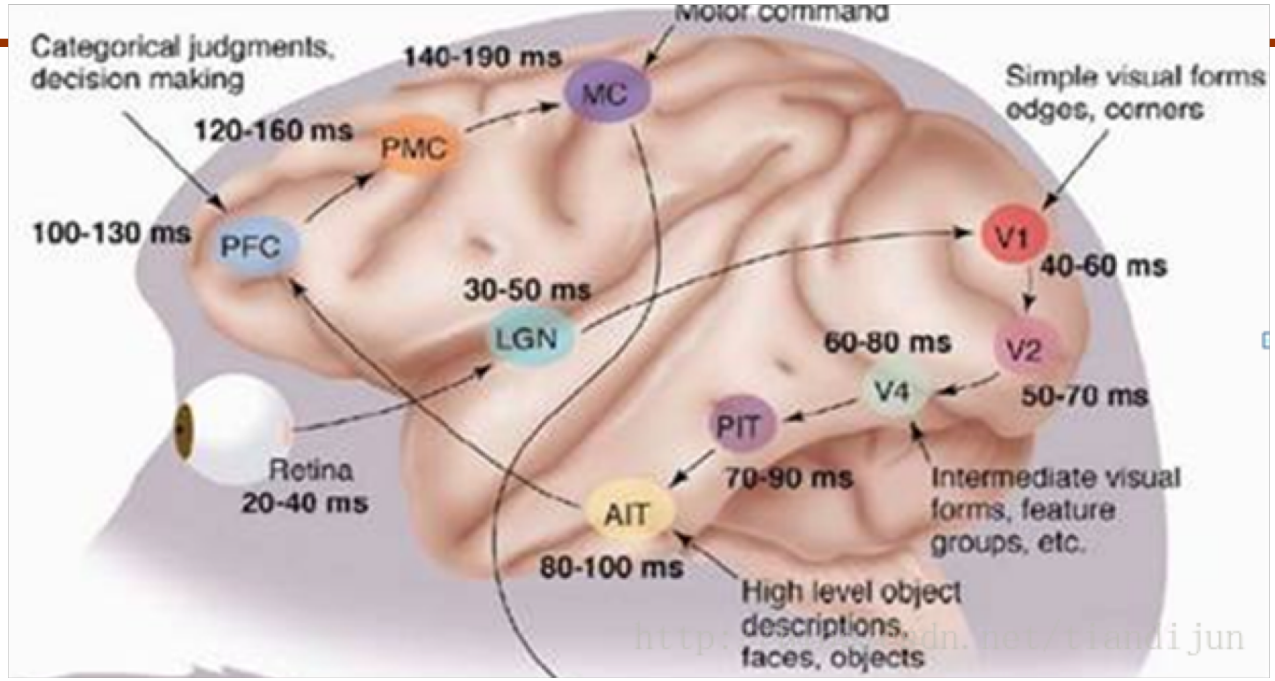
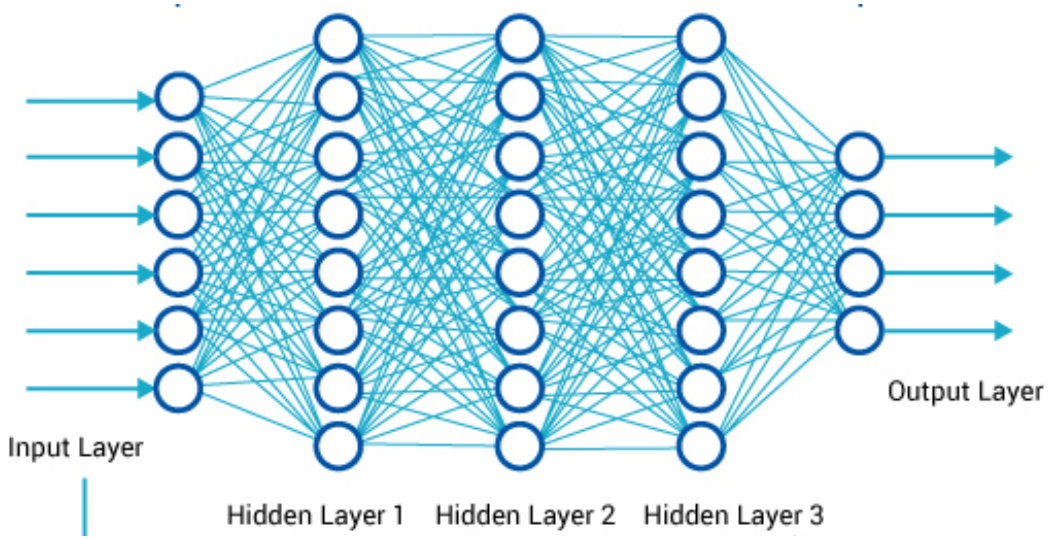
Introduction to Machine Learning (II)

Hong-Han Shuai
National Chiao Tung University
HCC
2019 Spring

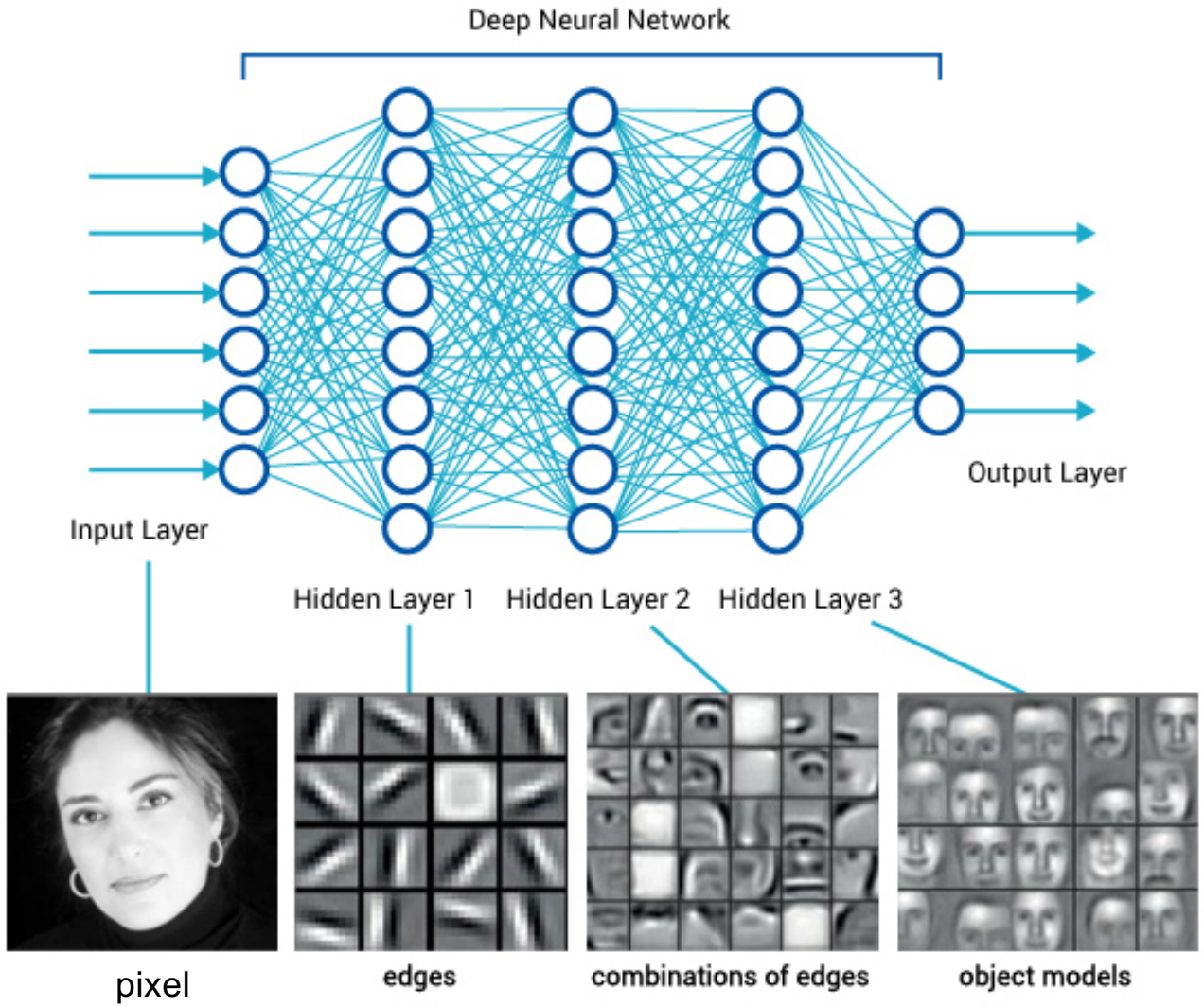
Thanks to the slides of Prof. P. Domingos from Washington University, Prof. H.-T. Lin and Prof. Lee Hung-Yi Lee from NTU.

Idea from human brain



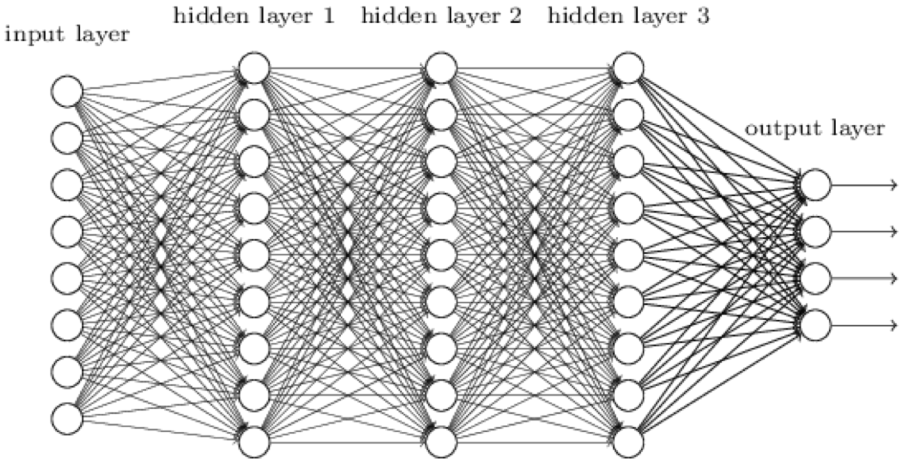


Artificial Neuron Network(ANN)

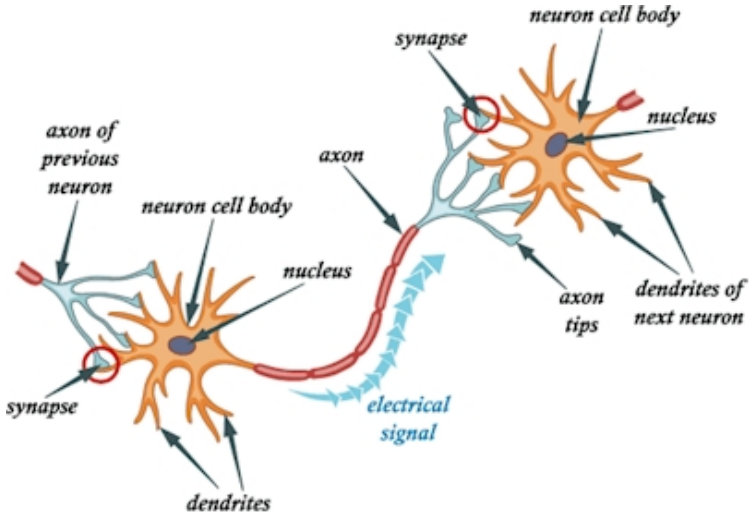


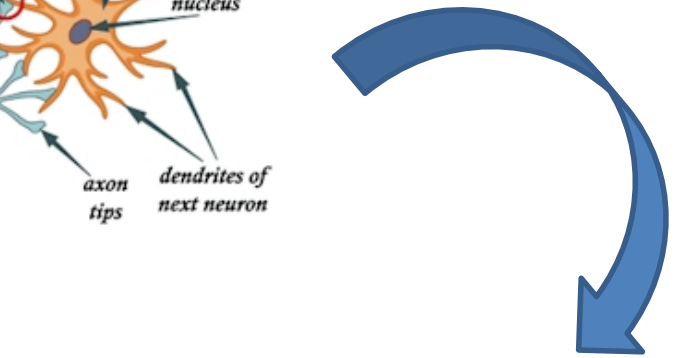
Source:http://www.amax.com/blog/?p=804

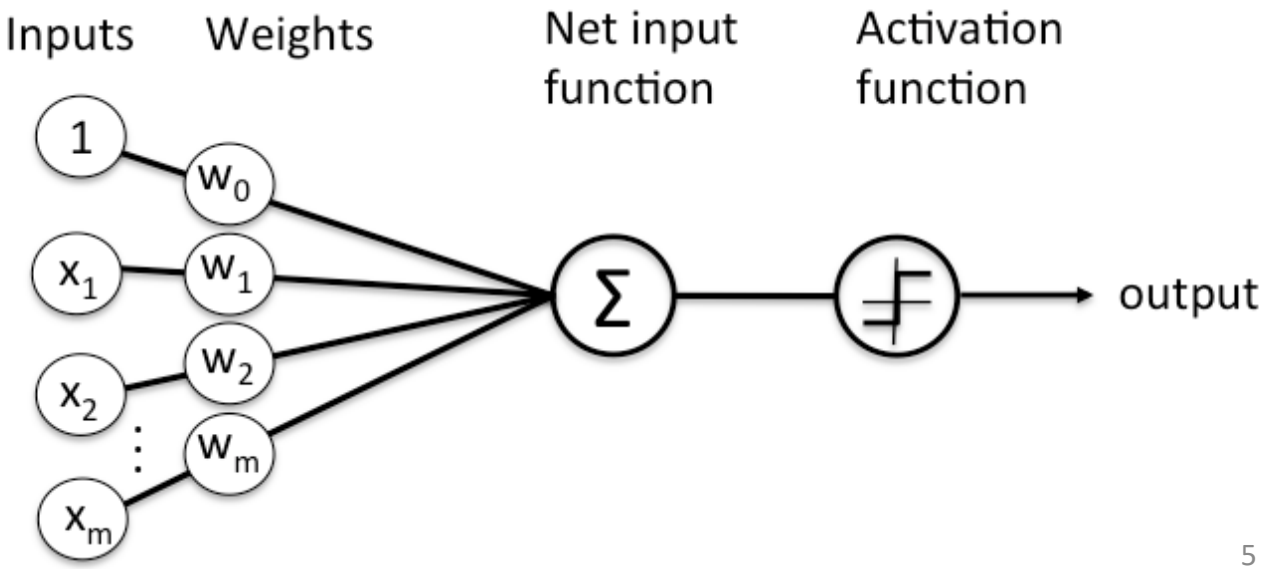
Neuron Network



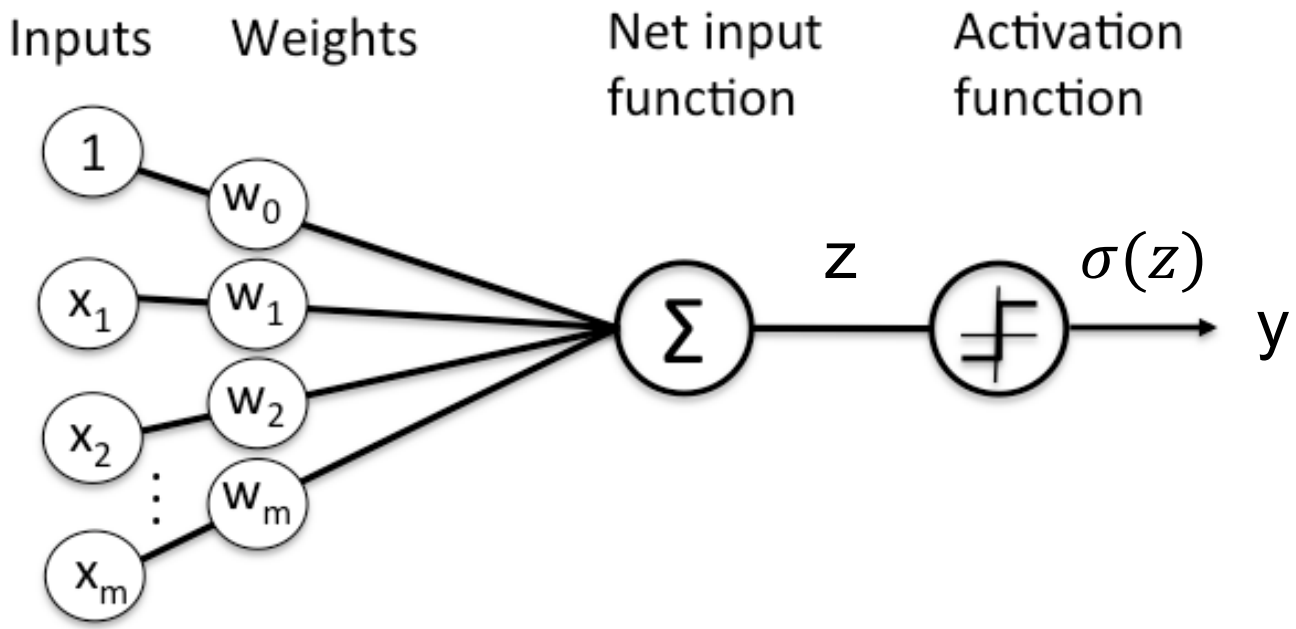
Neuron


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
w_2
x_2
w_m
x_m
Σ
output

Neuron

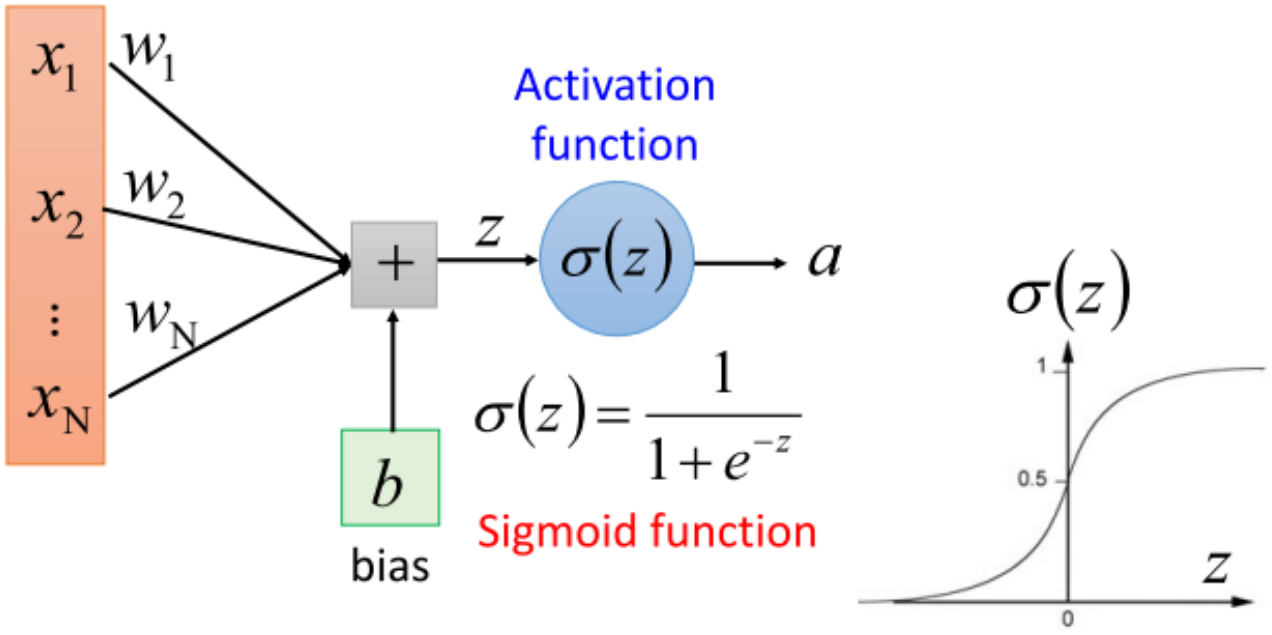


$z = 1^{*}w_0 + x_1{}^{*}w_1 + x_2{}^{*}w_2 + \ldots + x_m{}^{*}w_m$

$y = \sigma(z)$

A Neuron for Machine

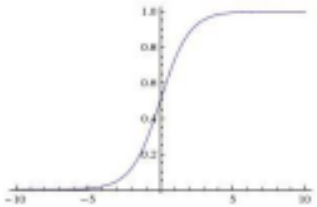
Each neuron is a very simple function



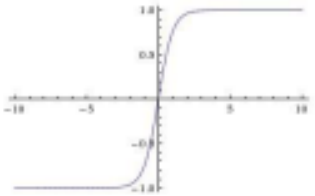
Activation Functions

Sigmoid

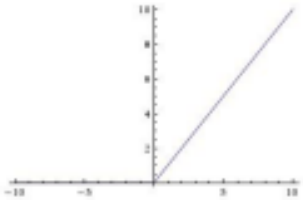
$$\sigma(x) = 1/(1+e^{-x})$$



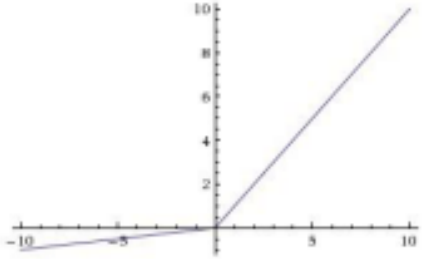
tanh tanh(x)



ReLU max(0,x)



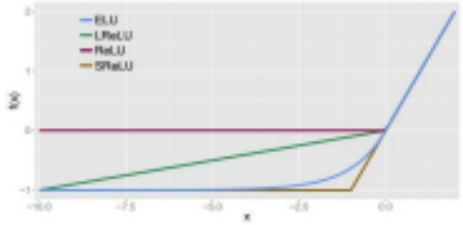
Leaky ReLU
max(0.1x, x)



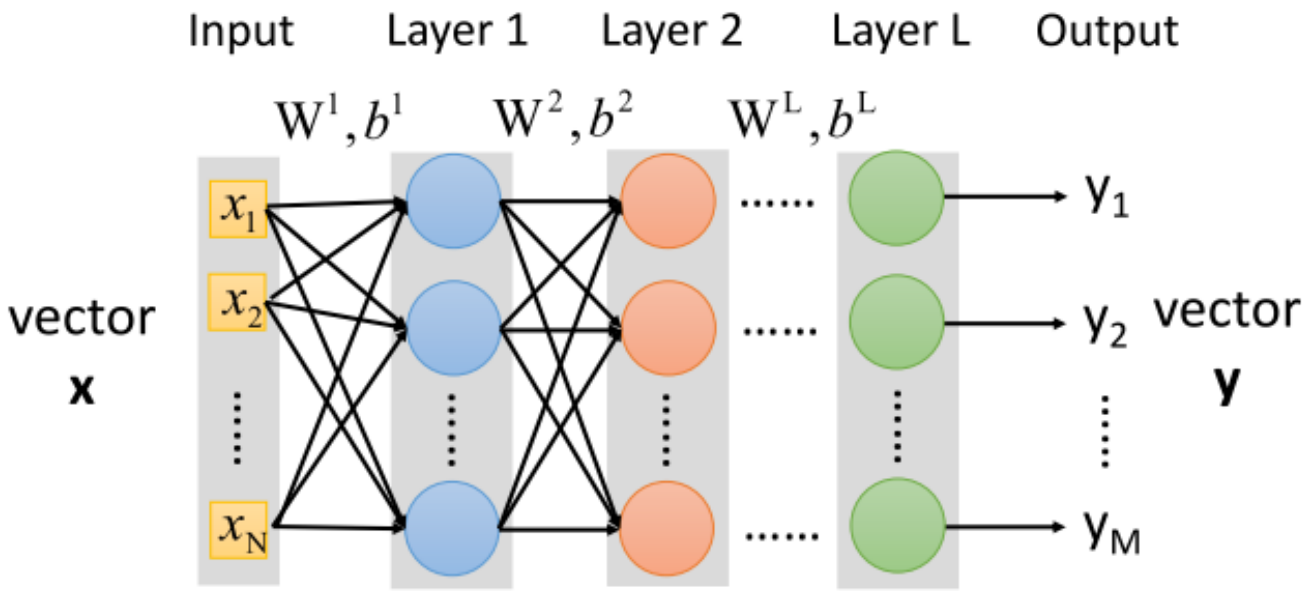
Maxout $\max(w_1^T x + b_1, w_2^T x + b_2)$

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*

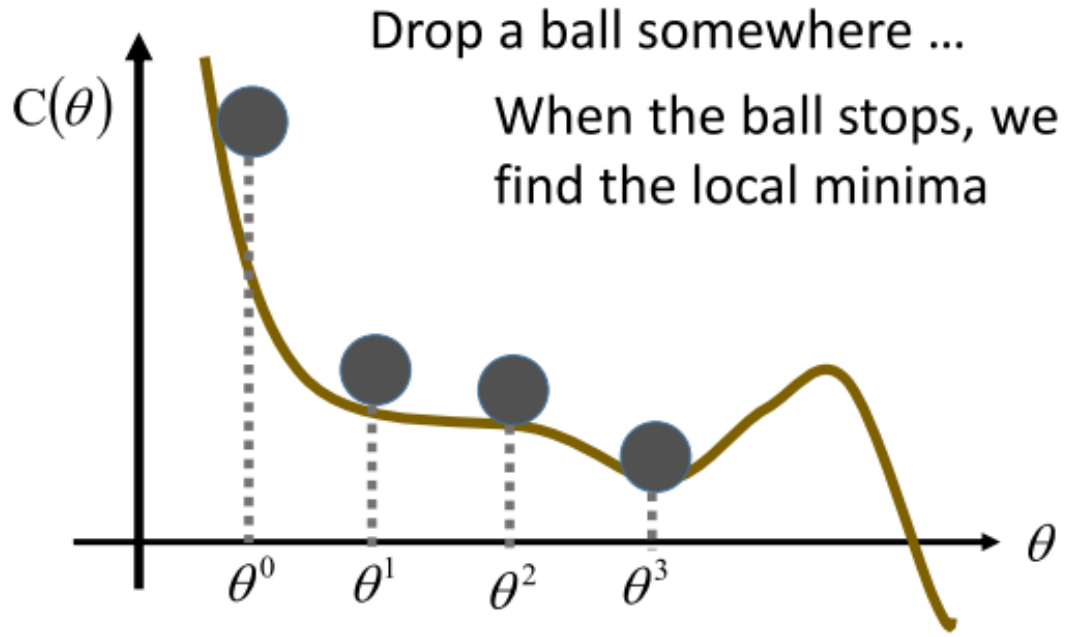
Pick the "best" parameter set θ*

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

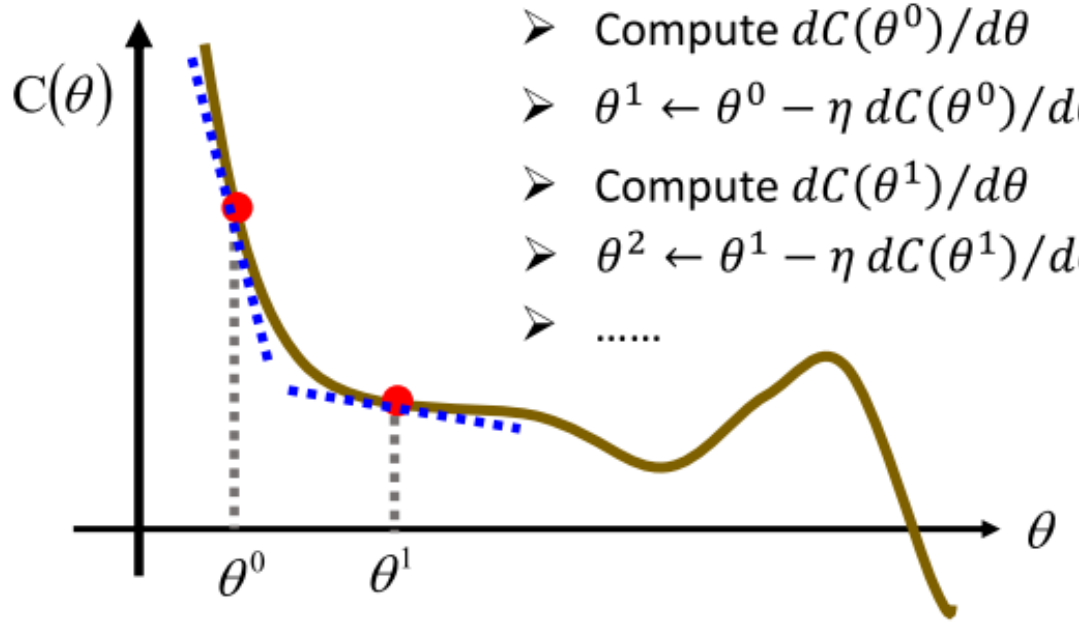
- For simplification, first consider that θ has only one variable



Gradient Descent – Idea

η is called *"learning rate"*

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

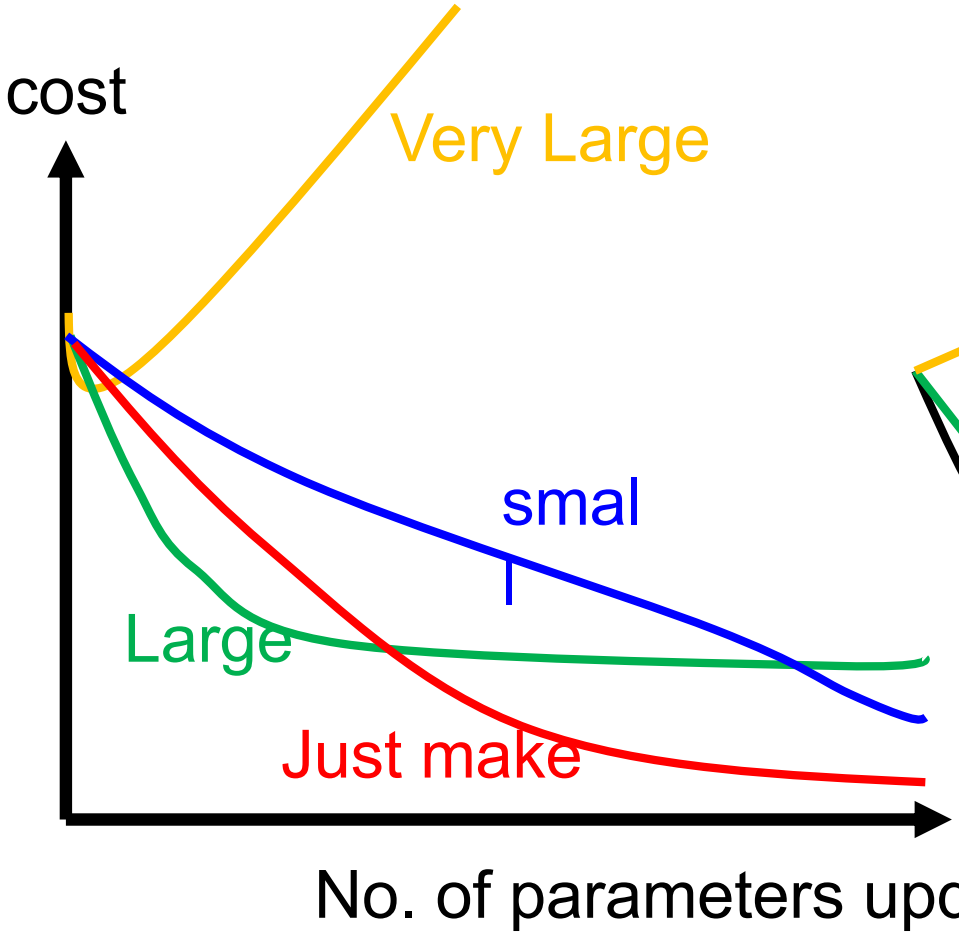


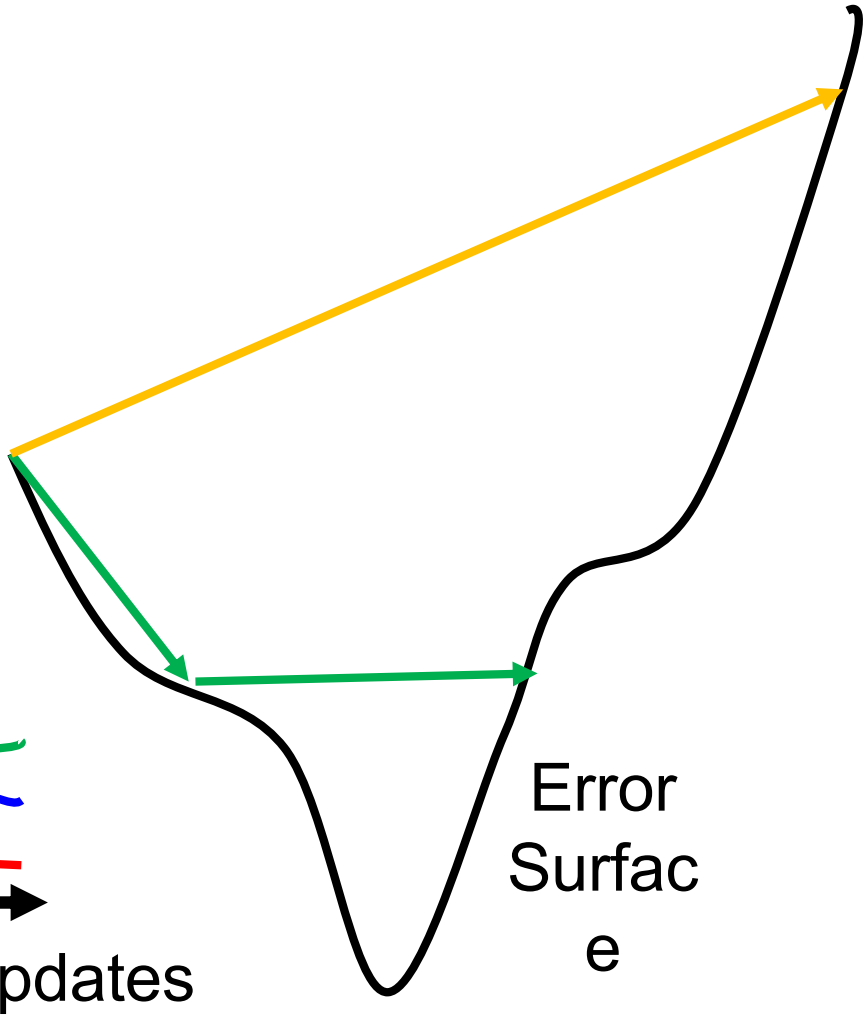
- Randomly start at θ^0
- Compute $dC(\theta^0)/d\theta$
- $\theta^1 \leftarrow \theta^0 - \eta\, dC(\theta^0)/d\theta$
- Compute $dC(\theta^1)/d\theta$
- $\theta^2 \leftarrow \theta^1 - \eta\, dC(\theta^1)/d\theta$
- ……

Learning Rate

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

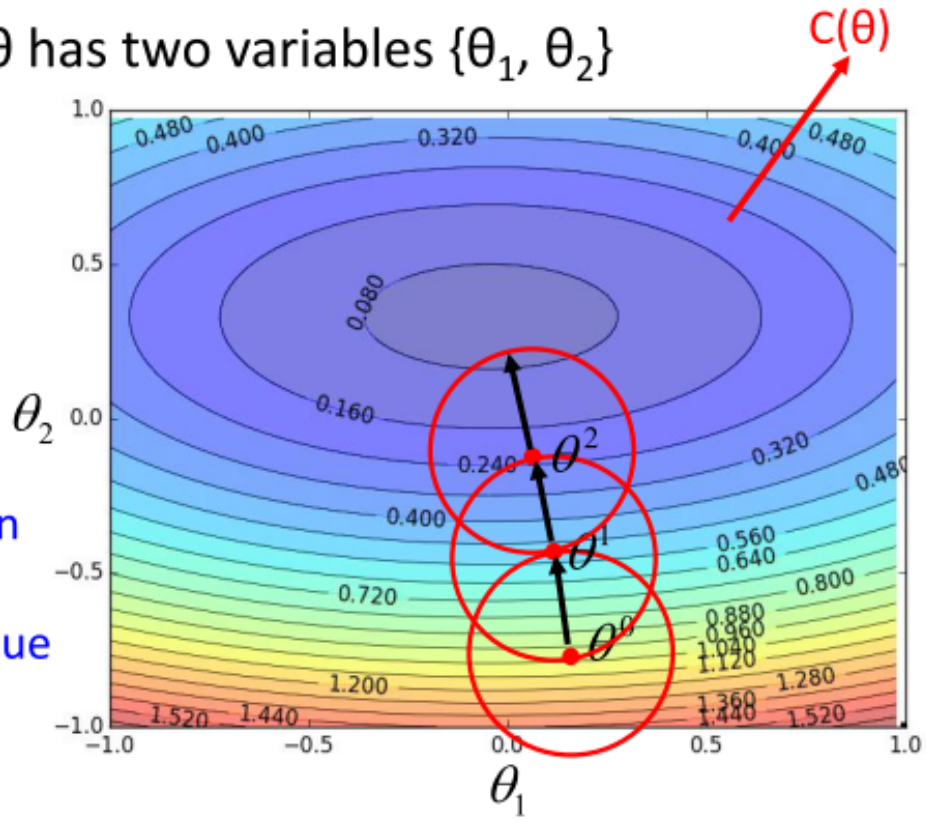
- Set the learning rate η carefully





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 $\quad \theta^1 = \theta^0 - \eta \nabla C^1\left(\theta^0\right)$

pick x^2 $\quad \theta^2 = \theta^1 - \eta \nabla C^2\left(\theta^1\right)$

$\vdots \qquad \vdots$

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

$\vdots \qquad \vdots$

pick x^R $\quad \theta^R = \theta^{R-1} - \eta \nabla C^R\left(\theta^{R-1}\right)$

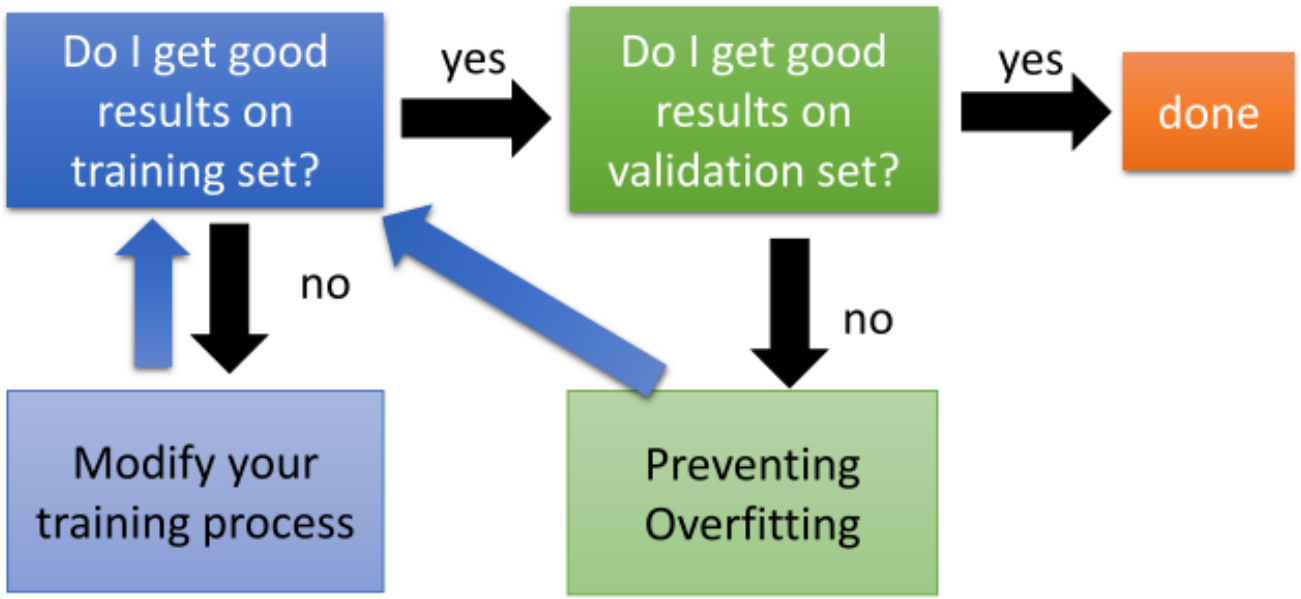
Seen all the examples once

One epoch

---

pick x^1 $\quad \theta^{R+1} = \theta^R - \eta \nabla C^1\left(\theta^R\right)$

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,

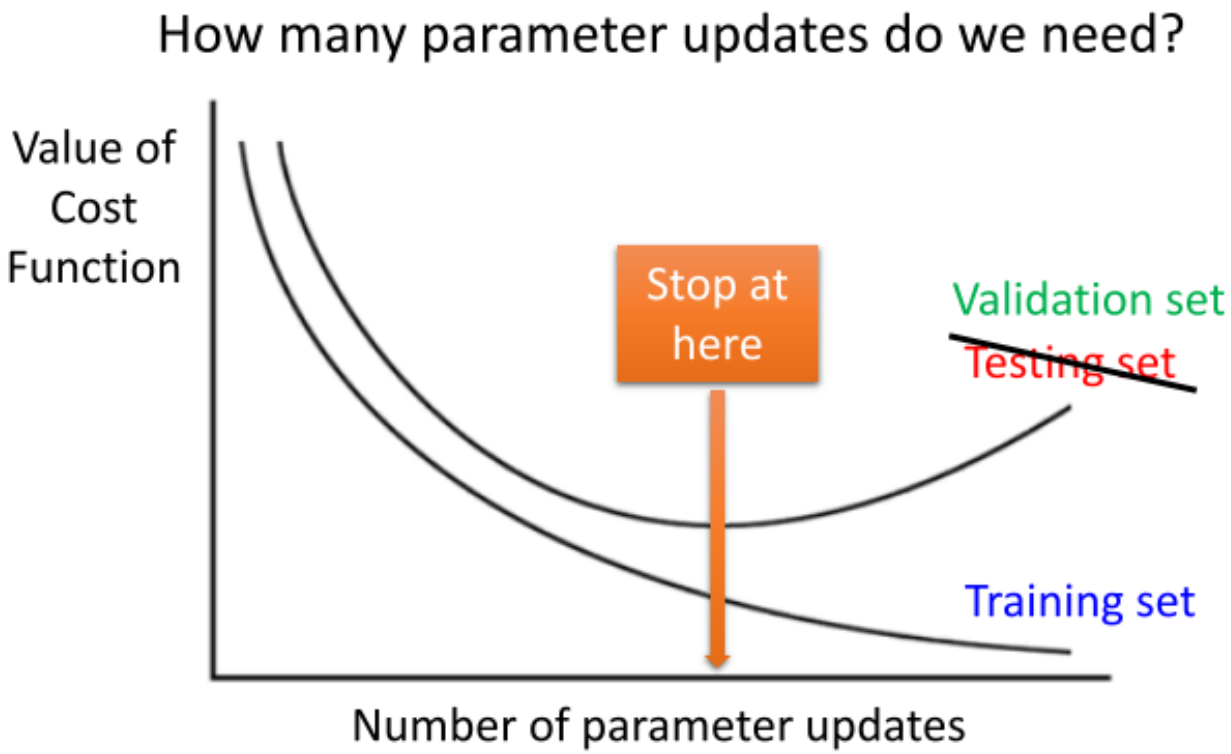
Testing Data: $\{\ldots x^u \ldots\} \Rightarrow f(x^u;\theta^*) \neq \hat{y}^u$

Training data and testing data have different distribution.

Training Data:

Testing Data:

Early Stopping

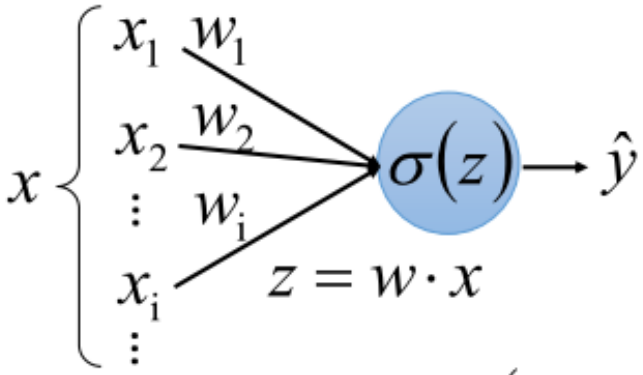


Weight Decay

- The parameters closer to zero is preferred.

Training data:

$$\{(x, \hat{y}), \ldots\}$$



Testing data:

$$\{(x', \hat{y}), \ldots\} \qquad x' = x + \varepsilon$$

$$\begin{aligned} z' &= w \cdot (x + \varepsilon) \\ &= w \cdot x + w \cdot \varepsilon \\ &= z + w \cdot \varepsilon \end{aligned}$$

To minimize the effect of noise, we want w close to zero.

Weight Decay

- New cost function to be minimized
  - Find a set of weight not only minimizing original cost but also close to zero

$$C'(\theta) = C(\theta) + \lambda \frac{1}{2} \|\theta\|^2$$

→ Regularization term:

Original cost
(e.g. minimize square error, cross entropy …)

$$\theta = \{W^1, W^2, \ldots\}$$

$$\|\theta\|^2 = \left(w_{11}^1\right)^2 + \left(w_{12}^1\right)^2 + \ldots + \left(w_{11}^2\right)^2 + \left(w_{12}^2\right)^2 + \ldots$$

(not consider biases. why?)

Weight Decay

$$\|\theta\|^2 = \left(w_{11}^1\right)^2 + \left(w_{12}^1\right)^2 + \ldots + \left(w_{11}^2\right)^2 + \left(w_{12}^2\right)^2 + \ldots$$

- New cost function to be minimized

$$C'(\theta) = C(\theta) + \lambda \frac{1}{2}\|\theta\|^2 \quad \text{Gradient:} \quad \frac{\partial C'}{\partial w} = \frac{\partial C}{\partial w} + \lambda w$$

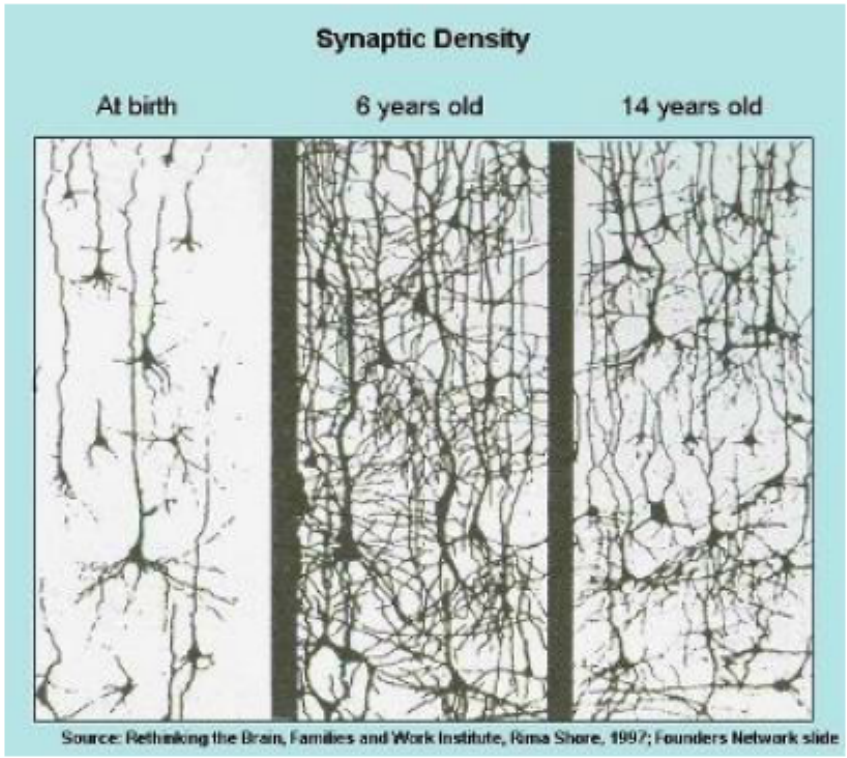
Update: $$w^{t+1} \rightarrow w^t - \eta \frac{\partial C'}{\partial w} = w^t - \eta\left(\frac{\partial C}{\partial w} + \lambda w^t\right)$$

$$= \underbrace{(1-\eta\lambda)w^t} - \eta \underbrace{\frac{\partial C}{\partial w}}$$

Smaller and smaller

Weight Decay

- Our Brain



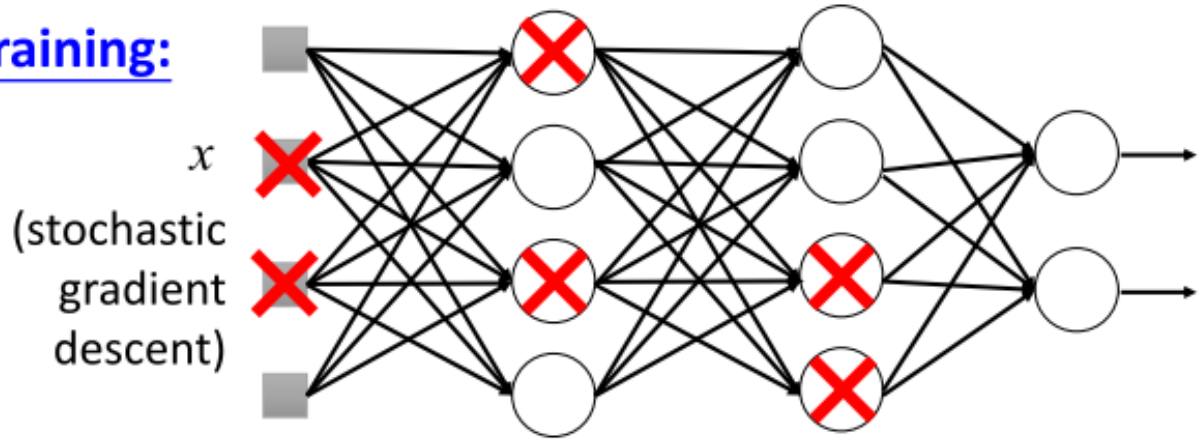
Dropout

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

Training:

x

(stochastic gradient descent)



- In each ***iteration***
  - Each neuron has p% to dropout

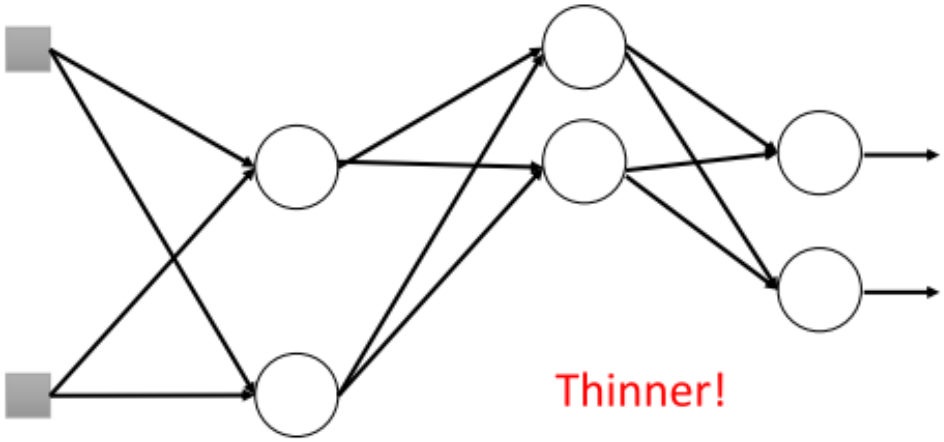
Dropout

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

Training:

x

(stochastic gradient descent)

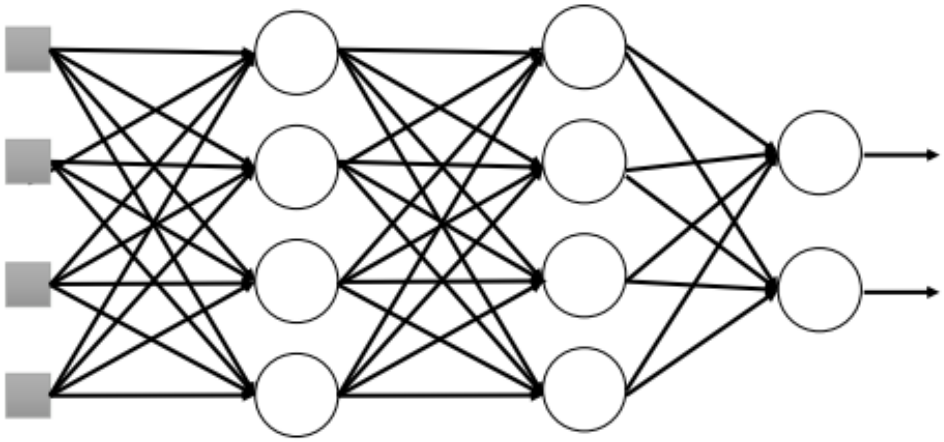


- In each ***iteration***
  - Each neuron has p% to dropout
    - **The structured of the network is changed.**
  - Using the new network for training

For each iteration, we resample the dropout neurons

Dropout

Testing:



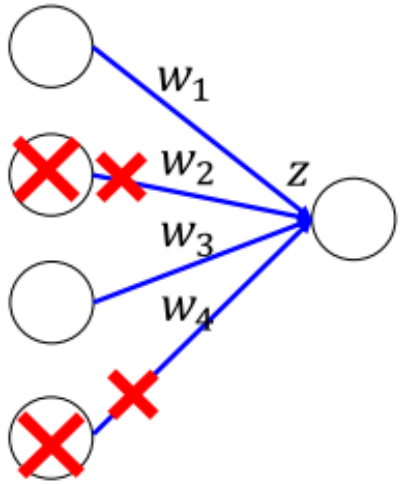
- **No dropout**
  - If the dropout rate at training is p%, all the weights times (1-p)%
  - Assume that the dropout rate is 50%. If $w_{ij}^{l} = 1$ from training, set $w_{ij}^{l} = 0.5$ for testing.

Dropout
- Intuitive Reason

- Why the weights should multiply (1-p)% (dropout rate) when testing?

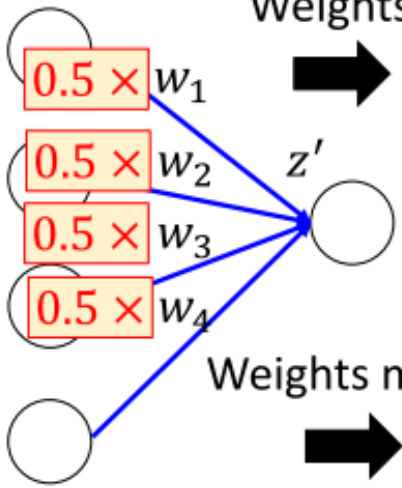
Training of Dropout

Assume dropout rate is 50%



Testing of Dropout

No dropout



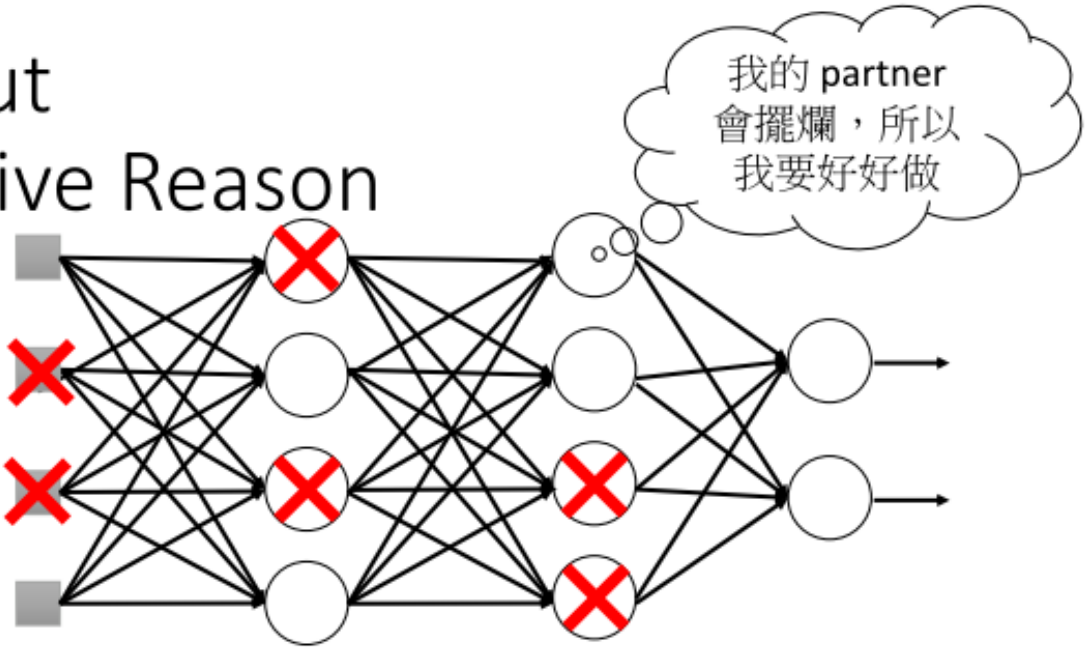
Weights from training

$z' \approx 2z$

Weights multiply (1-p)%

$z' \approx z$

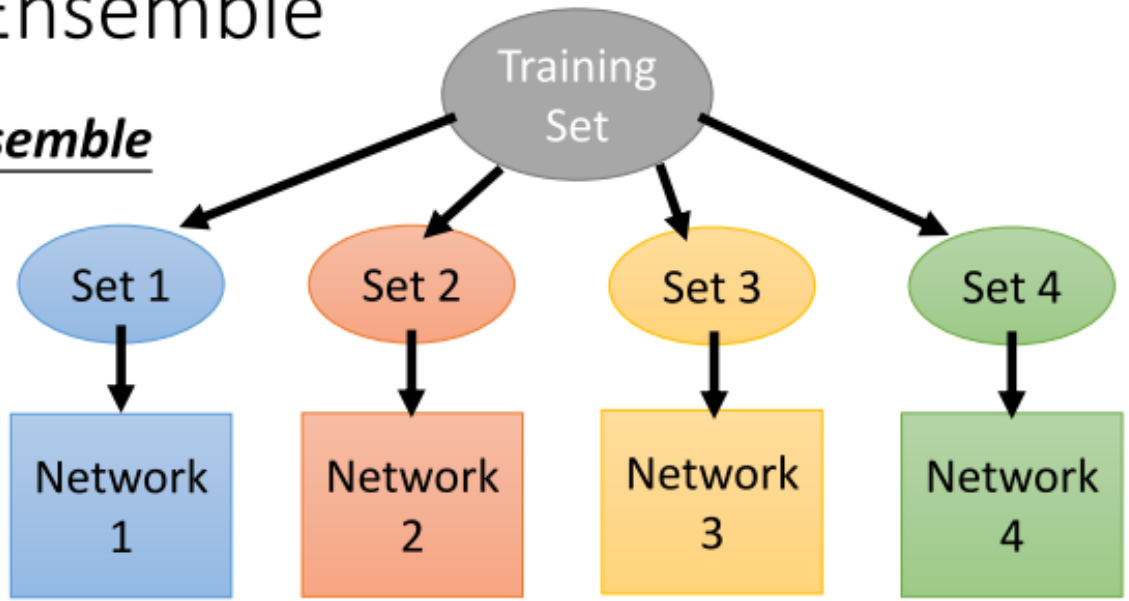
Dropout
- Intuitive Reason



- 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
- Ensemble

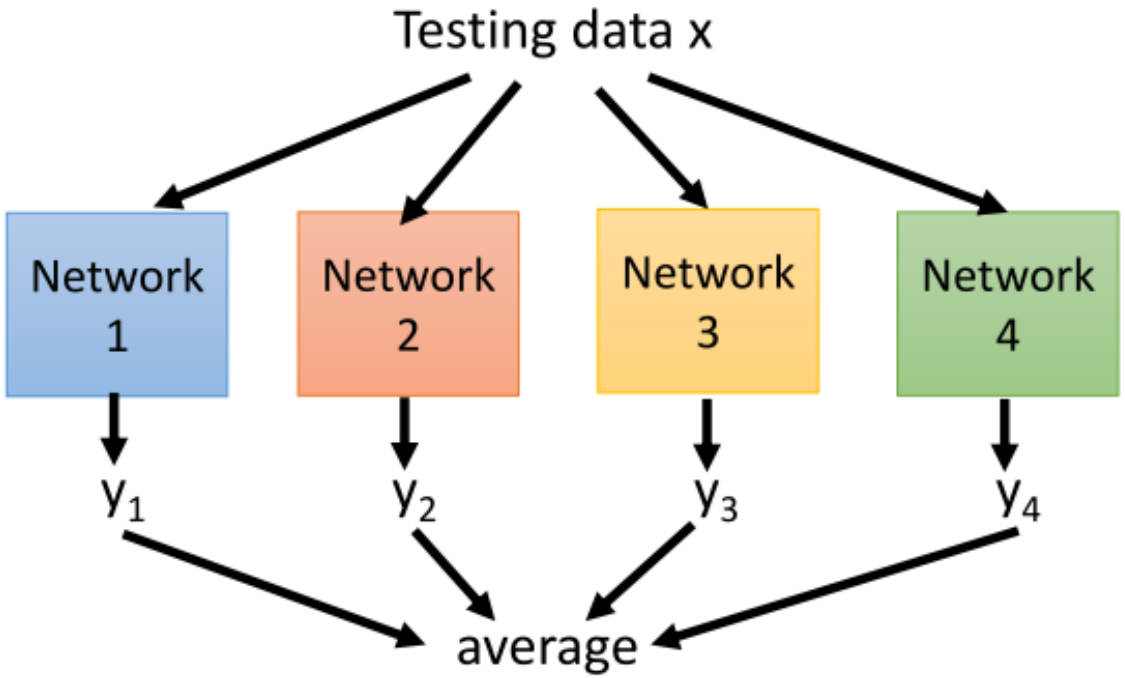
Ensemble



Train a bunch of networks with different structures

Dropout
- Ensemble

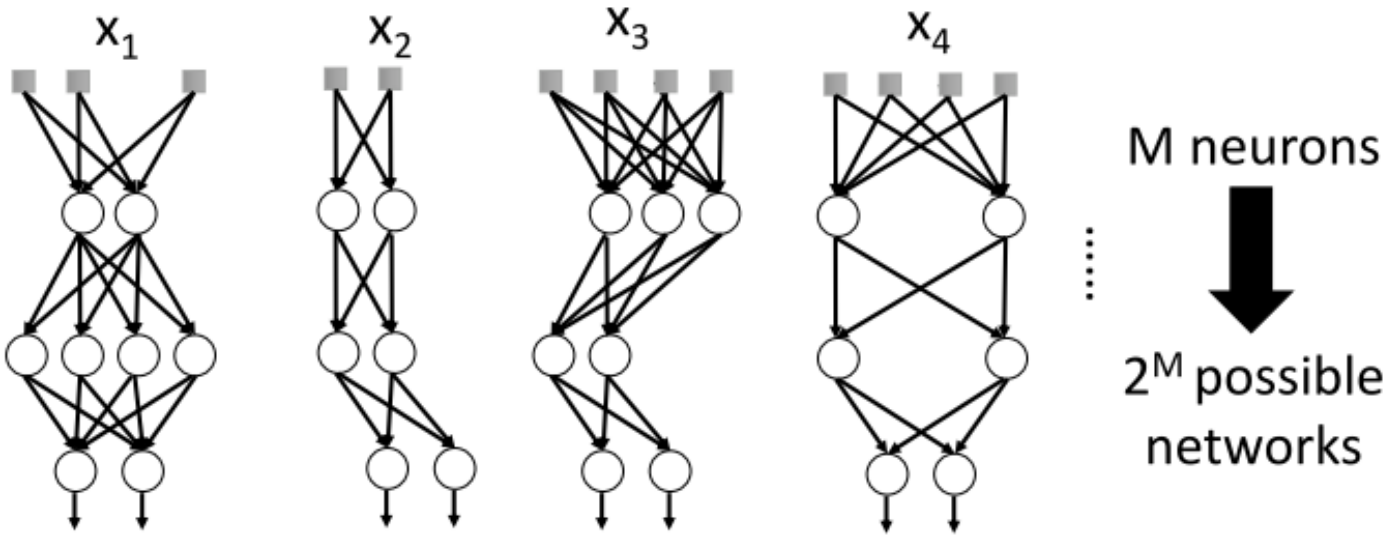
Ensemble



Dropout - Ensemble

Dropout ≈ ***Ensemble***.

Training of Dropout

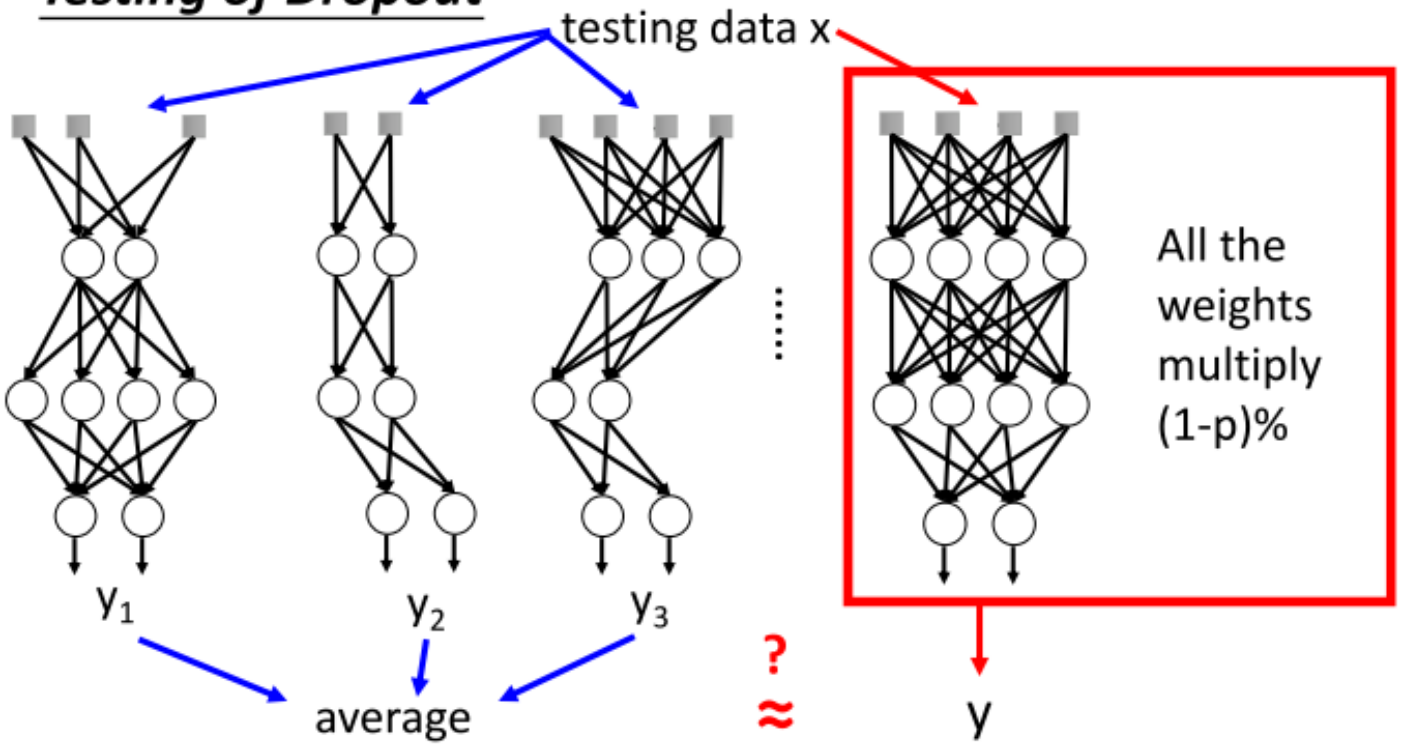


- Using one data to train one network
- Some parameters in the network are shared

Dropout - Ensemble

Dropout ≈ ***Ensemble***.

Testing of Dropout



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.

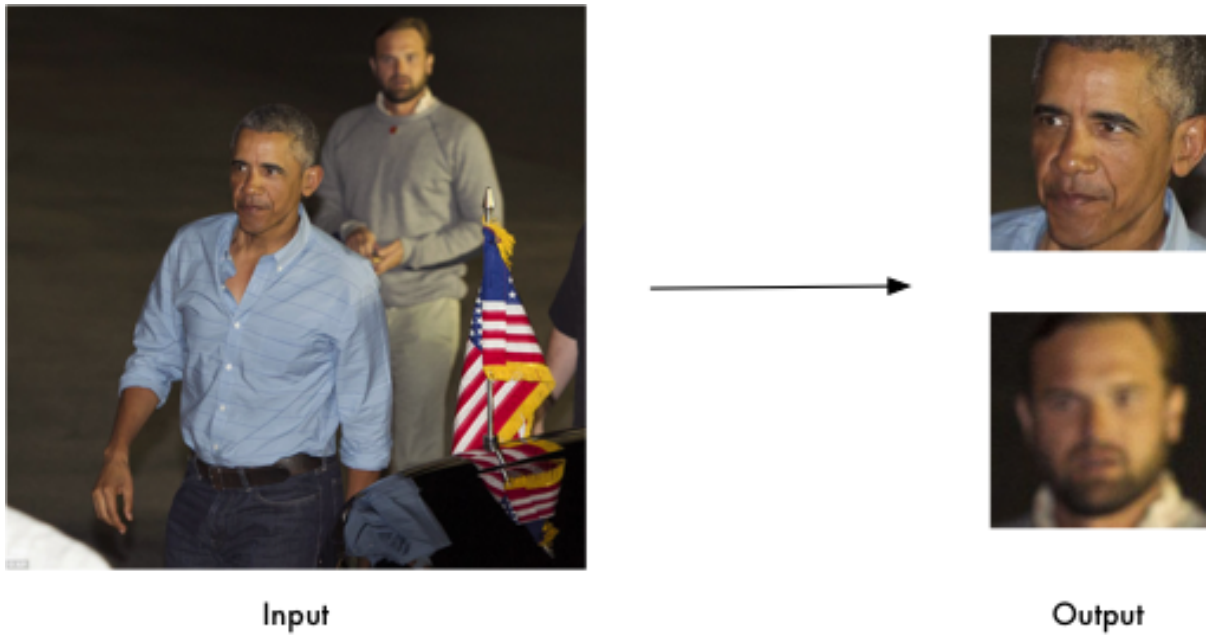
- https://github.com/facebookresearch/fastText?utm_source=mybridge&utm_medium=blog&utm_campaign=read_more

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

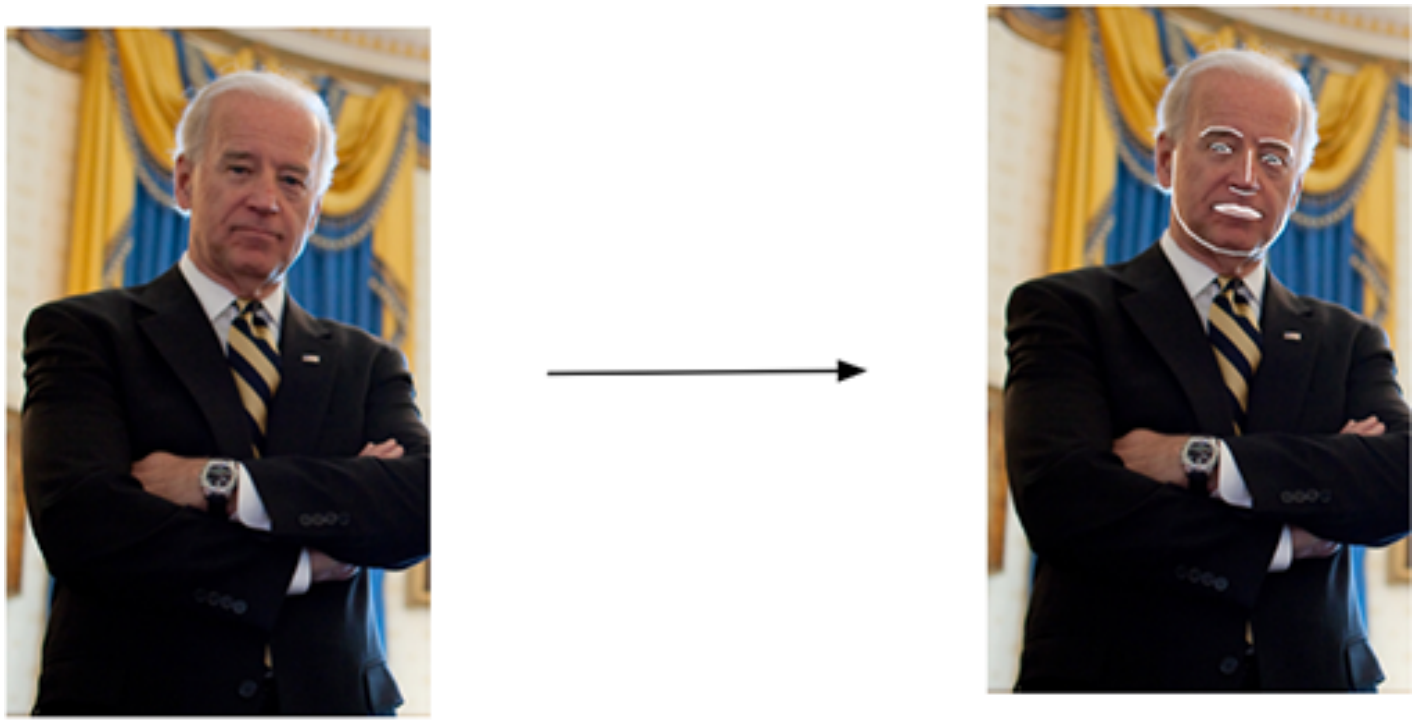
```
import face_recognition
image = face_recognition.load_image_file("your_file.jpg")
face_locations = face_recognition.face_locations(image)
```



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

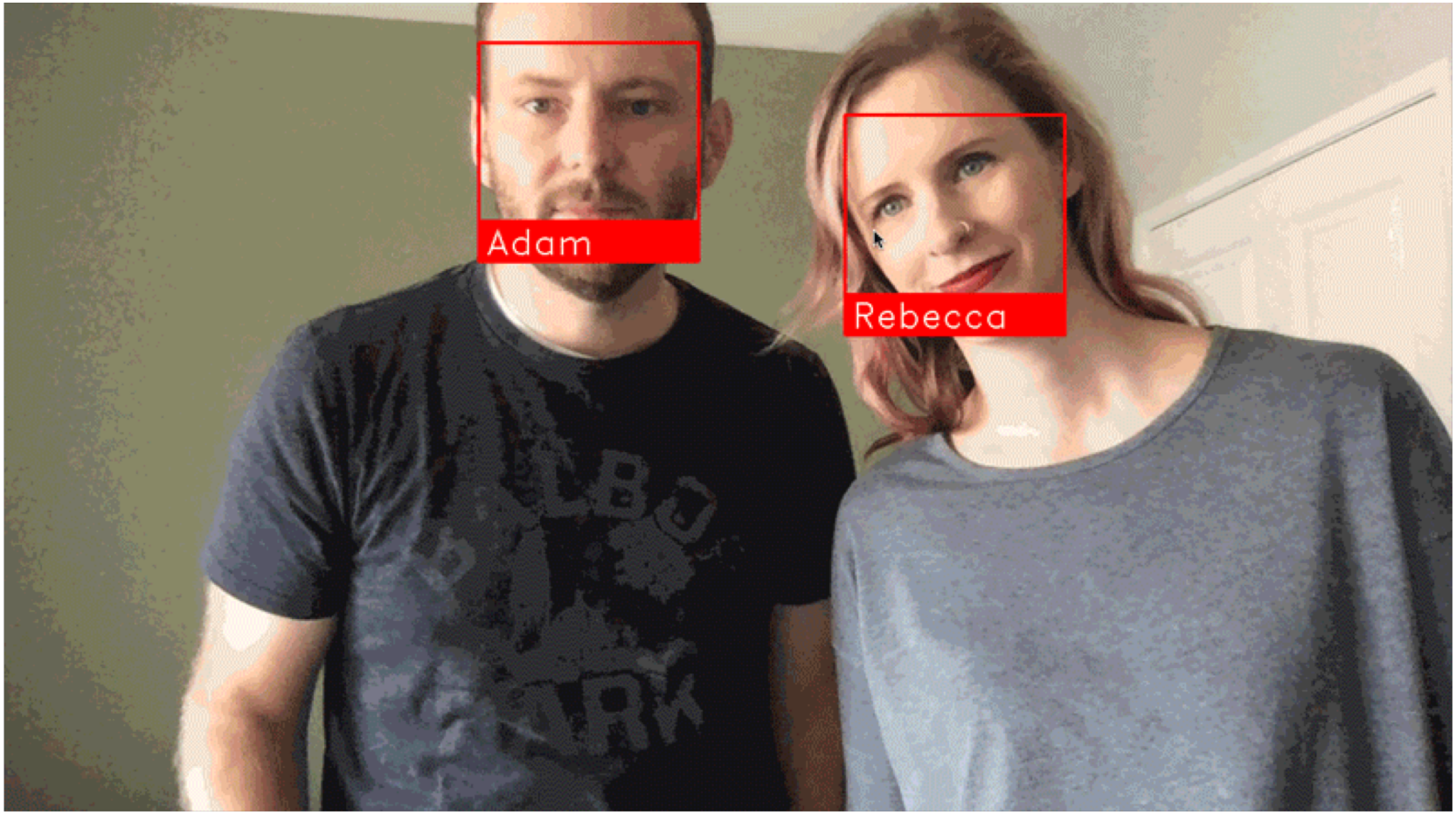
Output

Identify faces in pictures

Input

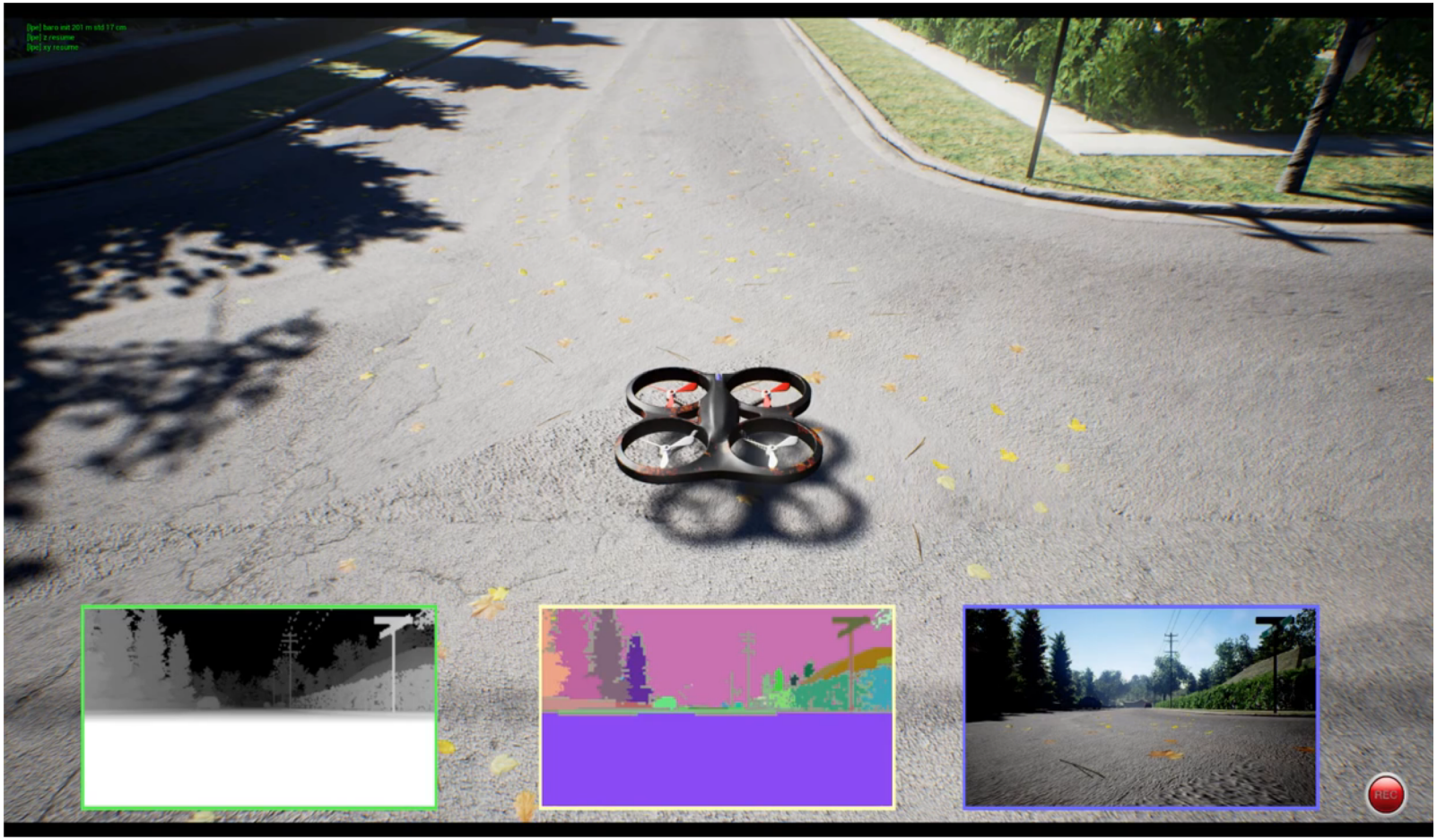
Picture contains
"Joe Biden"

Output


Adam
Rebecca

AirSim

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



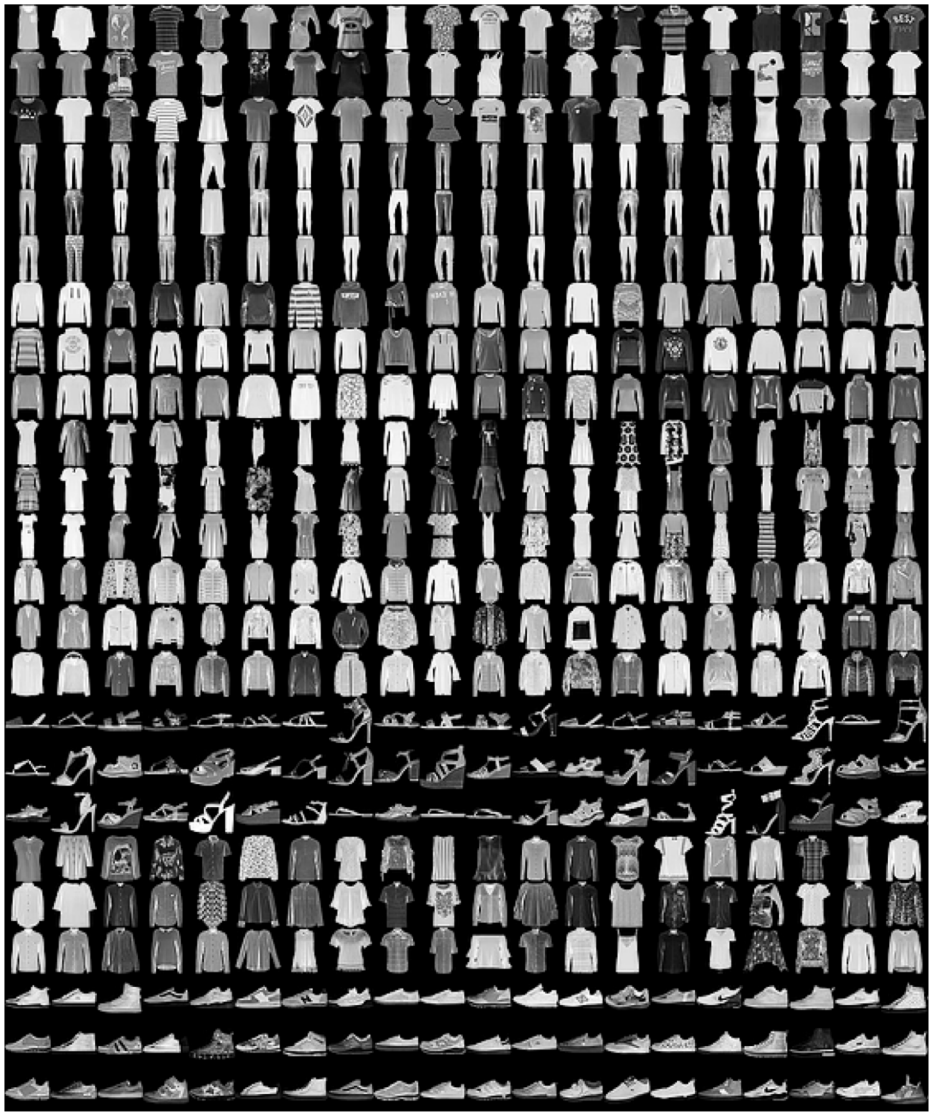
https://github.com/Microsoft/AirSim

CARLA Simulator

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

Fashion-MNIST

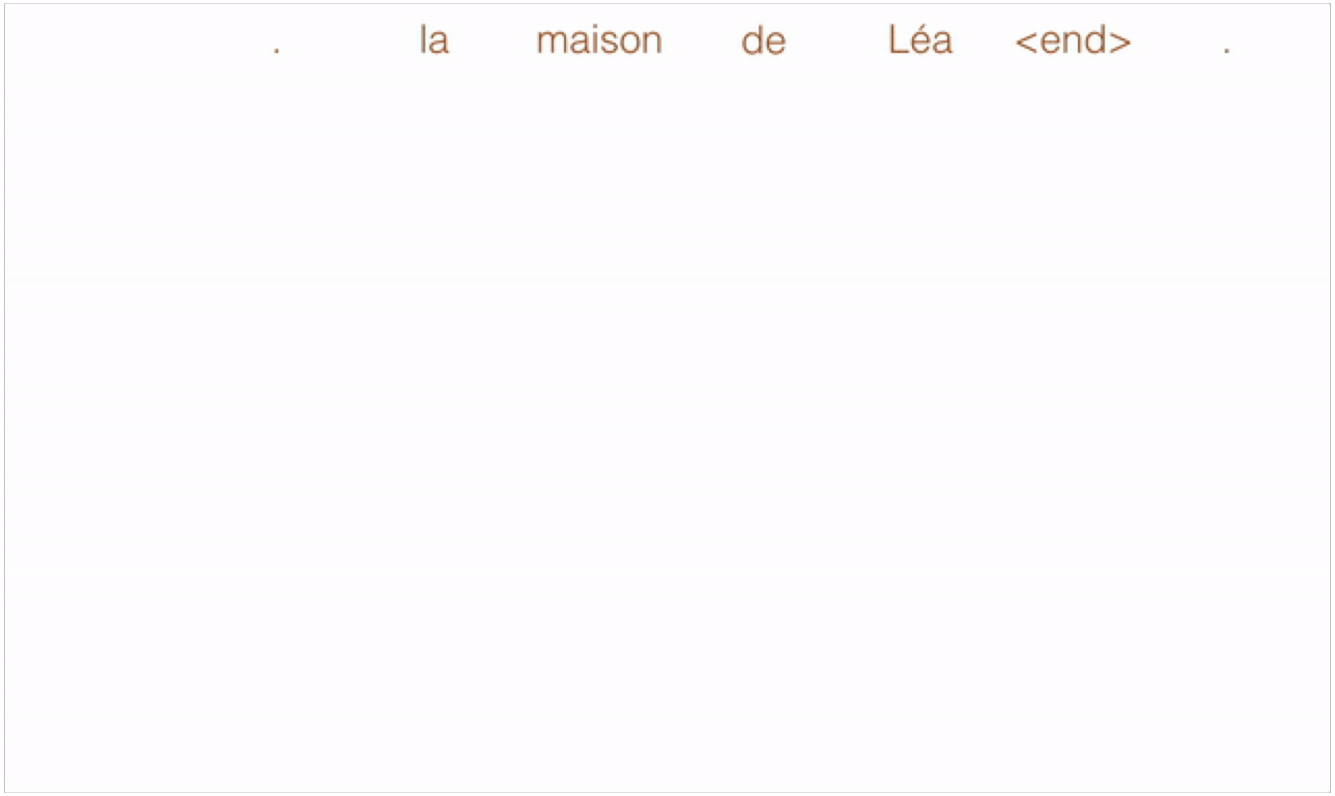
Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot



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

Fairseq

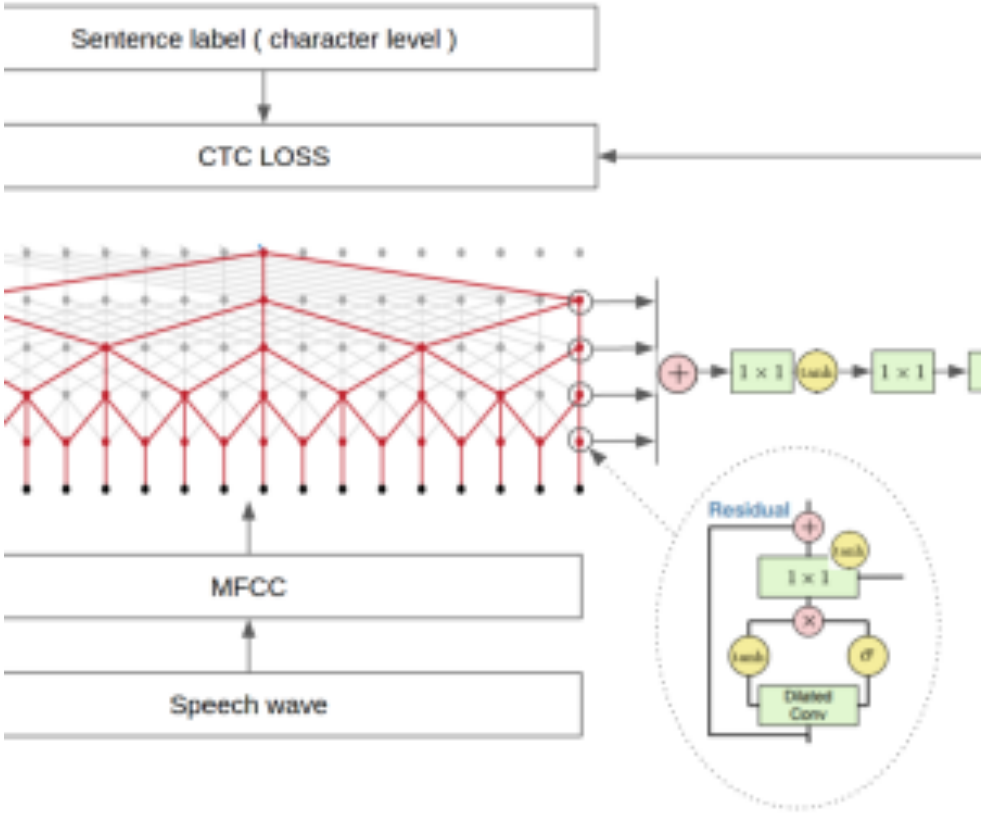
- sequence-to-sequence learning toolkit for Torch from Facebook AI Research
- English to French, English to German and English to Romanian translation



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

Feature Selection
-A Data Perspective

Li, J., Cheng, K., Wang, S., Morstatter, F., Trevino, R. P., Tang, J., & Liu, H. (2017). Feature selection: A data perspective. *ACM Computing Surveys*, *50*(6).

https://arxiv.org/abs/1601.07996

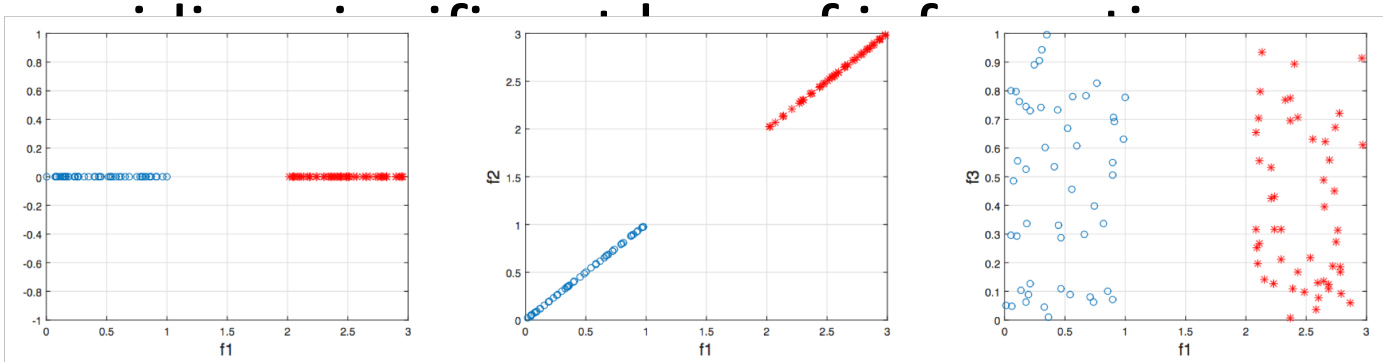
Intro

- The problems when handling **high dimensional data**
  - Curse of dimensionality: data becomes sparser in high dimensional space
  - Overfitting: cause performance degradation on unseen data

- Dimensionality reduction
  - Feature extraction
    - Projects original high dimensional feature space to a new one with low dimensionality
    - Principle Component Analysis, Linear Discriminant Analysis, etc.
  - **Feature selection**
    - Selects a subset of relevant features for the use model construction.
    - Lasso, Information Gain, Relif, Laplacian, etc.

Intro

- Why feature selection?
  - Feature extraction builds a set of new features, further analysis is problematic as we cannot get the physical meaning of these feauture in the transformed space.
  - Reducing storage and computational cost while egative



(a) relevant feature f_1 (b) redundant feature f_2 (c) irrelevant feature f_3

Intro

- Traditional Categorizations of Feature Selection Algorithms
  - Label Perspective

Feature Selection	Supervised	Unsupervised	Semi-Supervised
Availability of label information	when sufficient	do not require any label information	In many real-world applications, we usually have a small number of labeled samples and a large number of unlabeled samples.
Designed for type of problems	Classification or regression problem	Clustering problem	

Intro

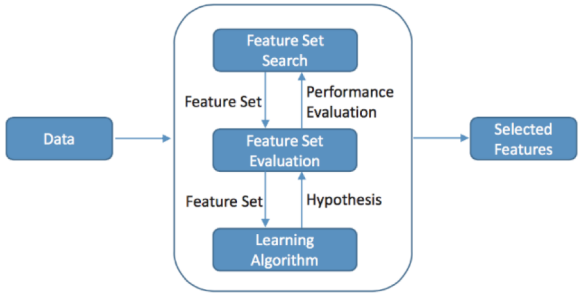


Figure 6: A General Framework of Wrapper Feature Selection Methods.

- Traditional Categorizations of Feature Selection Algorithms
  - **Wrapper Method**
    - Rely on the predictive performance of a predefined learning algorithm to evaluate the quality of selected features.
    - Steps: (1)Search for a subset of features and (2)Evaluate selected features.
    - Repeats (1) and (2) until some stopping criteria are satisfied or the desired learning performance is obtained.
    - Disadvantage: The search space for d features is 2^d, which

Intro

- Traditional Categorizations of Feature Selection Algorithms
  - **Filter Method**
    - Independent of any learning algorithms.
    - Typically more efficient than wrapper methods.
    - Rely on certain characteristics of data to assess the importance of features.
    - (1) Feature importance is ranked by a feature score according to some feature evaluation criteria.
    - (2) Low ranking features are filtered out and the remaining features are selected.

Disadvantages

- Due to the lack of a specific learning algorithm guiding the feature selection phase, the selected features may not be optimal for the target learning algorithms.

Intro

- Traditional Categorizations of Feature Selection Algorithms
  - **Embedded Method**
    - A trade-off solution between filter and wrapper methods which embed the feature selection with the model learning.
    - Include the interactions with the learning algorithm.
    - The most widely used embedded methods are the regularization models which targets to fit a learning model by minimizing the fitting errors and forcing the feature coefficients to be small (or exact zero) simultaneously.

Intro - Feature Selection Algorithms from A Data Perspective

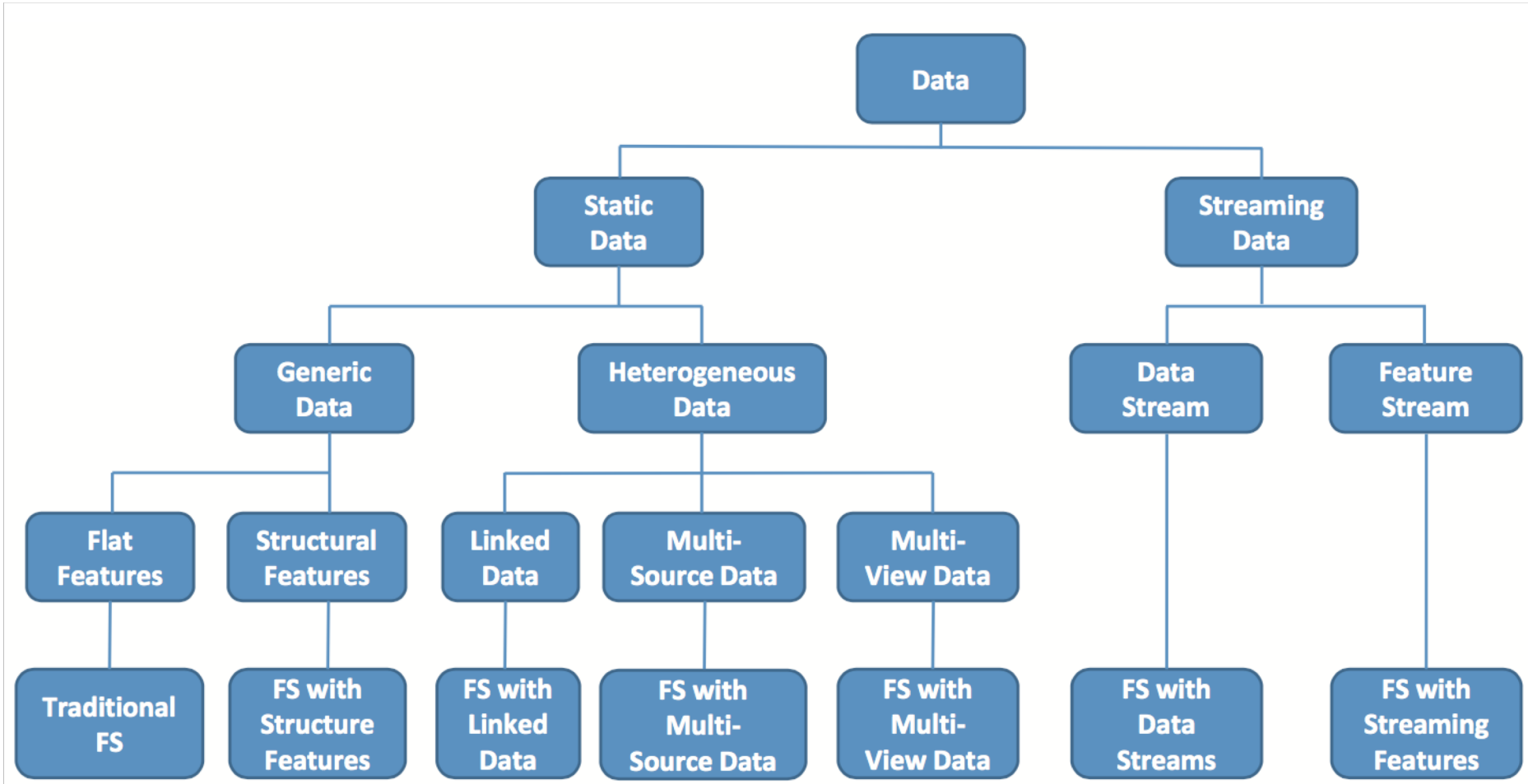


Figure 7: Feature Selection Algorithms from the Data Perspective.

Intro

- Feature Selection Algorithms from A Data Perspective
  - Streaming Data and Features
    - For example in Twitter, new data like posts and new features like slang words are continuously being generated. It is impractical to apply traditional batch-mode feature selection algorithms to find relevant features at each round when new data or new feature arrives.
  - Heterogeneous Data
    - Most existing algorithms of feature selection assume that the data is independent and identically distributed (i.i.d.). However, multi-source data is quite prevalent in many domains.
    - For instance, with the existence of link information, the widely adopted i.i.d. assumption in most machine learning algorithms does not hold. How to appropriately utilize link information for feature selection is still a challenging problem.

Intro

- Feature Selection Algorithms from A Data Perspective
  - Structures Between Features
    - Some well-known structures among features are group structure, tree structure, graph structure, etc.
    - Incorporating the prior knowledge of feature structures can possibly help select relevant features to greatly improve the learning performance.

Intro

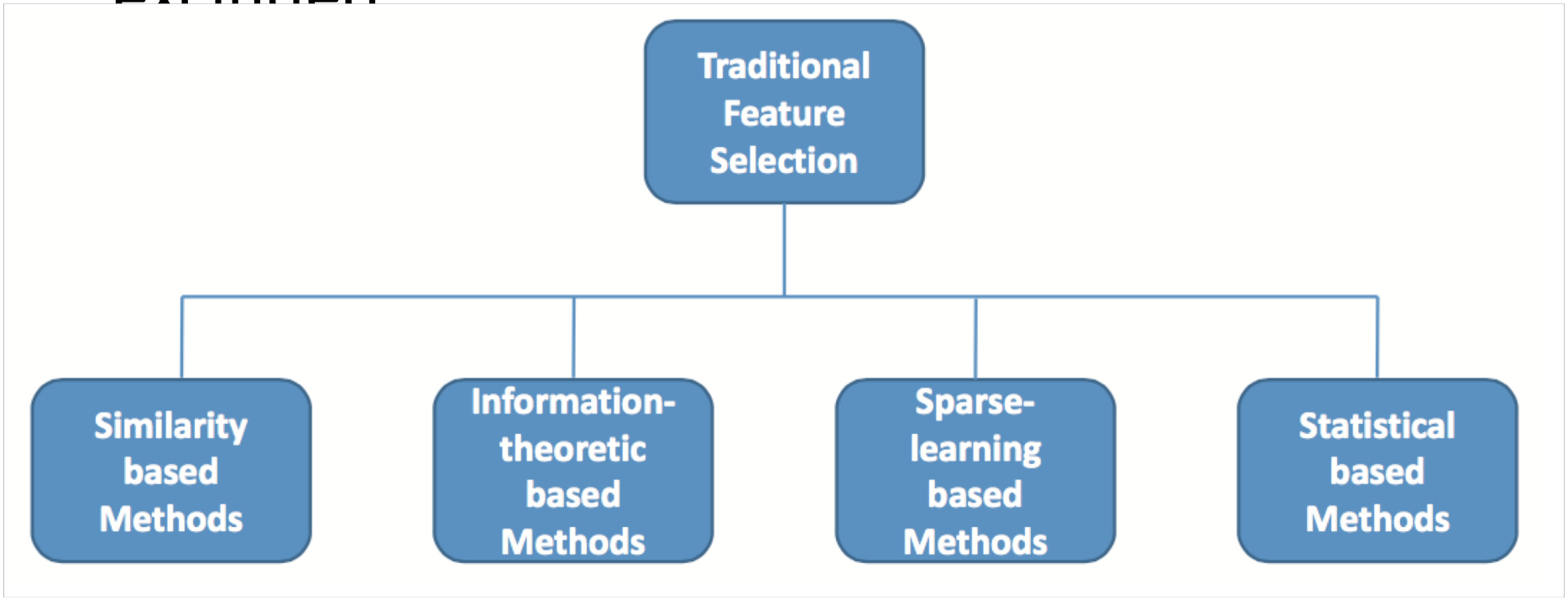
- A feature selection repository in Python named ***scikit-feast*** which is built upon the widely used machine learning package ***scikit-learn*** and two scientific computing packages Numpy and Scipy.
- Includes more than 40 representative feature selection algorithms.
- The website of the repository: http://featureselection.asu.edu/scikit-feast/.

Organization of the Survey

1. Feature Selection with Generic Data (Section 2)
   a) Similarity based Feature Selection Methods
   b) Information Theoretical based Feature Selection Methods
   c) Sparse Learning based Feature Selection Methods
   d) Statistical based Feature Selection Methods
2. Feature Selection with Structure Features (Section 3)
3. Feature Selection with Heterogeneous Data (Section 4)
4. Feature Selection with Streaming Data (Section 5)
5. Performance Evaluation (Section 6)
6. Open Problems and Challenges (Section 7)
7. Summary of the Survey (Section 8)

Section 2. Feature Selection on Generic Data

- Only involves filter methods and embedded methods while the wrapper methods are excluded



2.1 Similarity based Methods

- Different feature selection algorithms exploit various types of criteria to define the relevance of features
  - such as distance, separability, information, correlation, dependency, and reconstruction error.
- Among them, there is a family of methods assessing the importance of features by its ability to preserve data similarity.

Notation

Notations	Definitions or Descriptions
n	number of instances in the data
d	number of features in the data
k	number of selected features
c	number of classes (if exist)
$\mathcal{F}$	original feature set which contains d features
$\mathcal{S}$	selected feature set which contains k selected features
$\{i_1, i_2, ..., i_k\}$	index of k selected features in $\mathcal{S}$
$f_1, f_2, ..., f_d$	d features
$f_{i_1}, f_{i_2}, ..., f_{i_k}$	k selected features
$x_1, x_2, ..., x_n$	n data instances
$\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_d$	d feature vectors corresponding to $f_1, f_2, ..., f_d$
$\mathbf{f}_{i_1}, \mathbf{f}_{i_2}, ..., \mathbf{f}_{i_k}$	k feature vectors corresponding to $f_{i_1}, f_{i_2}, ..., f_{i_k}$
$\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$	n data vectors corresponding to $x_1, x_2, ..., x_n$
$y_1, y_2, ..., y_n$	class labels of all n instances (if exist)
$\mathbf{X} \in \mathbb{R}^{n \times d}$	data matrix with n instances and d features
$\mathbf{X}_{\mathcal{F}} \in \mathbb{R}^{n \times k}$	data matrix on the selected k features
$\mathbf{y} \in \mathbb{R}^n$	class label vector for all n instances (if exist)

Table 1: Symbols.

2.1 Similarity based Methods

- Given a dataset $\mathbf{X} \in \mathbb{R}^{n \times d}$ with n instances and d features, the pairwise similarity among instances can be encoded in an affinity matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$.
- The affinity matrix $\mathbf{S}$ is symmetric and its (i, j)-th entry indicates the similarity between the i-th instance x_i and the j-th instance x_j, the larger the value of $\mathbf{S}_{i,j}$ is, the more similarity x_i and x_j share.

2.1 Similarity based Methods

- Suppose we want to select k most relevant features from $\mathcal{F}$, then the utility of these k features is maximized as
  - $\max_{\mathcal{F}} \sum_{f \in \mathcal{F}} SC(f) = \max_{\mathcal{F}} \sum_{\mathbf{f} \in \mathcal{F}} \widehat{\mathbf{f}}' \hat{\mathbf{S}} \hat{\mathbf{f}}$

  where SC is a function that measures the utility of feature $\mathbf{f}$

  $\widehat{\mathbf{f}}'$ are the normalized feature

  $\hat{\mathbf{S}}$ are refined affinity matrix obtained from $\mathbf{f}$ and $\mathbf{S}$
- We would select a subset of features from $\mathcal{F}$ such that they can well preserve the data similarity structures defined in $\hat{\mathbf{S}}$.
- Usually solved by greedily selecting the top k features that maximize their individual utility $\widehat{\mathbf{f}}' \hat{\mathbf{S}} \hat{\mathbf{f}}$. Methods in this category vary in the way the similarity matrix $\mathbf{S}$ is designed.

2.1.1 Laplacian Score (He et al., 2005) (Unsupervised)

- An unsupervised feature selection algorithm which selects features that can best preserve the data manifold structure.
- Three phases:
  - First, it constructs a nearest neighbor graph $\mathcal{G}$ with n nodes where the i-th node corresponds to x_i. If x_i is among the p nearest neighbors of x_j or x_j is among the p nearest neighbors of x_i, nodes i and j are connected in $\mathcal{G}$ (p is a predefined number).
  - Second, if nodes i and j are connected, the entry in the
  - affinity matrix $\mathbf{S}_{\mathrm{ij}}$ is $\mathbf{S}(i,j) = e^{-\frac{\left\|x_i - x_j\right\|^2}{t}}$, where t is a constant, otherwise $\mathbf{S}(i,j) = 0$.

2.1.1 Laplacian Score (He et al., 2005) (Unsupervised)

- The diagonal matrix $\mathbf{D}$ is defined as $\mathbf{D}(i,i) = \sum_{j=1}^{n} \mathbf{S}(i,j)$
- The laplacian matrix $\mathbf{L}$ is $\mathbf{L} = \mathbf{D} - \mathbf{S}$
- the Laplacian Score of each feature f_i
  - $laplacian\ score(f_i) = \frac{\widetilde{\mathbf{f}}_i' \mathbf{L} \widetilde{\mathbf{f}}_i}{\widetilde{\mathbf{f}}_i' \mathbf{D} \widetilde{\mathbf{f}}_i}$, where $\widetilde{\mathbf{f}}_i = \mathbf{f}_i - \frac{\mathbf{f}_i' \mathbf{D1}}{\mathbf{1}' \mathbf{D1}} \mathbf{1}$

- Laplacian Score evaluates the importance of each feature individually, the task of selecting the k features can be solved by greedily picking the top k features with the **smallest** Laplacian Scores.

Justification

- A ”good” feature should the one on which two data points are close to each other if and only if there is an edge between these two points.

$$L_r = \frac{\sum_{ij}(f_{ri} - f_{rj})^2 S_{ij}}{Var(\mathbf{f}_r)}$$

- Var($\mathbf{f}_r$) is the estimated variance of the r-th feature

$$\sum_{ij} (f_{ri} - f_{rj})^2 S_{ij} = \sum_{ij} \left(f_{ri}^2 + f_{rj}^2 - 2 f_{ri} f_{rj}\right) S_{ij}$$

$$= \quad 2\sum_{ij} f_{ri}^2 S_{ij} - 2\sum_{ij} f_{ri} S_{ij} f_{rj} = 2\mathbf{f}_r^T D\mathbf{f}_r - 2\mathbf{f}_r^T S\mathbf{f}_r = 2\mathbf{f}_r^T L\mathbf{f}_r$$

$$Var(\mathbf{f}_r) = \sum_i (\mathbf{f}_{ri} - \mu_r)^2 D_{ii}$$

$$\mu_r = \sum_i \left(\mathbf{f}_{ri} \frac{D_{ii}}{\sum_i D_{ii}}\right) = \frac{1}{\left(\sum_i D_{ii}\right)} \left(\sum_i \mathbf{f}_{ri} D_{ii}\right) = \frac{\mathbf{f}_r^T D\mathbf{1}}{\mathbf{1}^T D\mathbf{1}}$$

2.1.1 Laplacian Score (He et al., 2005) (Unsupervised)

- Reformulate
  - $laplacian\ score(f_i) = \frac{\widetilde{\mathbf{f}}_i' \mathbf{L} \widetilde{\mathbf{f}}_i}{\widetilde{\mathbf{f}}_i' \mathbf{D} \widetilde{\mathbf{f}}_i} = \frac{\widetilde{\mathbf{f}}_i' (\mathbf{D}-\mathbf{S}) \widetilde{\mathbf{f}}_i}{\widetilde{\mathbf{f}}_i' \mathbf{D} \widetilde{\mathbf{f}}_i} = 1 - \frac{\widetilde{\mathbf{f}}_i' \mathbf{S} \widetilde{\mathbf{f}}_i}{\widetilde{\mathbf{f}}_i' \mathbf{D} \widetilde{\mathbf{f}}_i}$

$$= 1 - \left(\frac{\widetilde{\mathbf{f}}_i}{\left\| \mathbf{D}^{\frac{1}{2}} \widetilde{\mathbf{f}}_i \right\|} \right)' \mathbf{S} \left(\frac{\widetilde{\mathbf{f}}_i}{\left\| \mathbf{D}^{\frac{1}{2}} \widetilde{\mathbf{f}}_i \right\|} \right)$$

  - $\widetilde{\mathbf{f}}_i' \mathbf{D} \widetilde{\mathbf{f}}_i$ is the weighted data variance of feature f_i (denoted as σ_i^2)
  - $\left\| \mathbf{D}^{\frac{1}{2}} \widetilde{\mathbf{f}}_i \right\|$ is the standard data variance (denoted as σ_i)
  - $\frac{\widetilde{\mathbf{f}}_i}{\left\| \mathbf{D}^{\frac{1}{2}} \widetilde{\mathbf{f}}_i \right\|}$ is interpreted as a normalized feature vector $\widehat{\mathbf{f}}_i = \frac{\mathbf{f}_i - \mu_i \mathbf{1}}{\sigma}$

- Therefore, Laplacian Score feature selection can be reformulated by maximizing the following:

$$\max_{\mathcal{F}} \sum_{\mathbf{f} \in \mathcal{F}} \widehat{\mathbf{f}}' \widehat{\mathbf{S}} \widehat{\mathbf{f}}$$

Other methods

- Fisher score
- Trace Ratio Criterion
- ReliefF
- information gain

Q & A

Thanks.