Introduction to Machine Learning

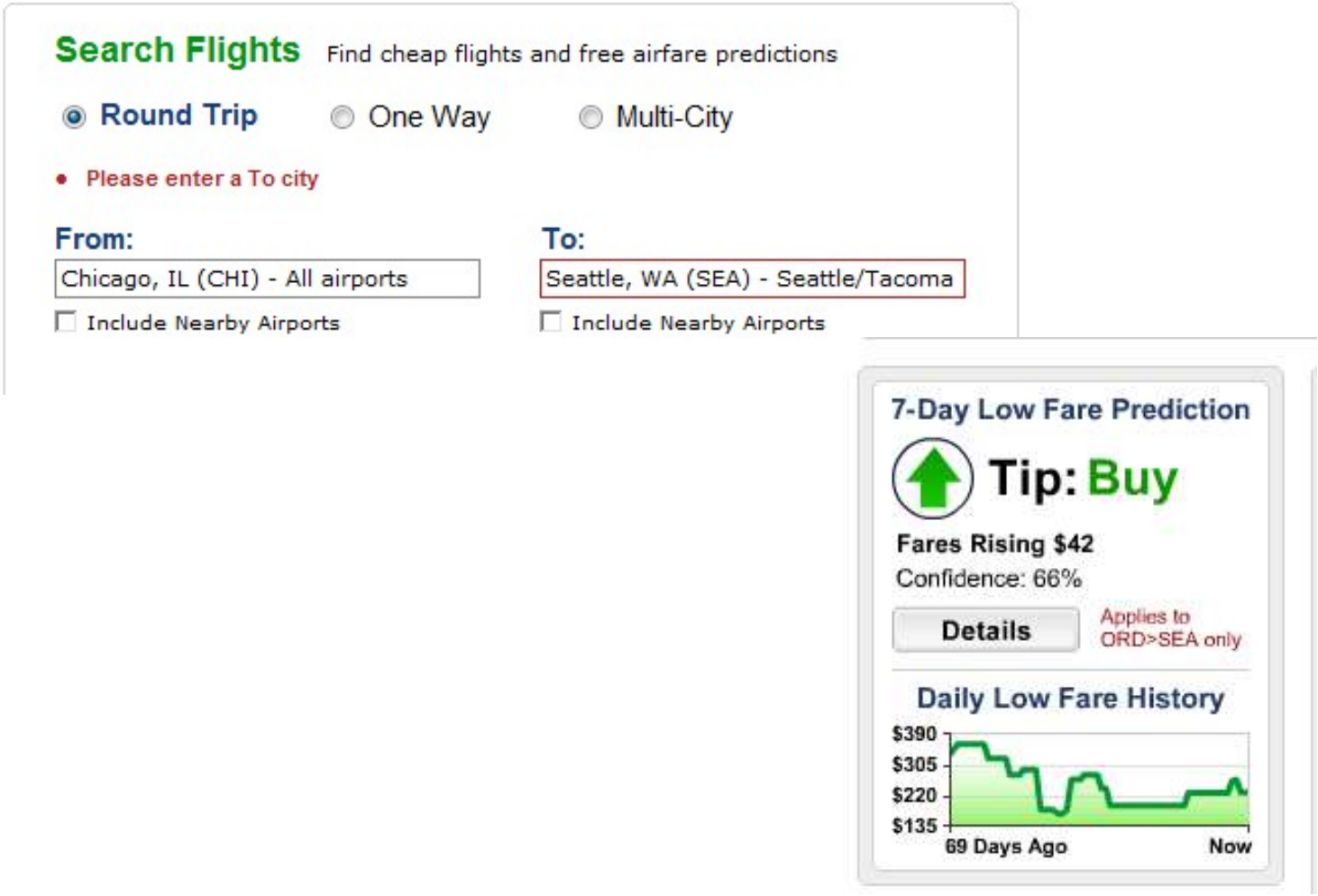
Hong-Han Shuai
National Chiao Tung University
HCC
2019 Spring

Thanks to the slides of Prof. P. Domingos from Washington University, Prof. H.-T. Lin and Prof. Lee Hung-Yi Lee from NTU.

Agenda

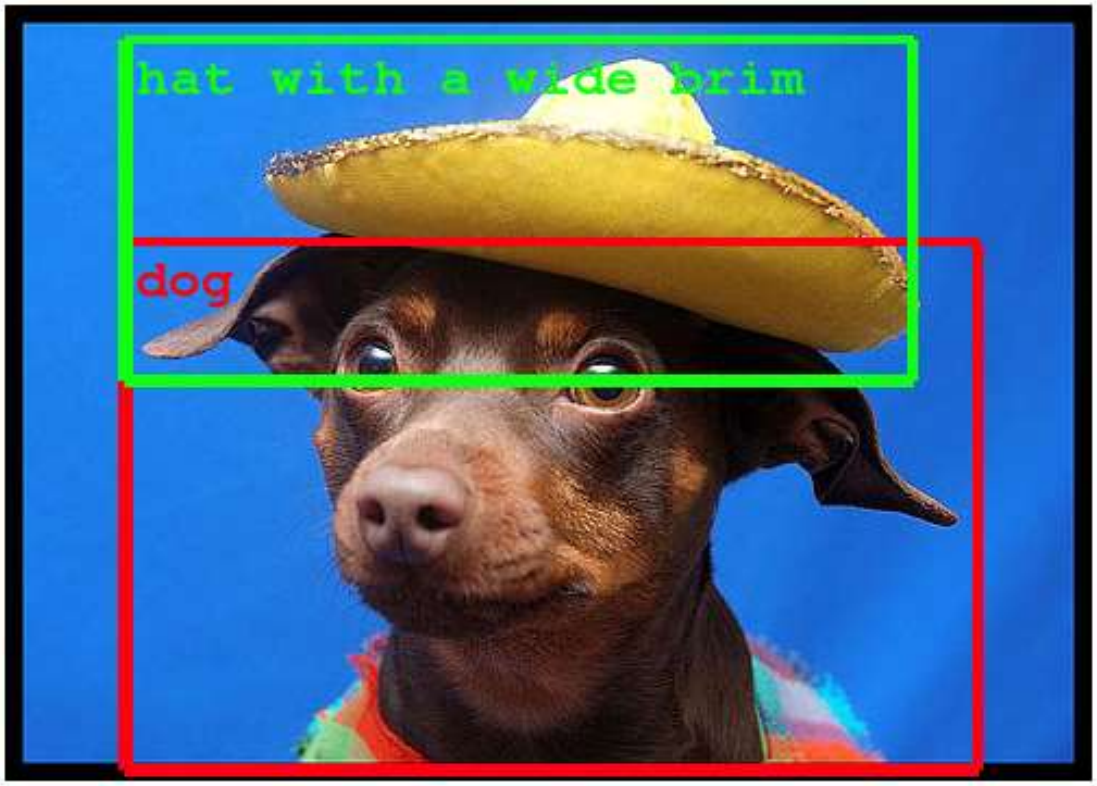
- Basics of Machine Learning
- Decision Tree and Random Forest
- Deep Learning
- Interesting Project Review

Applications



Applications

- Object Classification and Detection in Photographs



Source:http://machinelearningmastery.com/inspirational-applications-deep-learning

From Learning to Machine Learning

- What's learning?
  - knowledge or skill acquired by instruction or study

- What's machine learning?

http://www.csie.ntu.edu.tw/~htlin/mooc/

From Learning to Machine Learning

- What's learning?
  - knowledge or skill acquired by instruction or study

- What's machine learning?

A More Concrete Definition

- Skill: improve the performance measurements (e.g., prediction, 3pts shooting percentage)
- Therefore, machine learning is defined to be the improvements on some performance measurements by computing from data.
- For example,
  - Network data -> **ML** -> Better flow control
  - Signal data -> **ML** -> Faster antenna direction detection
  - Sequential web log data -> **ML** -> Higher accuracy of anomaly detection/efficiency of caching algorithm
  - Stock data -> **ML** -> More money

The Machine Learning Route

- ML: an **alternative route** to build complicated systems
- Some Use Scenarios
  - when human cannot program the system manually
    - navigating on Mars
  - when human cannot 'define the solution' easily
    - speech/visual recognition
  - when needing rapid decisions that humans cannot do
    - high-frequency trading
  - when needing to be user-oriented in a massive scale
    - consumer-targeted marketing

The Machine Learning Route

- ML: an **alternative route** to build complicated systems
- Some Use Scenarios
  - when human cannot program the system manually
    - navigating on Mars
  - when human cannot 'define the solution' easily
    - speech/visual recognition
  - when needing rapid decisions that humans cannot do
    - high-frequency trading
  - when needing to be user-oriented in a massive scale
    - consumer-targeted marketing

The Machine Learning Route

- ML: an **alternative route** to build complicated systems
- Some Use Scenarios
  - when human cannot program the system manually
    - navigating on Mars
  - when human cannot 'define the solution' easily
    - speech/visual recognition
  - when needing rapid decisions that humans cannot do
    - high-frequency trading
  - when needing to be user-oriented in a massive scale
    - consumer-targeted marketing

The Machine Learning Route

- ML: an **alternative route** to build complicated systems
- Some Use Scenarios
  - when human cannot program the system manually
    - navigating on Mars
  - when human cannot 'define the solution' easily
    - speech/visual recognition
  - when needing rapid decisions that humans cannot do
    - high-frequency trading
  - when needing to be user-oriented in a massive scale
    - consumer-targeted marketing

http://www.cartier.hk/content/dam/rcq/car/92/55/67/925567.png.scale.120.high.png

Key Essence of Machine Learning

- Exists some ‘underlying pattern’ to be learned
  - So performance measure can be improved
- But no programmable (easy) definition
  - So machine learning is required
- Somehow there is data about the pattern
  - So machine learning has inputs to learn from

Which of the following is best suited for machine learning?

- 1. predicting whether the next cry of the baby girl happens at an even-numbered minute or not?
- 2. determining whether a given graph contains a cycle
- 3. deciding whether to approve credit card to some customer
- 4. guessing whether the earth will be destroyed by the misuse of nuclear power in the next ten years

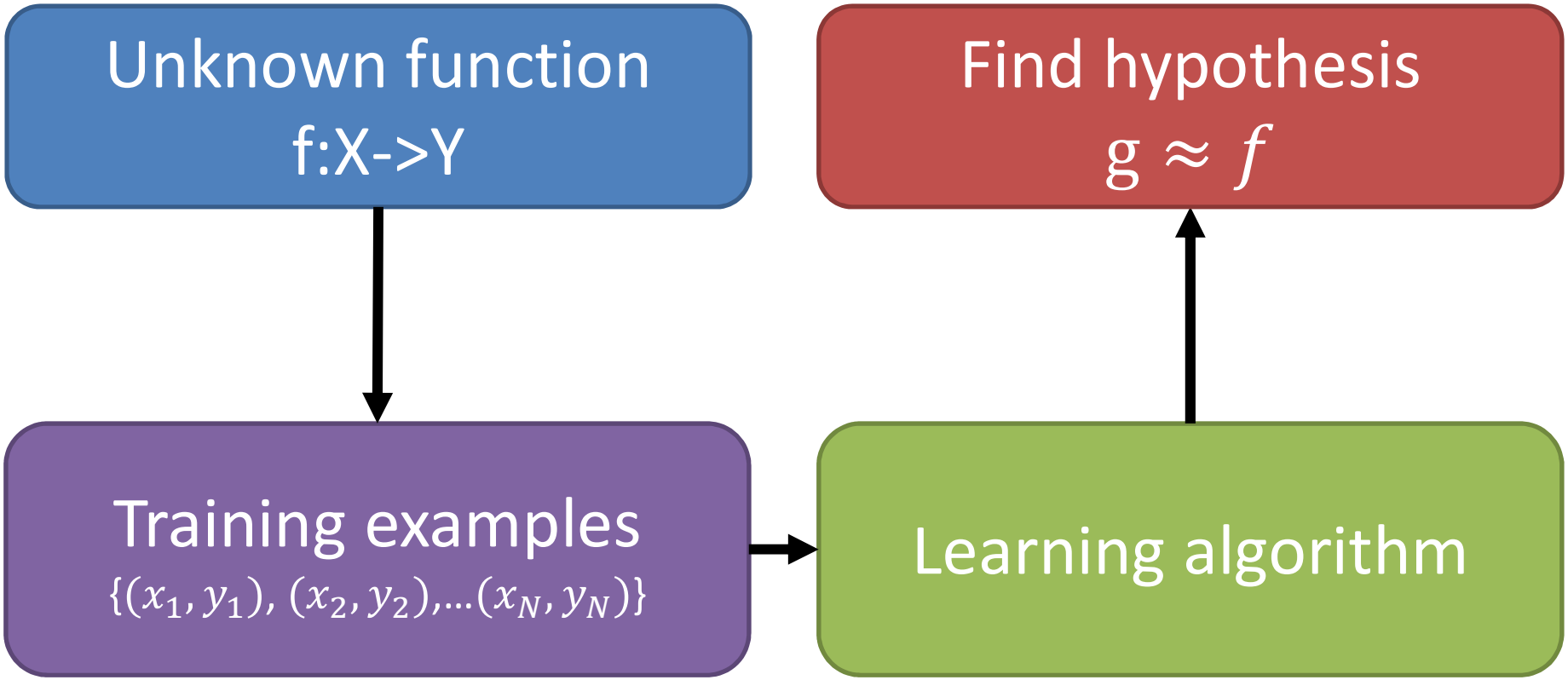
Which of the following field cannot use machine learning?

- 1. Medical
- 2. Animation
- 3. Law
- 4. none of the above

Learning Problem Formulation

- Notation
  - **Input:** $x \in X$(application)
  - **Output:** $y \in Y$(good/bad after approving)
  - **Unknown pattern to be learned can be formulated as a function**
    - f: X->Y (ideal function)
  - **Data:** D=$\{(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)\}$
  - **Hypothesis:**
    - g:X->Y (hopefully can be as close to f as possible)

Learning Flow



Learning is to find a function

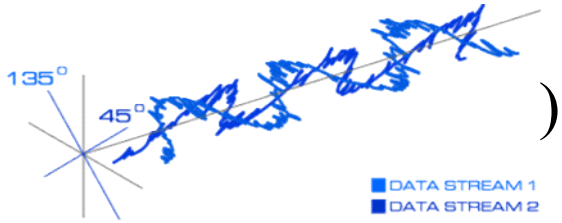
- Speech Recognition

f() = “我不知道你說什麼”

- Image recognition

f() = “Seafood”

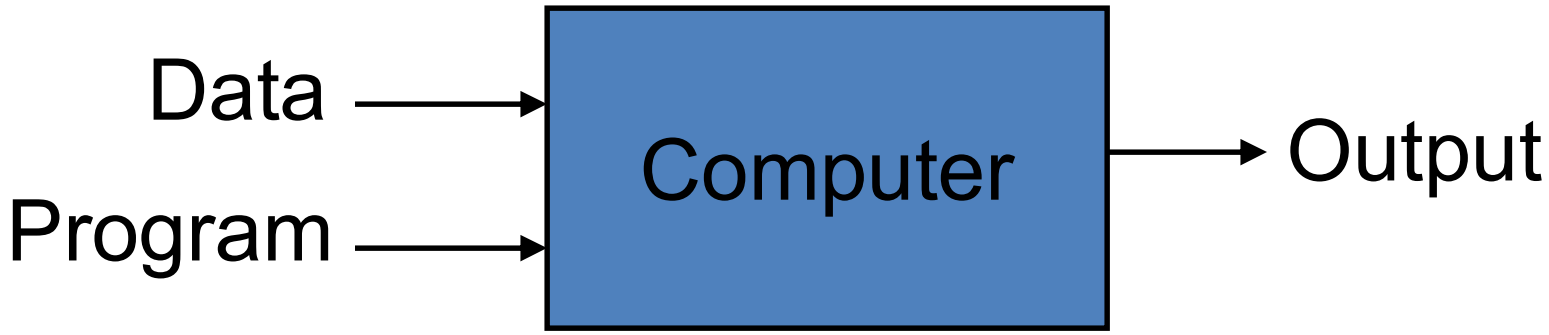
- Channel estimation

f(

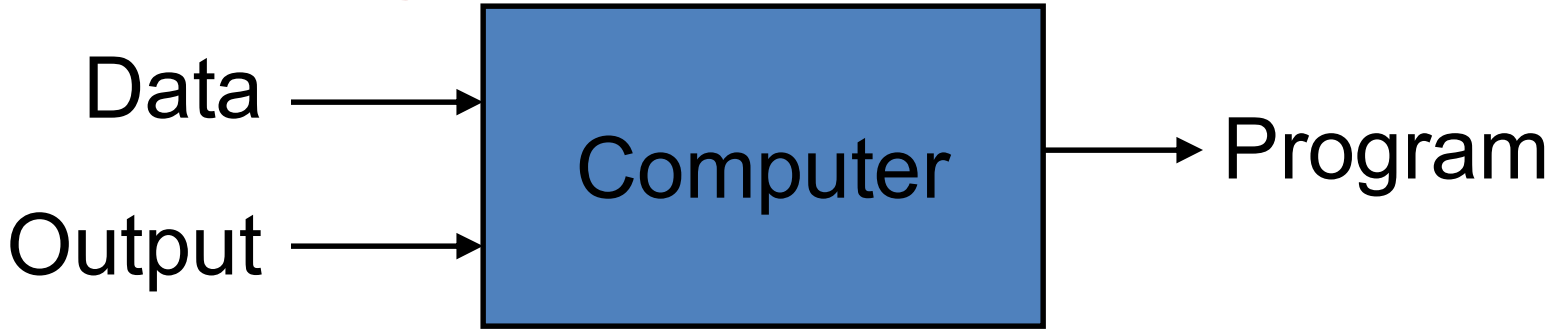
) = Channel parameters

http://cdn1.itpro.co.uk/sites/itpro/files/images/dir_176/it_photo_88225.jpg
http://www.berkeleywellness.com/sites/default/files/field/image/ThinkstockPhotos-520490716_field_img_hero_988_380.jp
https://telcoantennas.com.au/site/sites/default/files/images/4G-cross-polarisation-low-signal-areas.png

Traditional Programming



Machine Learning



Magic?

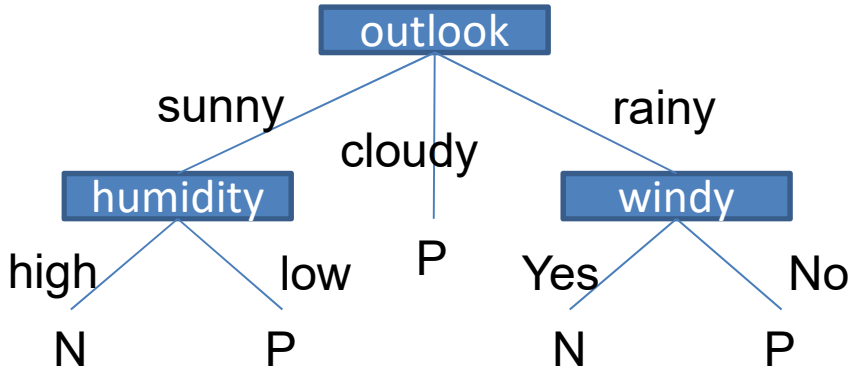
No, more like gardening

- **Seeds** = Algorithms
- **Nutrients** = Data
- **Gardener** = You
- **Plants** = Programs

Decision Tree/Random Forest

A Decision-Tree Based Classification

- A decision tree of whether going to play tennis or not:

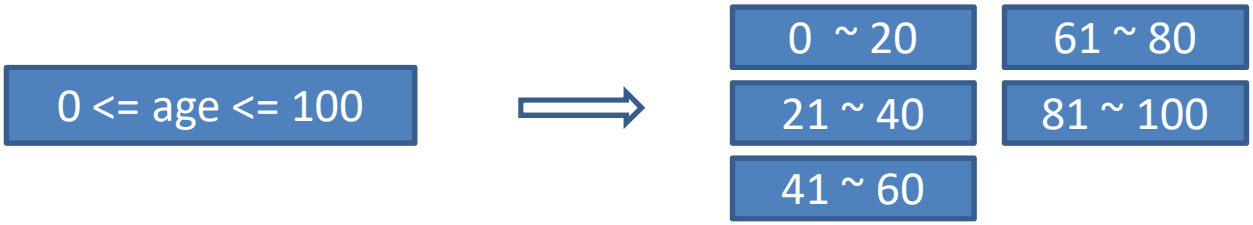


- **ID-3** and its extended version **C4.5** (Quinlan'93):
  A **top-down** decision tree generation algorithm

Algorithm for Decision Tree Induction (1/2)

- Basic algorithm (a **greedy algorithm**)
  - Tree is constructed in a top-down recursive divide-and-conquer manner.
  - Attributes are categorical.

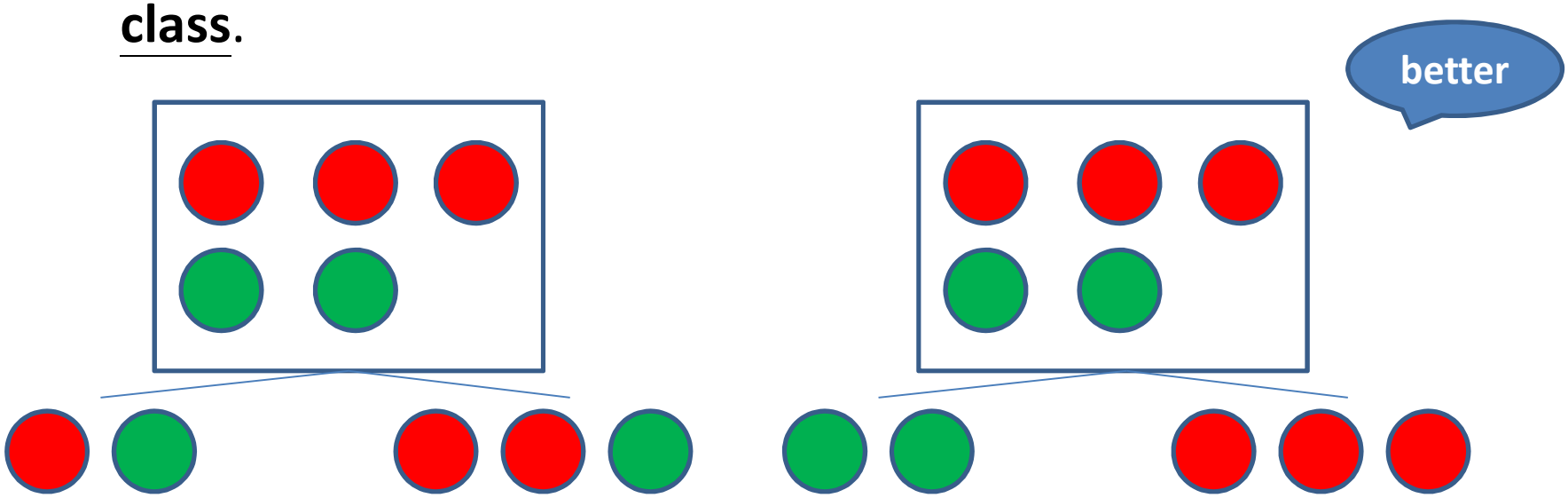
    (if an attribute is a continuous number, it needs to be discretized in advance.) E.g.



  - At start, all the training examples are at the root.
  - Examples are partitioned recursively based on selected attributes.

Algorithm for Decision Tree Induction (2/2)

- Basic algorithm (a **greedy algorithm**)
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain): maximizing an information gain measure, i.e., **favoring the partitioning which makes the majority of examples belong to a single class**.

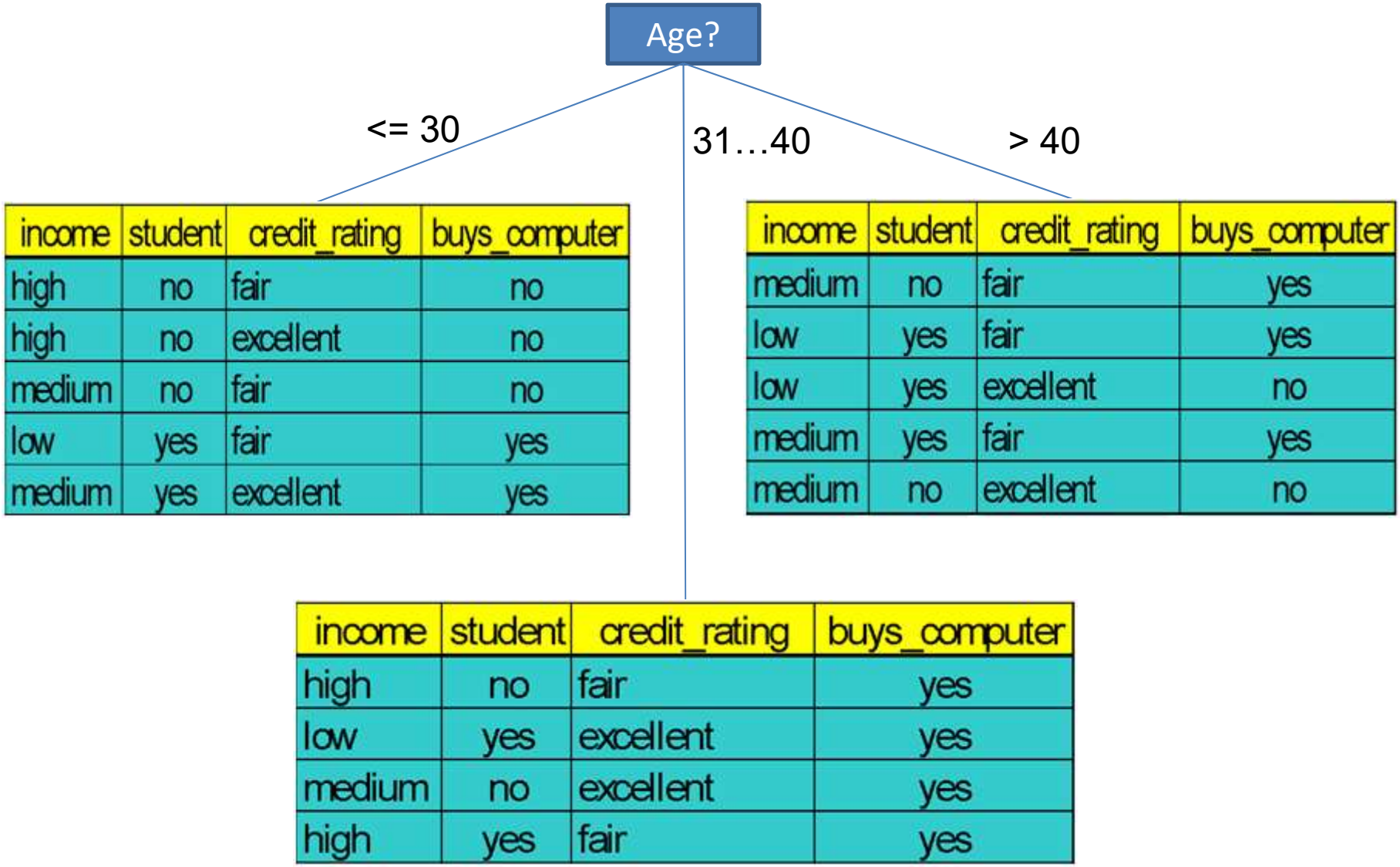


Algorithm for Decision Tree Induction (2/2)

- Basic algorithm (a **greedy algorithm**)
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain): maximizing an information gain measure, i.e., **favoring the partitioning which makes the majority of examples belong to a single class**.
  - Conditions for stopping partitioning:
    - All samples for a given node belong to the same class
    - There are no remaining attributes for further partitioning – majority voting is employed for classifying the leaf
    - There are no samples left

Decision Tree Induction: Training Dataset

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no



income	student	credit_rating	buys_computer
high	no	fair	no
high	no	excellent	no
medium	no	fair	no
low	yes	fair	yes
medium	yes	excellent	yes

income	student	credit_rating	buys_computer
medium	no	fair	yes
low	yes	fair	yes
low	yes	excellent	no
medium	yes	fair	yes
medium	no	excellent	no

income	student	credit_rating	buys_computer
high	no	fair	yes
low	yes	excellent	yes
medium	no	excellent	yes
high	yes	fair	yes

Primary Issues in Tree Construction (1/2)

- **Split criterion**: *Goodness function*
  - Used to select the attribute to be split at a tree node during the tree generation phase
  - Different algorithms may use different goodness functions:
    - Information gain (used in ID3/C4.5)
    - Gini index (used in CART)

Primary Issues in Tree Construction (2/2)

- **Branching scheme**:
  - Determining the tree branch to which a sample belongs
  - Binary vs. *k-ary* splitting
- When to stop the further splitting of a node? e.g. impurity measure

- Labeling rule: a node is labeled as the class to which most samples at the node belongs.

How to Use a Tree?

- Directly
  - Test the attribute value of unknown sample against the tree.
  - A path is traced from root to a leaf which holds the label.
- Indirectly
  - Decision tree is converted to classification rules.
  - One rule is created for each path from the root to a leaf.
  - **IF-THEN** is easier for humans to understand .

Attribute Selection Measure: **Information Gain** (ID3/C4.5)

- **Select the attribute with the highest information gain**
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- Expected information (**entropy**) needed to classify a tuple in D:

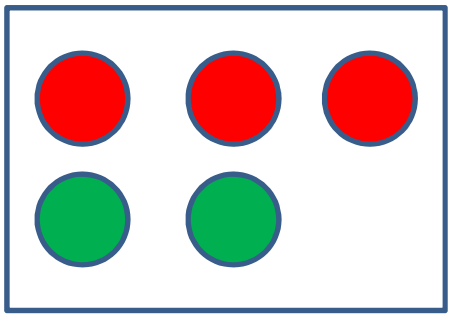
$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

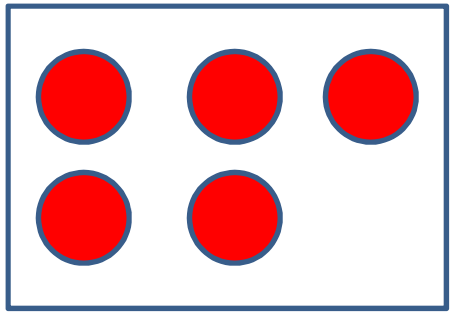
- Expected information (**entropy**):
  - Entropy is a measure of how "mixed up" an attribute is.
  - It is sometimes equated to the purity or impurity of a variable.
  - High Entropy means that we are sampling from a uniform (boring) distribution.

Expected Information (Entropy)

- Expected information (**entropy**) needed to classify a tuple in D:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$ (m: number of labels)



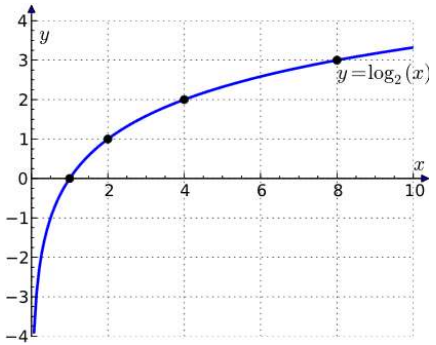


$$Info(D) = I(3,2) = -\frac{3}{5}\log_2(\frac{3}{5}) - \frac{2}{5}\log_2(\frac{2}{5})$$

$$\approx -\frac{3}{5}\times(-0.737) - \frac{2}{5}\times(-1.322)$$

$$Info(D) = I(5,0) = -\frac{5}{5}\log_2(\frac{5}{5}) - \frac{0}{5}\log_2(\frac{0}{5})$$

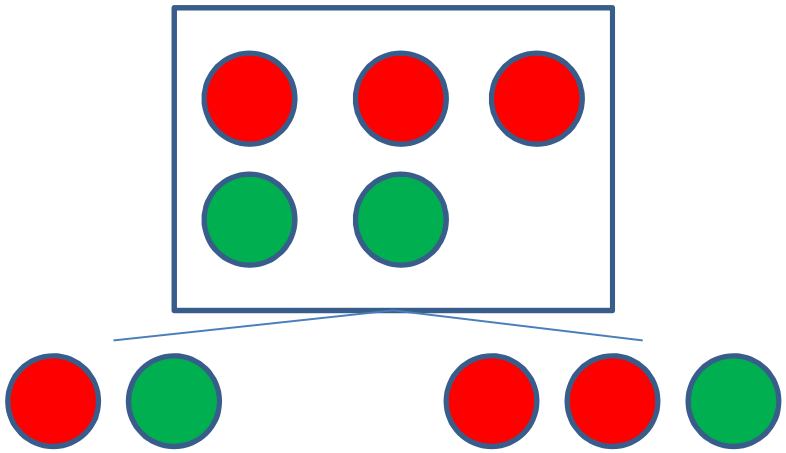
$$= 0 - 0 = 0$$



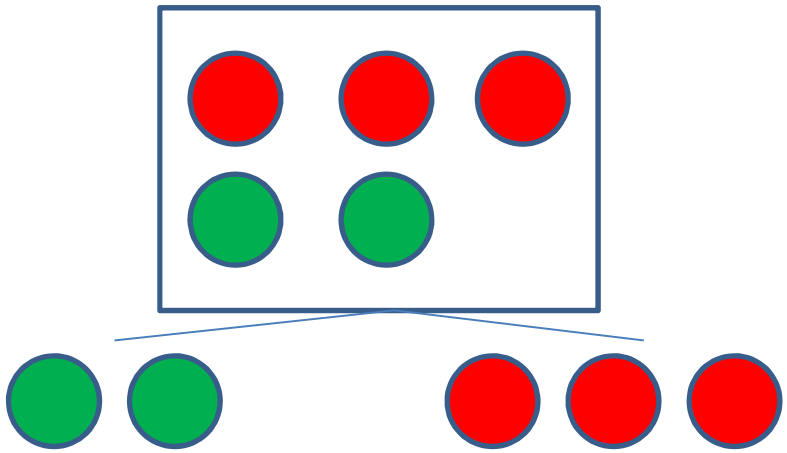
Expected Information (Entropy)

- Information needed (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times I(D_j)$$



$$Info(D) = \frac{2}{5} Info(1,1) + \frac{3}{5} Info(2,1)$$



$$Info(D) = \frac{2}{5} Info(2,0) + \frac{3}{5} Info(3,0)$$

Attribute Selection Measure: **Information Gain** (ID3/C4.5)

- **Select the attribute with the highest information gain**
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i, estimated by $|C_{i,D}|/|D|$
- Expected information (**entropy**) needed to classify a tuple in D:

$$Info(D) = I(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

- Information needed (after using A to split D into v partitions) to classify D:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times I(D_j)$$

- Information gained by branching on attribute A

$$Gain(A) = Info(D) - Info_A(D)$$

Attribute Selection: Information Gain

- Class P: buys_computer = "yes"
- Class N: buys_computer = "no"

$$Info(D) = I(9,5) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$$

$$Info_{age}(D) = \frac{5}{14}I(2,3) + \frac{4}{14}I(4,0) + \frac{5}{14}I(3,2) = 0.694$$

age	p_i	n_i	$I(p_i, n_i)$
<=30	2	3	0.971
31…40	4	0	0
>40	3	2	0.971

$\frac{5}{14}I(2,3)$ means "age <=30" has 5 out of 14 samples, with 2 yes'es and 3 no's. Hence

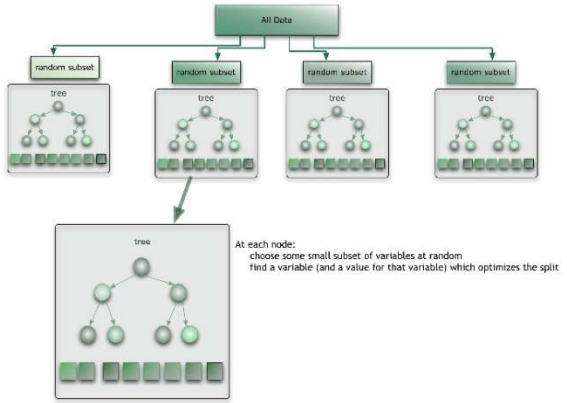
$$Gain(age) = Info(D) - Info_{age}(D) = 0.246$$

Similarly,

$$Gain(income) = 0.029$$
$$Gain(student) = 0.151$$
$$Gain(credit\_rating) = 0.048$$

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Random Forest

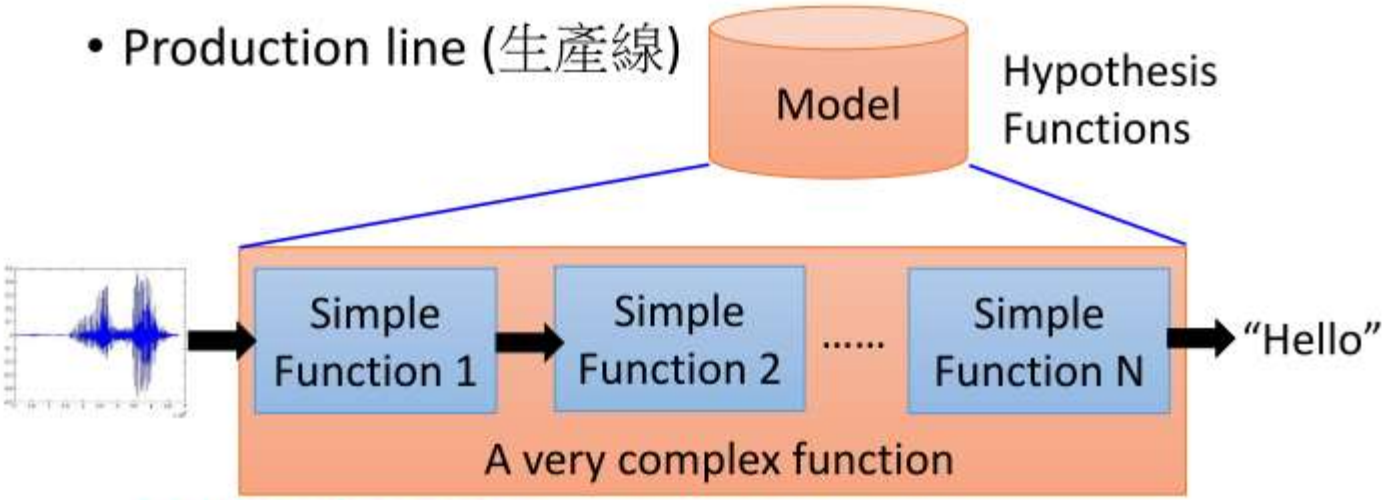


- Random Forest:
  - Each classifier in the ensemble is a ***decision tree*** classifier and is generated using a random selection of attributes at each node to determine the split
  - During classification, each tree votes and the most popular class is returned
- Two Methods to construct Random Forest:
  - Forest-RI (*random input selection*): Randomly select, at each node, F attributes as candidates for the split at the node. The CART methodology is used to grow the trees to maximum size
  - Forest-RC (*random linear combinations*)*:* Creates new attributes (or features) that are a linear combination of the existing attributes (reduces the correlation between individual classifiers)
- Comparable in accuracy to Adaboost, but more robust to errors and outliers
- Insensitive to the number of attributes selected for consideration at each split, and faster than bagging or boosting

Deep Learning

What is Deep Learning?

- Production line (生產線)

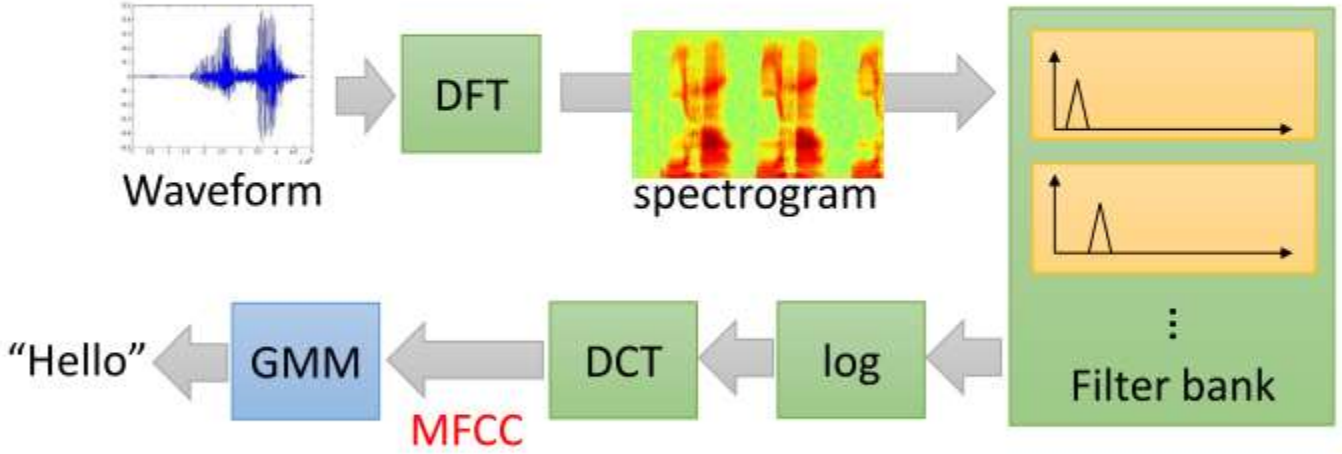


End-to-end training:

What each function should do is learned automatically

Deep v.s. Shallow - Speech Recognition

- Shallow Approach



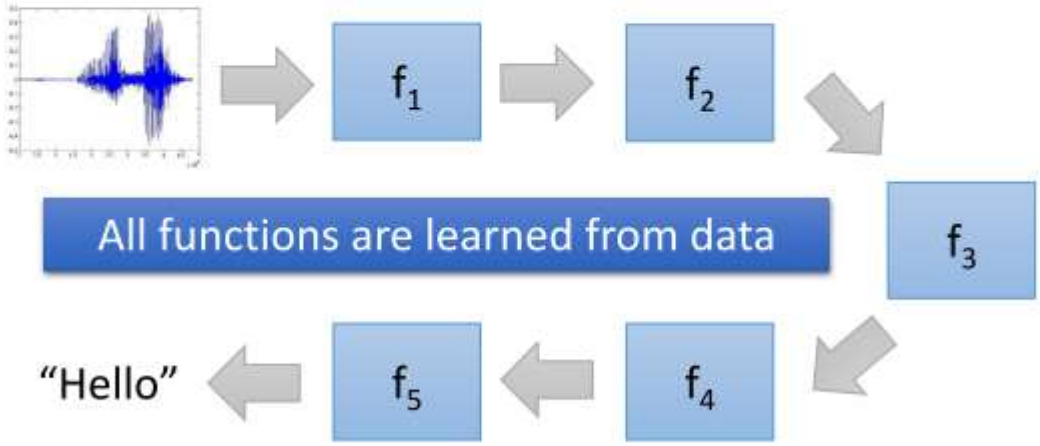
Each box is a simple function in the production line:

:hand-crafted :learned from data

Deep v.s. Shallow - Speech Recognition

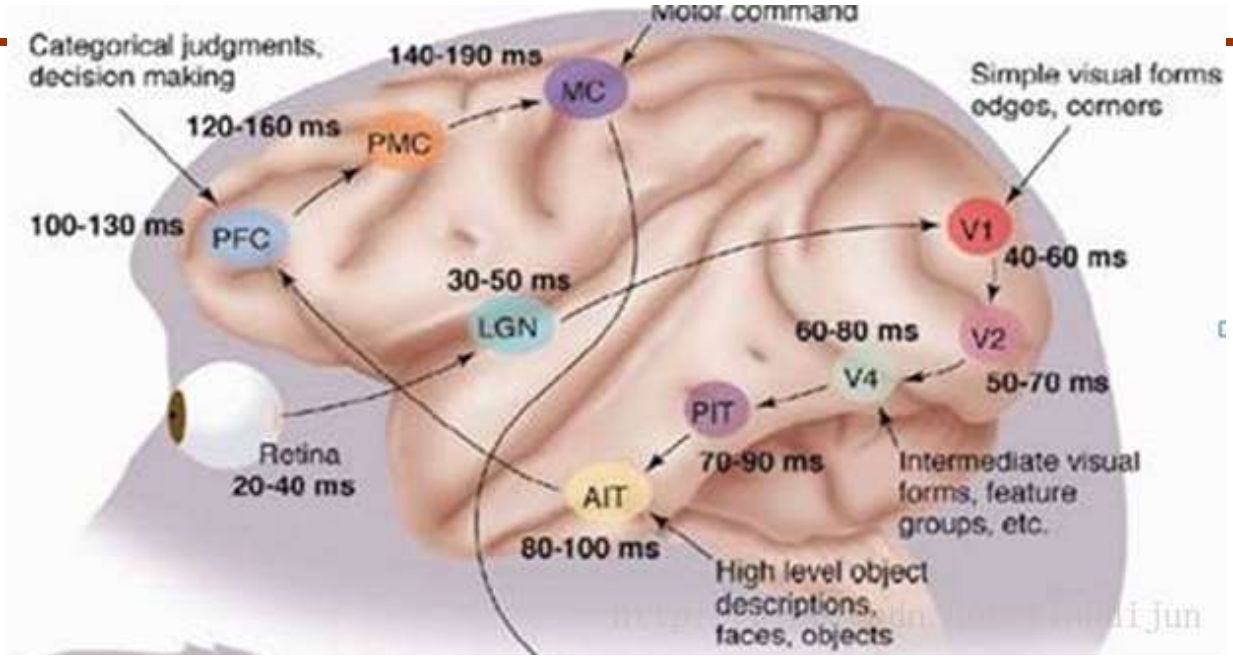
- Deep Learning

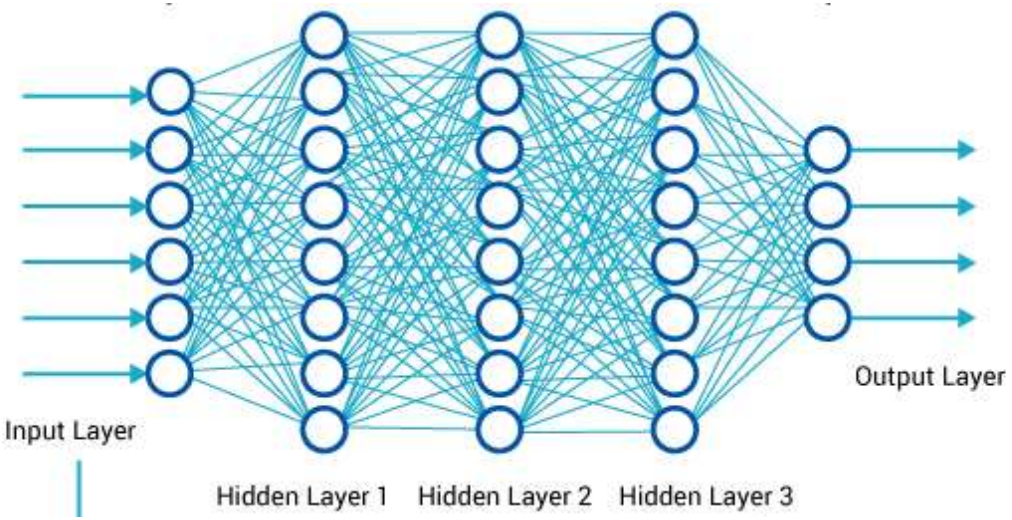
"Bye bye, MFCC"
- Deng Li in Interspeech 2014



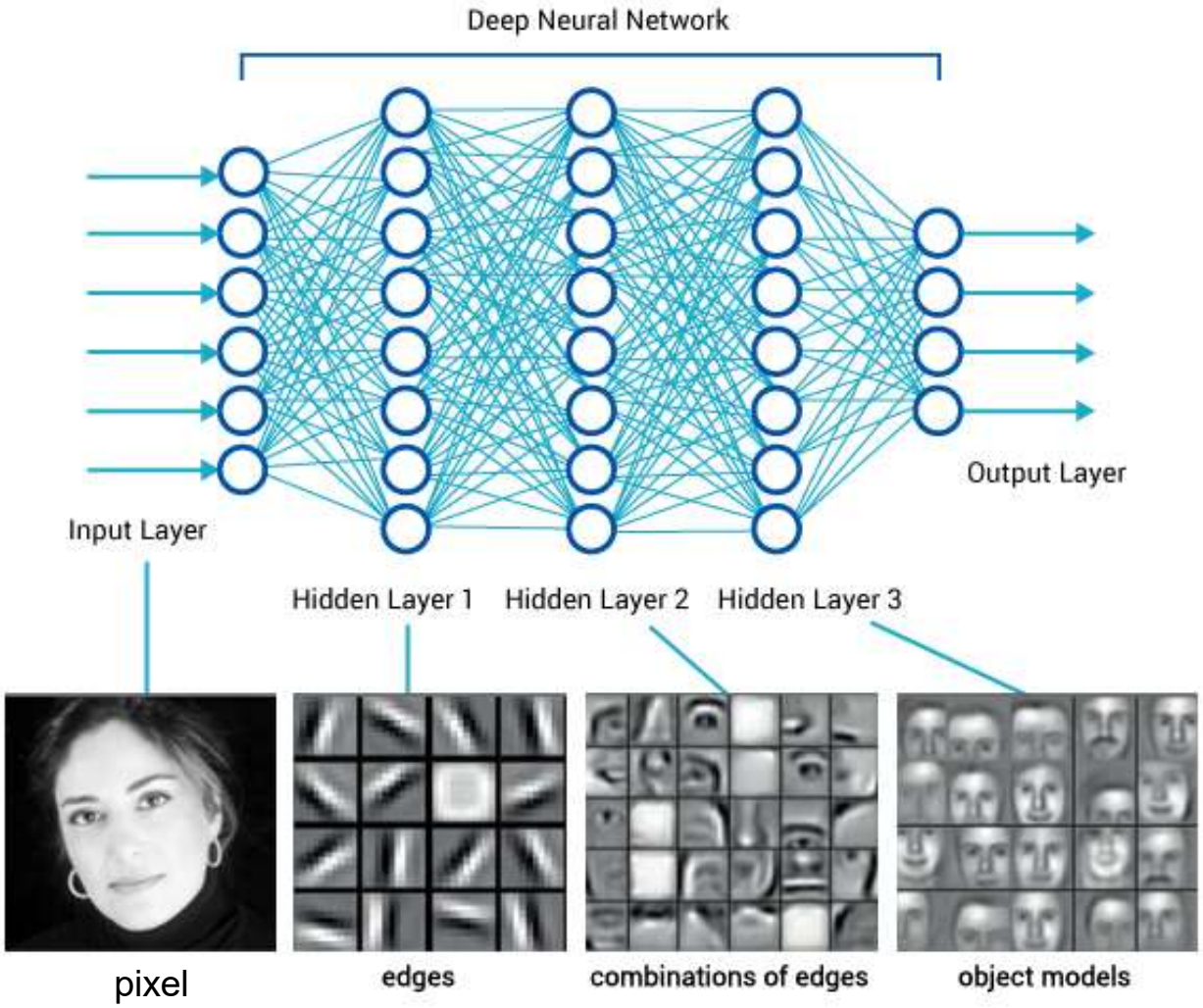
Less engineering labor, but machine learns more

Idea from human brain



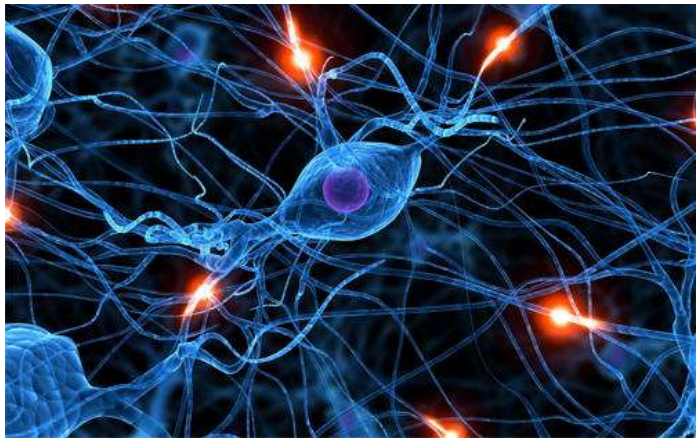


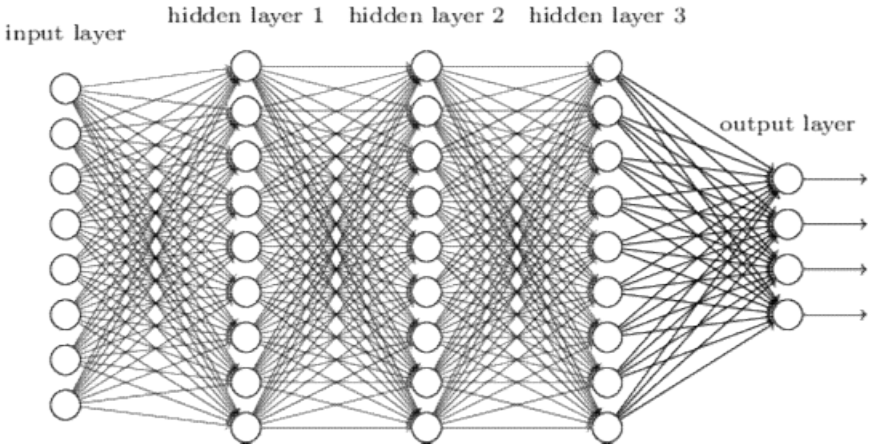
Artificial Neuron Network(ANN)



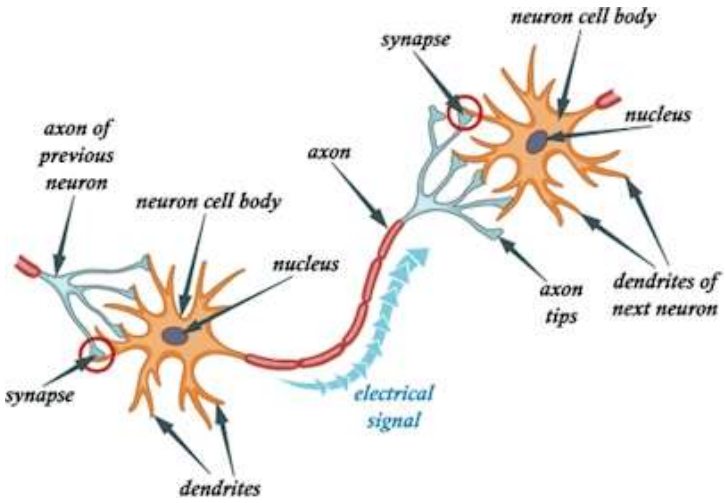
Source:http://www.amax.com/blog/?p=804

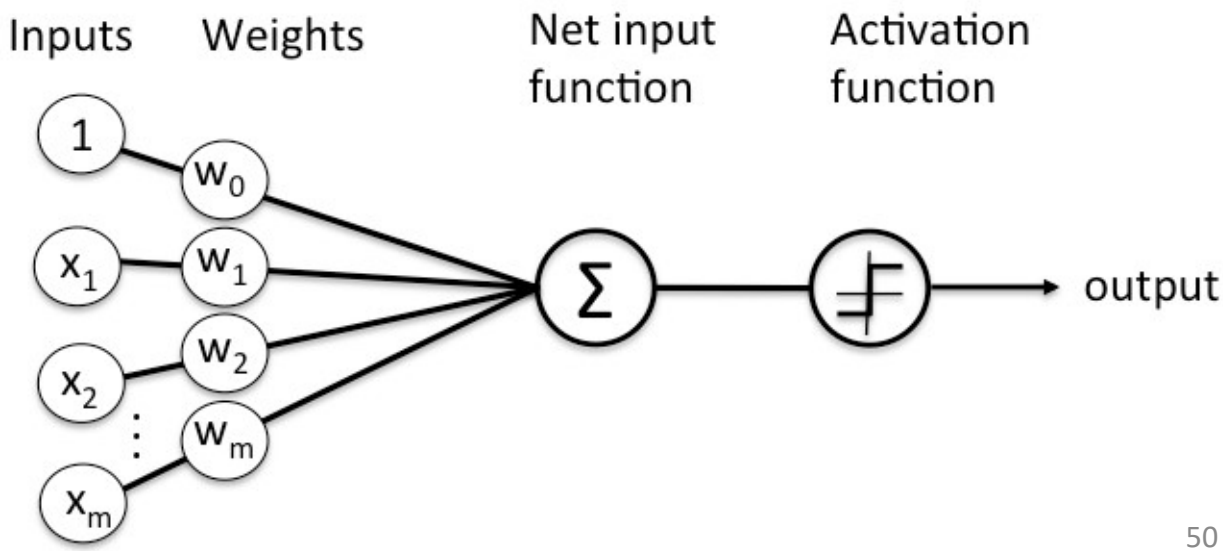
Neuron Network



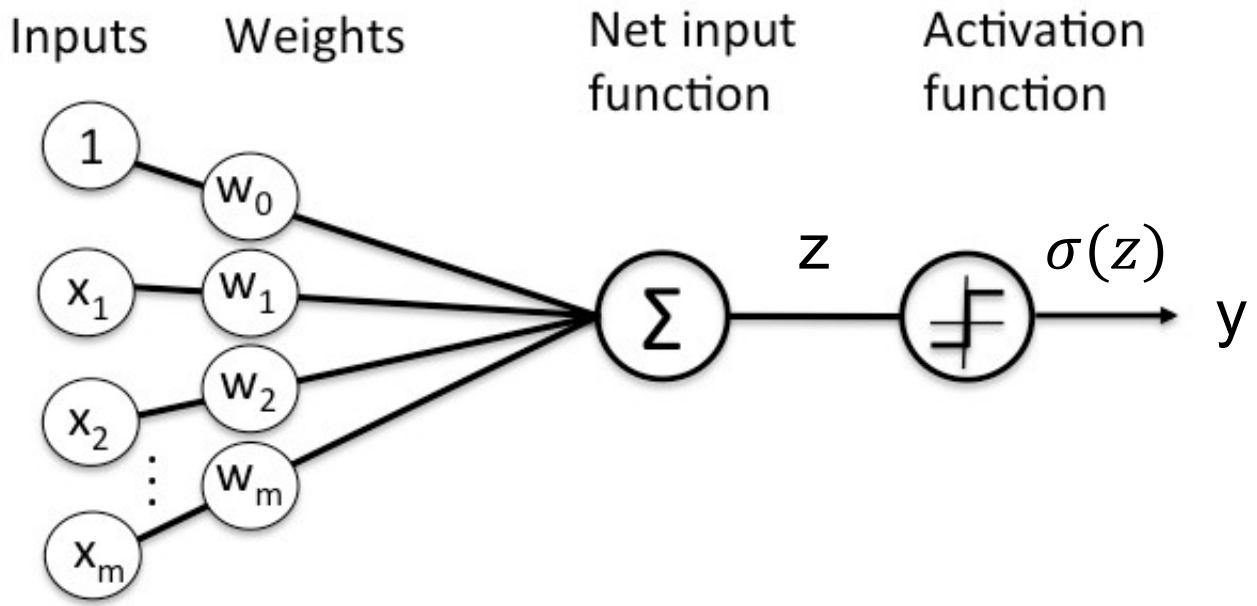


Neuron


synapse
neuron cell body
axon of previous neuron
axon
nucleus
neuron cell body
nucleus
axon tips
dendrites of next neuron
synapse
electrical signal
dendrites


Inputs
Weights
Net input function
Activation function
1
w_0
x_1
w_1
x_2
w_2
w_m
x_m
Σ
output

Neuron

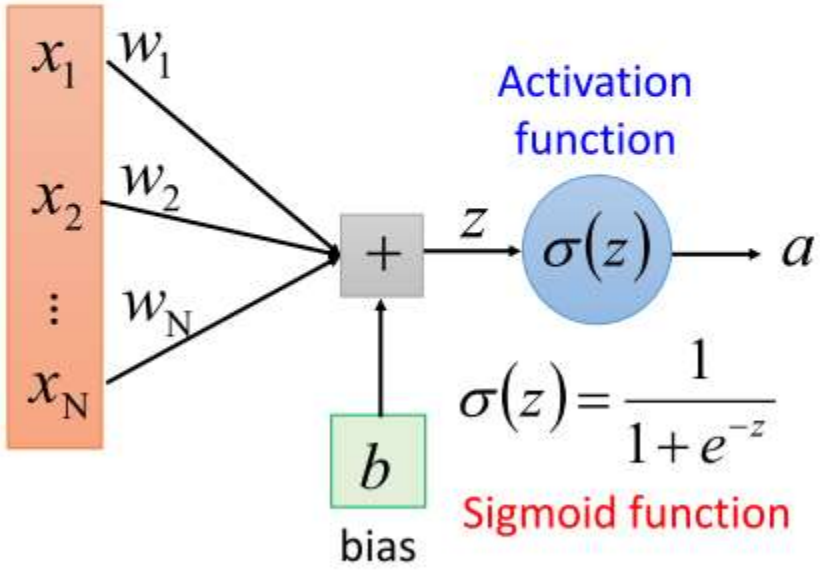


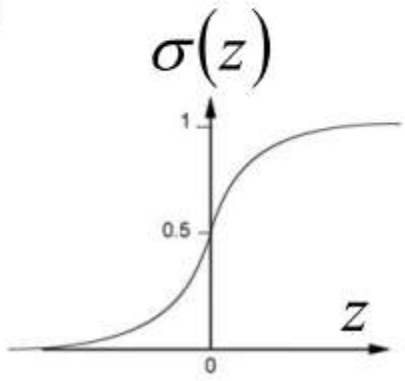
$z = 1^{*}w_0 + x_1{}^{*}w_1 + x_2{}^{*}w_2 + \ldots + x_m{}^{*}w_m$

$y = \sigma(z)$

A Neuron for Machine

Each neuron is a very simple function

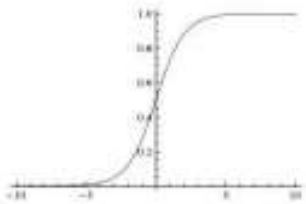




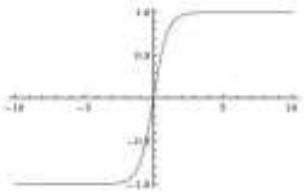
Activation Functions

Sigmoid

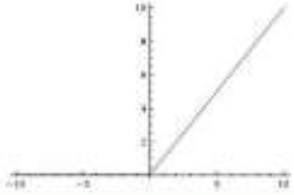
$$\sigma(x) = 1/(1+e^{-x})$$



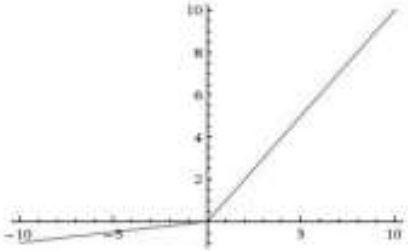
tanh tanh(x)



ReLU max(0,x)



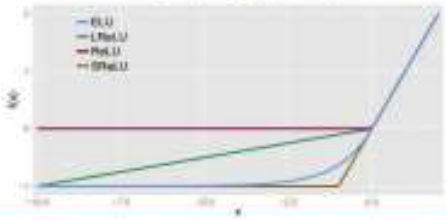
Leaky ReLU
max(0.1x, x)



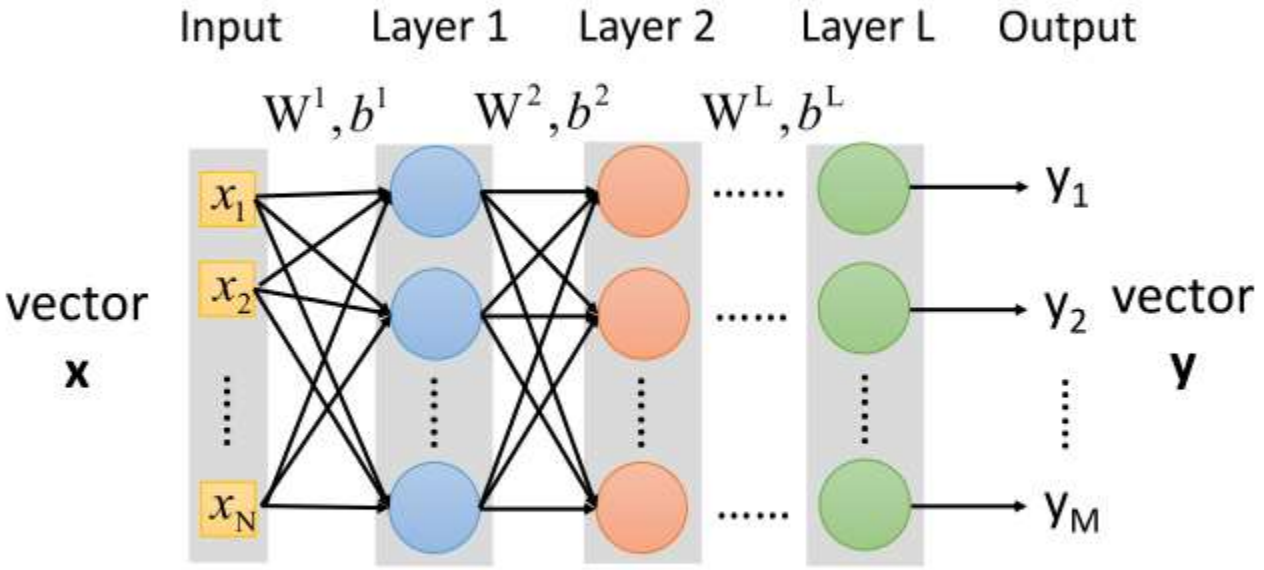
Maxout $\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha\,(\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$$



Function of Neural Network



$$y = f(x)$$
$$= \sigma\left(W^L \ldots \sigma\left(W^2 \sigma\left(W^1 x + b^1\right) + b^1\right) \ldots + b^L\right)$$

Best Function = Best Parameters

$$y = f(x) = \sigma\left(W^L \ldots \sigma\left(W^2 \sigma\left(W^1 x + b^1\right) + b^1\right) \ldots + b^L\right)$$

function set

because different parameters W and b lead to different function

Formal way to define a function set:

$f(x;\theta)$ → parameter set

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

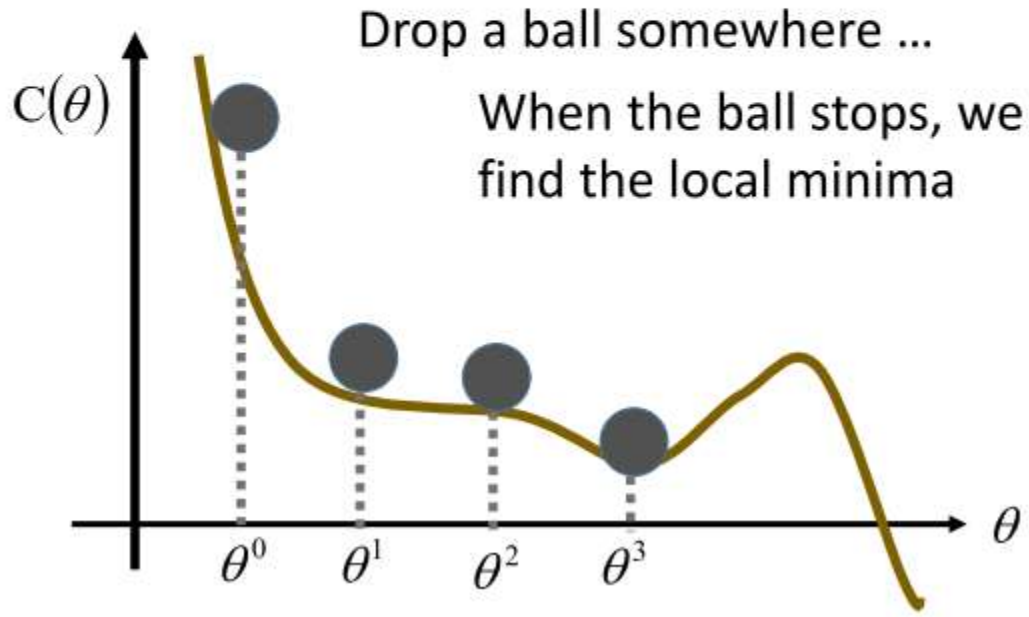
Pick the "best" function f* → Pick the "best" parameter set θ*

Statement of Problems

- Statement of problems:
  - There is a function C(θ)
    - θ represents parameter set
    - $\theta = \{\theta_1, \theta_2, \theta_3, \ldots\ldots\}$
  - Find θ^* that minimizes C(θ)
- Brute force?
  - Enumerate all possible θ
- Calculus?
  - Find θ^* such that $\left.\frac{\partial C(\theta)}{\partial \theta_1}\right|_{\theta=\theta^*} = 0, \left.\frac{\partial C(\theta)}{\partial \theta_2}\right|_{\theta=\theta^*} = 0, \ldots$

Gradient Descent – Idea

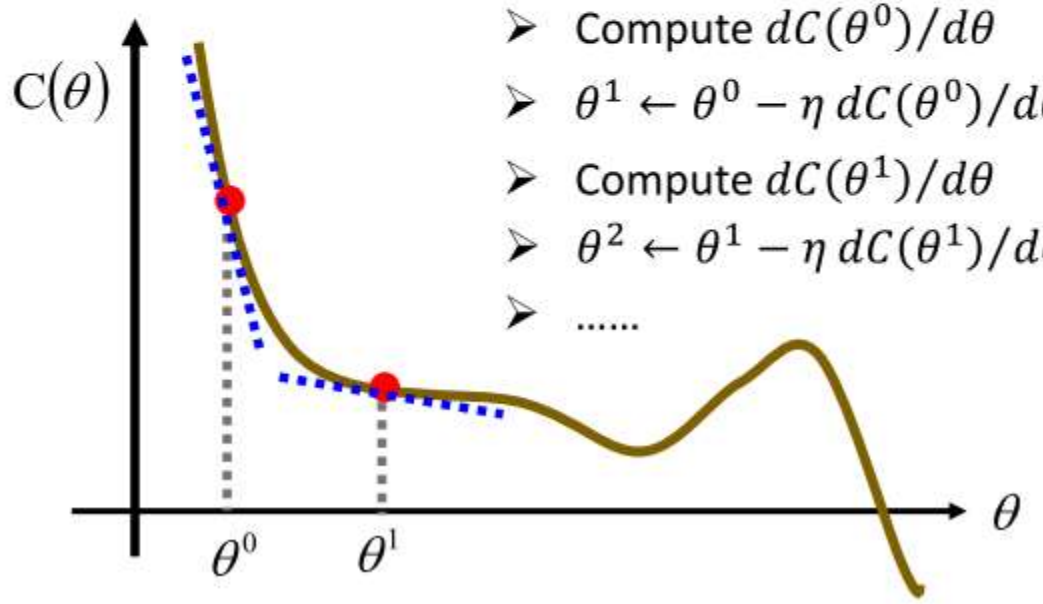
- For simplification, first consider that θ has only one variable



Gradient Descent – Idea

η is called *"learning rate"*

- For simplification, first consider that θ has only one variable

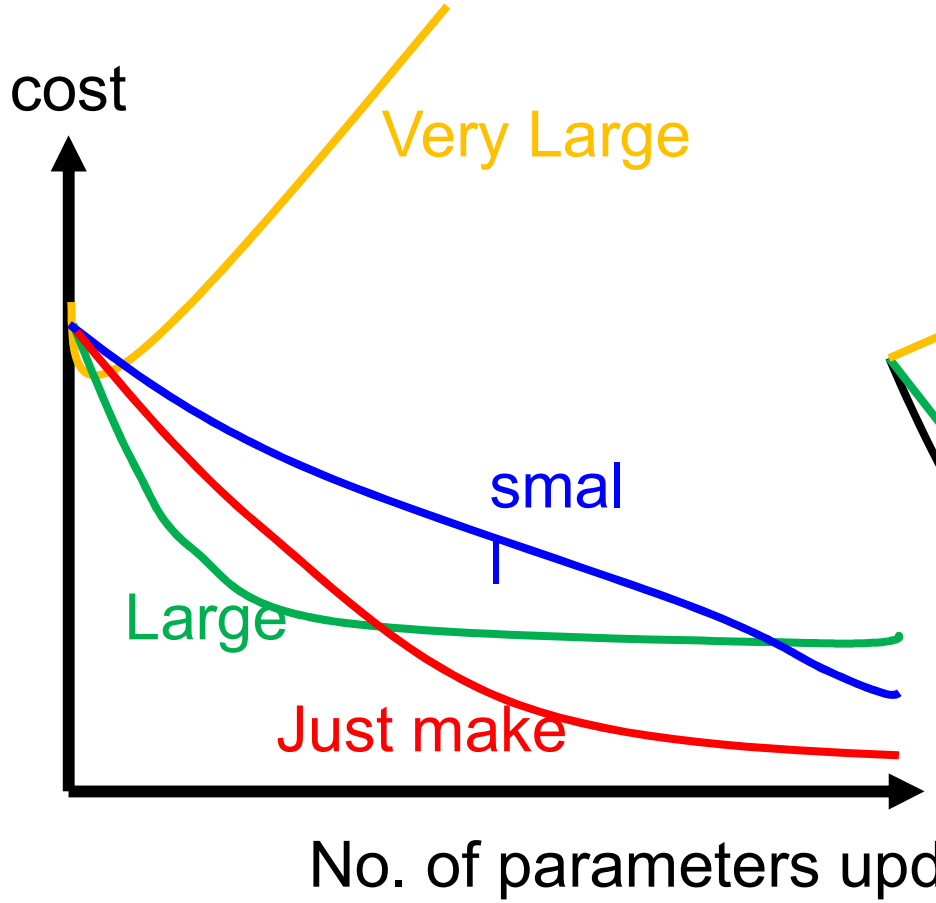


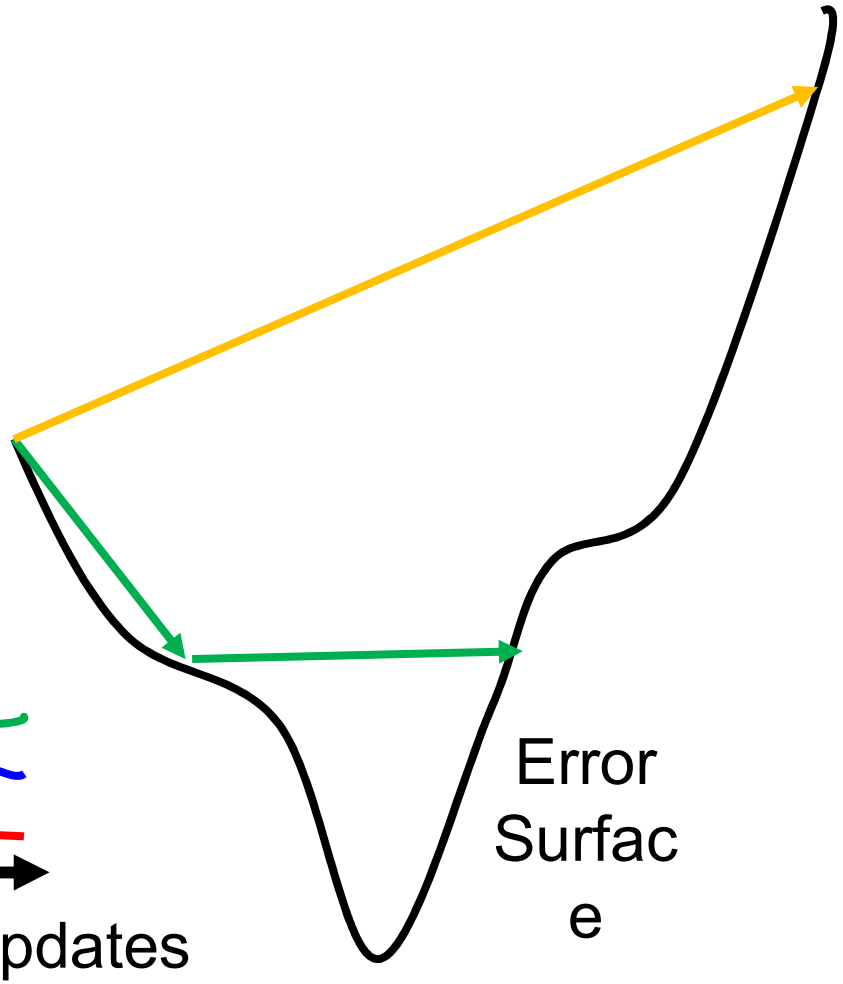
- Randomly start at θ^0
- Compute $dC(\theta^0)/d\theta$
- $\theta^1 \leftarrow \theta^0 - \eta\, dC(\theta^0)/d\theta$
- Compute $dC(\theta^1)/d\theta$
- $\theta^2 \leftarrow \theta^1 - \eta\, dC(\theta^1)/d\theta$
- .....

Learning Rate

$$\theta^i = \theta^{i-1} - \eta \nabla C\left(\theta^{i-1}\right)$$

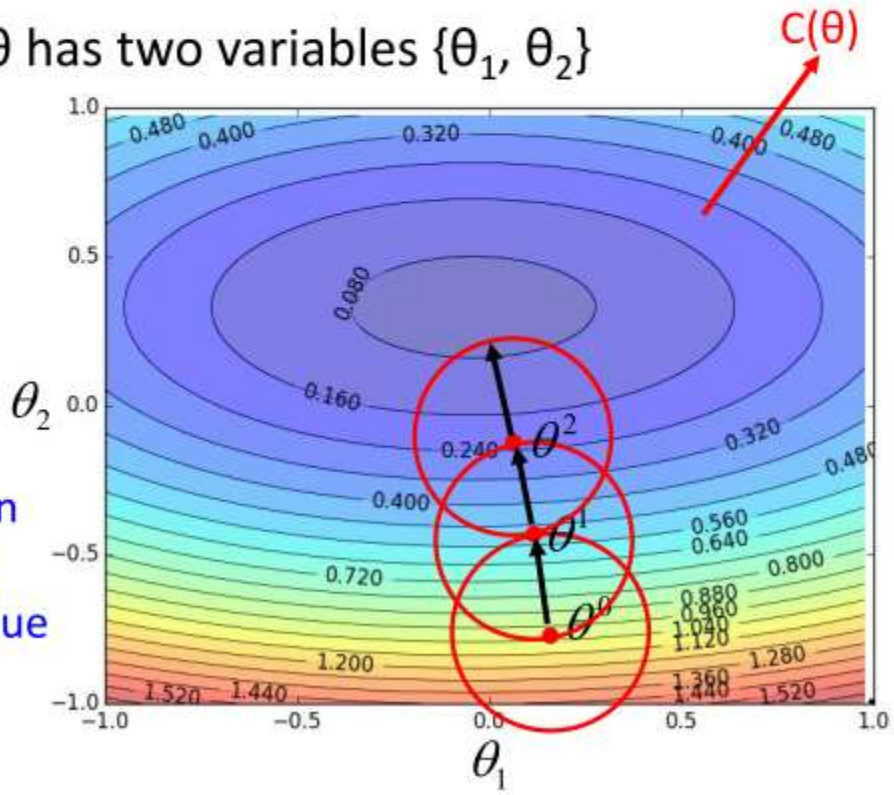
- Set the learning rate η carefully





Formal Derivation of Gradient Descent

- Suppose that θ has two variables $\{\theta_1, \theta_2\}$



Given a point, we can easily find the point with the smallest value nearby. How?

Stochastic Gradient Descent and Mini-batch

$$C(\theta)=\frac{1}{R}\sum_{r}\left\|f\left(x^r;\theta\right)-\hat{y}^r\right\|$$

$$=\frac{1}{R}\sum_{r}C^r\left(\theta\right)$$

◆ ***Gradient Descent***

$$\theta^i=\theta^{i-1}-\eta\nabla C\left(\theta^{i-1}\right)$$

$$\nabla C\left(\theta^{i-1}\right)=\frac{1}{R}\sum_{r}\nabla C^r\left(\theta^{i-1}\right)$$

◆ ***Stochastic Gradient Descent*** Faster! Better!

Pick an example x^r

$$\theta^i=\theta^{i-1}-\eta\nabla C^r\left(\theta^{i-1}\right)$$

If all example x^r have equal probabilities to be picked

$$E\left[\nabla C^r\left(\theta^{i-1}\right)\right]=\frac{1}{R}\sum_{r}\nabla C^r\left(\theta^{i-1}\right)$$

Stochastic Gradient Descent and Mini-batch

What is epoch?

Training Data: $\left\{\left(x^1, \hat{y}^1\right), \left(x^2, \hat{y}^2\right), \cdots \left(x^r, \hat{y}^r\right), \cdots \left(x^R, \hat{y}^R\right)\right\}$

When using stochastic gradient descent

Starting at θ_0

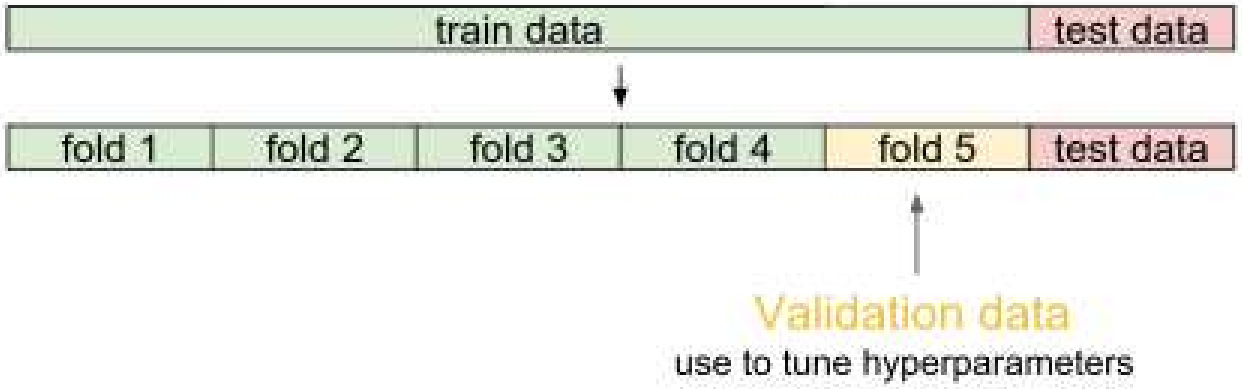
pick x^1	$\theta^1 = \theta^0 - \eta \nabla C^1\left(\theta^0\right)$	
pick x^2	$\theta^2 = \theta^1 - \eta \nabla C^2\left(\theta^1\right)$	
⋮	⋮	Seen all the examples once
pick x^r	$\theta^r = \theta^{r-1} - \eta \nabla C^r\left(\theta^{r-1}\right)$	
⋮	⋮	One epoch
pick x^R	$\theta^R = \theta^{R-1} - \eta \nabla C^R\left(\theta^{R-1}\right)$	
pick x^1	$\theta^{R+1} = \theta^R - \eta \nabla C^1\left(\theta^R\right)$	

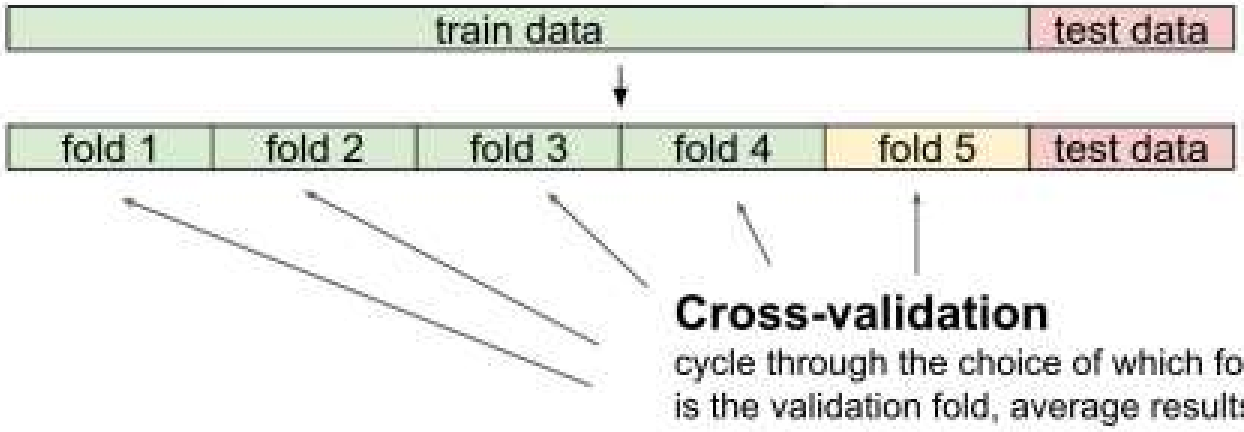
Try out what hyperparameters work best on test set.

Trying out what hyperparameters work best on test set:

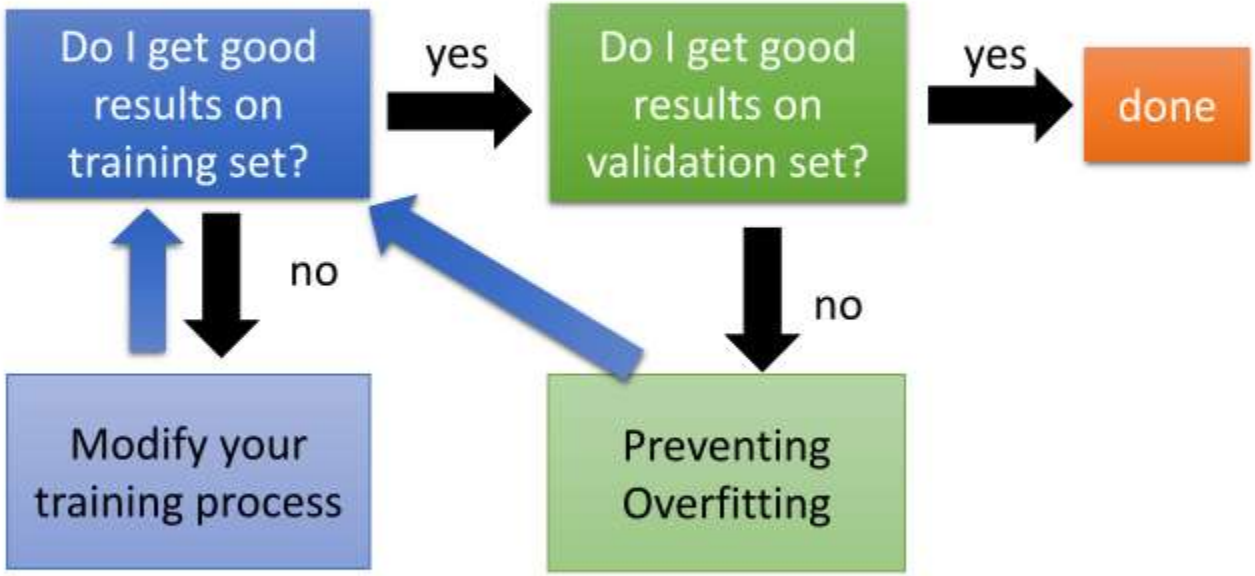
Very bad idea. The test set is a proxy for the generalization performance!

Use only **VERY SPARINGLY**, at the end.


train data
test data
fold 1
fold 2
fold 3
fold 4
fold 5
test data
Validation data
use to tune hyperparameters


train data
test data
fold 1
fold 2
fold 3
fold 4
fold 5
test data
Cross-validation
cycle through the choice of which fold
is the validation fold, average results.

Recipe for Learning



- Your code usually do not have bug at this situation.

Recipe for Learning - Overfitting

- You pick a "best" parameter set θ^*

Training Data: $\{\ldots(x^r, \hat{y}^r)\ldots\} \Rightarrow \forall r : f(x^r; \theta^*) = \hat{y}^r$

However,

Testing Data: $\{\ldots x^u \ldots\} \Rightarrow f(x^u; \theta^*) \neq \hat{y}^u$

Training data and testing data have different distribution.

Training Data:

Testing Data:

Interesting Project Review (with Python)

2017-2018

FastText

- Library for fast text representation and classification
  - Courtesy of Facebook Research
- Recent state-of-the-art English word vectors.
- Word vectors for 157 languages trained on Wikipedia and Crawl.
- Models for language identification and various supervised tasks.

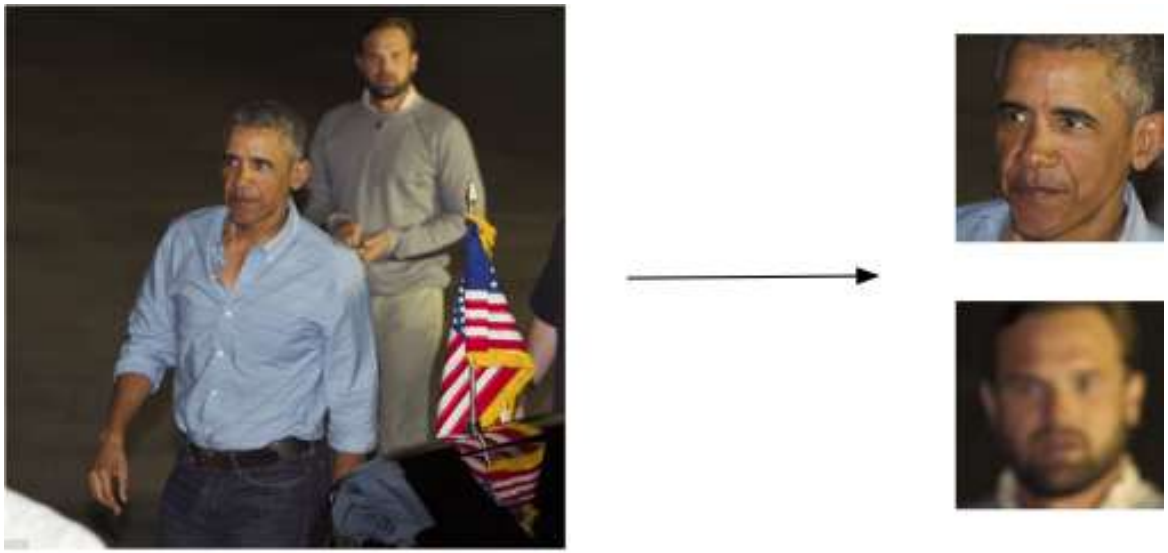
- https://github.com/facebookresearch/fastText?utm_source=mybridge&utm_medium=blog&utm_campaign=read_more

face_recognition

- Built using dlib's state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark.
- Features:
  - **Find faces in pictures**
  - **Find and manipulate facial features in pictures**
  - **Identify faces in pictures**
- https://github.com/ageitgey/face_recognition?utm_source=mybridge&utm_medium=blog&utm_campaign=read_more

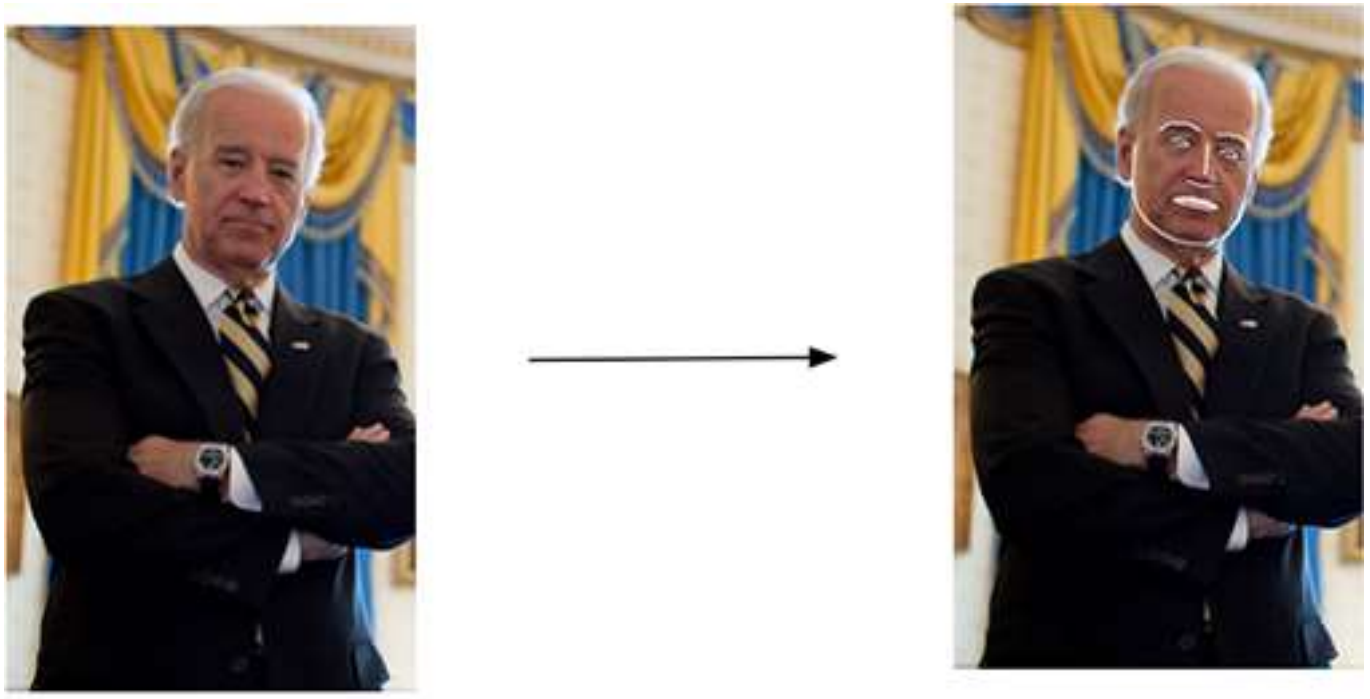
Find faces in pictures

```
import face_recognition
image = face_recognition.load_image_file("your_file.jpg")
face_locations = face_recognition.face_locations(image)
```



Input Output

Find and manipulate facial features in pictures



Input

Output

```
import face_recognition
image = face_recognition.load_image_file("your_file.jpg")
face_landmarks_list = face_recognition.face_landmarks(image)
```

Input

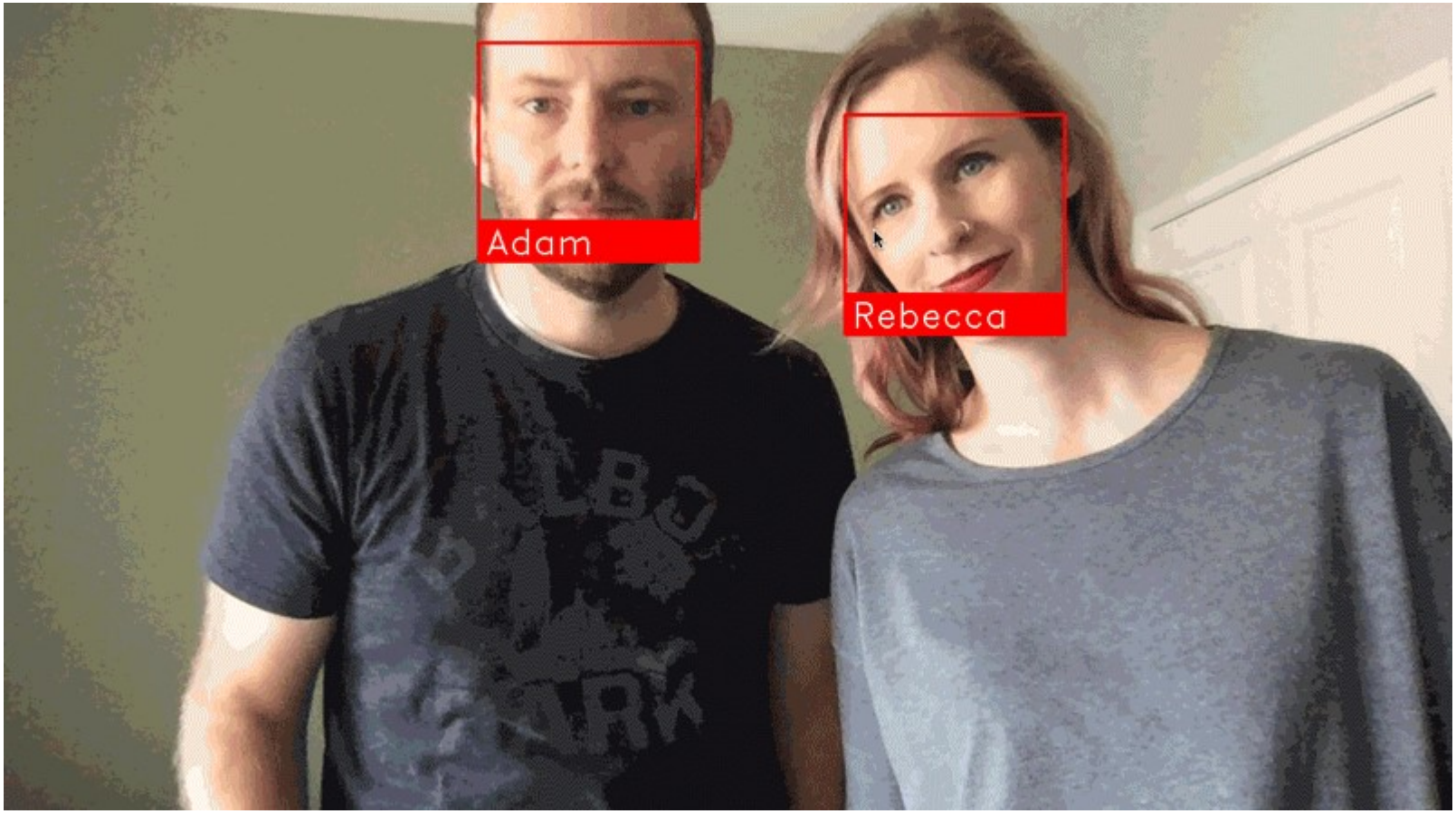
Output

Identify faces in pictures

Input

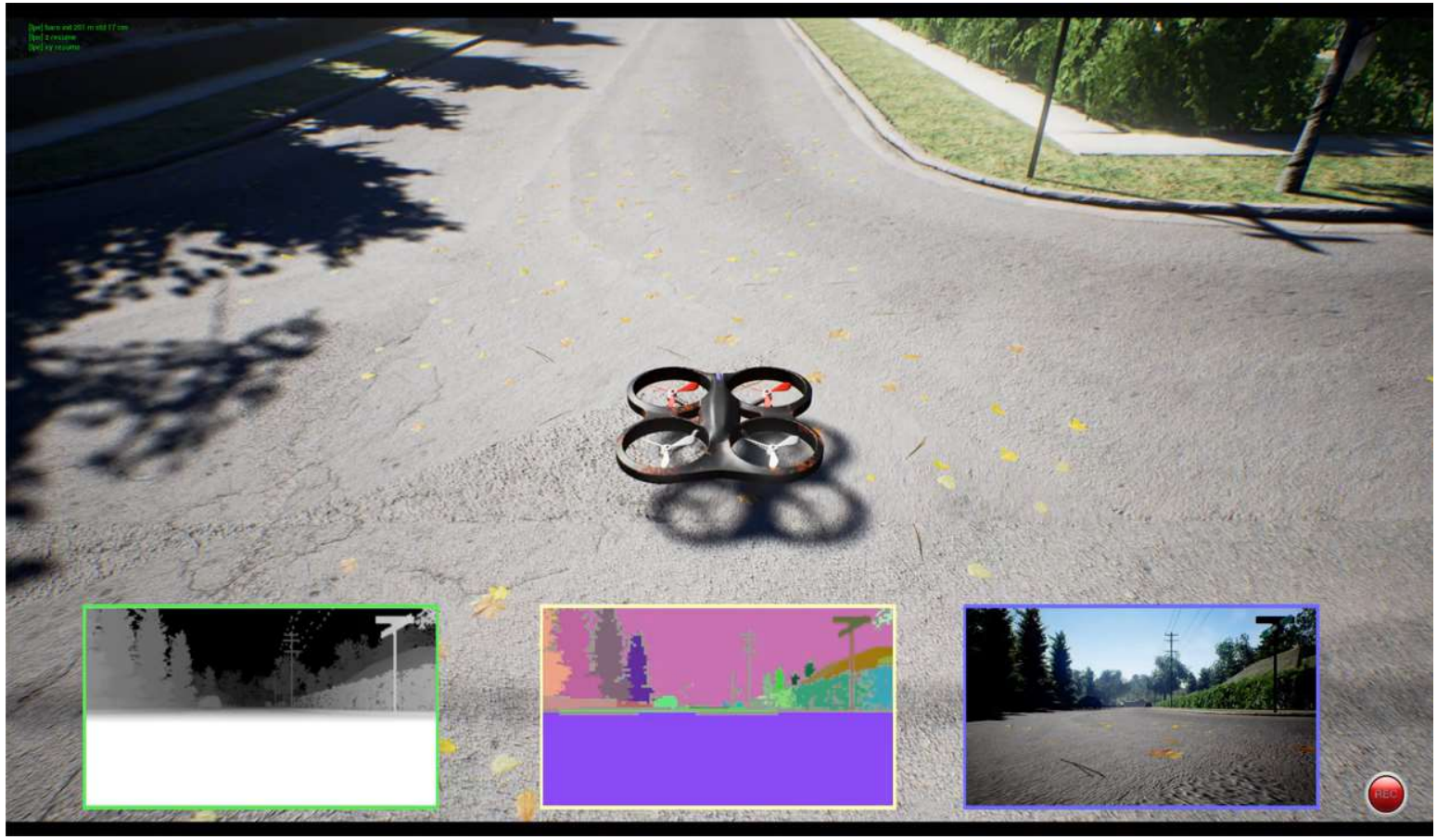
Picture contains
"Joe Biden"

Output


Adam
Rebecca

AirSim

- Open source simulator based on Unreal Engine for autonomous vehicles from Microsoft AI & Research



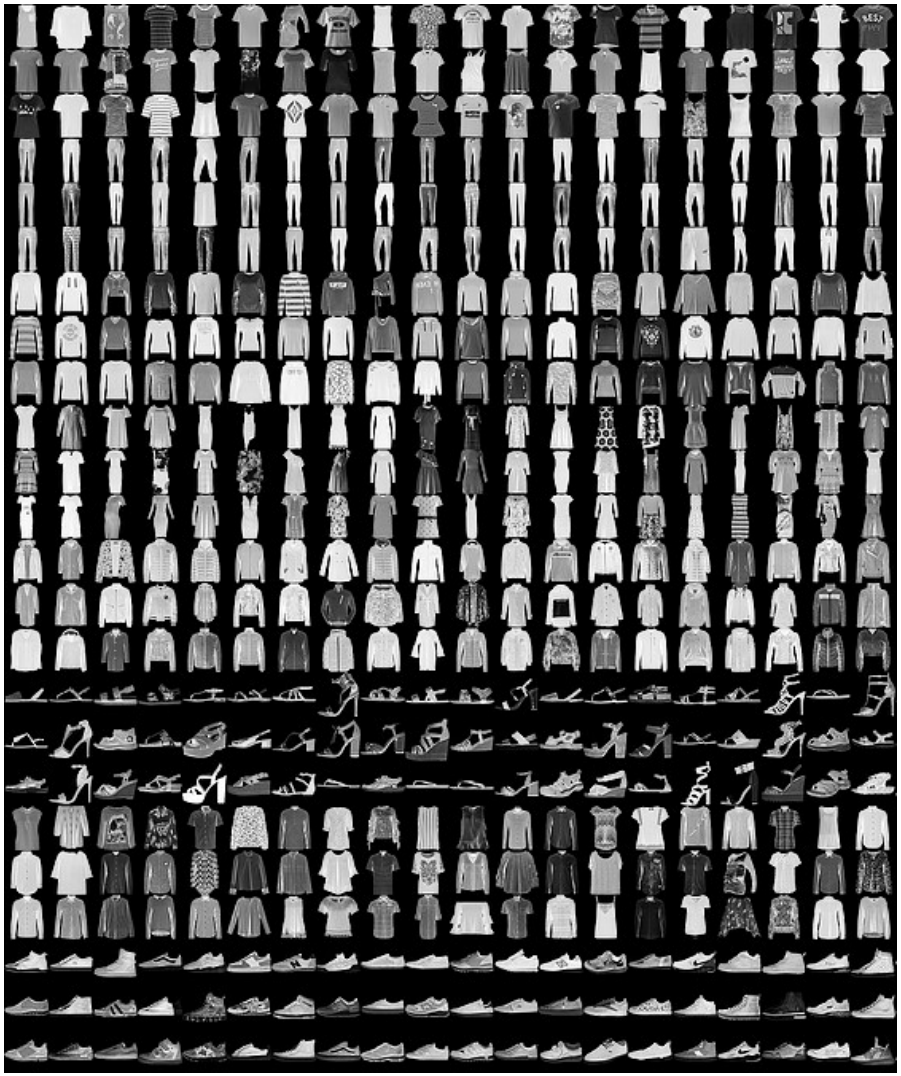
https://github.com/Microsoft/AirSim

CARLA Simulator

https://github.com/carla-simulator/carla

Fashion-MNIST

Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot



https://github.com/zalandoresearch/fashion-mnist

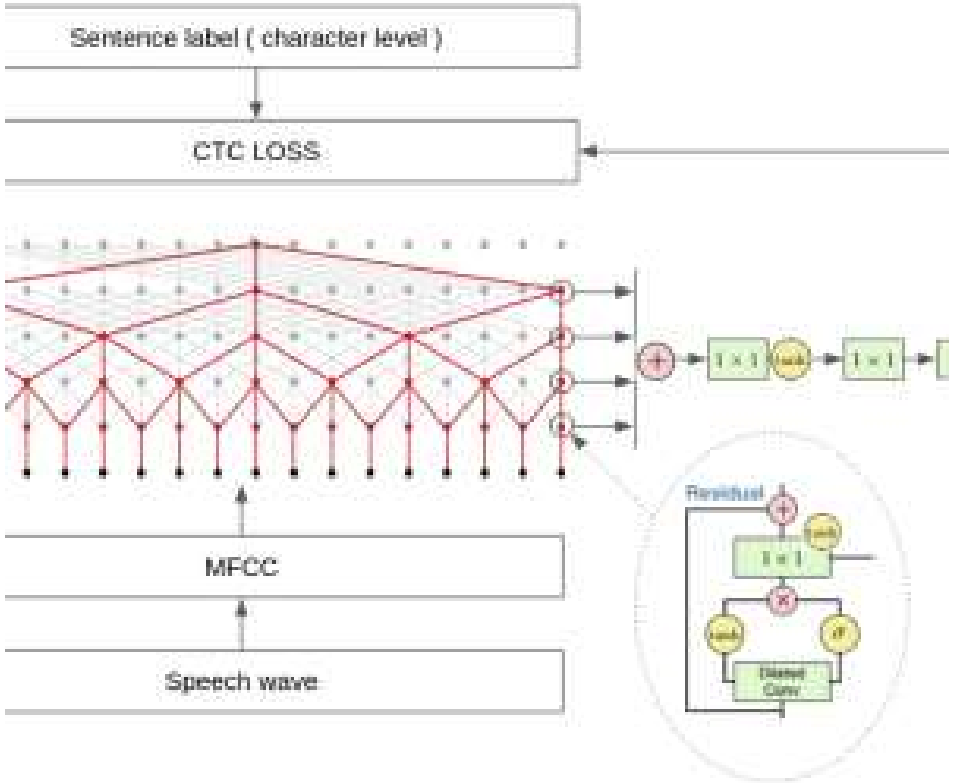
Fairseq

- sequence-to-sequence learning toolkit for Torch from Facebook AI Research
- English to French, English to German and English to Romanian translation

https://github.com/facebookresearch/fairseq?utm_source=mybridge&utm_medium=blog

Speech-to-Text-WaveNet

- End-to-end sentence level English speech recognition using DeepMind's WaveNet and tensorflow



https://github.com/buriburisuri/speech-to-text-wavenet

Dimension Reduction

Speaker: Hong-Han Shuai

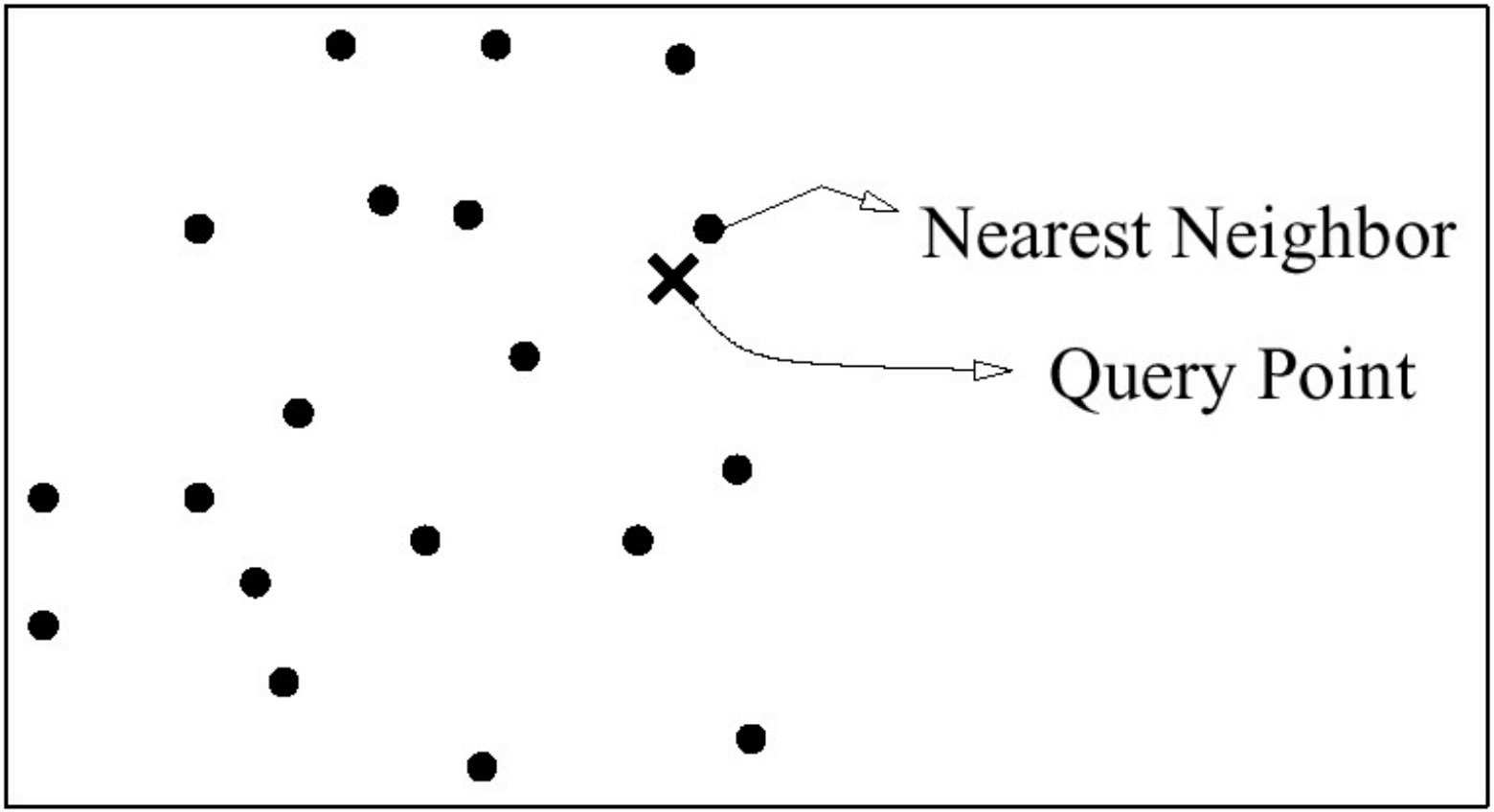
ECE, NCTU

Thanks to the slides made by Prof. Hung-Yi Lee from NTU.

Preliminaries : Nearest Neighbor Search

- Given a collection of data points and a query point in m-dimensional metric space, find the data point that is closest to the query point
- Variation: k-nearest neighbor
- Relevant to clustering and similarity search
- Applications: Geographical Information Systems, similarity search in multimedia databases

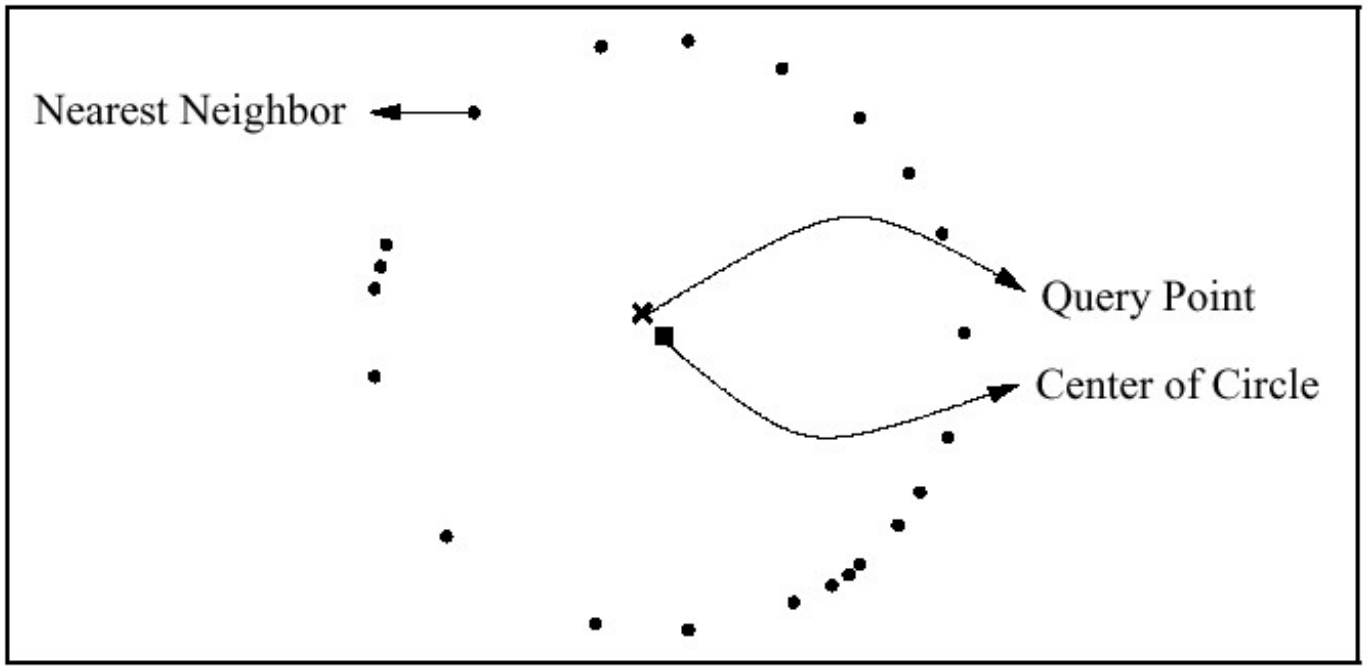
NN Search Con't



Source: [2]

Problems with High Dimensional Data

- A point's nearest neighbor (NN) loses meaning



Source: [2]

Problems (Con't)

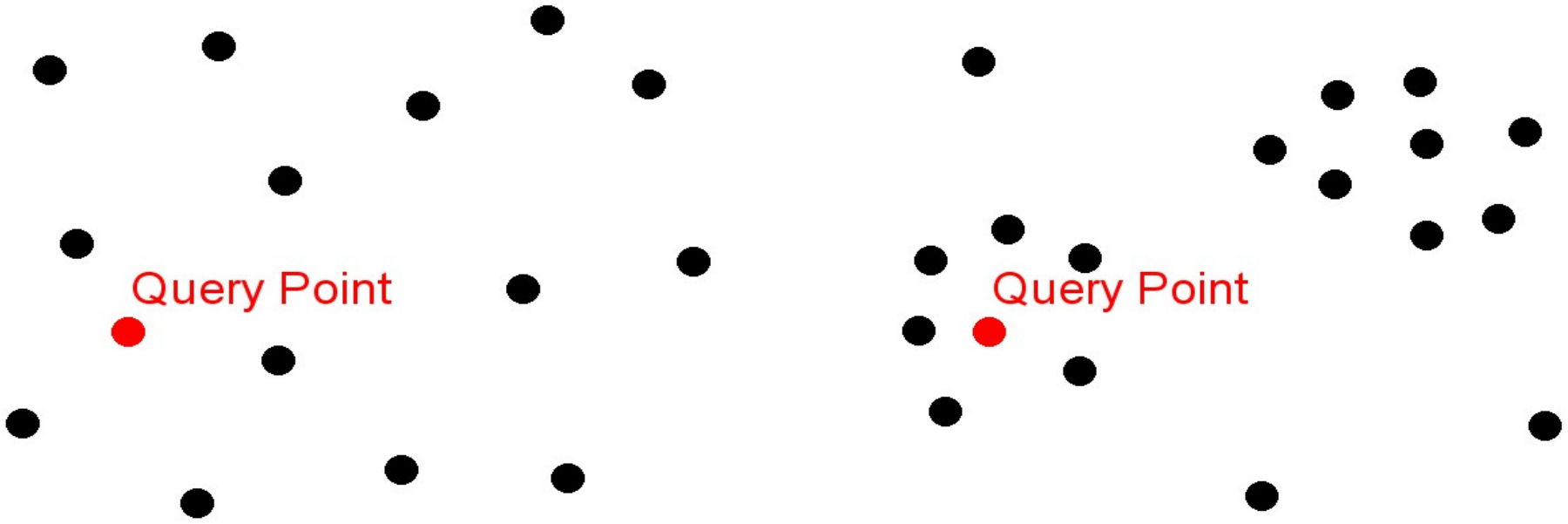
- NN query cost degrades – more strong candidates to compare with
- In as few as 10 dimensions, linear scan outperforms some multidimensional indexing structures (e.g. KD-tree, R* tree, SR tree)
- Biology and genomic data can have dimensions in the 1000's.

Problems (Con't)

- The presence of irrelevant attributes decreases the tendency for clusters to form
- Points in high dimensional space have high degree of freedom; they could be so scattered that they appear uniformly distributed

Problems Con't

- In which cluster does the query point fall?



The Curse

- Refers to the decrease in performance of query processing when the dimensionality increases
- In particular, under certain conditions, the distance between the nearest point and the query point equals the distance between the farthest and query point as dimensionality approaches infinity

Curse (Con't)

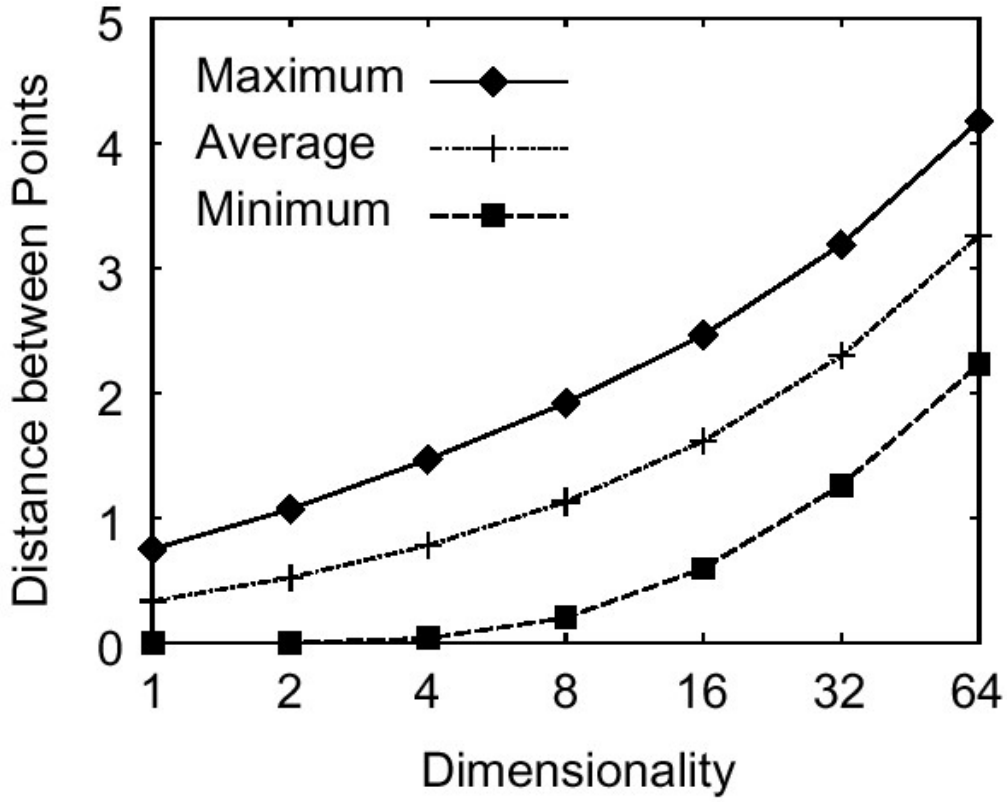
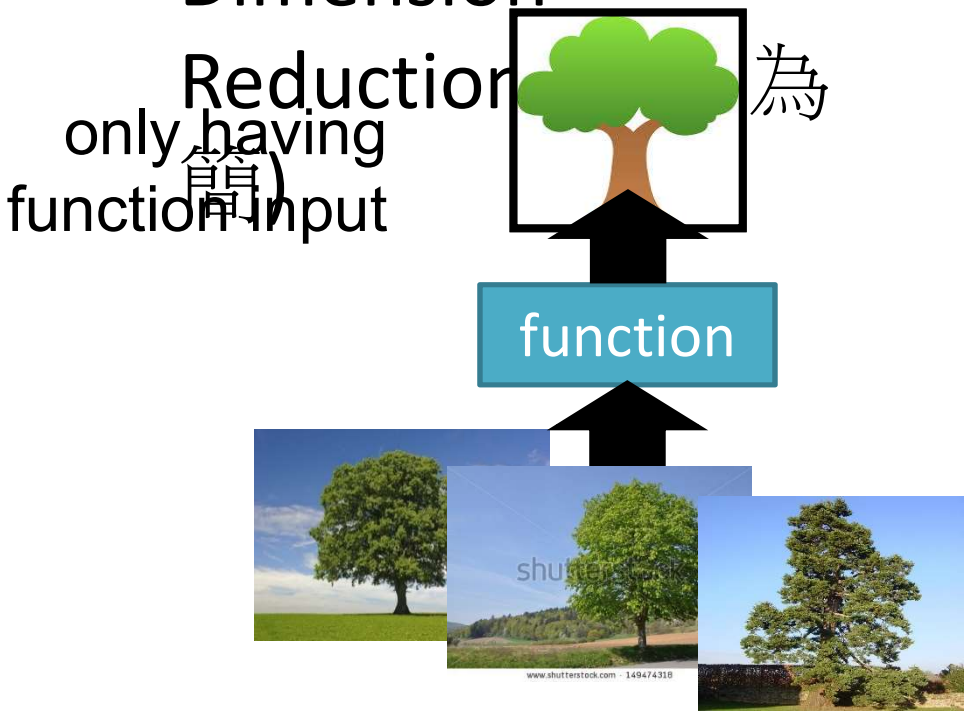


Figure 1. Distances among 100k points generated at random in a unit hypercube

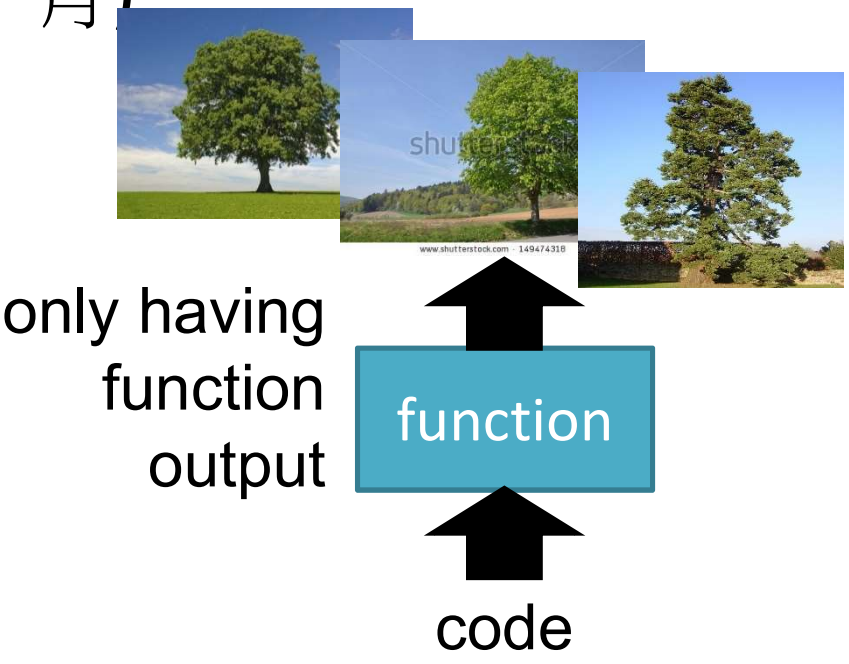
Source: N. Katayama, S. Satoh. Distinctiveness Sensitive Nearest Neighbor Search for Efficient Similarity Retrieval of Multimedia Information. ICDE Conference, 2001.

Unsupervised Learning

- Clustering & Dimension Reduction (化繁為簡)



- Generation (無中生有)



Clustering & Dimension Reduction in these slides

Clust

- K-means
  - Clustering $X = \{x^1, \cdots, x^n, \cdots, x^N\}$ into K clusters
  - Initialize cluster center c^i, i=1, 2, …, K (K random x^n from X)
  - Repeat
    - For all x^n in X:

$$b_i^n \begin{cases} 1 & x^n \text{ is most close to } c^i \\ 0 & \text{Otherwise} \end{cases}$$

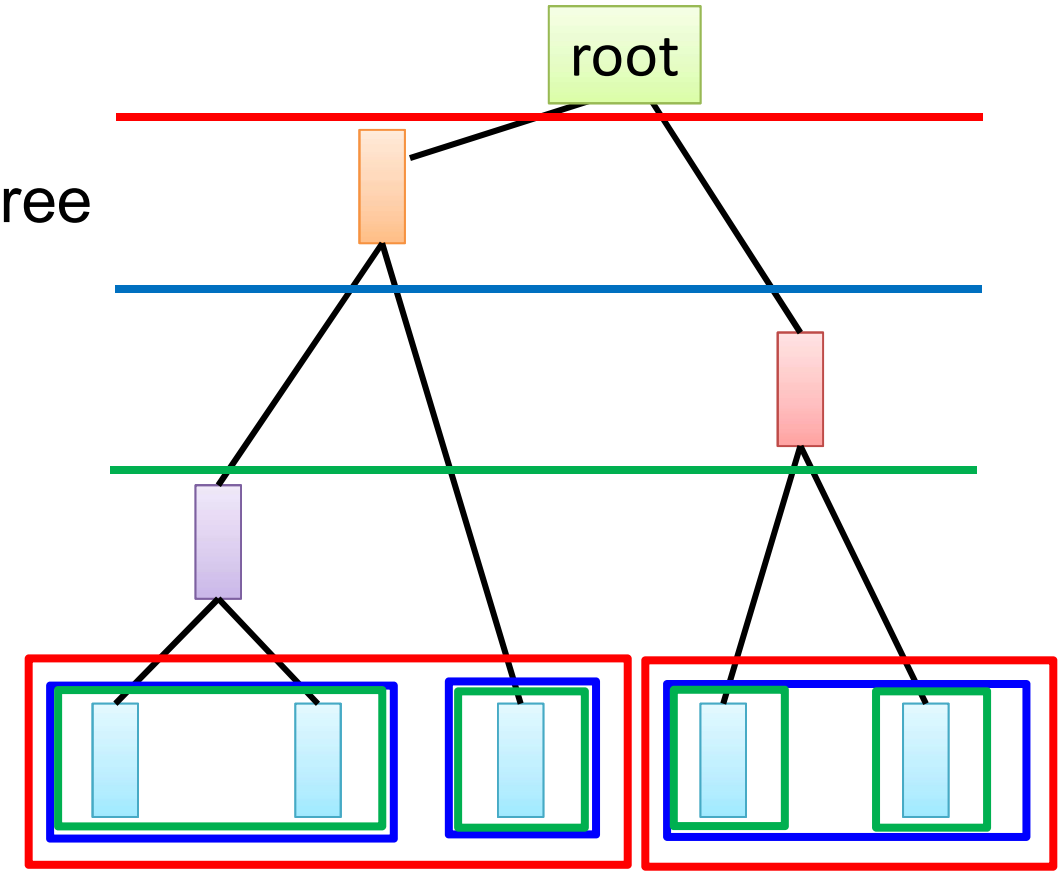
$$c^i = \sum_{x^n} b_i^n x^n \Big/ \sum_{x^n} b_i^n$$

Clustering

- Hierarchical Agglomerative Clustering (HAC)

Step 1: build a tree

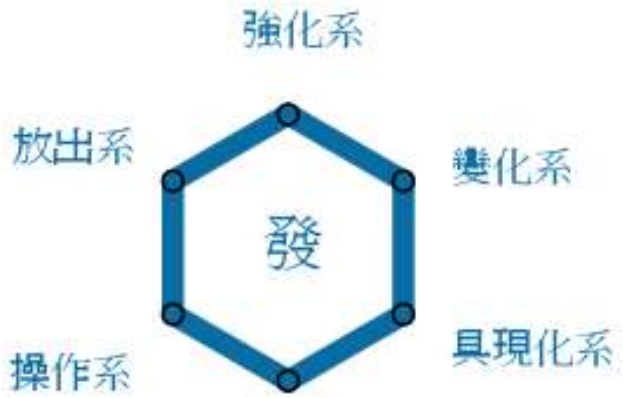
Step 2: pick a threshold



Distributed Representation

- Clustering: an object must belong to one cluster

小傑是強化系



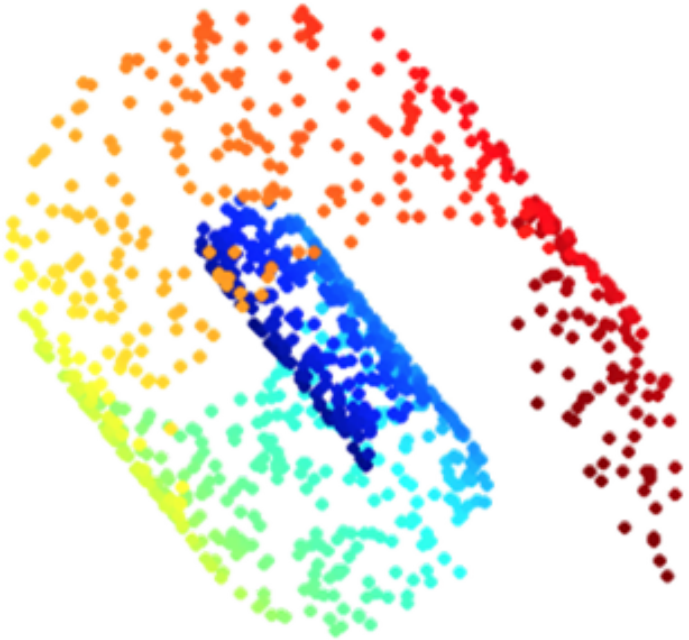
- Distributed representation

Dimension Reduction

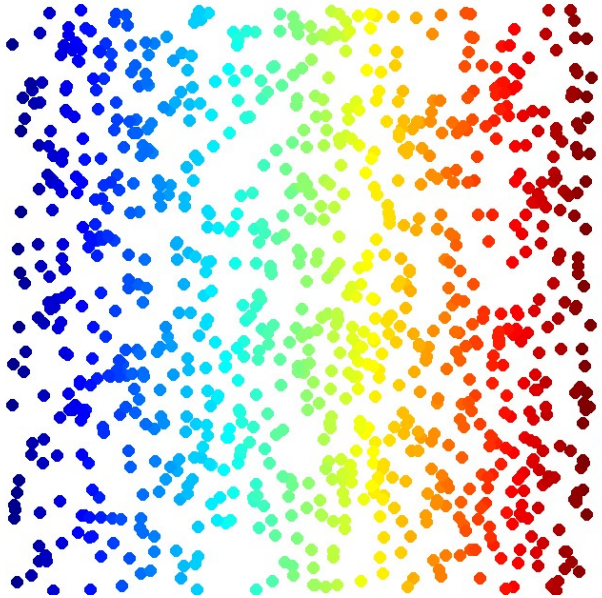
小傑是

強化系	0.70
放出系	0.25
變化系	0.05
操作系	0.00
具現化系	0.00
特質系	0.00

Dimension Reduction



Looks like 3-D



Actually, 2-D

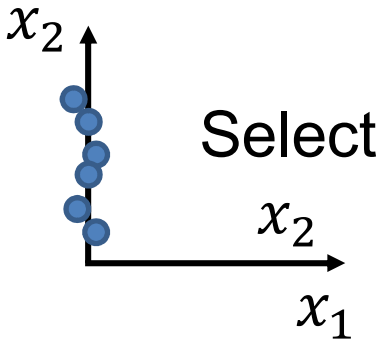
http://reuter.mit.edu/blue/images/research/manifold.png

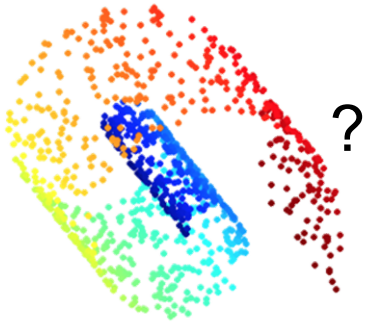
http://archive.cnx.org/resources/51a9b2052ae167db310fda5600b89badea85eae5/isomapCNXtrue1.png

Dimension Reduction

The dimension of z would be smaller than x

- Feature selection





- Principle component analysis (PCA) $z = Wx$

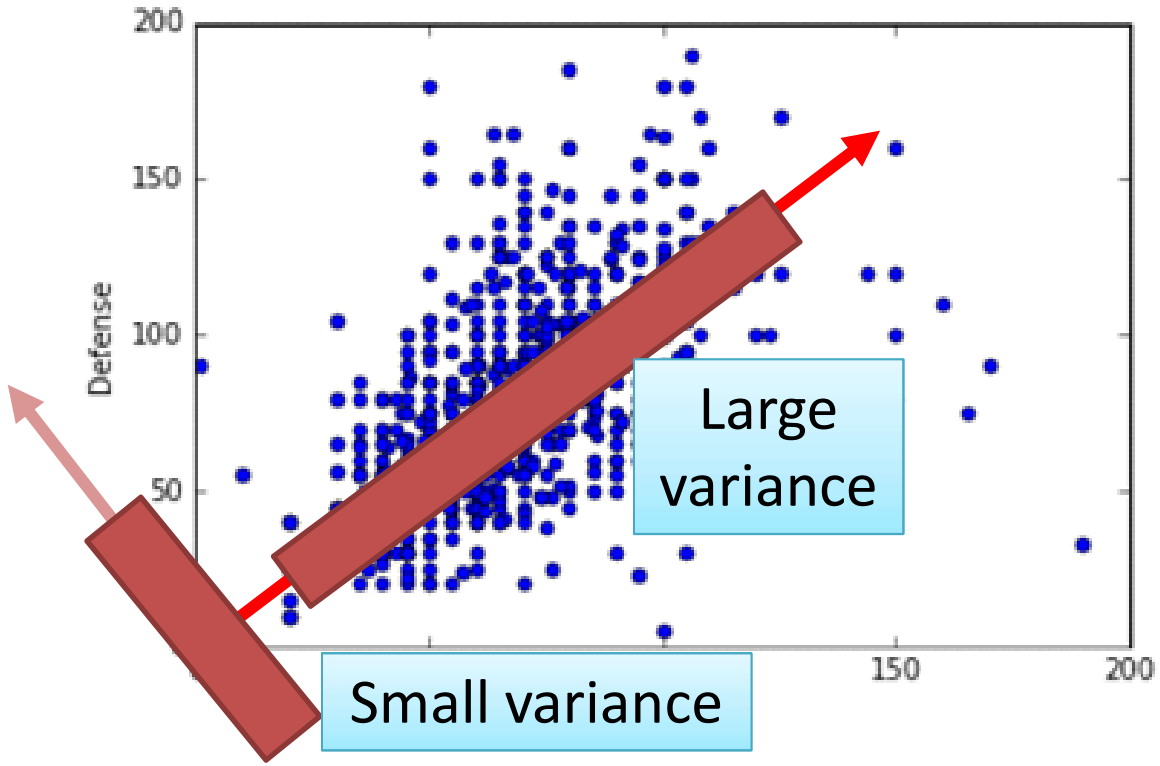
[Bishop, Chapter 12]

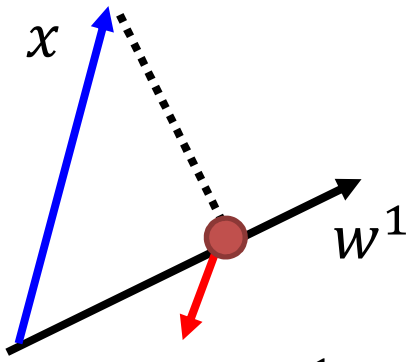
Principle Component Analysis (PCA)

$z = Wx$

Reduce to 1-D:

$z_1 = w^1 \cdot x$





$z_1 = w^1 \cdot x$

Project all the data points x onto w^1, and obtain a set of z_1

We want the variance of z_1 as large as possible

$$Var(z_1) = \sum_{z_1} (z_1 - \bar{z_1})^2 \qquad \|w^1\|_2 = 1$$

PCA

$$z = Wx$$

Reduce to 1-D:

$$z_1 = w^1 \cdot x$$

$$z_2 = w^2 \cdot x$$

$$W = \begin{bmatrix} (w^1)^T \\ (w^2)^T \\ \vdots \end{bmatrix}$$

Orthogonal matrix

Project all the data points x onto w^1, and obtain a set of z_1

We want the variance of z_1 as large as possible

$$Var(z_1) = \sum_{z_1} (z_1 - \bar{z_1})^2 \quad \|w^1\|_2 = 1$$

We want the variance of z_2 as large as possible

$$Var(z_2) = \sum_{z_2} (z_2 - \bar{z_2})^2 \quad \|w^2\|_2 = 1$$

$$w^1 \cdot w^2 = 0$$

Warning of Math

PCA

$$z_1 = w^1 \cdot x$$

$$\bar{z_1} = \frac{1}{N}\sum z_1 = \frac{1}{N}\sum w^1 \cdot x = w^1 \cdot \frac{1}{N}\sum x = w^1 \cdot \bar{x}$$

$$Var(z_1) = \frac{1}{N}\sum_{z_1}(z_1 - \bar{z_1})^2$$

$$= \frac{1}{N}\sum_{x}(w^1 \cdot x - w^1 \cdot \bar{x})^2$$

$$= \frac{1}{N}\sum\left(w^1 \cdot (x - \bar{x})\right)^2$$

$$= \frac{1}{N}\sum (w^1)^T (x - \bar{x})(x - \bar{x})^T w^1$$

$$= (w^1)^T \boxed{\frac{1}{N}\sum (x - \bar{x})(x - \bar{x})^T} w^1$$

$$= (w^1)^T Cov(x) w^1$$

$S = Cov(x)$

$$(a \cdot b)^2 = (a^T b)^2 = a^T b a^T b$$

$$= a^T b (a^T b)^T = a^T b b^T a$$

Find w^1 maximizing $(w^1)^T S w^1$

$$\|w^1\|_2 = (w^1)^T w^1 = 1$$

Find w^1 maximizing $(w^1)^T S w^1$ $(w^1)^T w^1 = 1$

$S = Cov(x)$ Symmetric positive-semidefinite (non-negative eigenvalues)

Using Lagrange multiplier [Bishop, Appendix E]

$$g(w^1) = (w^1)^T S w^1 - \alpha\left((w^1)^T w^1 - 1\right)$$

$$\partial g(w^1)/\partial w_1^1 = 0$$
$$\partial g(w^1)/\partial w_2^1 = 0$$
$$\vdots$$

$$Sw^1 - \alpha w^1 = 0$$
$$Sw^1 = \alpha w^1$$

w^1 : eigenvector

$$(w^1)^T S w^1 = \alpha (w^1)^T w^1$$
$$= \alpha$$

Choose the maximum one

w^1 is the eigenvector of the covariance matrix S Corresponding to the largest eigenvalue λ_1

Find w^2 $(w^2)^T S w^2$ $(w^2)^T w^2 = 1$ $(w^2)^T w^1 = 0$

maximizing

$$g(w^2) = (w^2)^T S w^2 - \alpha\left((w^2)^T w^2 - 1\right) - \beta\left((w^2)^T w^1 - 0\right)$$

$\partial g(w^2)/\partial w_1^2 = 0$

$\partial g(w^2)/\partial w_2^2 = 0$

$\vdots$

$$Sw^2 - \alpha w^2 - \beta w^1 = 0$$

$$0 - \alpha\, 0 - \beta\, 1 = 0$$

$$= \left((w^1)^T S w^2\right)^T = (w^2)^T S^T w^1$$

$$= (w^2)^T S w^1 = \lambda_1 (w^2)^T w^1 = 0$$

$$Sw^1 = \lambda_1 w^1$$

$\beta = 0$: $Sw^2 - \alpha w^2 = 0$ $Sw^2 = \alpha w^2$

w^2 is the eigenvector of the covariance matrix S Corresponding to the 2nd largest eigenvalue

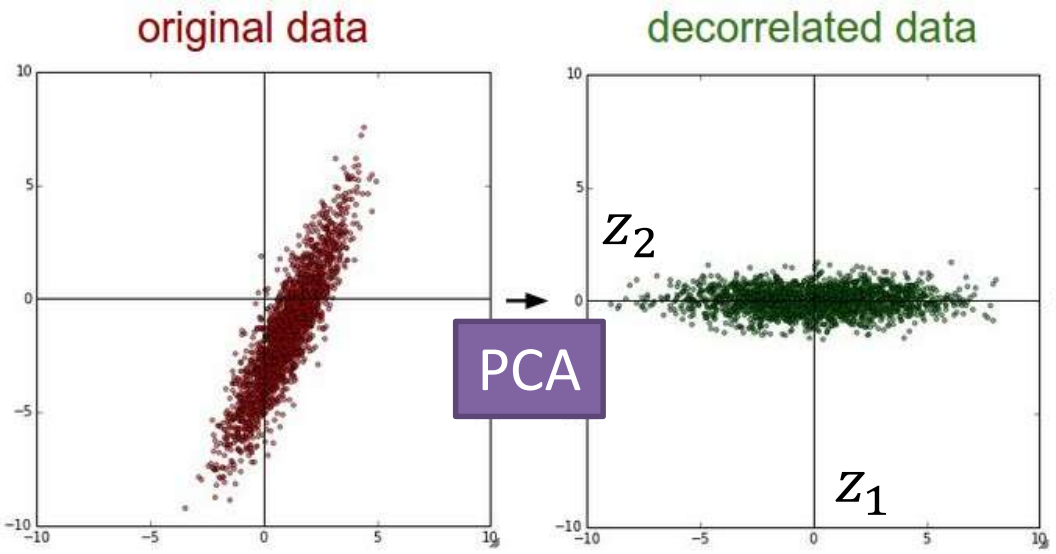
λ_2

PCA - decorrelatio

$$z = Wx$$

$$Cov(z) = D$$

Diagonal matrix



$$Cov(z) = \frac{1}{N}\sum (z - \bar{z})(z - \bar{z})^T = WSW^T \qquad S = Cov(x)$$

$$= WS[w^1 \quad \cdots \quad w^K] = W[Sw^1 \quad \cdots \quad Sw^K]$$

$$= W[\lambda_1 w^1 \quad \cdots \quad \lambda_K w^K] = [\lambda_1 Ww^1 \quad \cdots \quad \lambda_K Ww^K]$$

$$= [\lambda_1 e_1 \quad \cdots \quad \lambda_K e_K] = D$$

Diagonal matrix

End of Warning

More Related Approaches Not Introduced

- Multidimensional Scaling (MDS) [Alpaydin, Chapter 6.7]
  - Only need distance between objects
- Probabilistic PCA [Bishop, Chapter 12.2]
- Kernel PCA [Bishop, Chapter 12.3]
  - non-linear version of PCA
- Canonical Correlation Analysis (CCA) [Alpaydin, Chapter 6.9]
- Independent Component Analysis (ICA)
  - Ref: http://cis.legacy.ics.tkk.fi/aapo/papers/IJCNN99_tutorialweb/
- Linear Discriminant Analysis (LDA) [Alpaydin, Chapter 6.8]
  - Supervised

Q & A

Thanks.