1964		✓
AdaGrad	2011	✓	
RMSprop	2012	✓	
Adadelta	2012	✓	
Nesterov	2013		✓
Adam	2014	✓	✓
AdaMax	2015	✓	✓
Nadam	2015	✓	✓
AMSGrad	2018	✓	✓



Mini-batch



- Randomly initialize θ^0
- Pick the 1st batch

$$C = L^1 + L^{31} + \cdots$$

$$\theta^1 \leftarrow \theta^0 - \eta \nabla C(\theta^0)$$

- Pick the 2nd batch

$$C = L^2 + L^{16} + \cdots$$

$$\theta^2 \leftarrow \theta^1 - \eta \nabla C(\theta^1)$$

⋮

C is different each time when we update parameters!

Mini-batch

Gradient Decent

학습데이터

full-batch

Stochastic Gradient Decent

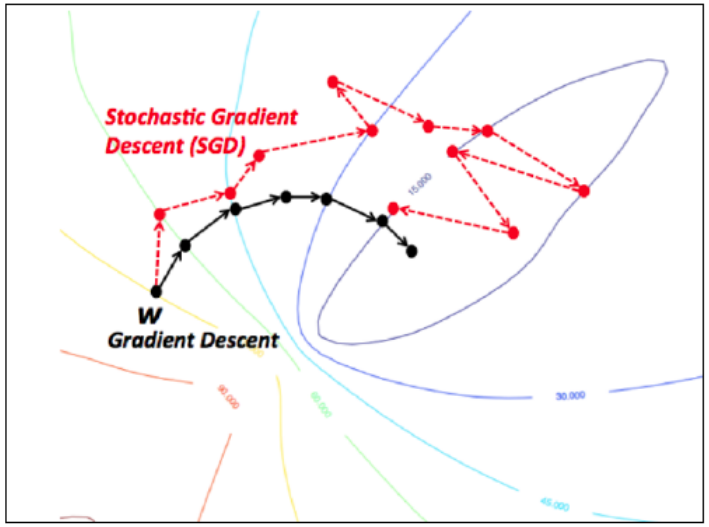
mini-batch

mini-batch

mini-batch

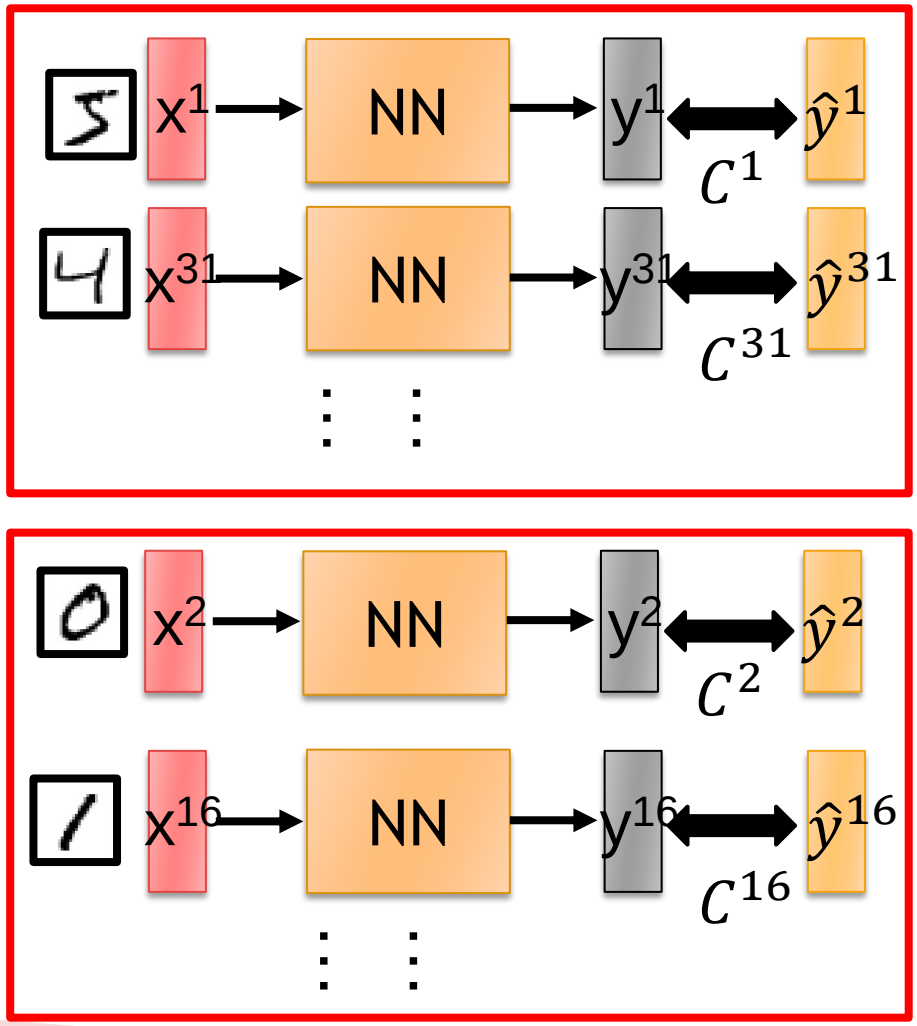
mini-batch

mini-batch



Mini-batch

Faster **Better!**



- Randomly initialize θ^0
- Pick the 1st batch

  $C = C^1 + C^{31} + \cdots$

  $\theta^1 \leftarrow \theta^0 - \eta \nabla C(\theta^0)$
- Pick the 2nd batch

  $C = C^2 + C^{16} + \cdots$

  $\theta^2 \leftarrow \theta^1 - \eta \nabla C(\theta^1)$

  ⋮
- Until all mini-batches have been picked

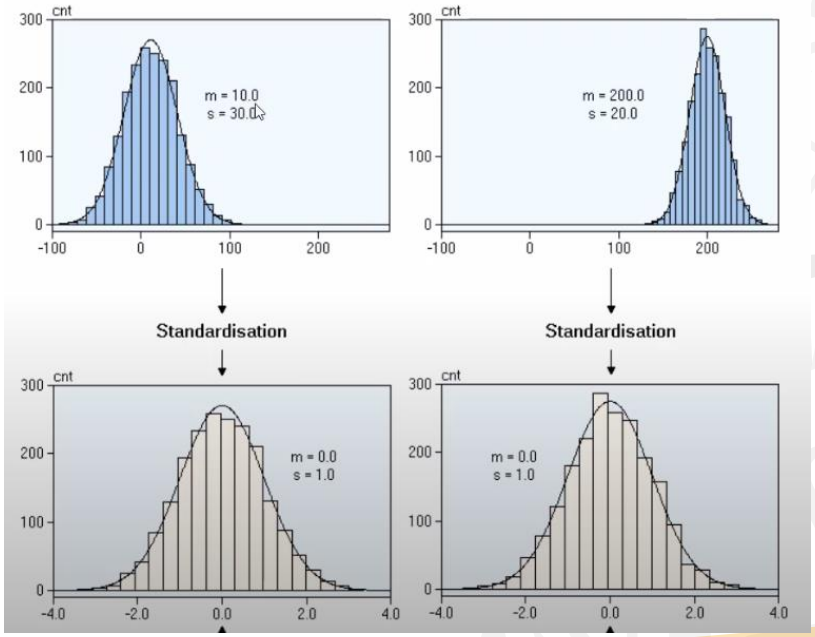
one epoch

Repeat the above process

Standardization

- 각 관찰값이 평균을 기준으로 어느정도 떨어져 있는지를 나타냄
  - 값의 scale이 다른 두 변수가 있을 때 scale 차이를 제거해주는 효과

$$x_{new} = \frac{x - \mu}{\sigma}$$



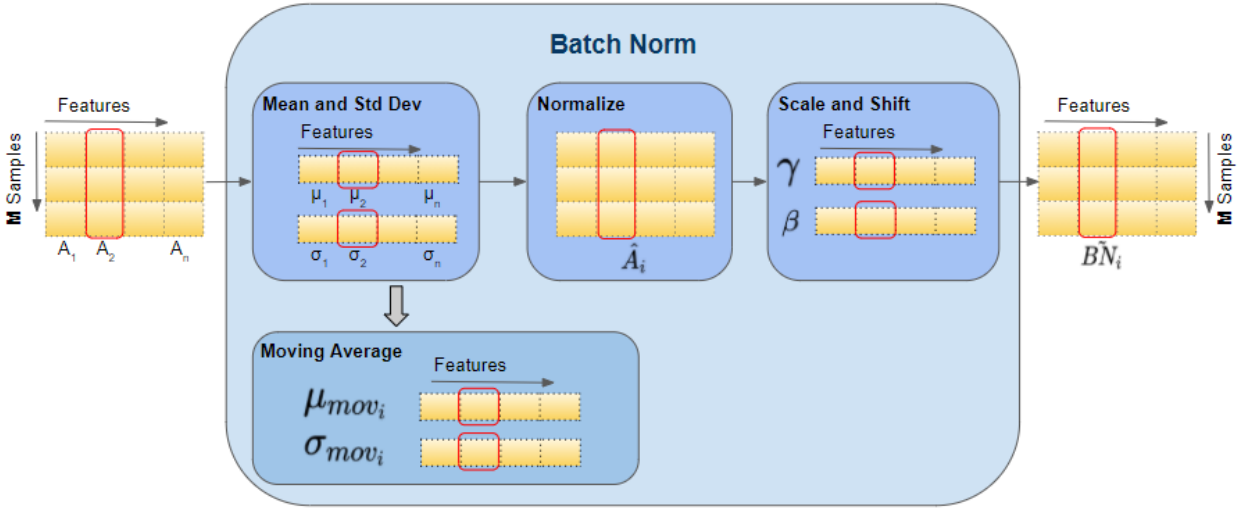
Normalization

- 정규화는 데이터의 범위를 0과 1로 변환하여 데이터 분포를 조정하는 방법
  - (해당 값- 최소값) / (최대값-최소값)
  - MinMax, Feature scaling 이라고도 함

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Batch Normalization

- To make training of neural networks more stable through normalization of the layers' inputs by re-centering and re-scaling (proposed in 2015)



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

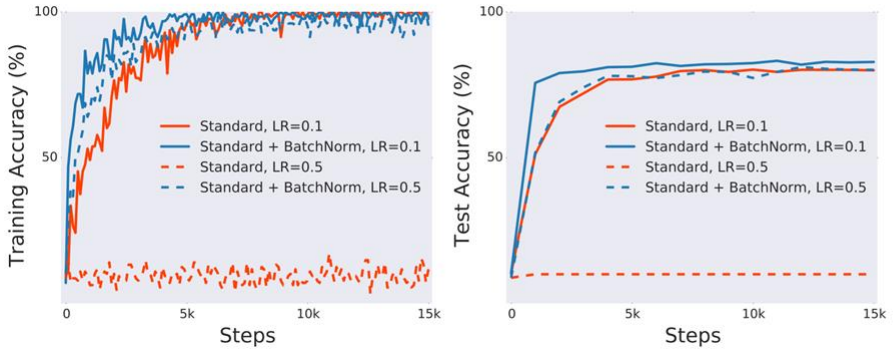
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

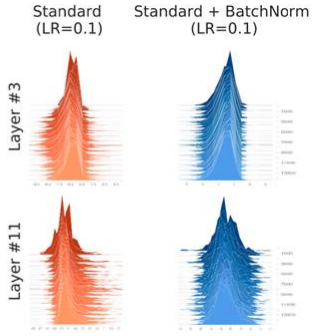
$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

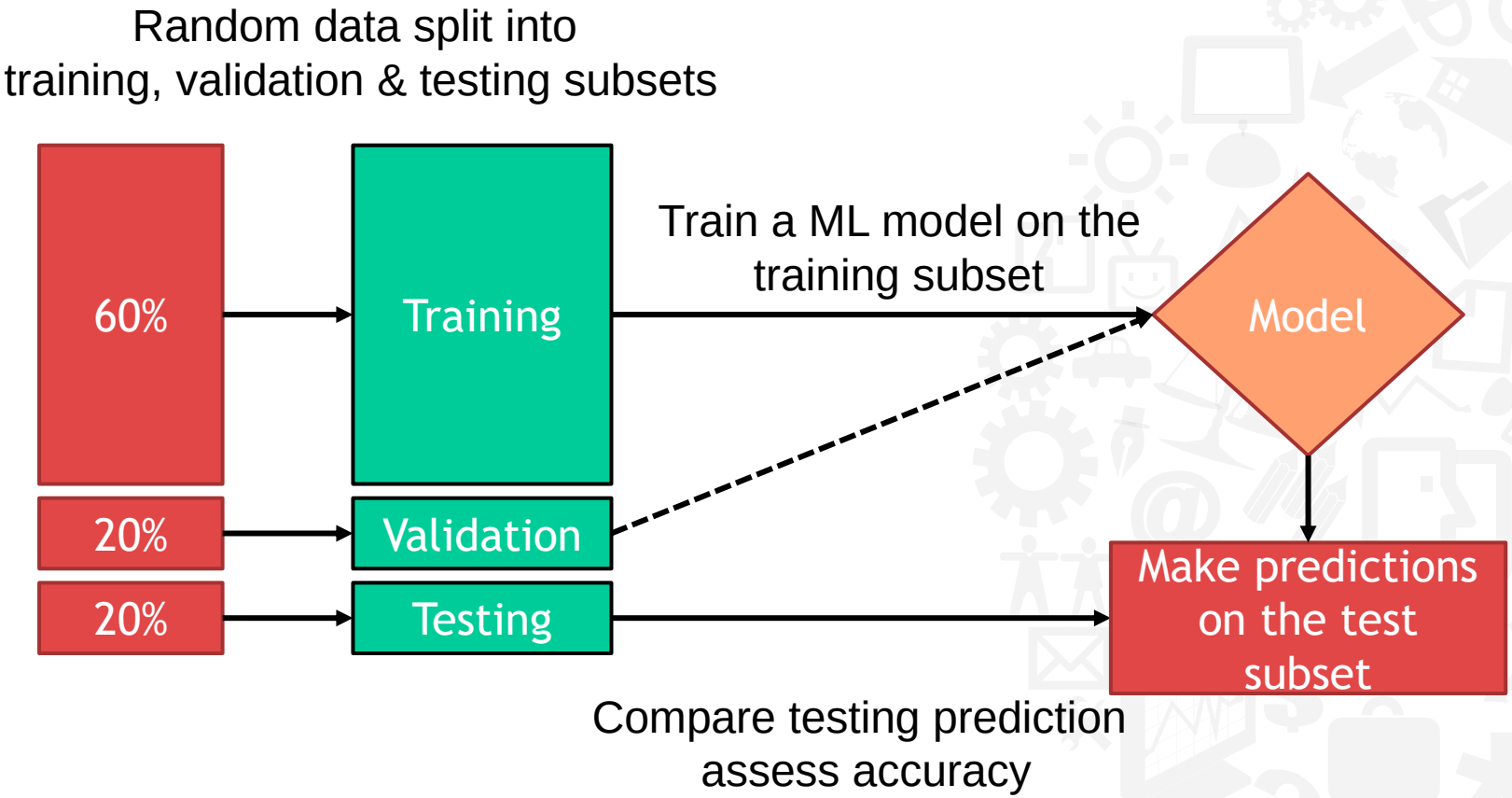
$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.



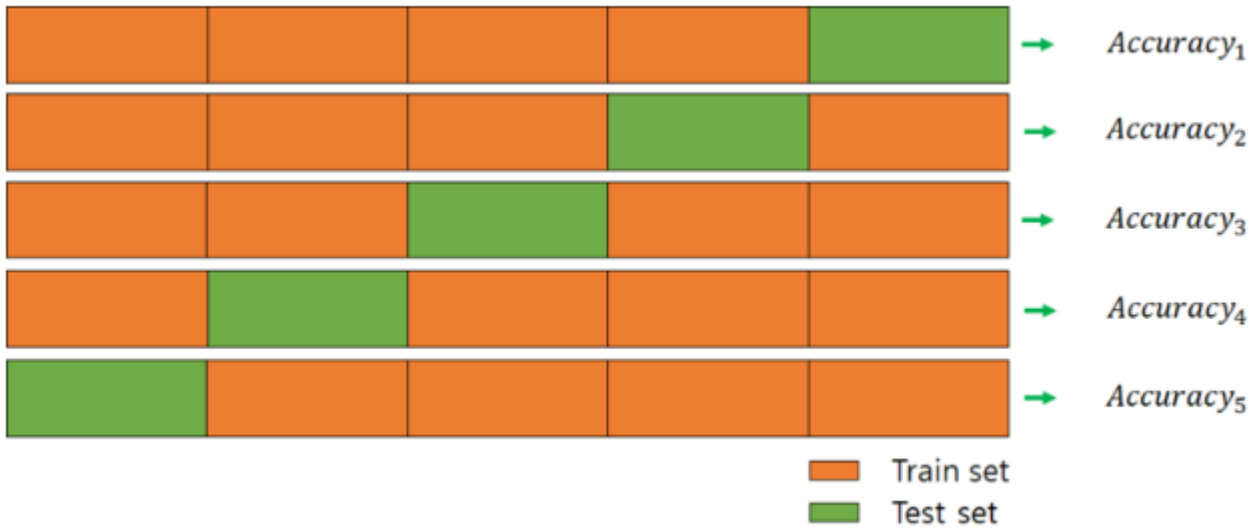


Training and testing splitting

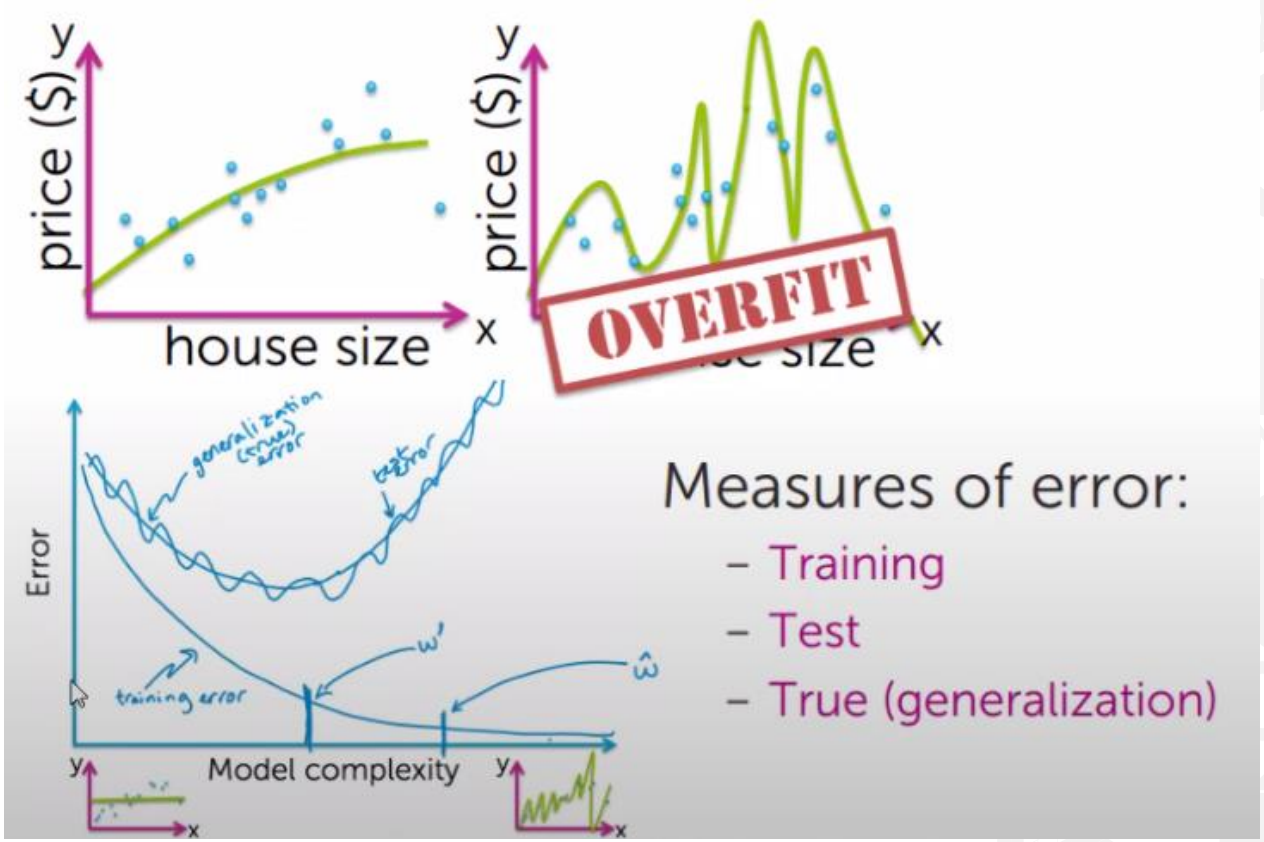


K-Folds Cross Validation

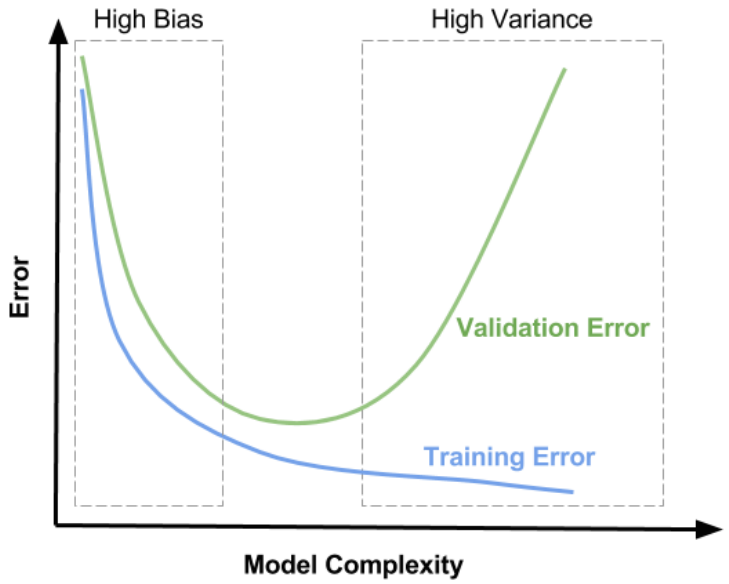
- 사용 이유
  - Test data 가 고정되어 있는 경우 test set에 overfit 될 수 있음
  - Dataset이 작은 경우 validation과 test set으로 데이터를 빼면 학습 데이터가 줄어드는 경우
-



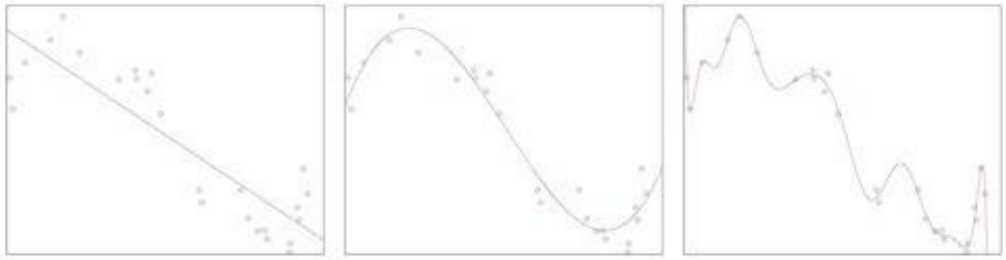
How do we determine the best fit line



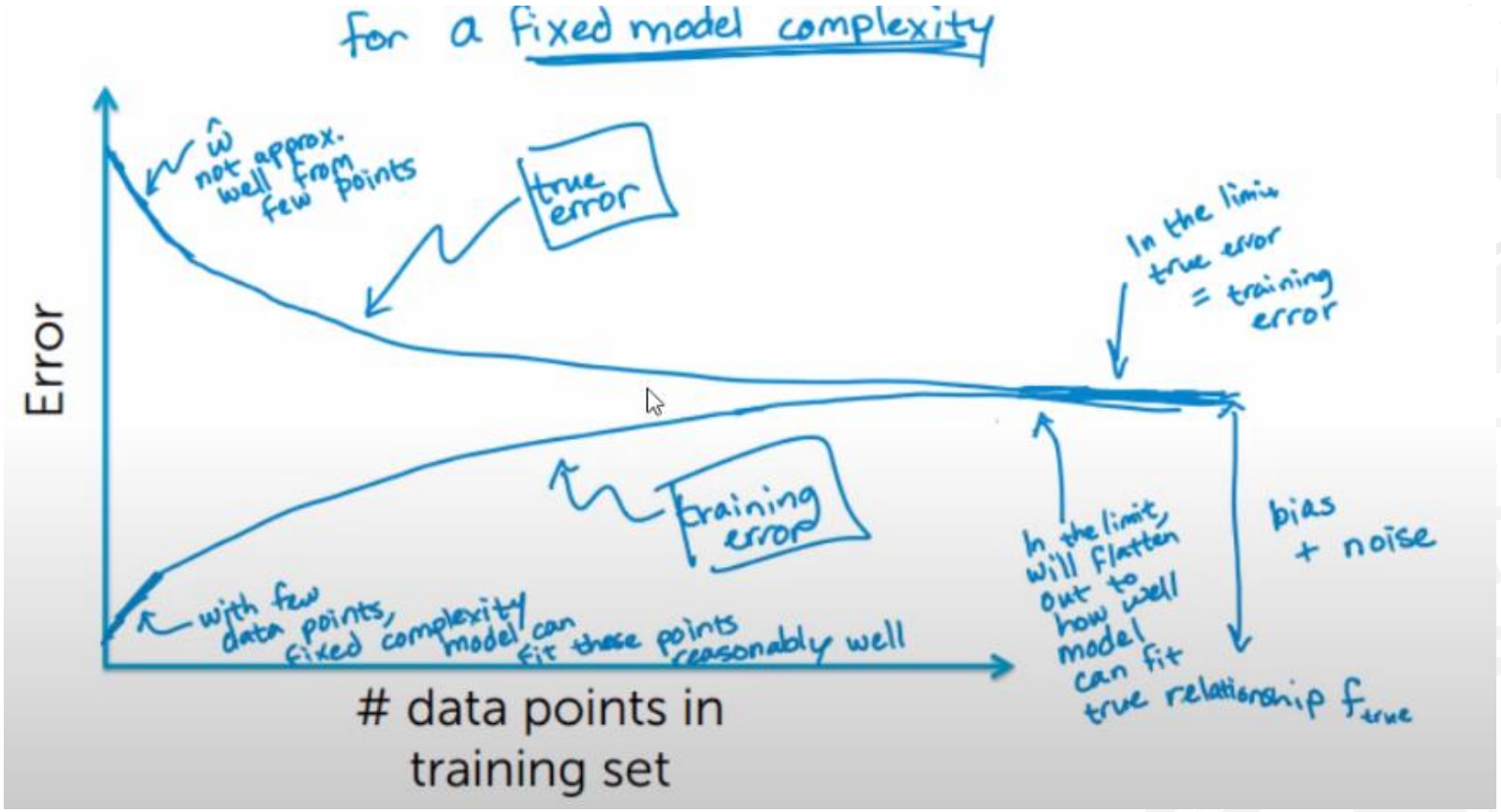
Bias-Variance tradeoff



Model complexity



Error vs. amount of data



Bias-Variance tradeoff

- Bias (underfitting)
  - 추정값의 평균과 참값의 차이
  - 데이터 내에 있는 모든 정보를 고려하지 않음으로 인해 발생
- Variance (Overfitting)
  - 추정값의 평균과 추정값들간의 차이
  - 데이터 내에 있는 error나 noise까지 잘 잡아내도록 model fitting하여, 실제 관계없는 것까지 학습하는 경향

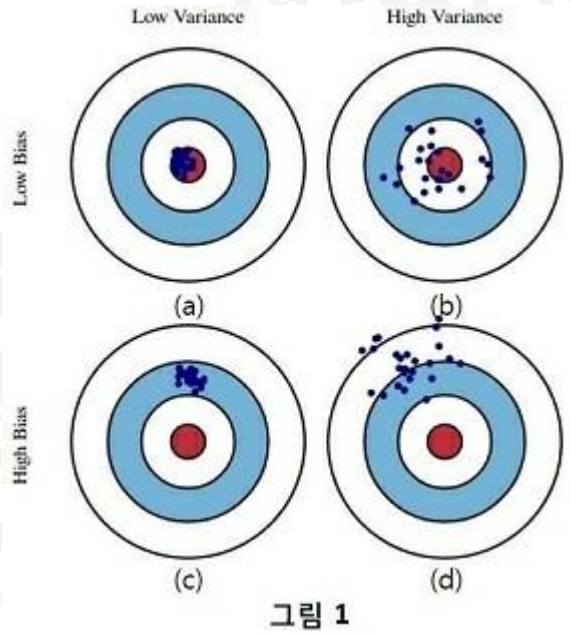


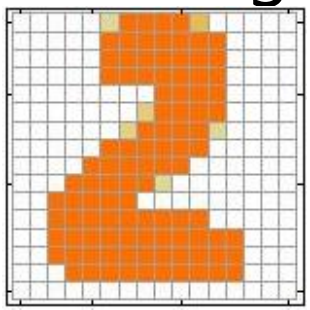
그림 1

LAB 1 : Housing Price Prediction

Classification Example

Deep learning example

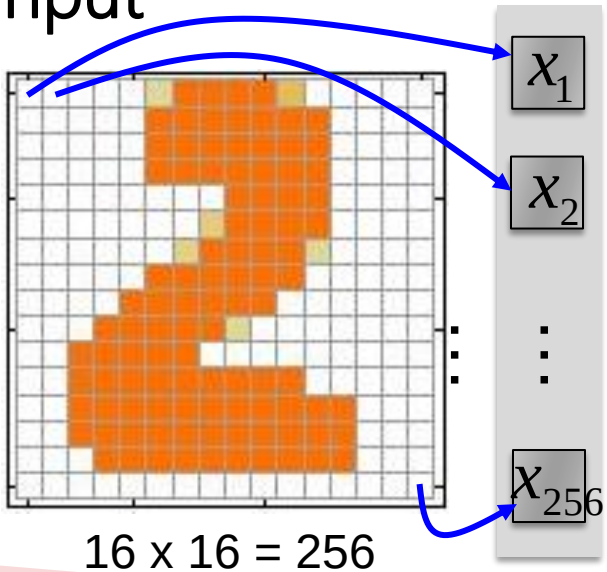
- Handwriting digit recognition



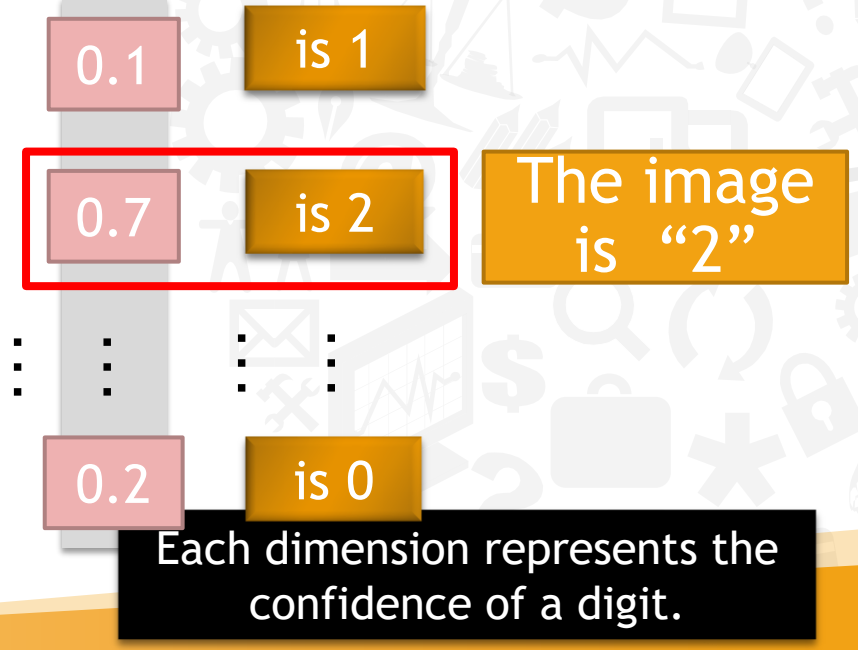
Machine

"2"

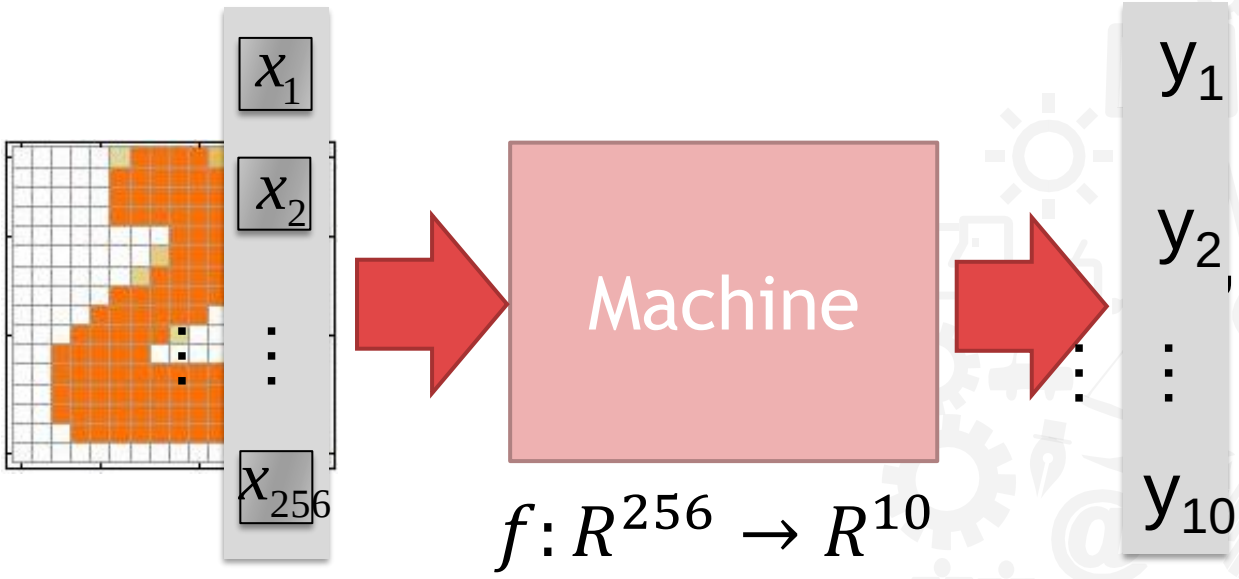
- Input



- Output



Handwriting digit recognition

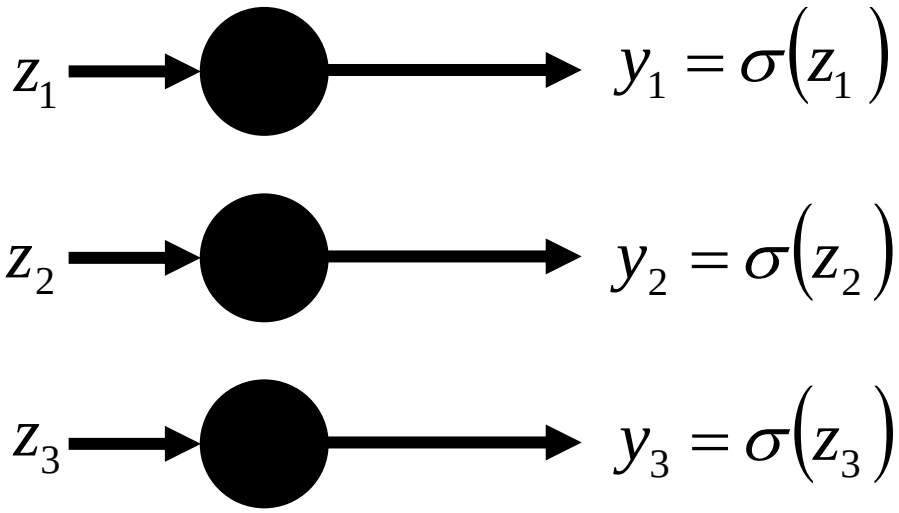


In deep learning, the function f is represented by neural network

Softmax

- Softmax layer as the output layer

Ordinary Layer



In general, the output of network can be any value

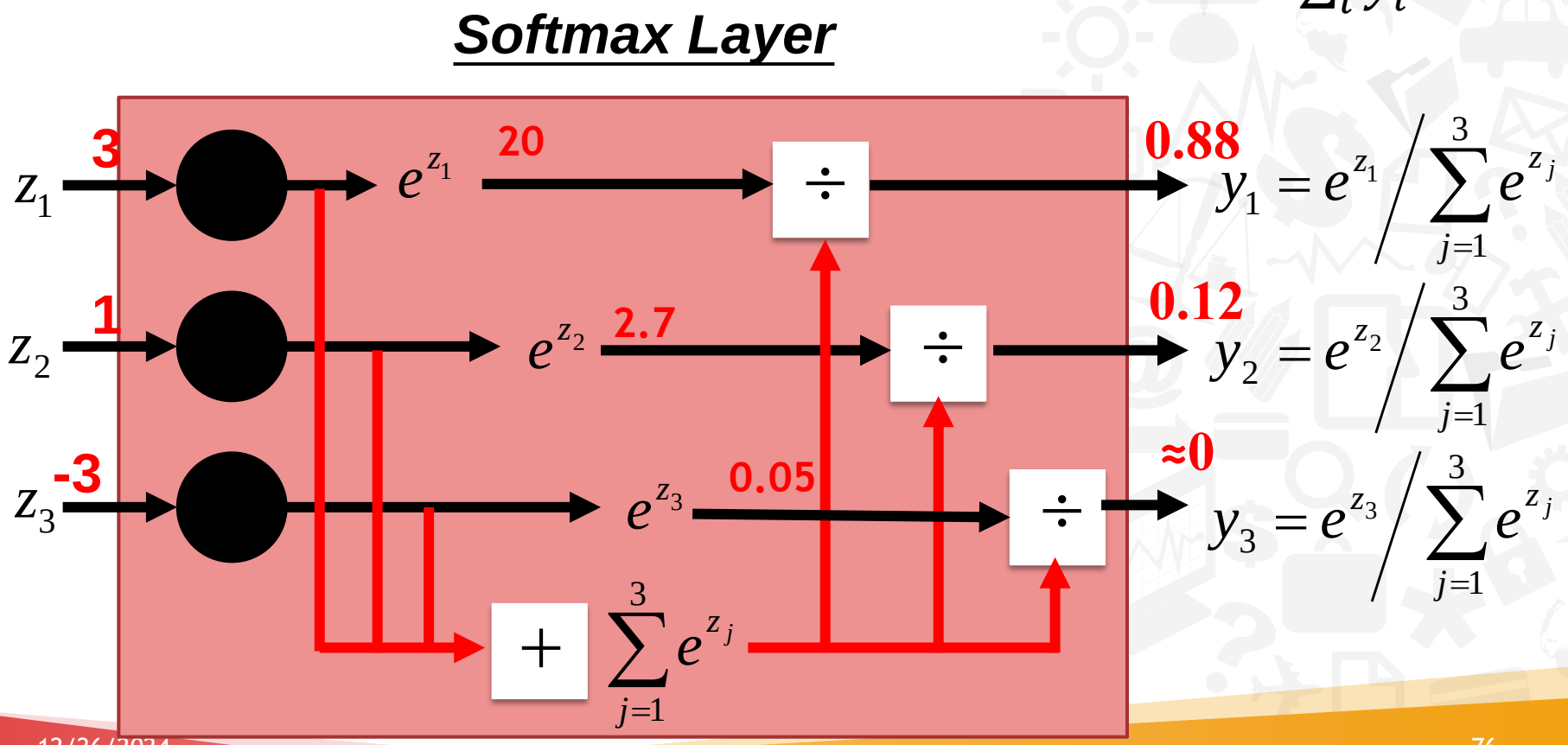
may not be easy to interpret

softmax

- Softmax as the output layer

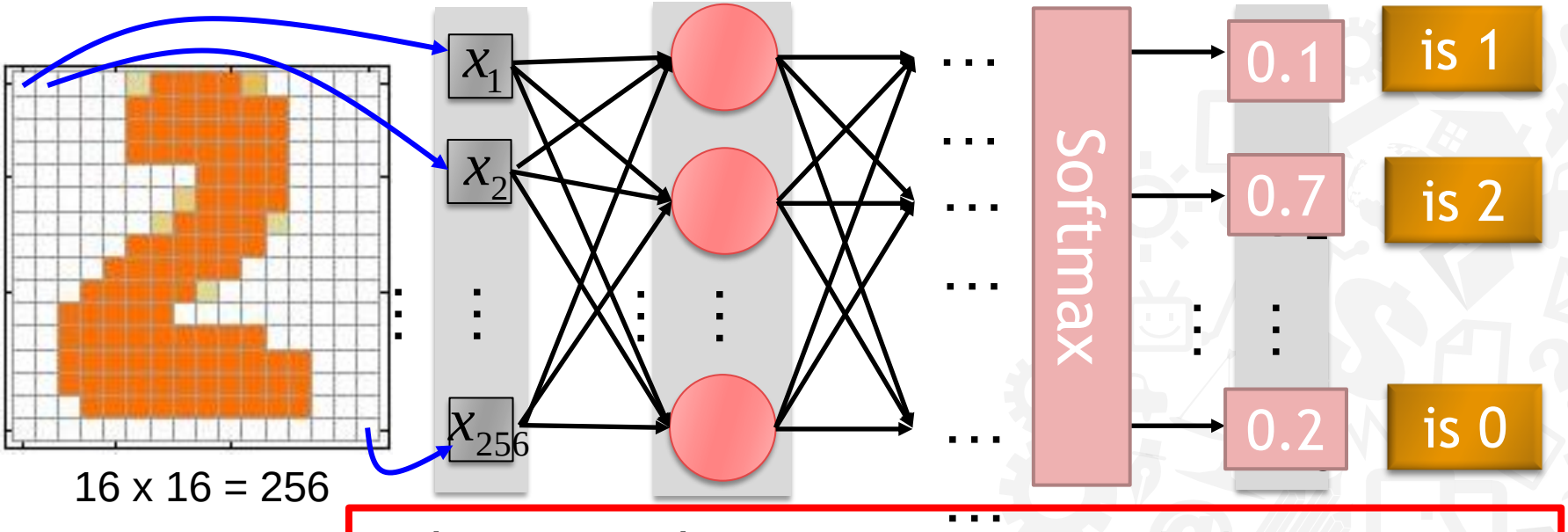
Probability:

- $1 > y_i > 0$
- $\sum_i y_i = 1$



How to set network parameters

$$\theta = \{W^1, b^1, W^2, b^2, \cdots W^L, b^L\}$$



16 x 16 = 256

Ink → 1

No ink → 0

Set the network parameters θ such that ……

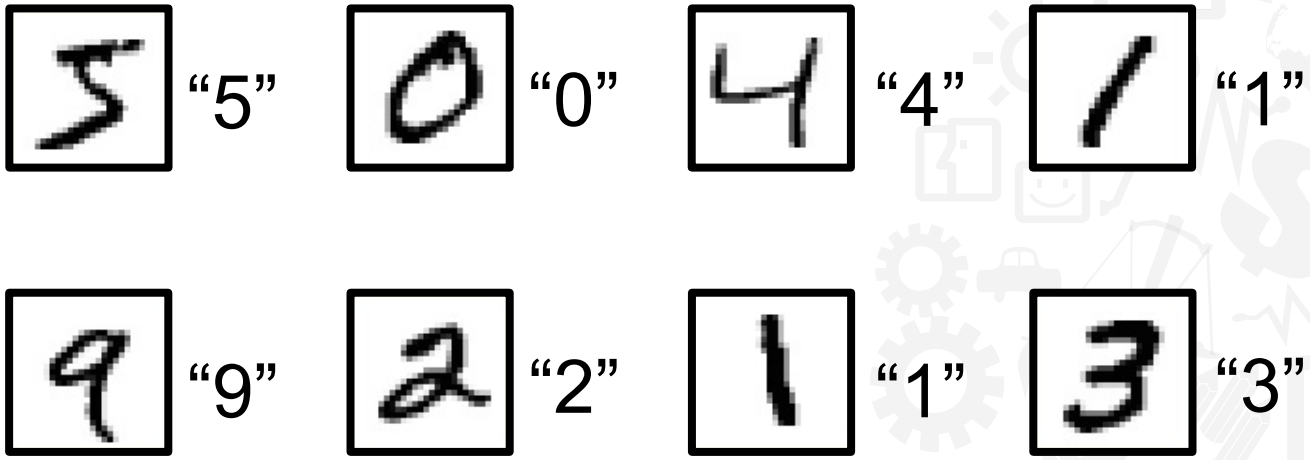
Input: y_1 has the maximum value

Input: y_2 has the maximum value

How to let the neural network achieve this

Training Data

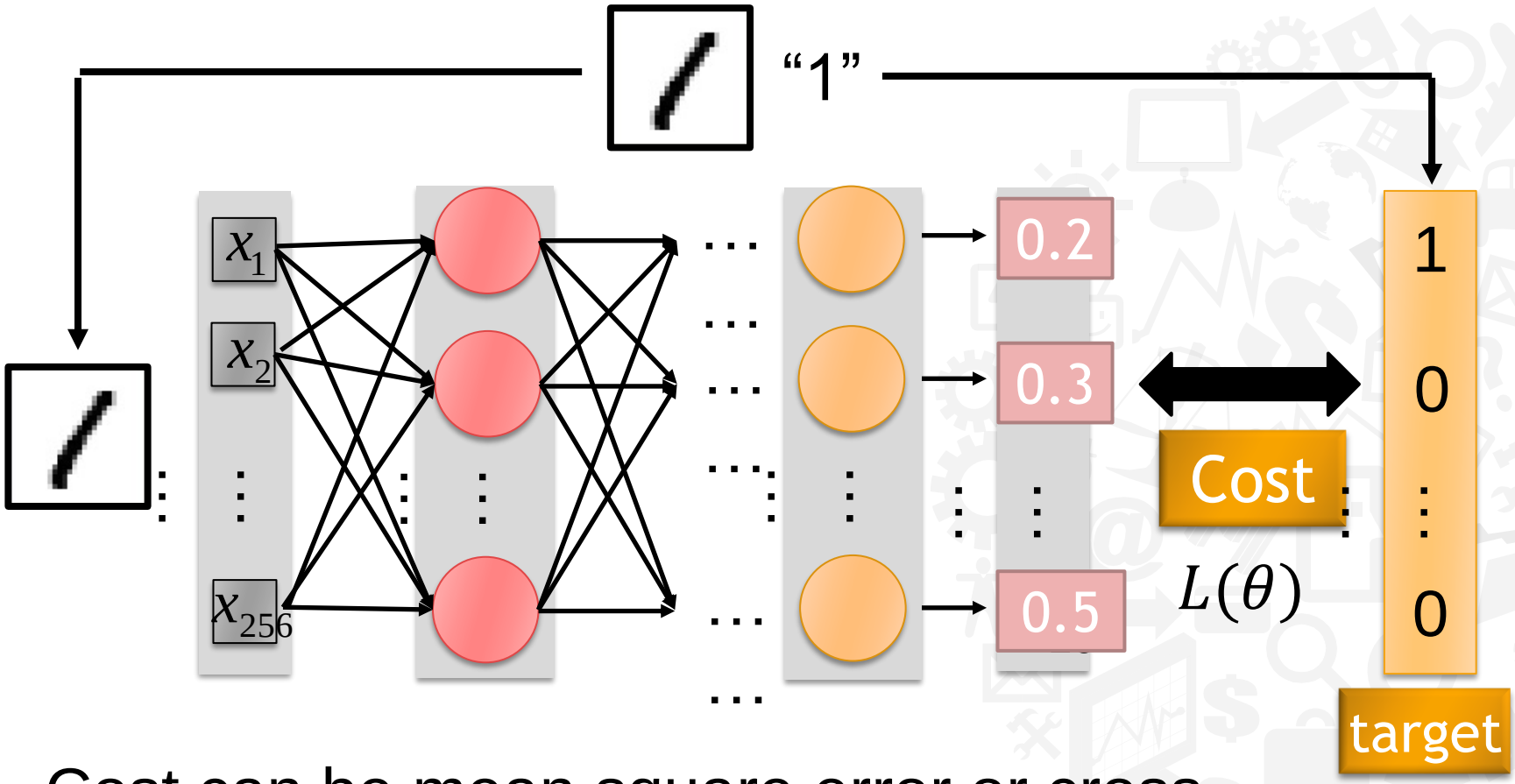
- Preparing training data: images and their labels



Using the training data to find the network parameters.

Cost

Given a set of network parameters θ, each example has a cost value.

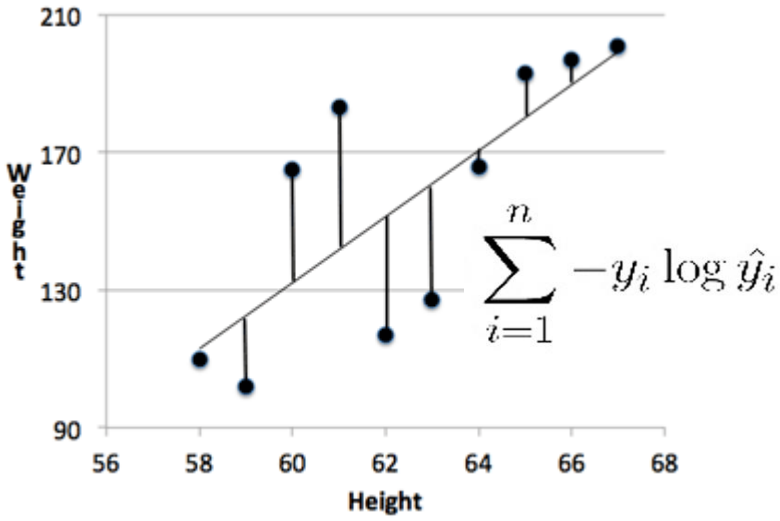


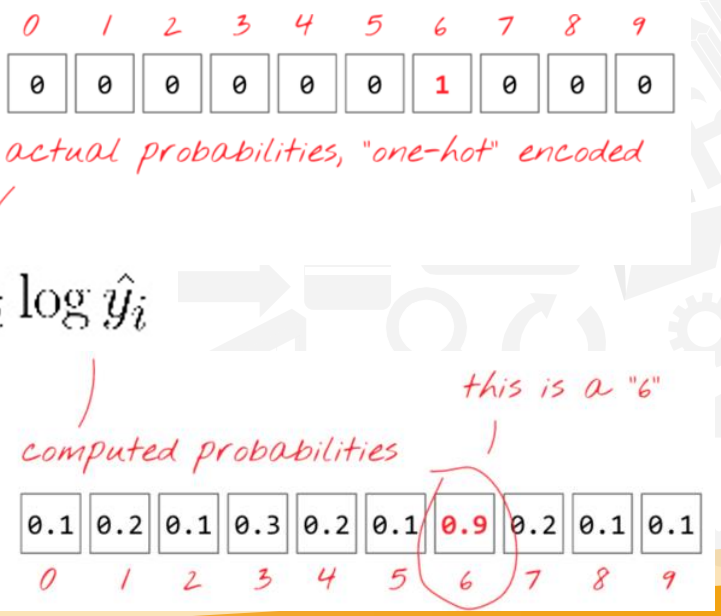
Cost can be mean square error or cross-entropy of the network output and target

Cost function

Mean Squared Error: $$\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

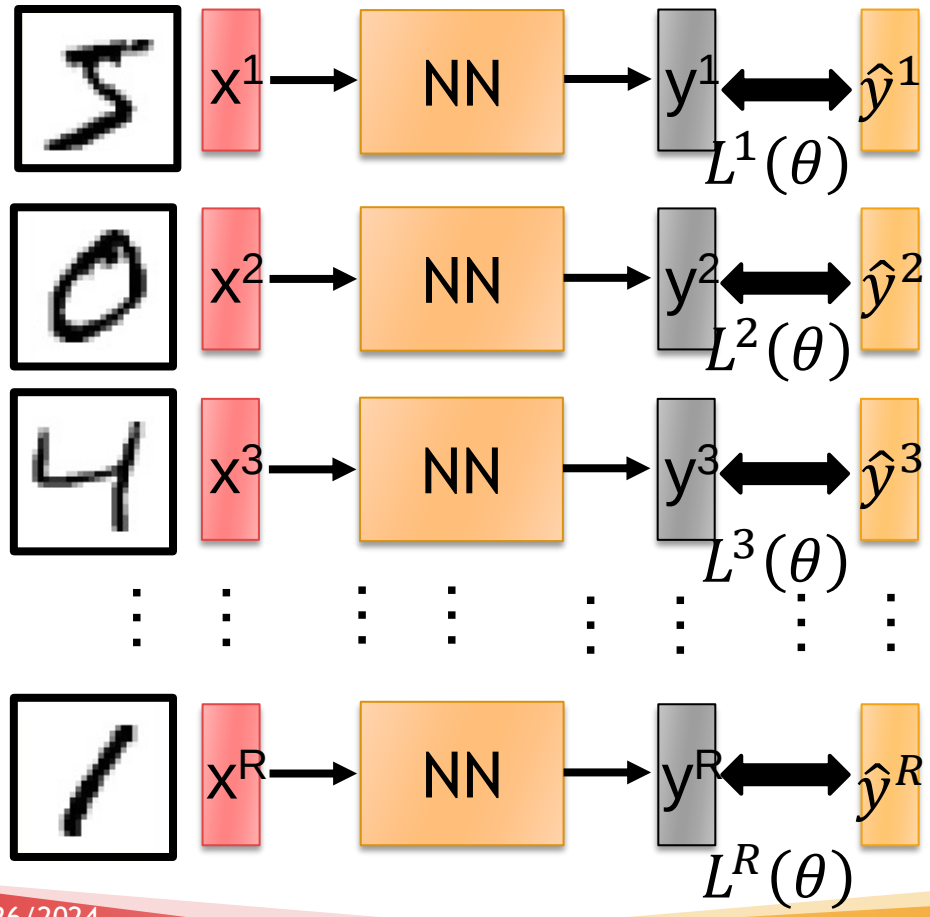
Cross Entropy: $$\mathcal{L}(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_{i=1}^{n} -y_i \log \hat{y}_i$$





Total Cost

For all training data …



Total Cost:

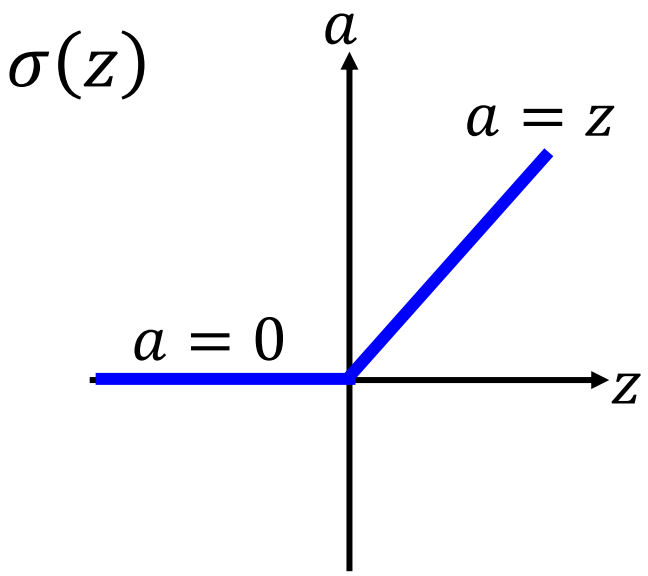
$$C(\theta) = \sum_{r=1}^{R} L^r(\theta)$$

How bad the network parameters θ is on this task

Find the network parameters θ^* that minimize this value

RELU

- Rectified Linear Unit(ReLU)

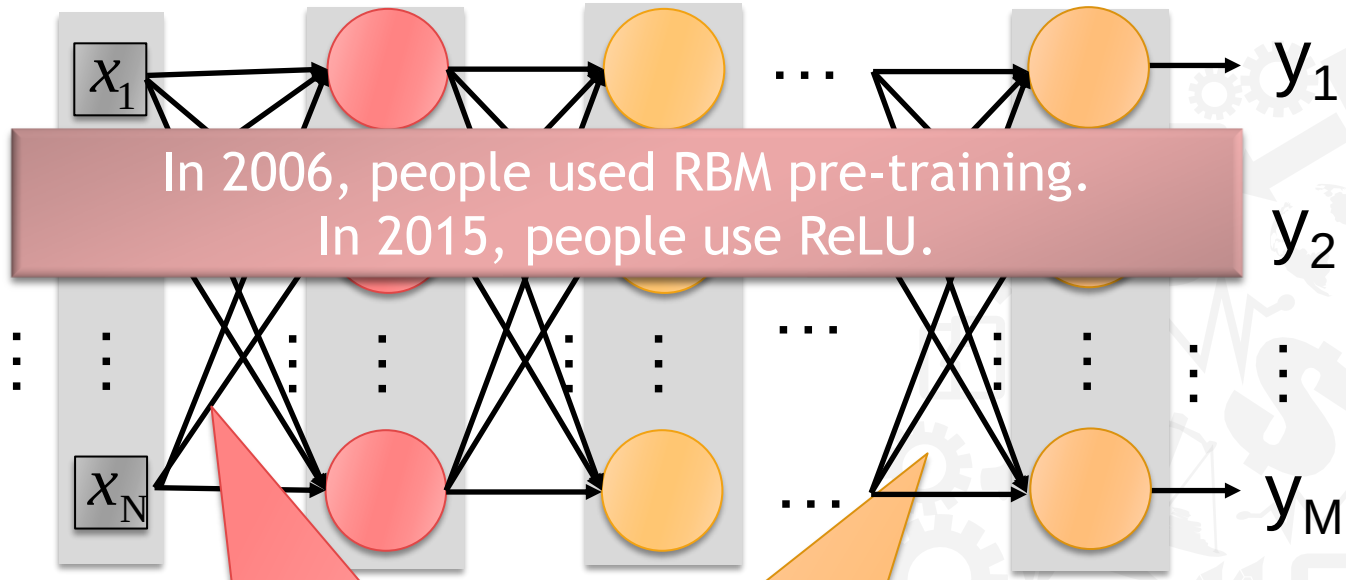


[Xavier Glorot, AISTATS'11]
[Andrew L. Maas, ICML'13]
[Kaiming He, arXiv'15]

Reason:

1. Fast to compute
2. Reducing Vanishing gradient problem

Vanishing gradient problem



Smaller gradients

Learn very slow

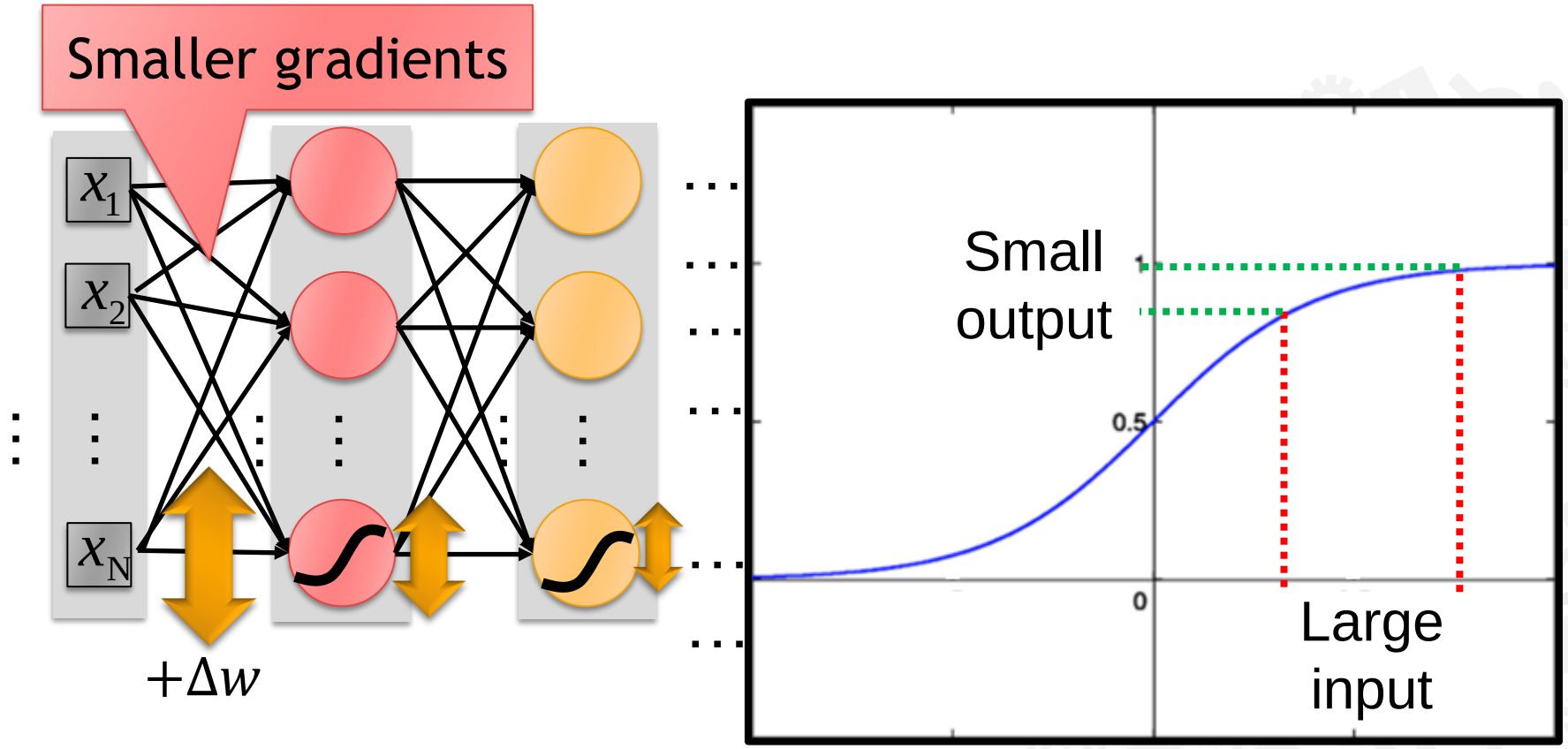
Almost random

Larger gradients

Learn very fast

Already converge

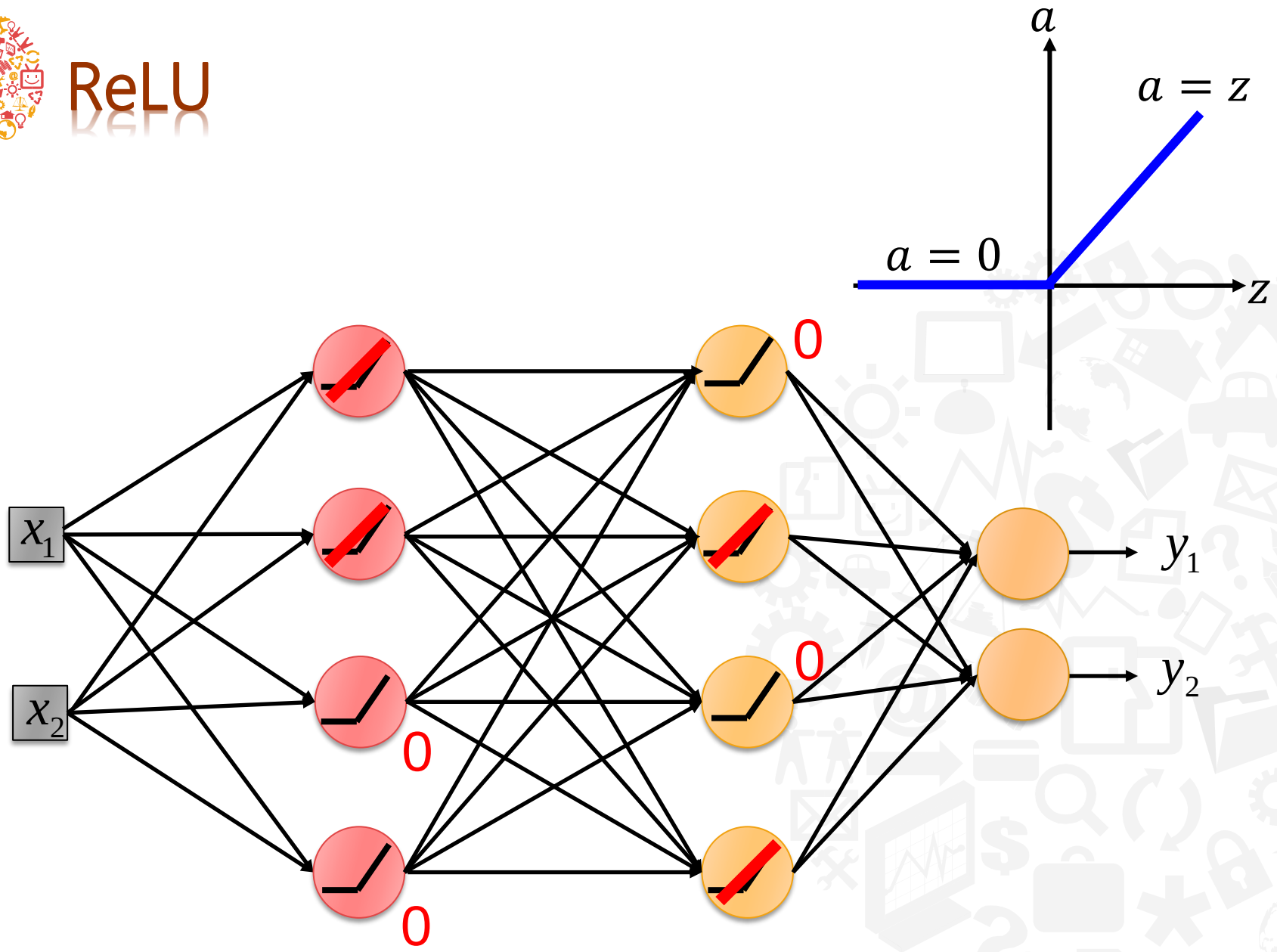
Vanishing gradient problem



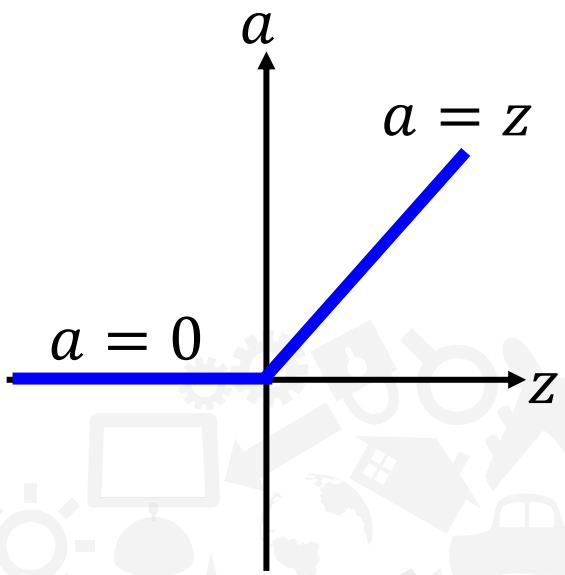
Intuitive way to compute the gradient …

$$\frac{\partial C}{\partial w} =? \frac{\Delta C}{\Delta w}$$

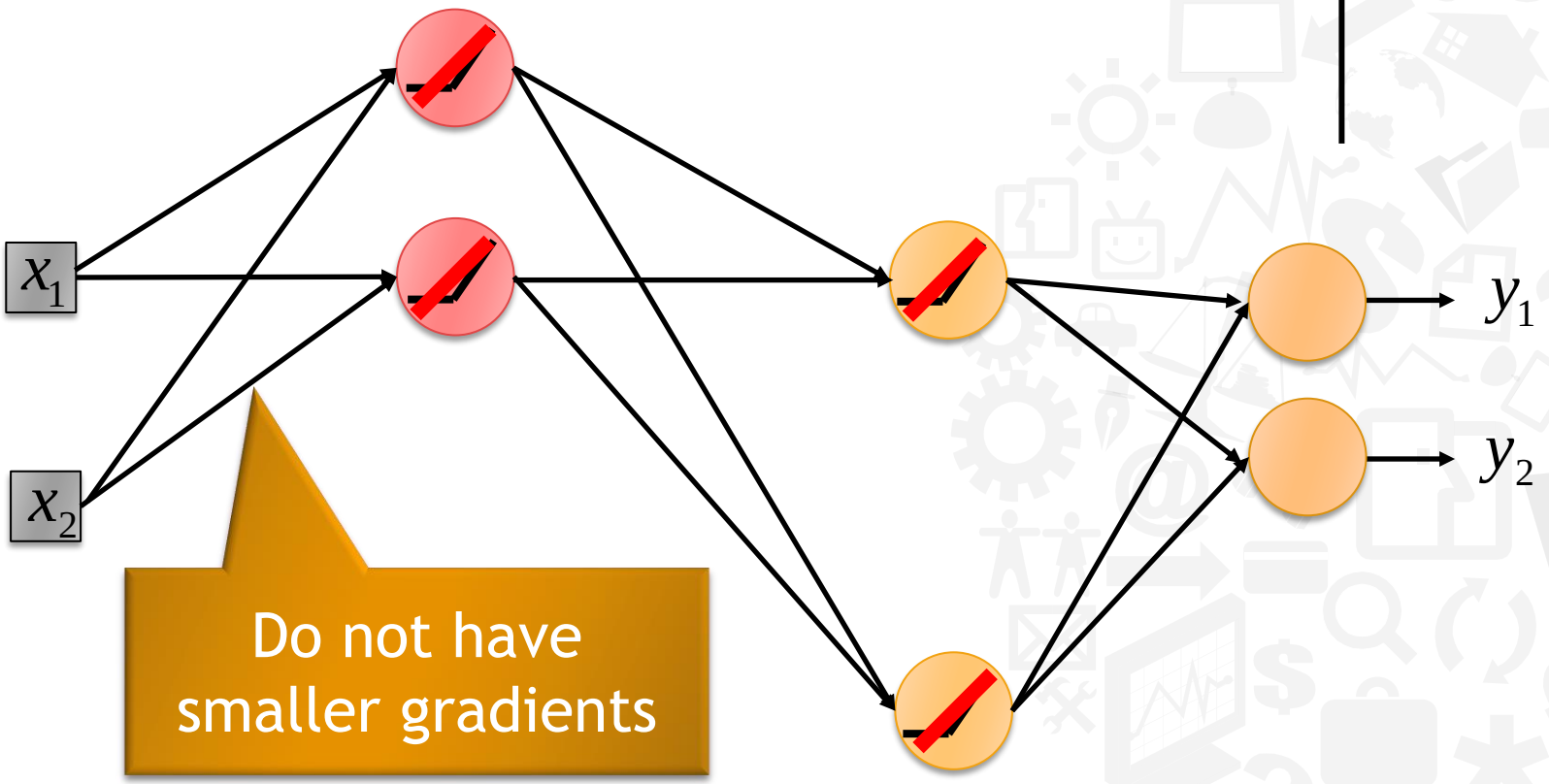
ReLU



ReLU



A Thinner linear network



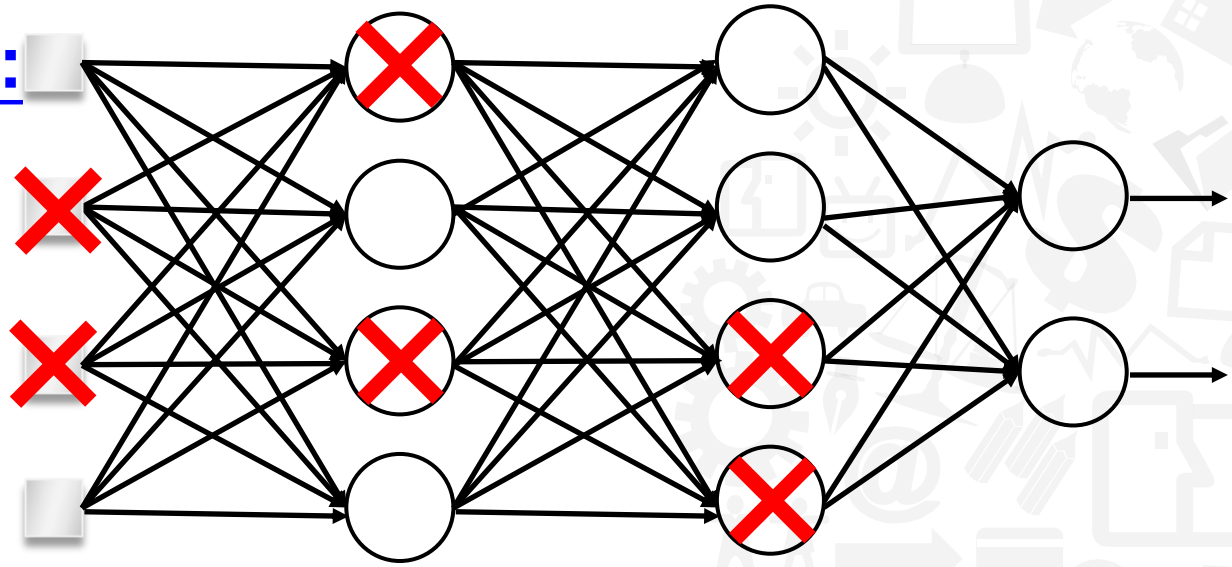
Do not have smaller gradients

DropOut

Pick a mini-batch

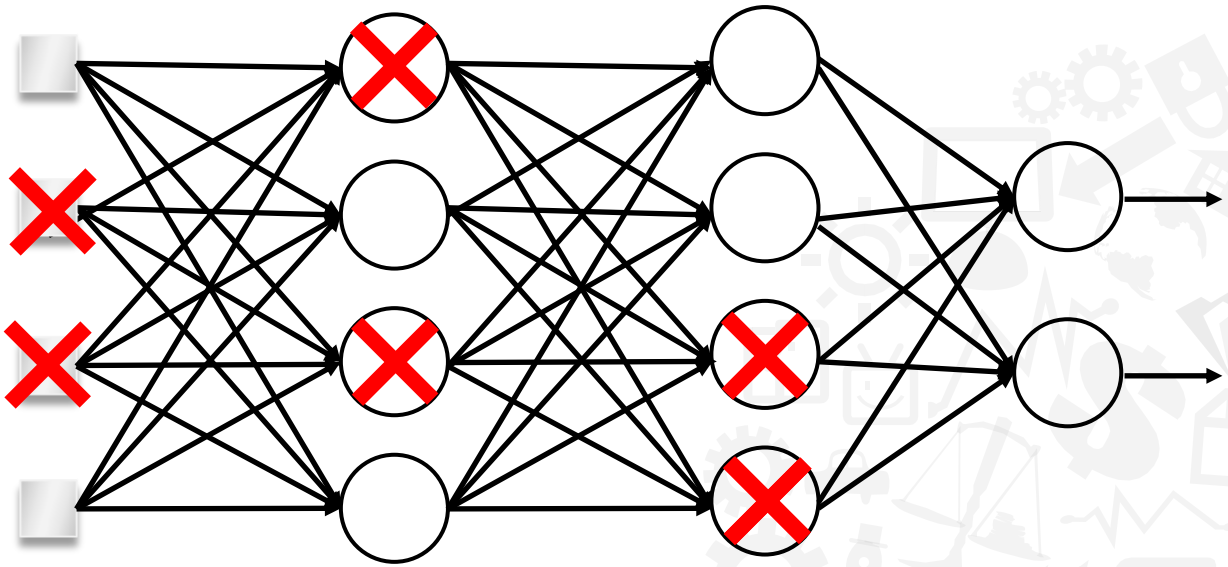
$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C(\theta^{t-1})$$

Training:



- **Each time before computing the gradients**
  - Each neuron has p% to dropout

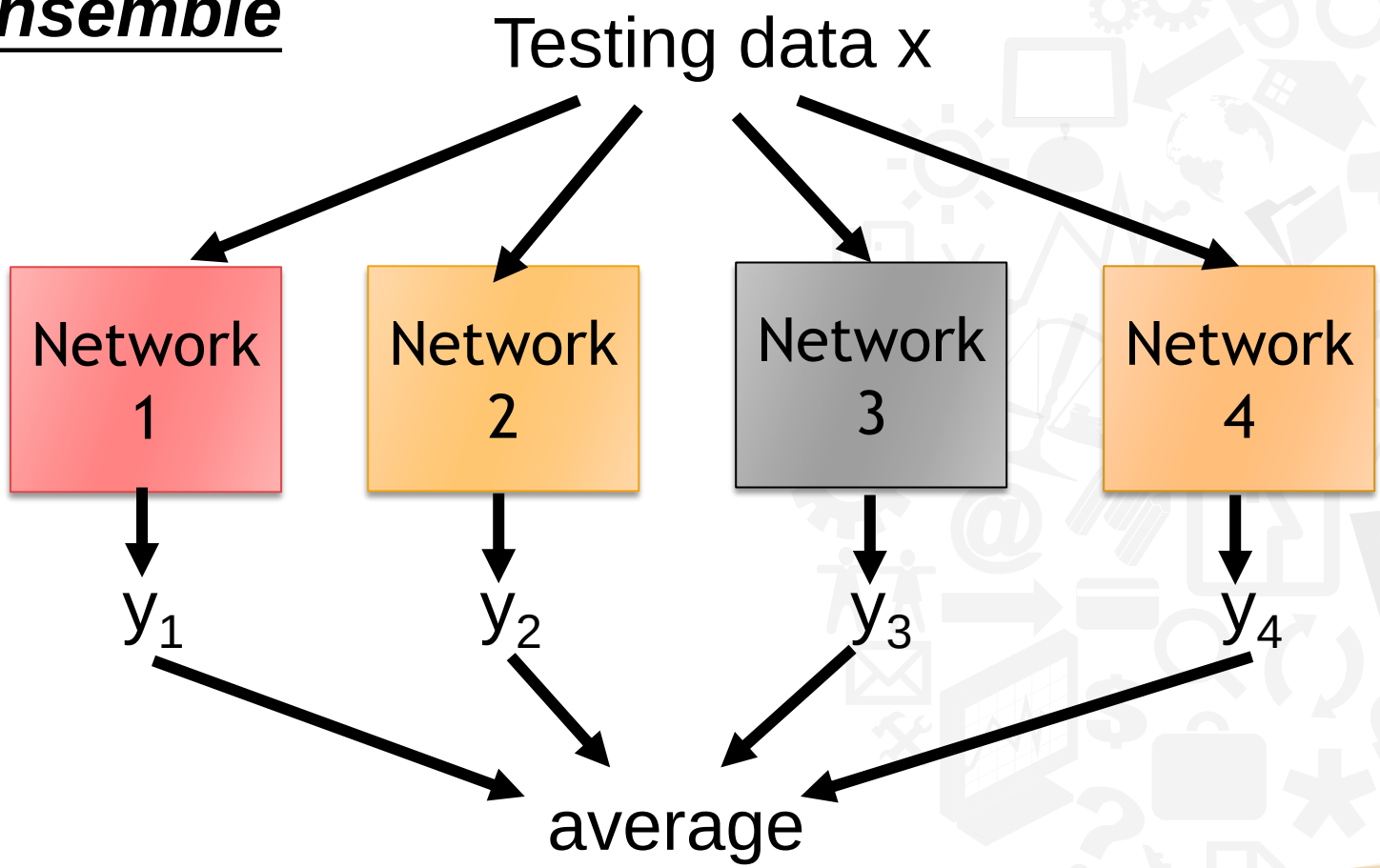
DropOut



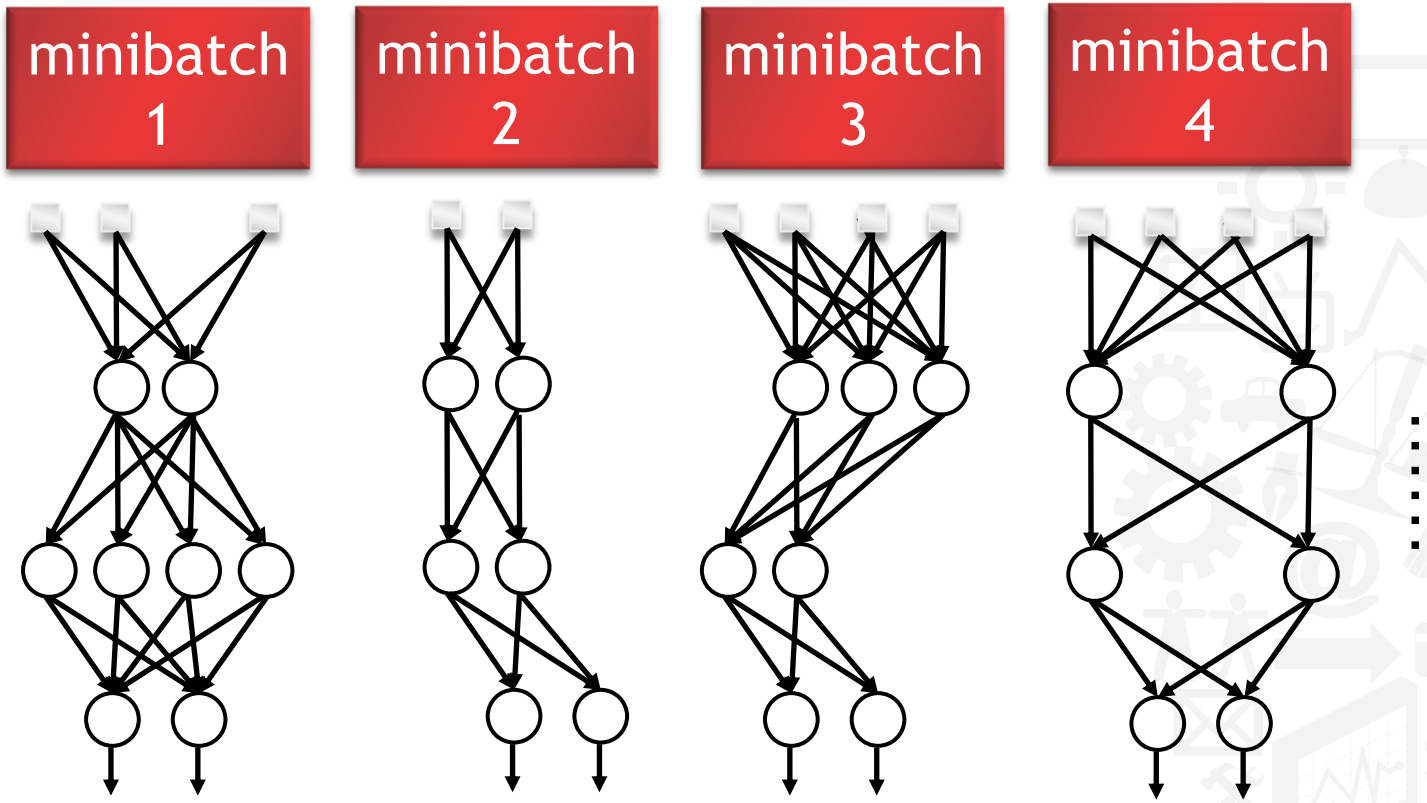
- When teams up, if everyone expect the partner will do the work, nothing will be done finally.
- However, if you know your partner will dropout, you will do better.
- When testing, no one dropout actually, so obtaining good results eventually.

DropOut is a kind of ensemble

Model Ensemble



DropOut is a kind of ensemble



Training of Dropout

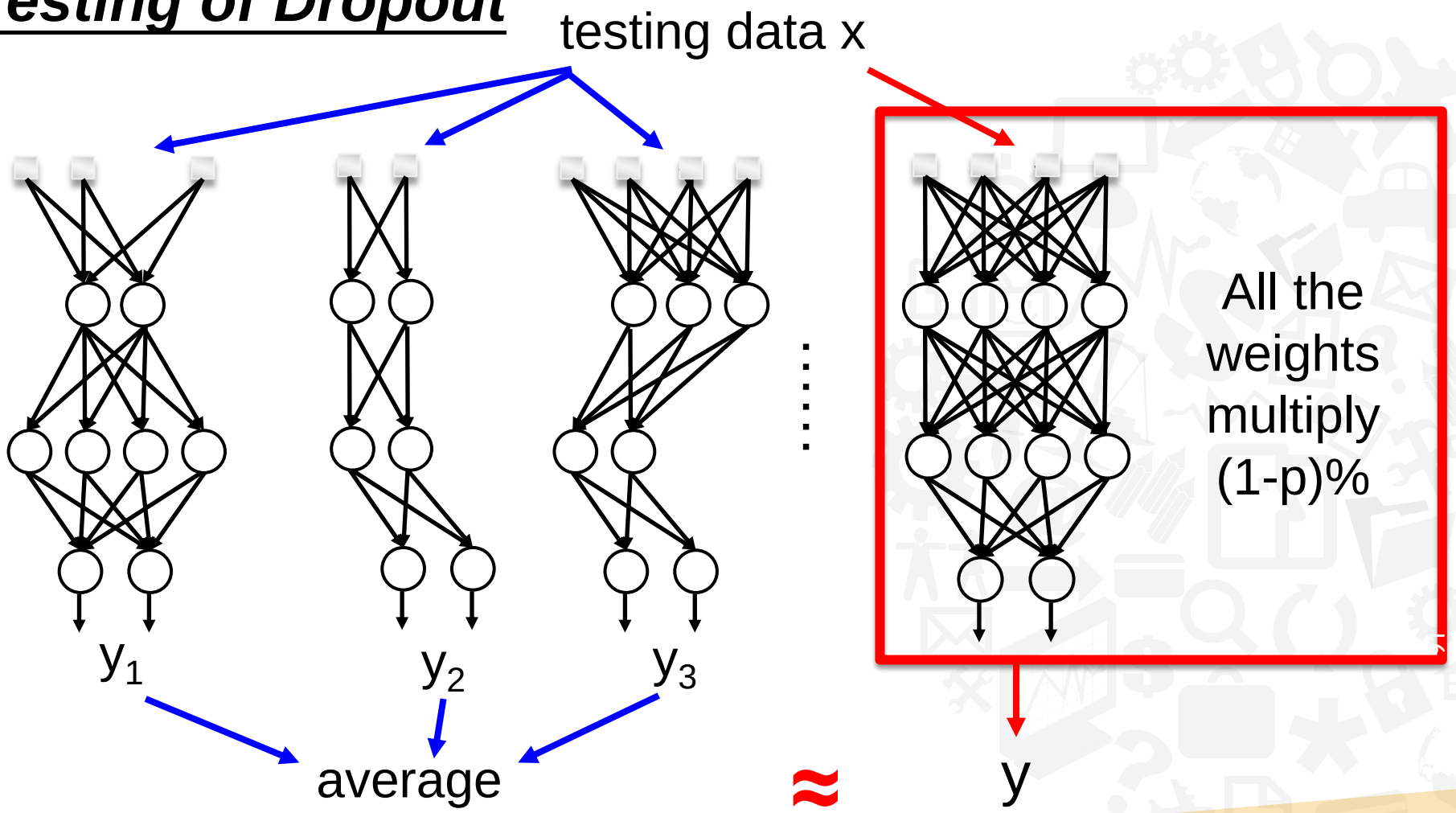
M neurons

2^M possible networks

- Using one mini-batch to train one network
- Some parameters in the network are shared

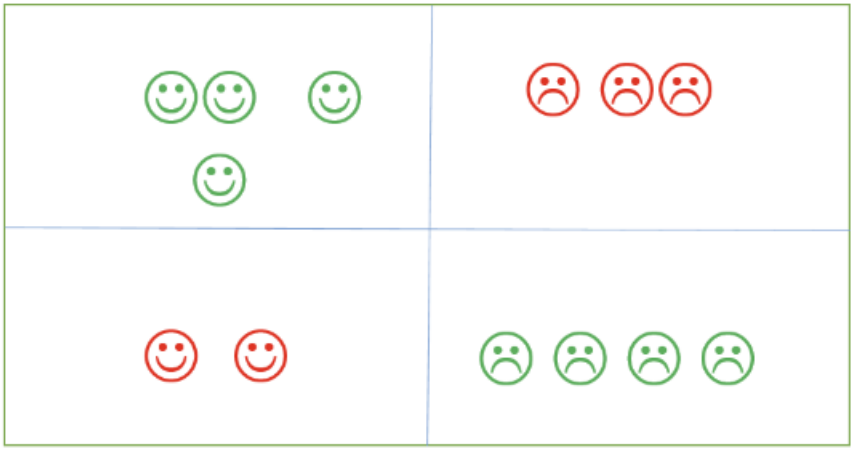
Dropout is a kind of ensemble.

Testing of Dropout



Metrics

- Precision
  - TP/(TP+FP)
- Recall
  - TP/(TP+FN)
- Accuracy
  - $\frac{TP+TN}{TP+TN+FP+FN}$
- F1 Score
  - $\frac{2 \times precision \times Recall}{precision+recall}$



Predicted

	Positive	Negative	
Positive	True positive(TP)	False Negative(FN)	Sensitivity or Recall or True Positive Rate=TP/(TP+FN)
Negative	False Positive (FP)	True Negative(TN)	Specificity or True Negative Rate=TN/(TN+FP)
	Precision or Positive Predictive Value=TP/(TP+FP)	Negative Predictive Value=FN/(FN+TN)	Accuracy=TP+TN/TP+TN+FP+FN

Actual

Metrics

- Assume that we are classifying an email into one of the three groups: urgent, normal and spam

		gold labels urgent	normal	spam	
system output	urgent	8	10	1	$\text{precision}_u = \frac{8}{8+10+1}$
	normal	5	60	50	$\text{precision}_n = \frac{60}{5+60+50}$
	spam	3	30	200	$\text{precision}_s = \frac{200}{3+30+200}$
		$\text{recall}_u = \frac{8}{8+5+3}$	$\text{recall}_n = \frac{60}{10+60+30}$	$\text{recall}_s = \frac{200}{1+50+200}$	

- How can we derive a single metric that tells us how well the system is doing?

Macro vs. Micro

- **Macro average**
  - we compute the performance for each class, and then average over classes.
- **Micro average**
  - we collect the decisions for all classes into a single confusion matrix, and then compute precision and recall from that table.

Class 1: Urgent

	true urgent	true not
system urgent	8	11
system not	8	340

$$\text{precision} = \frac{8}{8+11} = .42$$

Class 2: Normal

	true normal	true not
system normal	60	55
system not	40	212

$$\text{precision} = \frac{60}{60+55} = .52$$

Class 3: Spam

	true spam	true not
system spam	200	33
system not	51	83

$$\text{precision} = \frac{200}{200+33} = .86$$

Pooled

	true yes	true no
system yes	268	99
system no	99	635

$$\text{microaverage precision} = \frac{268}{268+99} = \mathbf{.73}$$

$$\text{macroaverage precision} = \frac{.42+.52+.86}{3} = \mathbf{.60}$$

Weighted Average

- Proportional Average that each score is multiplied by proportion of the class in your dataset

Label	Per-Class F1 Score	Support	Support Proportion	Weighted Average F1 Score
 Airplane	0.67	3	0.3	$(0.67 * 0.3) + (0.40 * 0.1) + (0.67 * 0.6)$ **= 0.64**
 Boat	0.40	1	0.1	
 Car	0.67	6	0.6	
Total	-	10	1.0	

LAB 2: Classifying Diabetes with NN

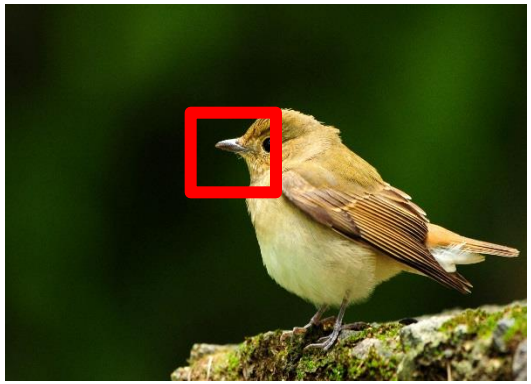
- Pima Indians Diabetes Database | Kaggle
  - Csv file download
    - https://vo.la/VOEZgg
- Python Notebook file download
  - https://vo.la/O7zrKO

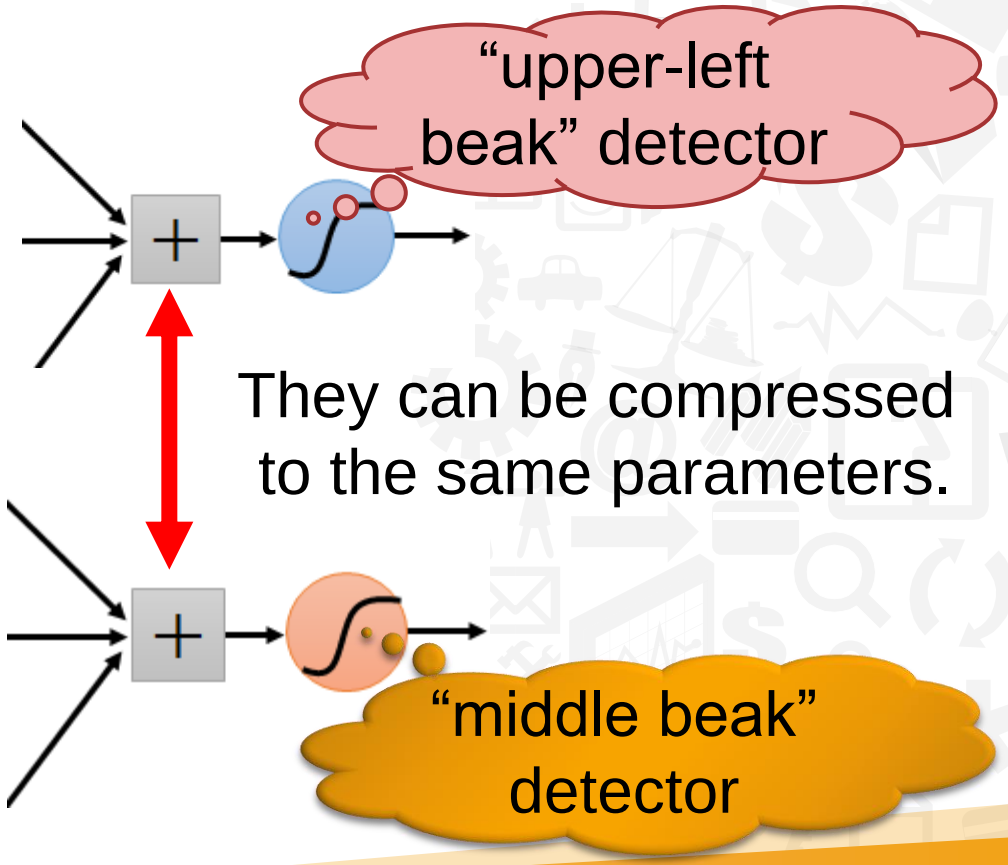
Convolutional Neural Network

- Some patterns are much smaller than the whole image
  - Can represent a small region with fewer parameters

Same pattern appears in different places: They can be compressed!

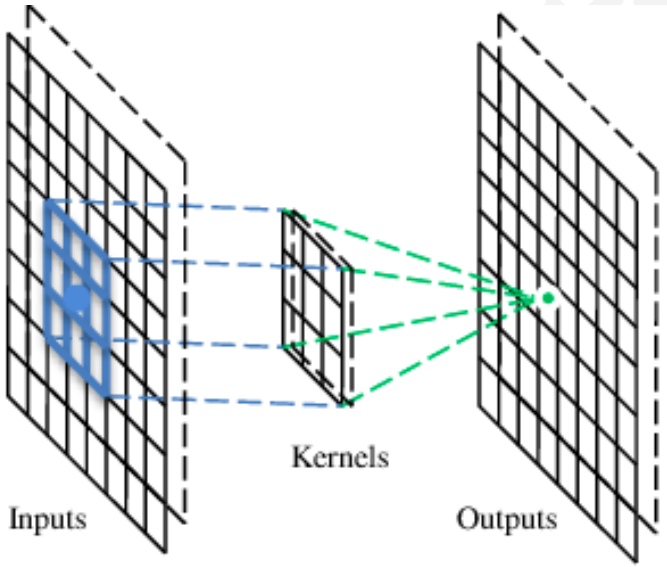
- What about training a lot of such "small" detectors and each detector must "move around".





A convolutional layer

- A CNN is a neural network with some convolutional layers (and some other layers).
- A convolutional layer has a number of filters that does convolutional operation



Convolution

These are the network parameters to be learned.

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

⋮ ⋮

Each filter detects a small pattern (3 x 3).

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

Dot product

6 x 6 image

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

If stride=2

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

3 -3

6 x 6 image

Convolution

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

3	-1	-3	-1
-3	1	0	-3
-3	-3	0	1
3	-2	-2	-1

Convolution

-1	1	-1
-1	1	-1
-1	1	-1

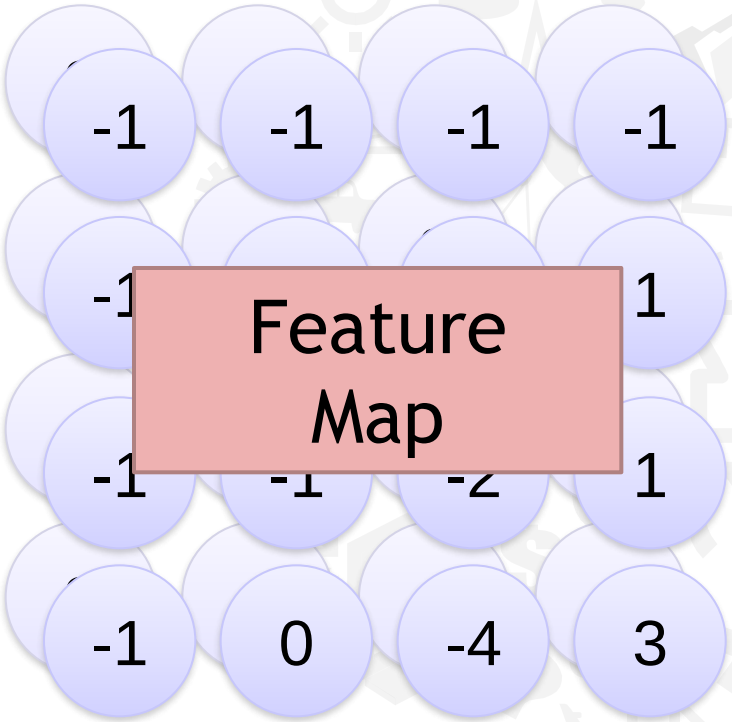
Filter 2

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

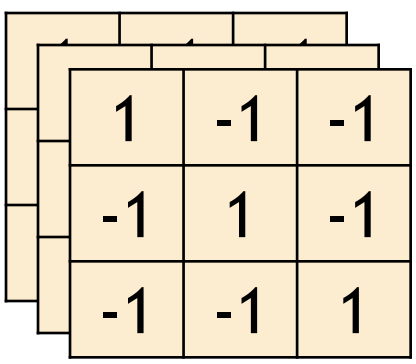
6 x 6 image

Repeat this for each filter

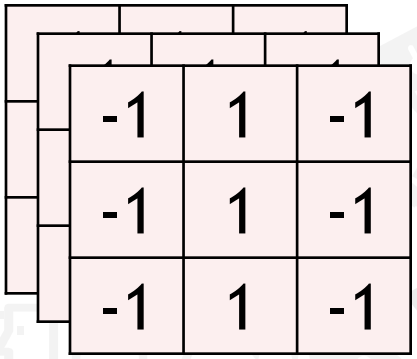


Two 4 x 4 images
Forming 2 x 4 x 4 matrix

Color image: RGB 3 Channels

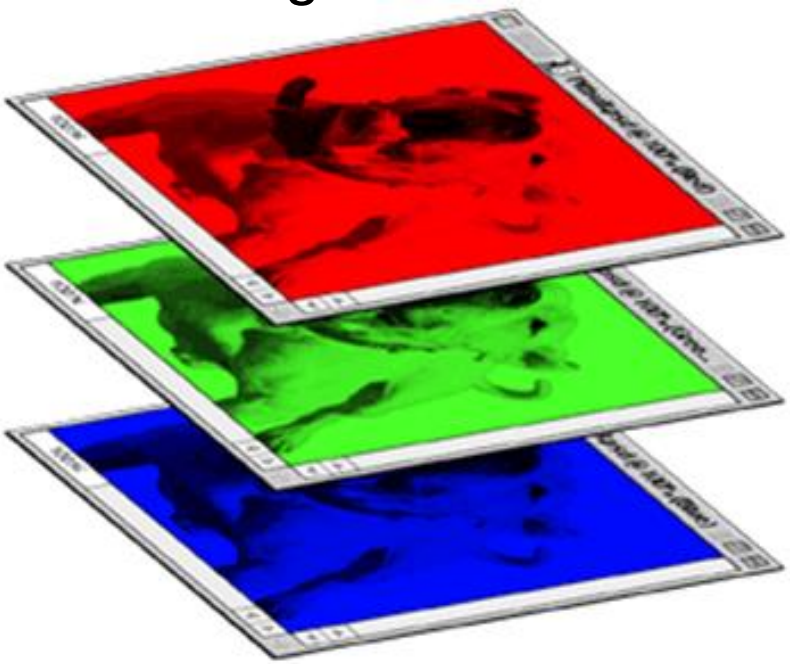


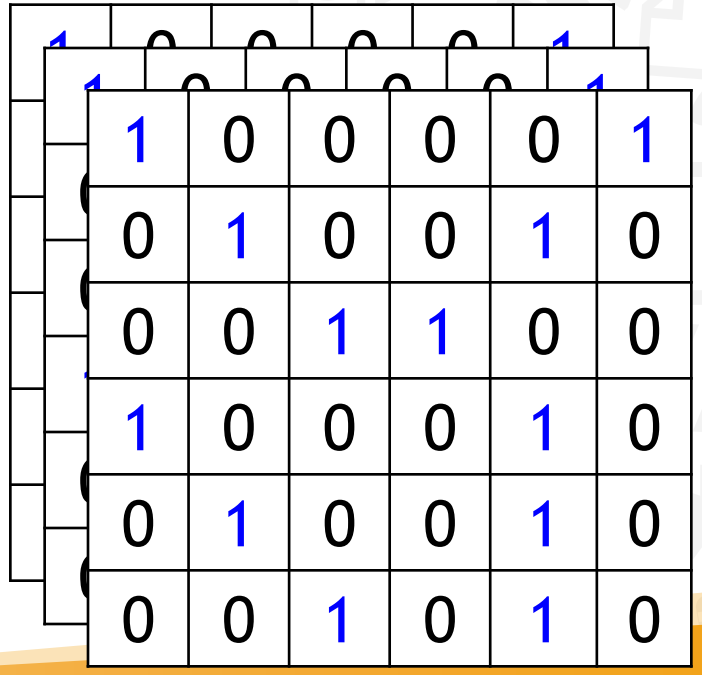
Filter 1



Filter 2

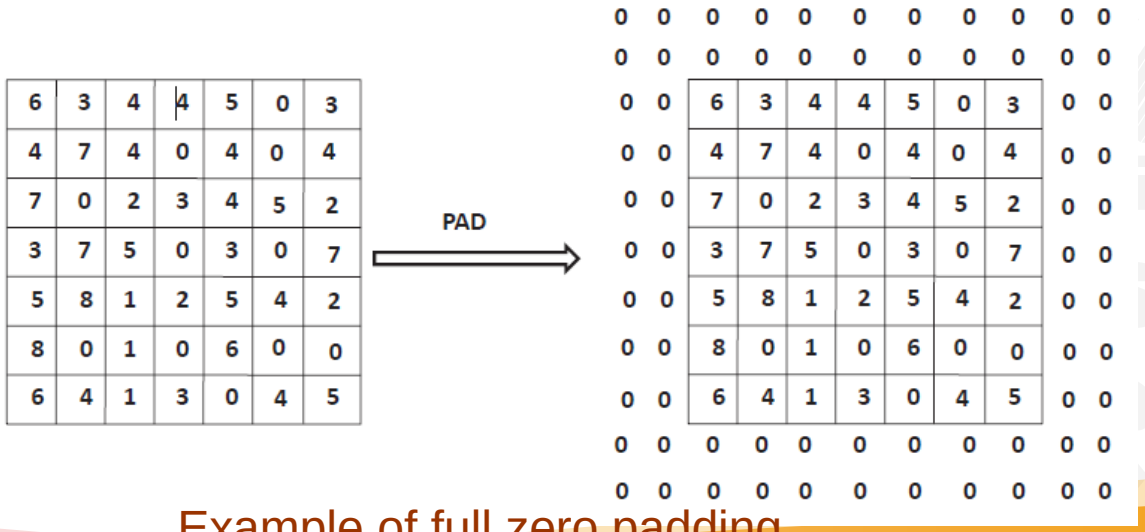
Color image





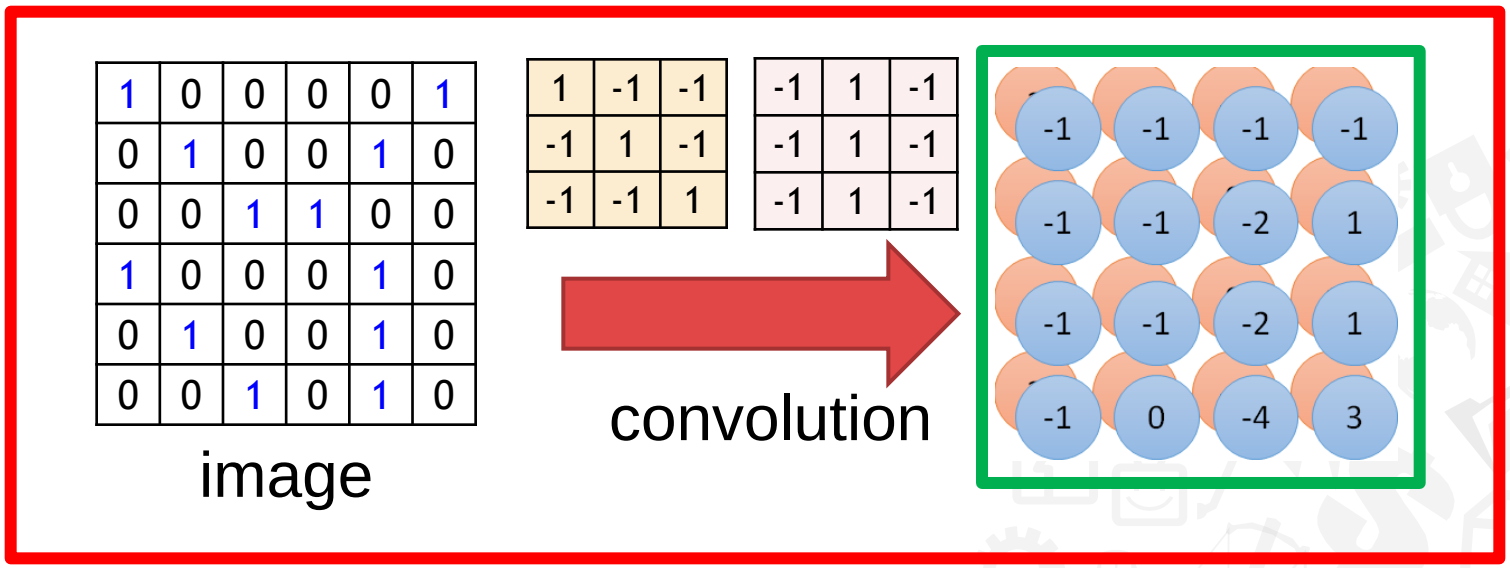
Padding

- Convolution operation reduces the size of the (q+1)-th layer with the size of the q-th layer
- By adding pixes all around the borders of the feature map, one can maintain the size of the spatial image

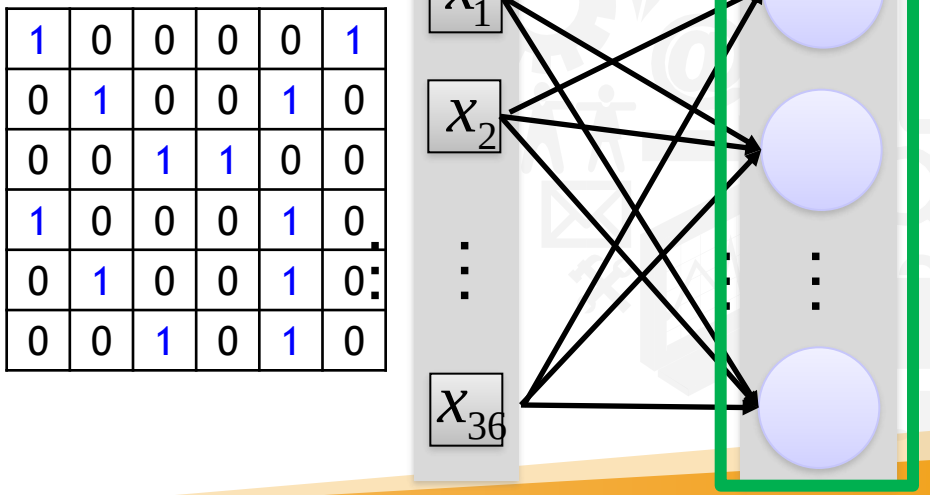


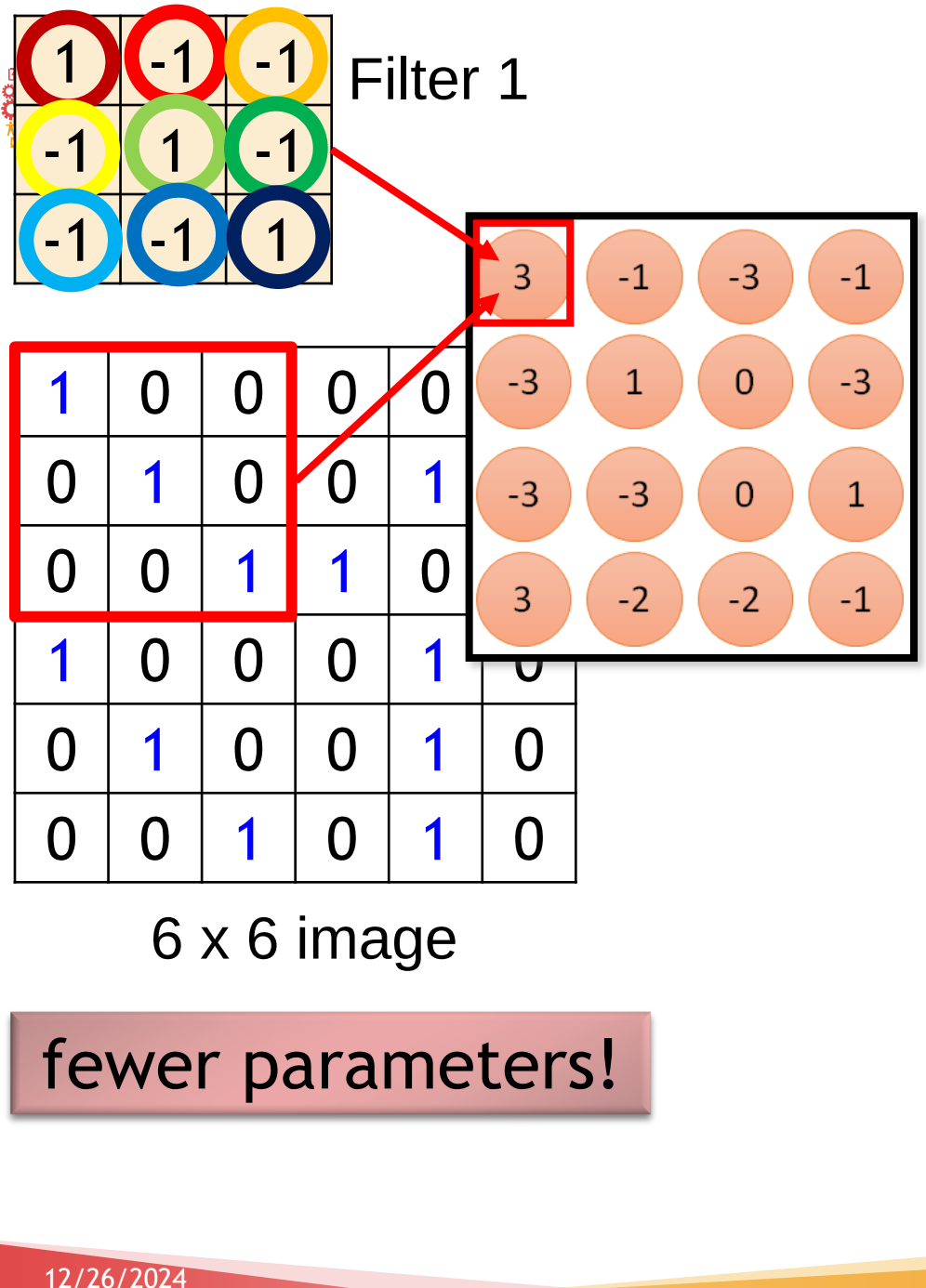
Example of full zero padding

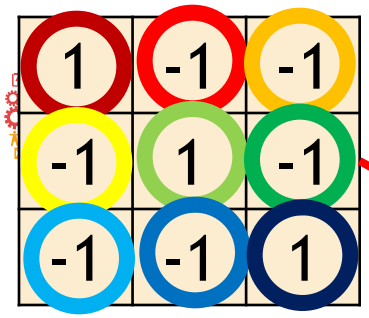
Convolution vs. Fully-Connected NN



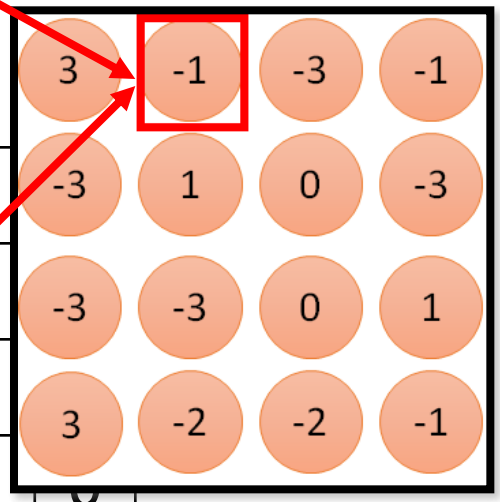
Fully-connected




1 -1 -1
-1 1 -1
-1 -1 1
Filter 1
3 -1 -3 -1
-3 1 0 -3
-3 -3 0 1
3 -2 -2 -1
1 0 0 0 0
0 1 0 0 1
0 0 1 1 0
1 0 0 0 1
0 1 0 0 1 0
0 0 1 0 1 0
6 x 6 image
fewer parameters!
1 1
2 0
3 0
4: 0
0
8 1
9 0
10: 0
1 0
14 0
15 1
16 1
3
Only connect to 9 inputs, not fully connected



Filter 1

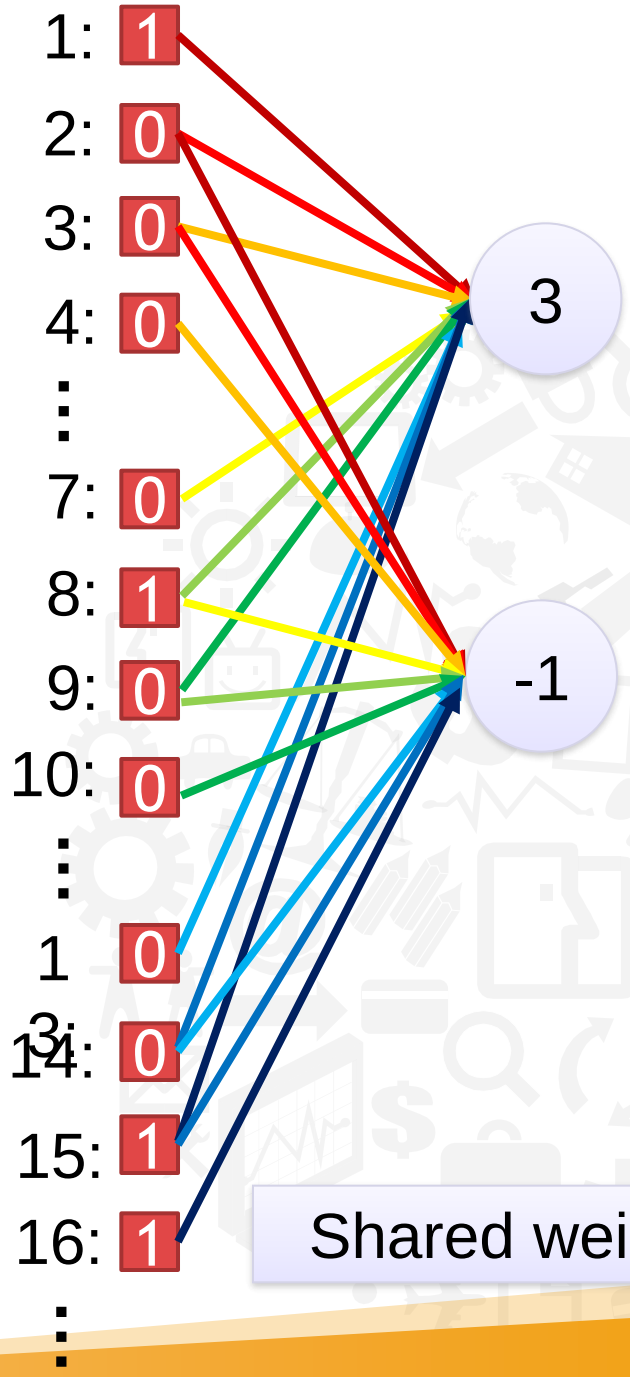


1	0	0	0	0	
0	1	0	0	1	
0	0	1	1	0	
1	0	0	0	1	
0	1	0	0	1	0
0	0	1	0	1	0

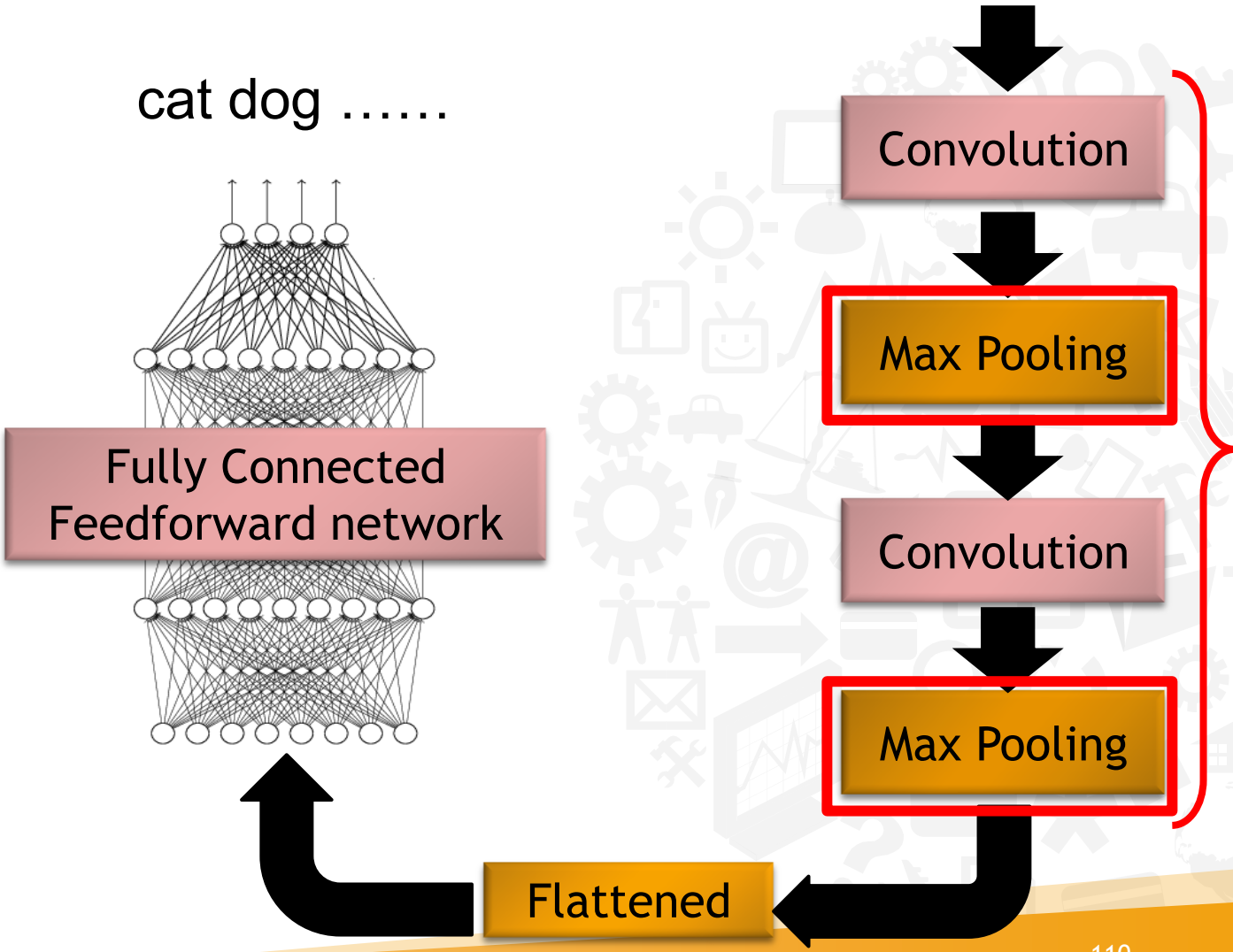
6 x 6 image

Fewer parameters

Even fewer parameters



Typical CNN Architecture



Max Pooling

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

3	-1
-3	1

-3	-1
0	-3

-3	-3
3	-2

0	1
-2	-1

-1	-1
-1	-1

-1	-1
-2	1

-1	-1
-1	0

-2	1
-4	3

Why Pooling?

- Subsampling pixels will not change the object

bird

Subsampling

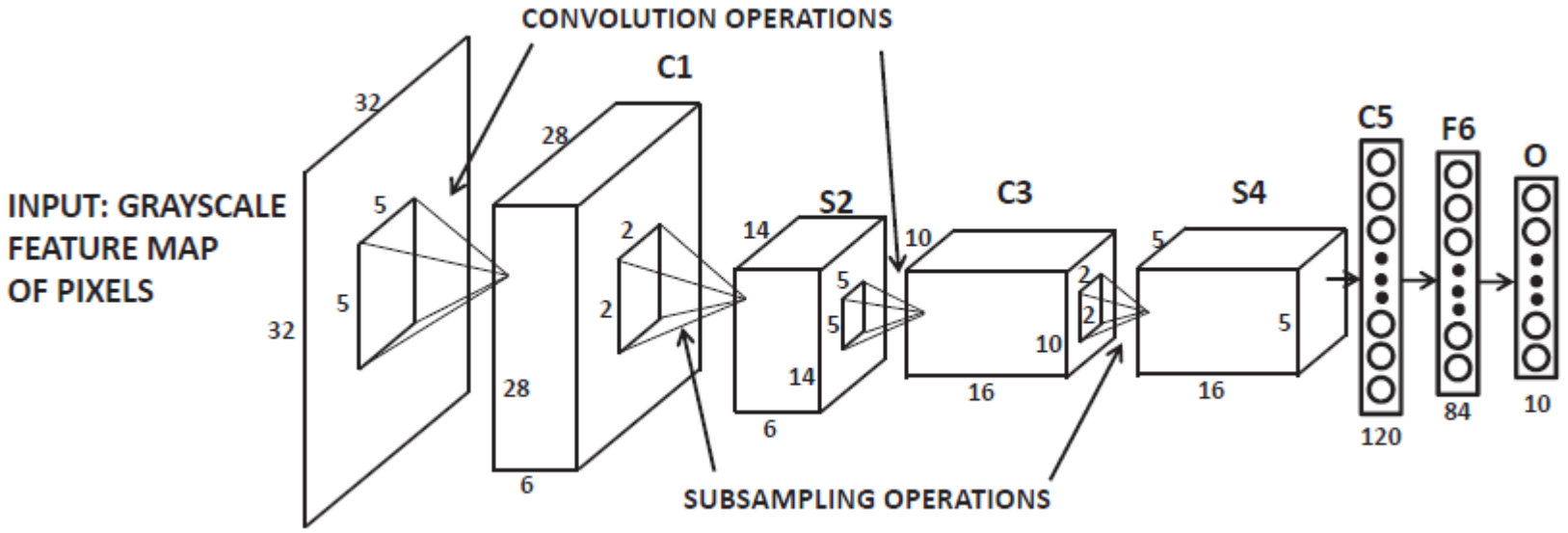
bird

We can subsample the pixels to make image smaller

→ fewer parameters to characterize the image

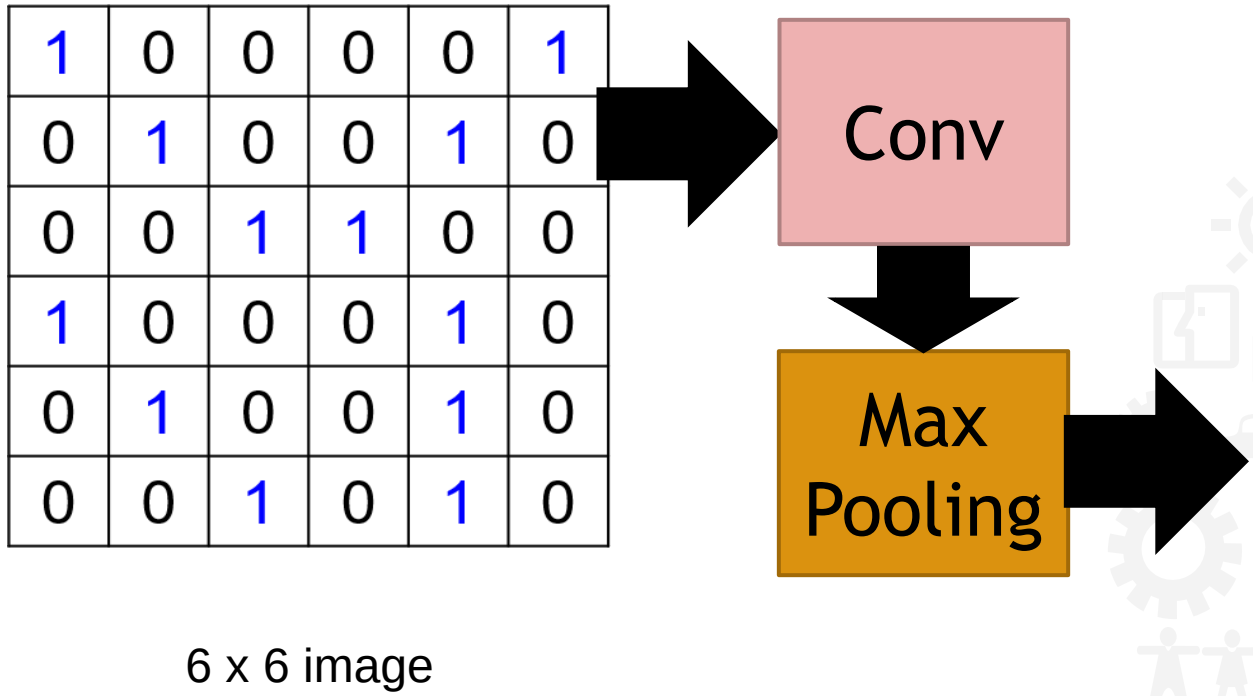
A CNN compresses a fully connected network in two ways:

- Reducing number of connections
- Shared weights on the edges
- Max pooling further reduces the complexity



LeNet-5 : The earlist CNN

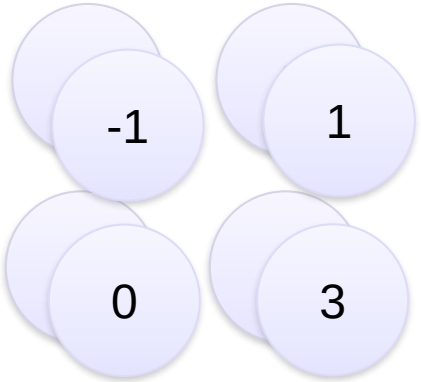
Max Pooling



6 x 6 image

Each filter is a channel

The whole CNN



A new image

Convolution

Max Pooling

Convolution

Max Pooling

Smaller than the original image

The number of channels is the number of filters

The whole CNN

cat dog ……

Convolution

Max Pooling

A new image

Convolution

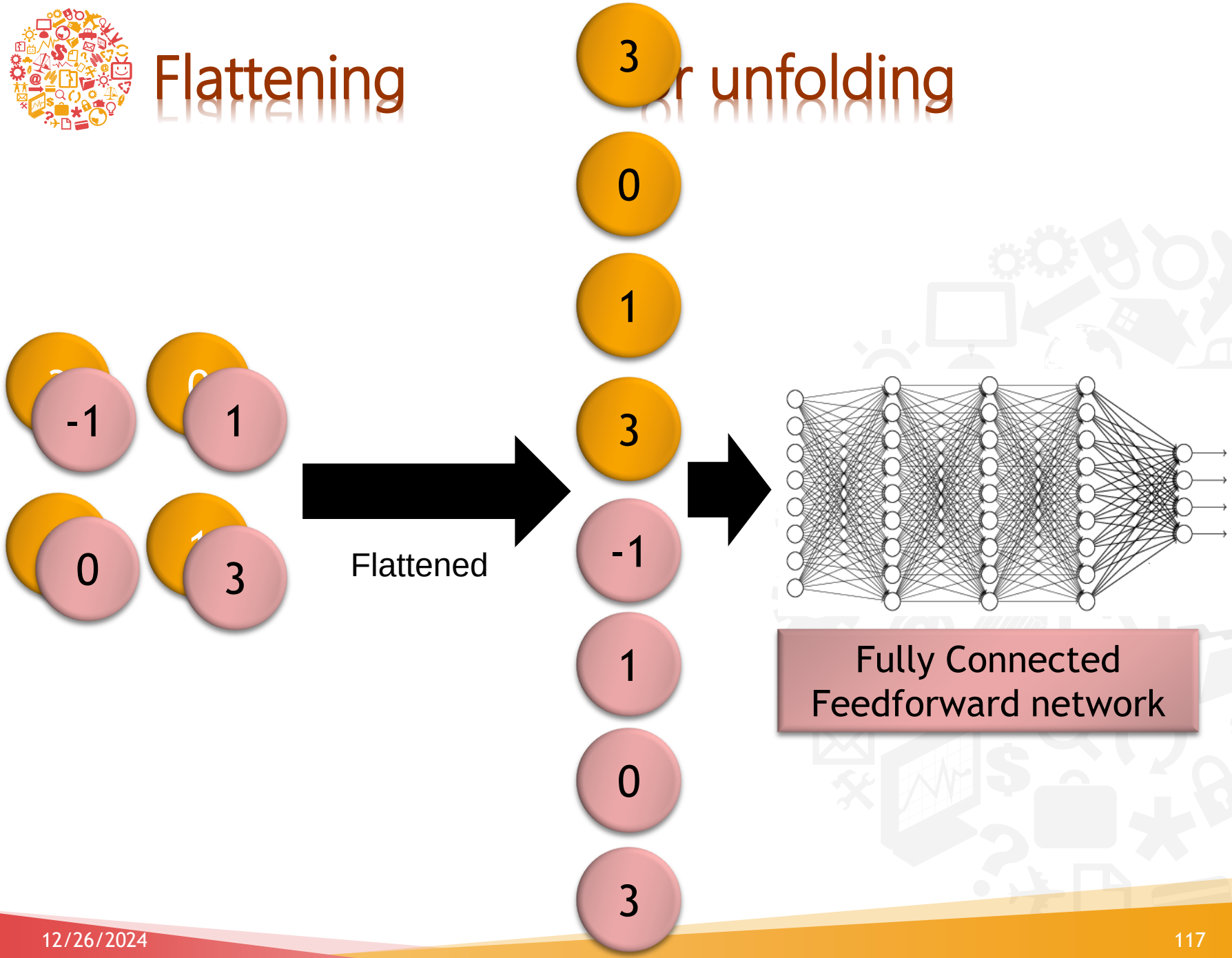
Max Pooling

A new image

Flattened

Fully Connected Feedforward network

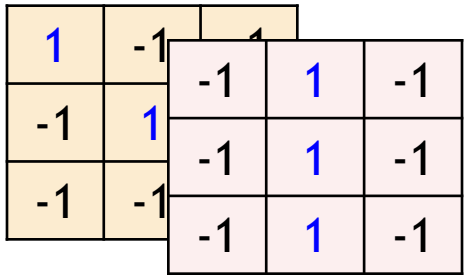
Flattening or unfolding



CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D tensor)***

```
model2.add( Convolution2D( 25,3,3,
            input_shape=(28,28,1)) )
```



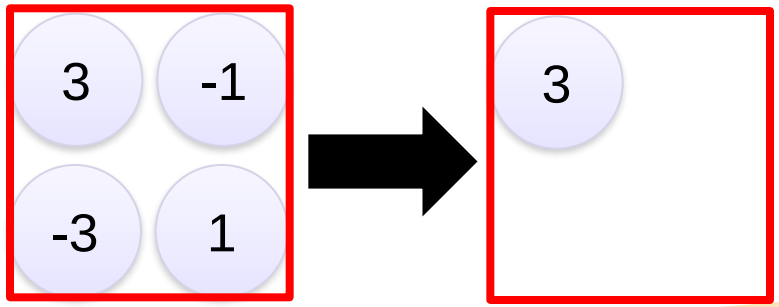
There are 25 3x3 filters.

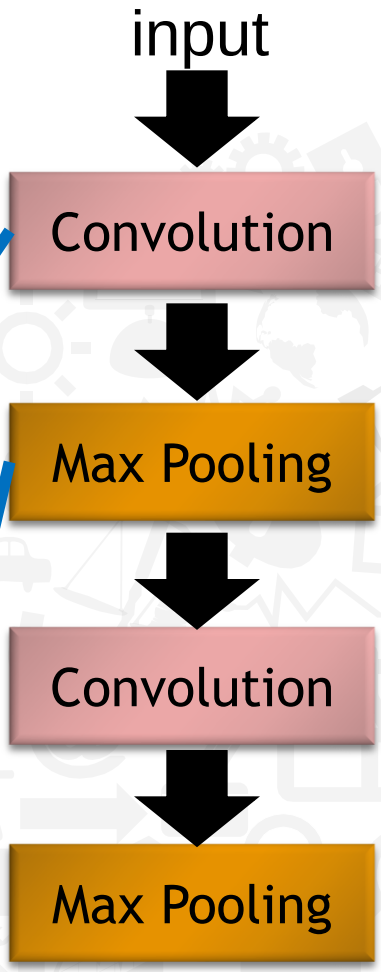
Input_shape = (28 , 28 , 1)

28 x 28 pixels

1: black/white, 3: RGB

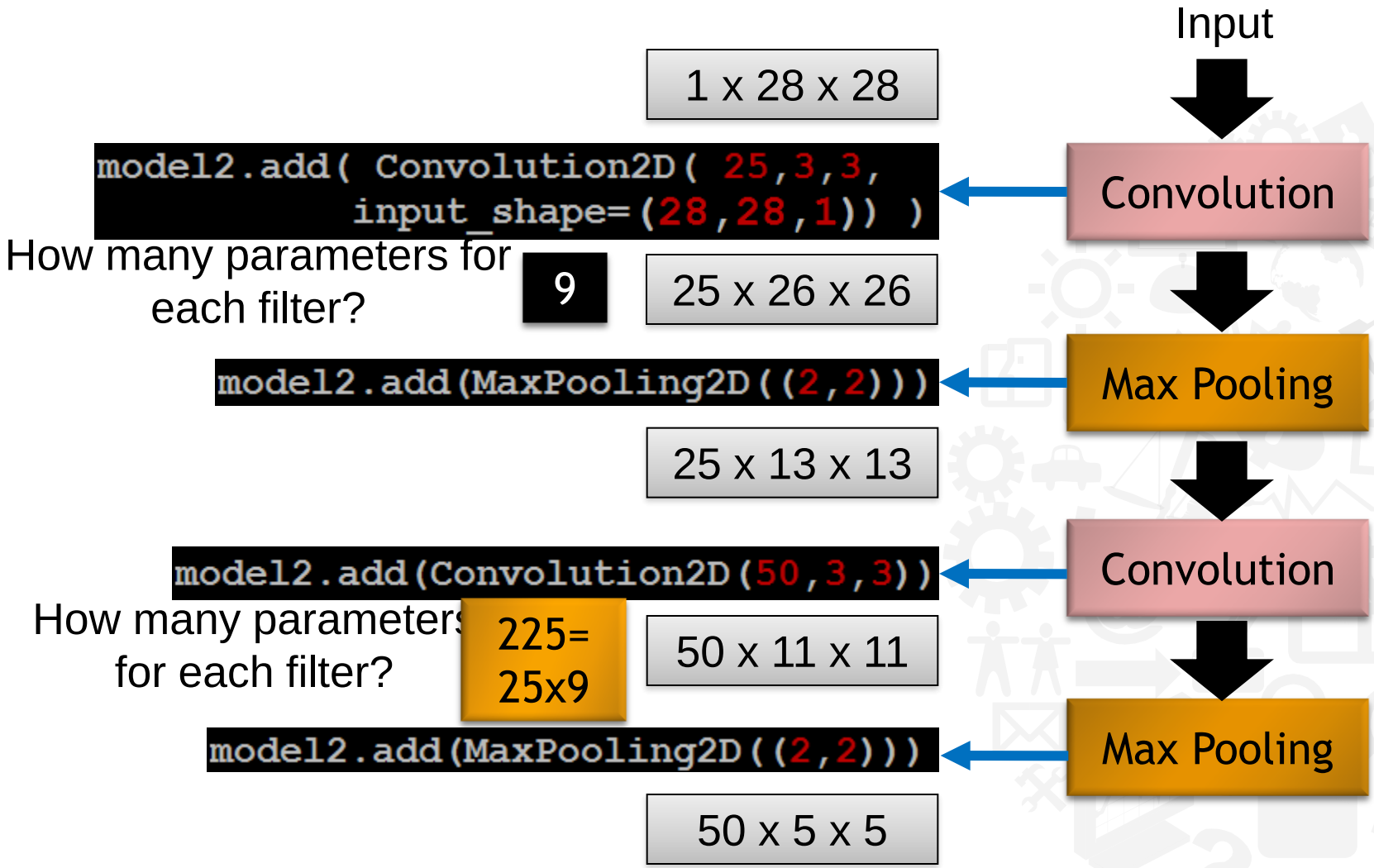
```
model2.add(MaxPooling2D((2,2)))
```





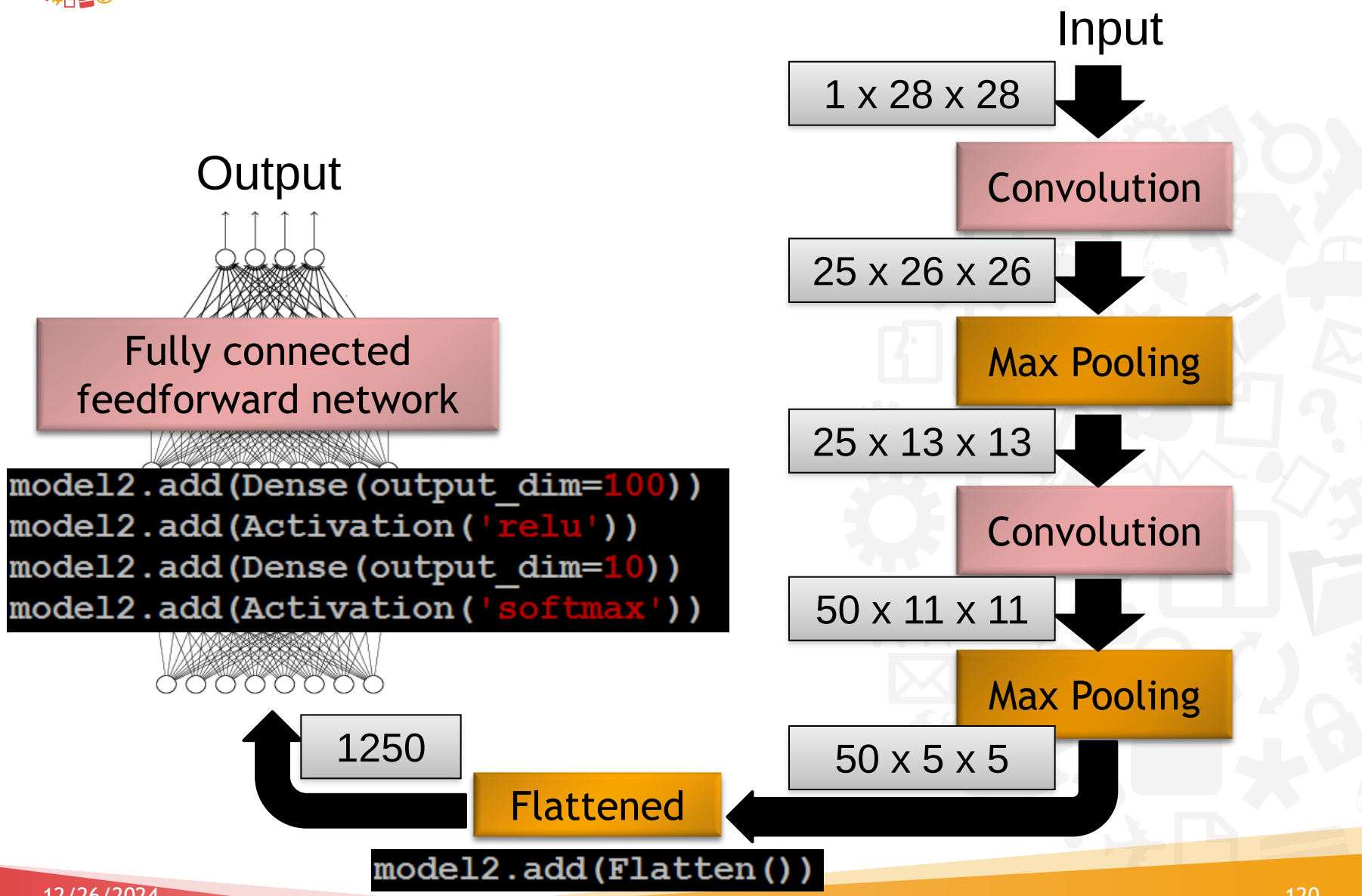
CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D array)***



CNN in Keras

Only modified the ***network structure*** and ***input format (vector -> 3-D array)***



AlphaGo

19 x 19 matrix

Black: 1

white: -1

none: 0

Neural Network

Next move

(19 x 19 positions)

Fully-connected feedforward network can be used

But CNN performs much better

AlphaGo's policy network

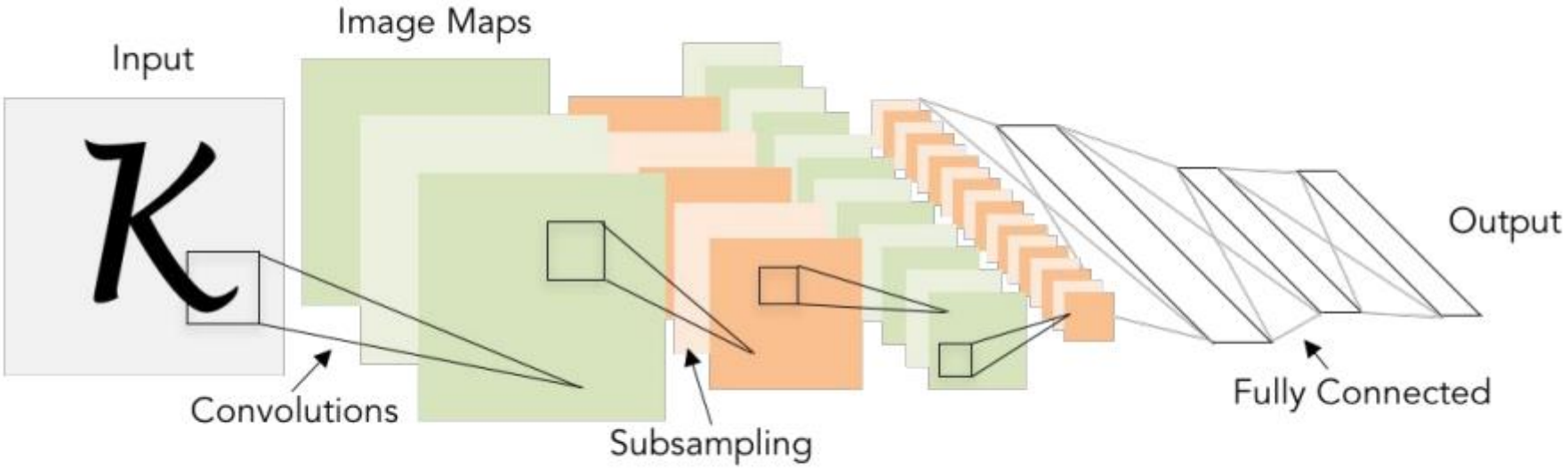
The following is quotation from their Nature article:

Note: AlphaGo does not use Max Pooling.

Neural network architecture. The input to the policy network is a $19 \times 19 \times 48$ image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a 23×23 image, then convolves k filters of kernel size 5×5 with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a 21×21 image, then convolves k filters of kernel size 3×3 with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size 1×1 with stride 1, with a different bias for each position, and applies a softmax function. The match version of AlphaGo used $k = 192$ filters; Fig. 2b and Extended Data Table 3 additionally show the results of training with $k = 128$, 256 and 384 filters.

ReNet-5

[LeCun et al., 1998]

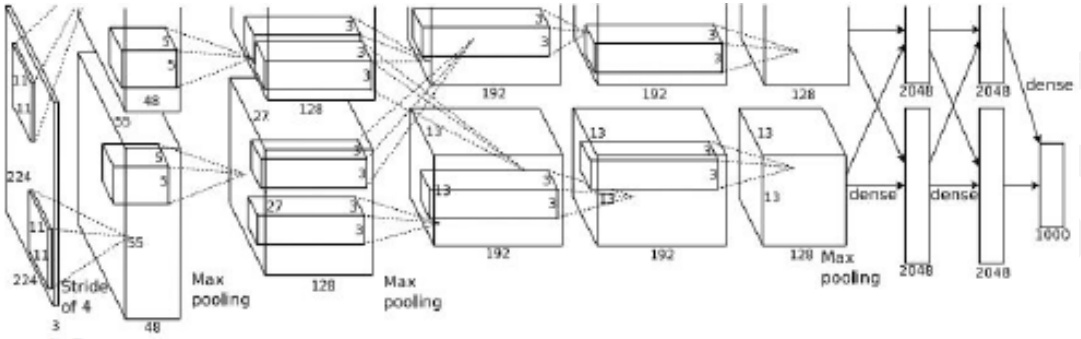


Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-FC-FC]

Family of CNNs

Case Study: AlexNet

[Krizhevsky et al. 2012]



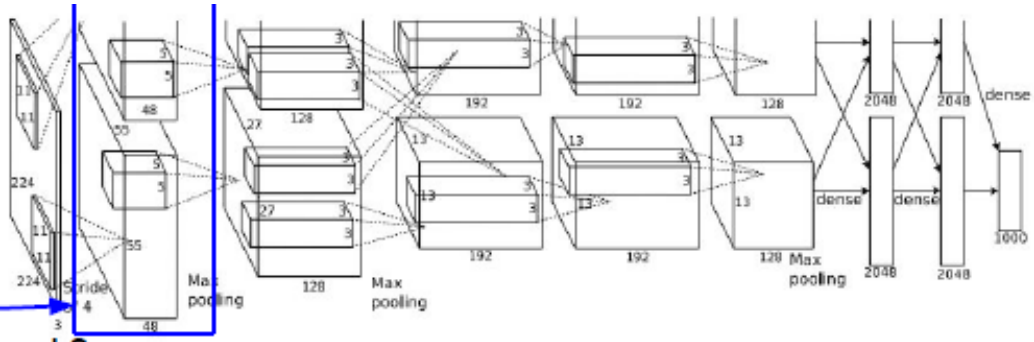
Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

Details/Retrospectives:
- first use of ReLU
- used Norm layers (not common anymore)
- heavy data augmentation
- dropout 0.5
- batch size 128
- SGD Momentum 0.9
- Learning rate 1e-2, reduced by 10 manually when val accuracy plateaus
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% -> 15.4%

Family of CNNs

Case Study: AlexNet

[Krizhevsky et al. 2012]

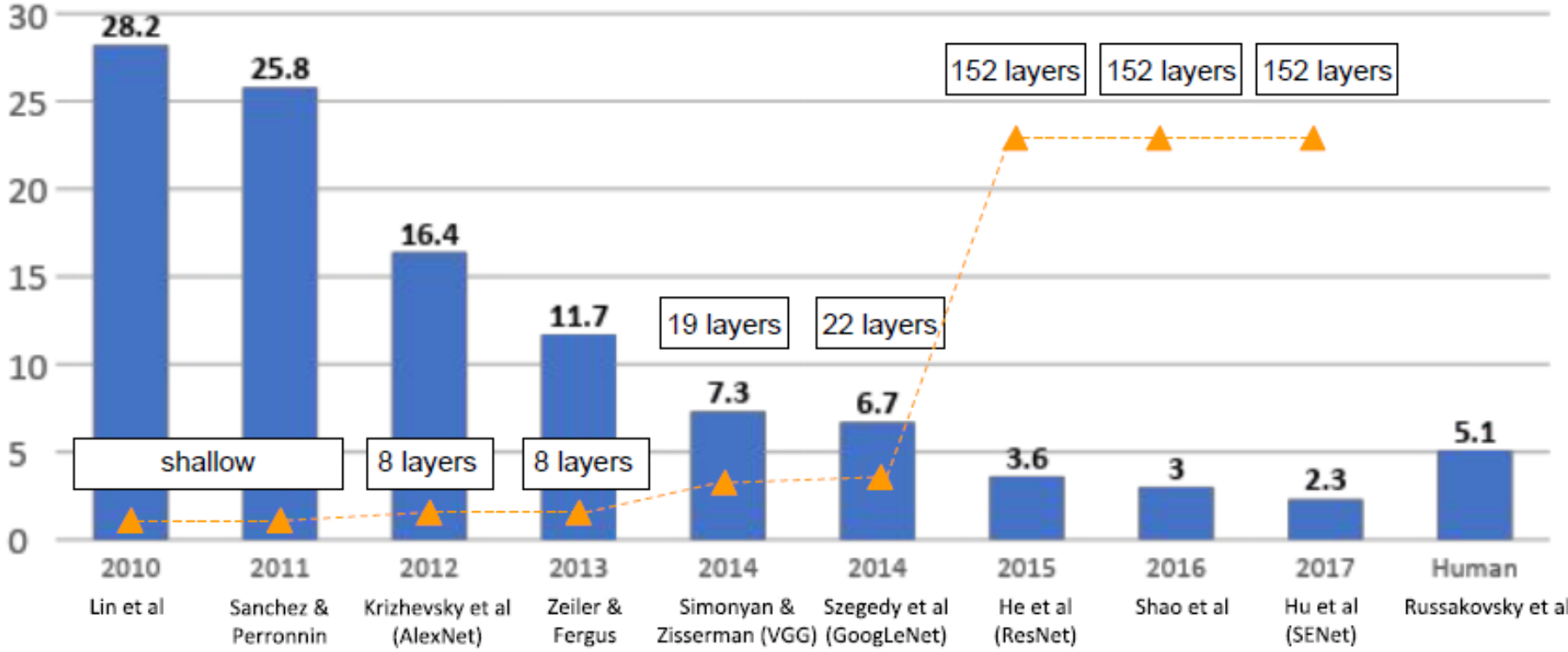


[55x55x48] x 2

Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

Historical note: Trained on GTX 580 GPU with only 3 GB of memory. Network spread across 2 GPUs, half the neurons (feature maps) on each GPU.

ImageNet Large Scale Visual Recognition Challenge(ILSVRC)



Family of CNNs

Case Study: VGGNet

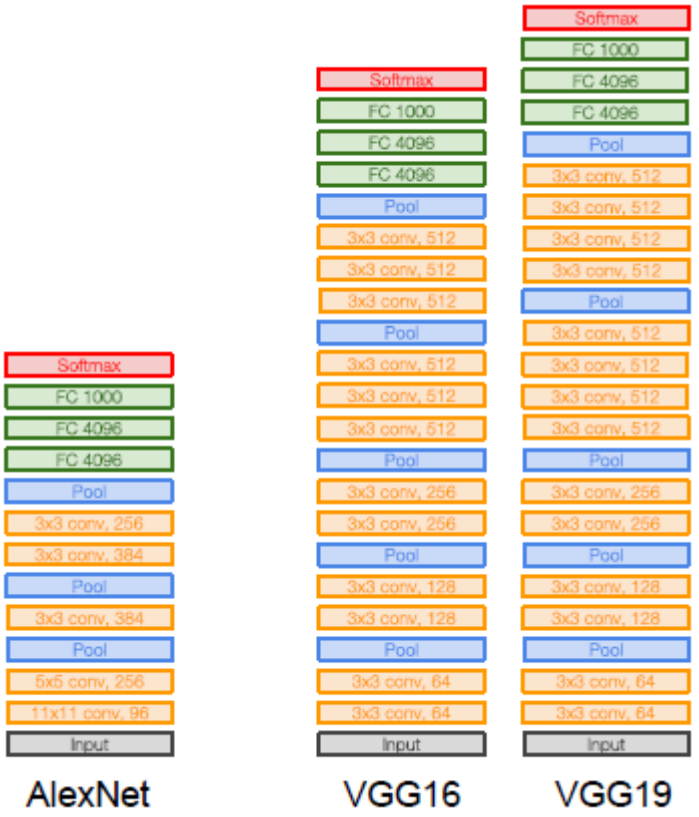
[Simonyan and Zisserman, 2014]

Small filters, Deeper networks

8 layers (AlexNet)
-> 16 - 19 layers (VGG16Net)

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

11.7% top 5 error in ILSVRC'13
(ZFNet)
-> 7.3% top 5 error in ILSVRC'14



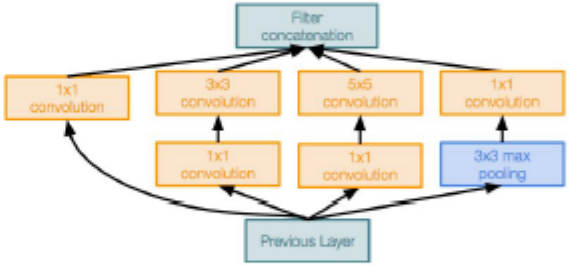
Family of CNNs

Case Study: GoogLeNet

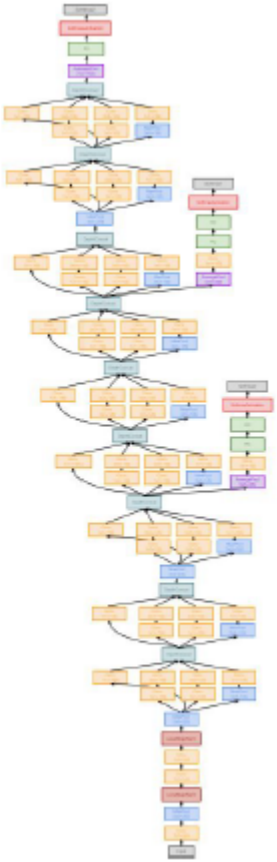
[Szegedy et al., 2014]

Deeper networks, with computational efficiency

- 22 layers
- Efficient "Inception" module
- No FC layers
- Only 5 million parameters!
  12x less than AlexNet
- ILSVRC'14 classification winner
  (6.7% top 5 error)

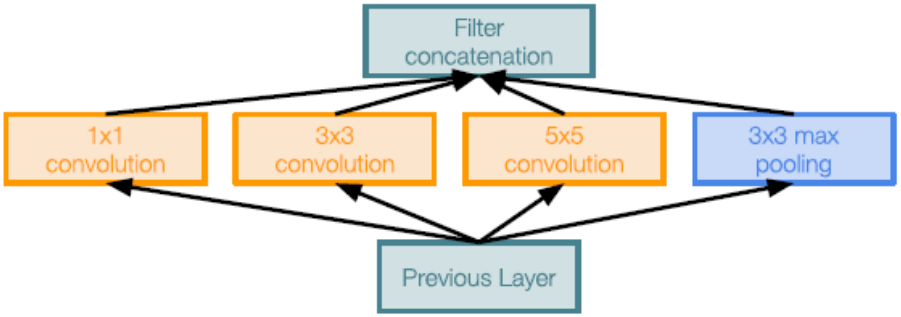


Inception module



GoogLeNet

- Inception module



Naive Inception module

Apply parallel filter operations on the input from previous layer:

- Multiple receptive field sizes for convolution (1x1, 3x3, 5x5)
- Pooling operation (3x3)

Concatenate all filter outputs together depth-wise

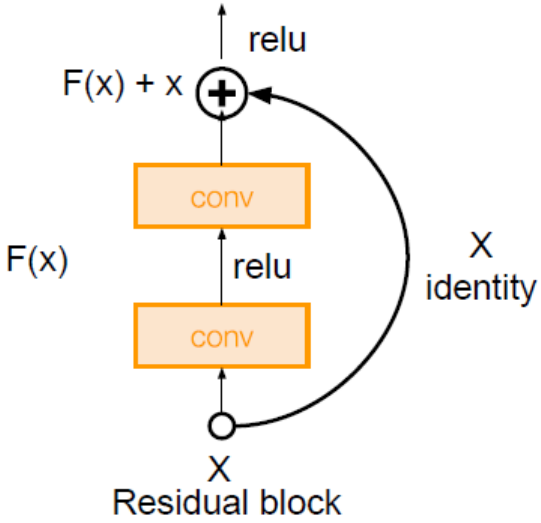
Family of CNNs

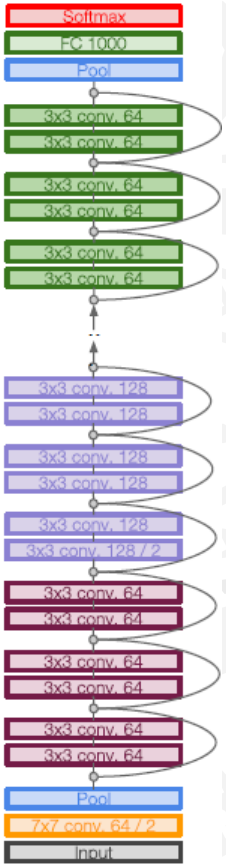
Case Study: ResNet

[He et al., 2015]

Very deep networks using residual connections

- 152-layer model for ImageNet
- ILSVRC'15 classification winner (3.57% top 5 error)
- Swept all classification and detection competitions in ILSVRC'15 and COCO'15!

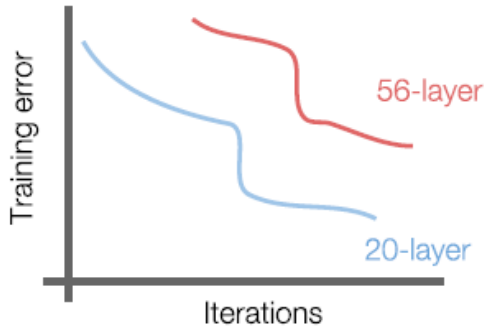


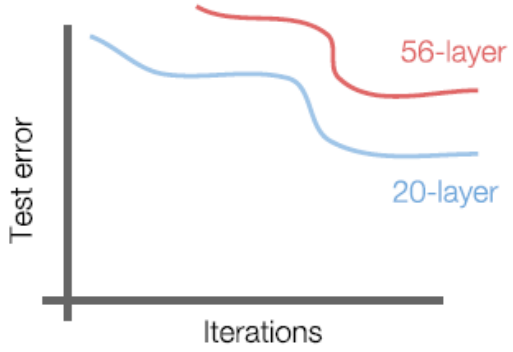


Case Study: ResNet

[He et al., 2015]

What happens when we continue stacking deeper layers on a “plain” convolutional neural network?





Q: What’s strange about these training and test curves?
[Hint: look at the order of the curves]

Case Study: ResNet

[He et al., 2015]

Hypothesis: the problem is an *optimization* problem, deeper models are harder to optimize

The deeper model should be able to perform at least as well as the shallower model.

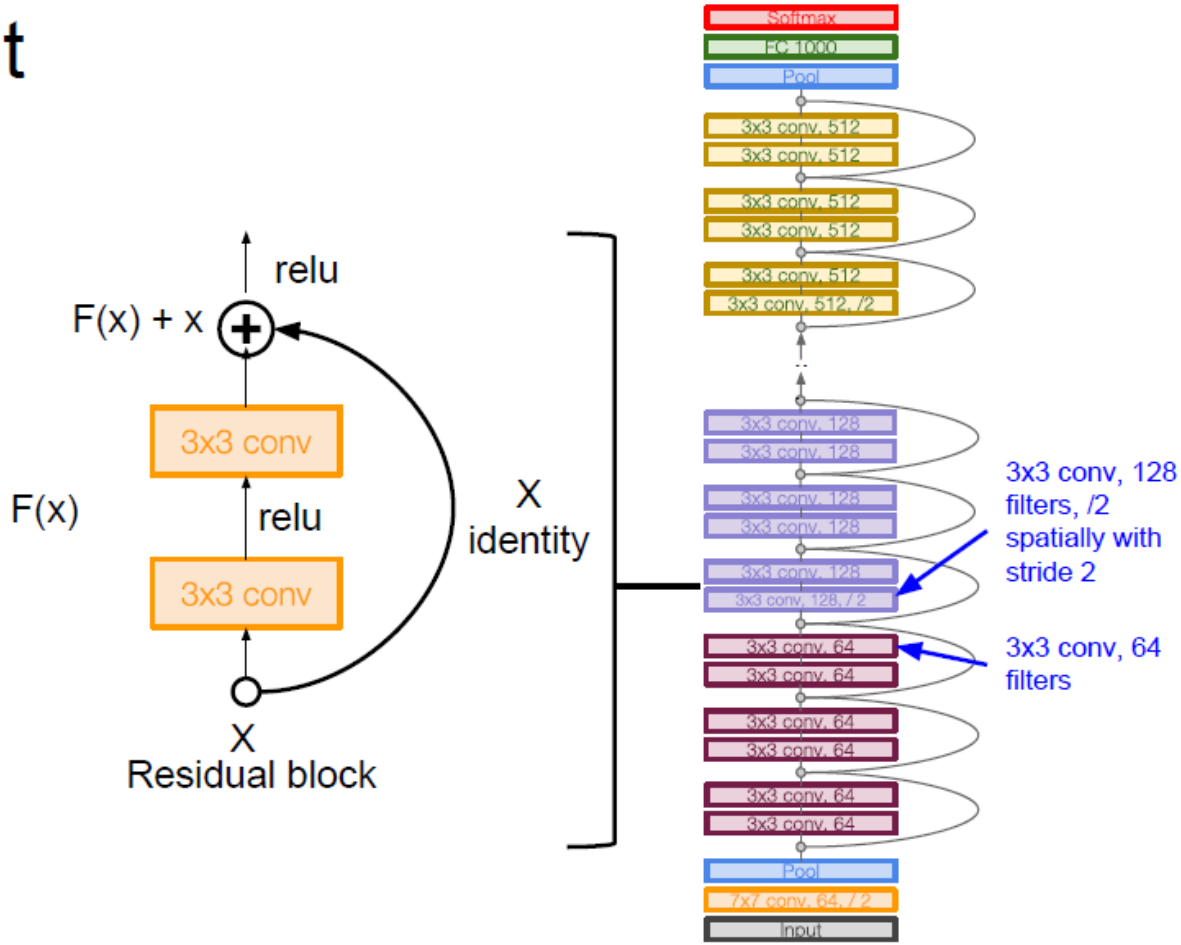
A solution by construction is copying the learned layers from the shallower model and setting additional layers to identity mapping.

Case Study: ResNet

[He et al., 2015]

Full ResNet architecture:

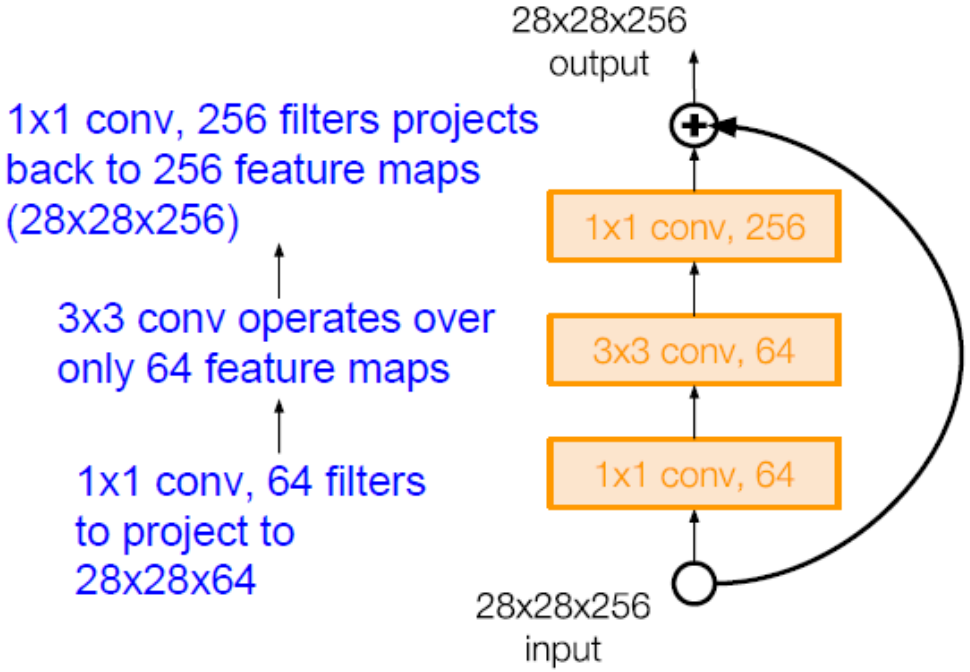
- Stack residual blocks
- Every residual block has two 3x3 conv layers
- Periodically, double # of filters and downsample spatially using stride 2 (/2 in each dimension)



Case Study: ResNet

[He et al., 2015]

For deeper networks (ResNet-50+), use "bottleneck" layer to improve efficiency (similar to GoogLeNet)

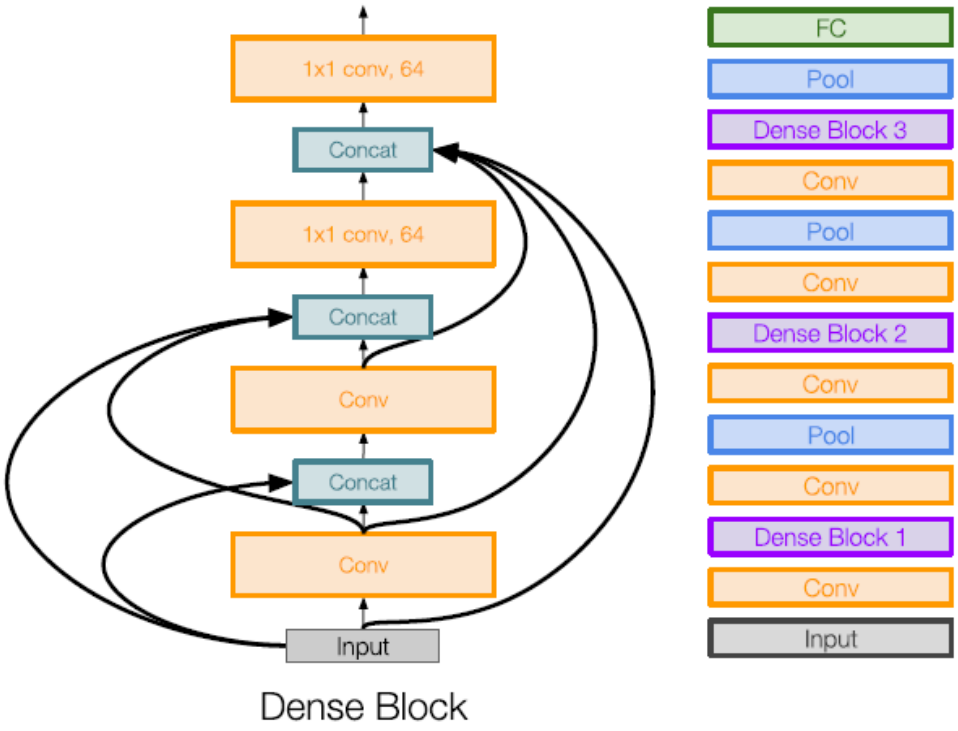


Family of CNNs

Densely Connected Convolutional Networks

[Huang et al. 2017]

- Dense blocks where each layer is connected to every other layer in feedforward fashion
- Alleviates vanishing gradient, strengthens feature propagation, encourages feature reuse



LAB 3 : Analyzing Chest X-ray images to diagnose Pneumonia

- Register into Kaggle and create a token for using Open API
  - https://Kaggle.com
- Read Chest X-ray Penumonia vs . Normal
  - Chest X-Ray: Pneumonia Vs Normal – Radiology In Plain English
- Dataset
  - Chest X-Ray Images (Pneumonia) | Kaggle
- Python Notebook file
  - https://vo.la/fETIct

Thank You!