Shuang Leung ML note

Table of Contents

1. Introduction & Intuition

How do human make decisions?

1. **trained** from experience
2. **learn** to make good decisions

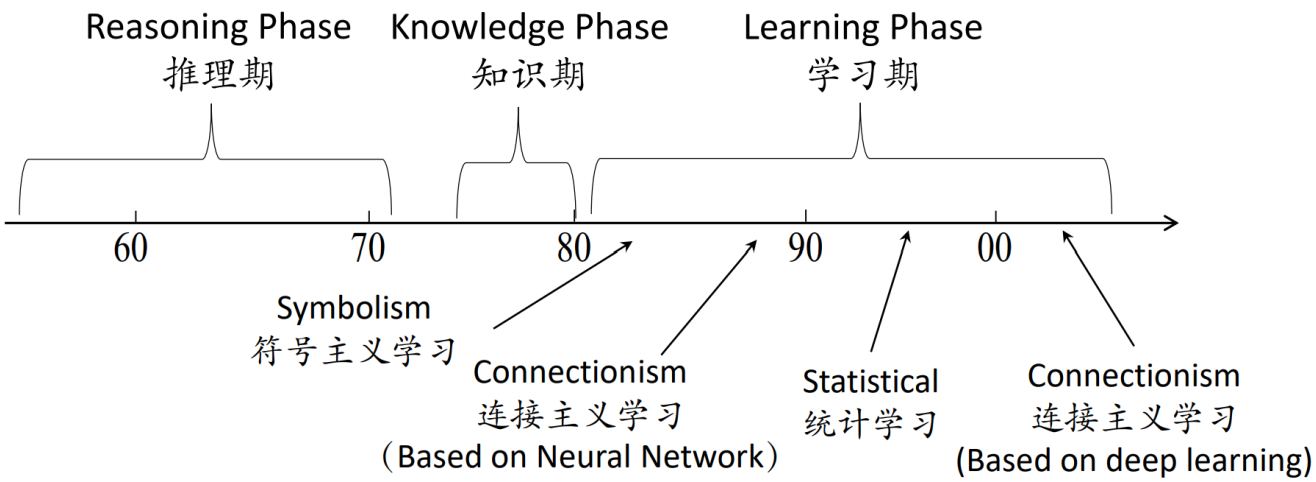
AI(goal)->pattern recognition(task,70/80s)->machine learning->deep learning

a. history

60-70s: reasoning phase

late 70s-80s: knowledge phase

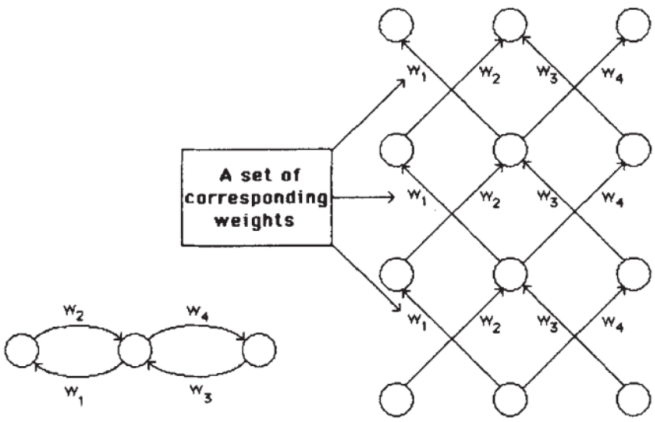
80s-now: learning phase



Connectionism-Phase 1

Neural Network

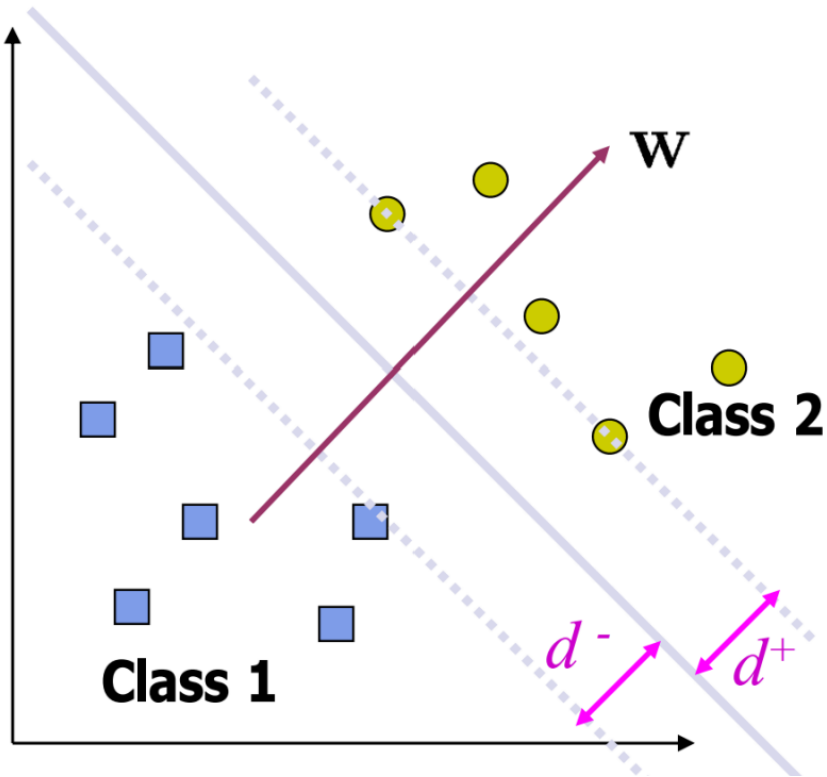
- **Connectionism – Phase 1 连接主义学习**
- Popular before 1990s
- Based on neural networks
- The backpropagation algorithm made a profound impact



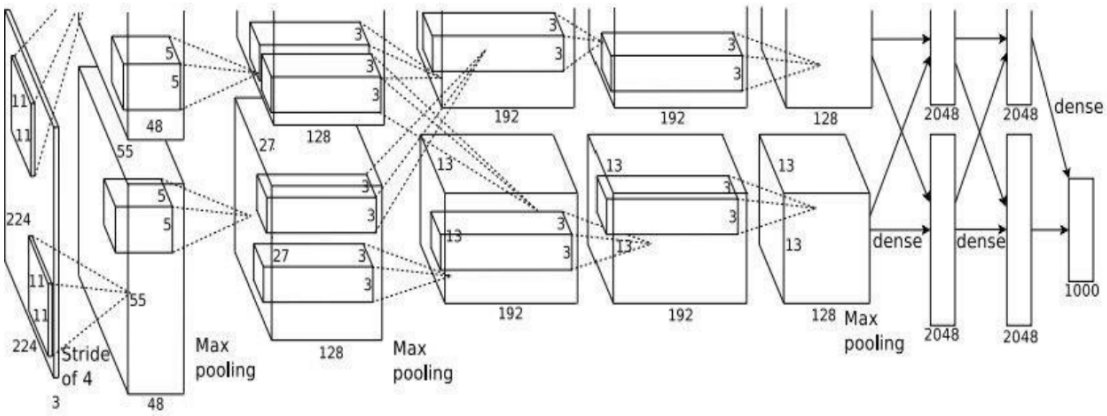
Rumelhart D E, Hinton G E, Williams R J. Learning representations by back-propagating errors[J]. nature, 1986, 323(6088): 533-536.

Statistic

SVM

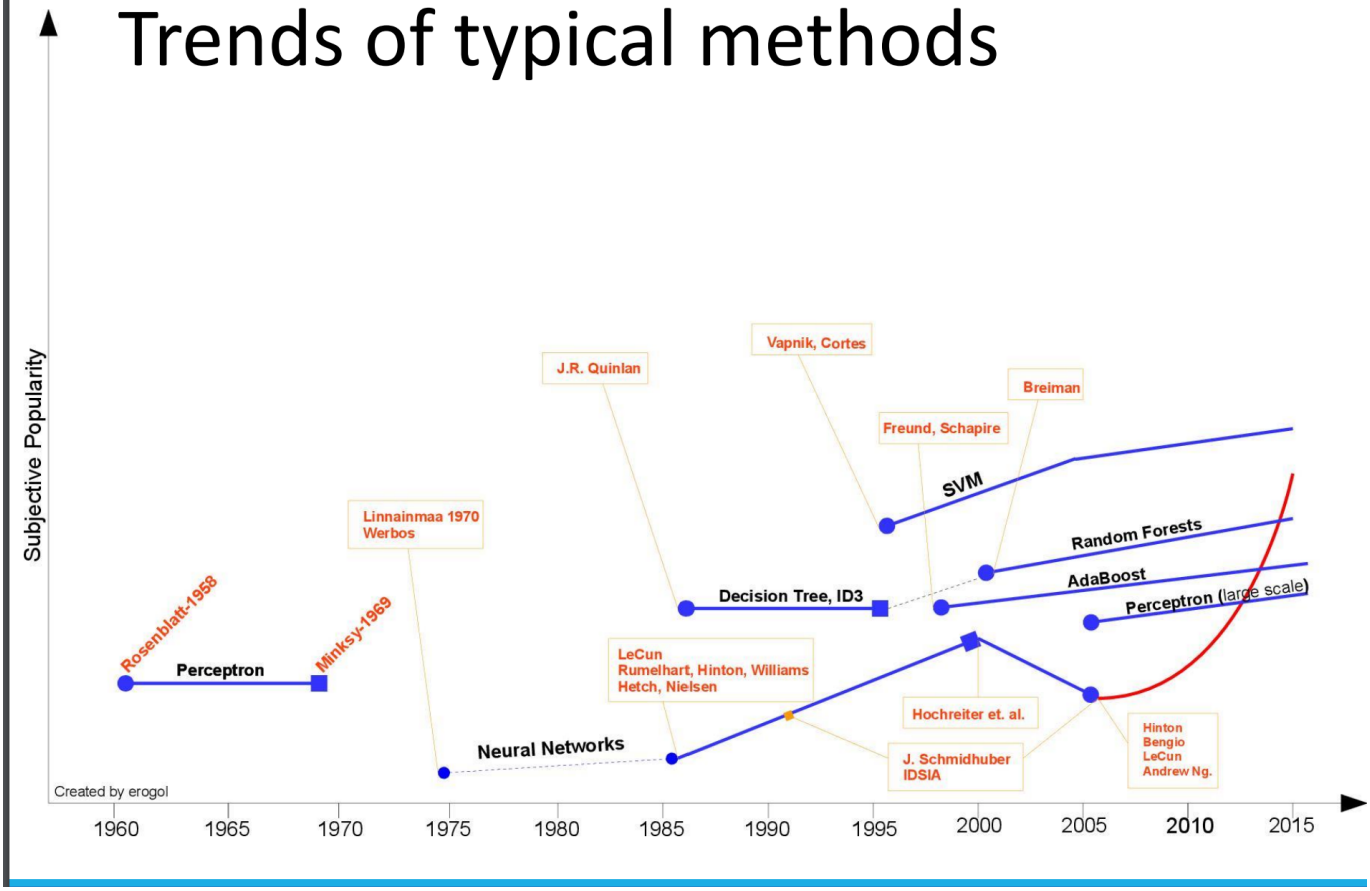


Connectionism-Phase 2



trend

Trends of typical methods



challenge

Challenges

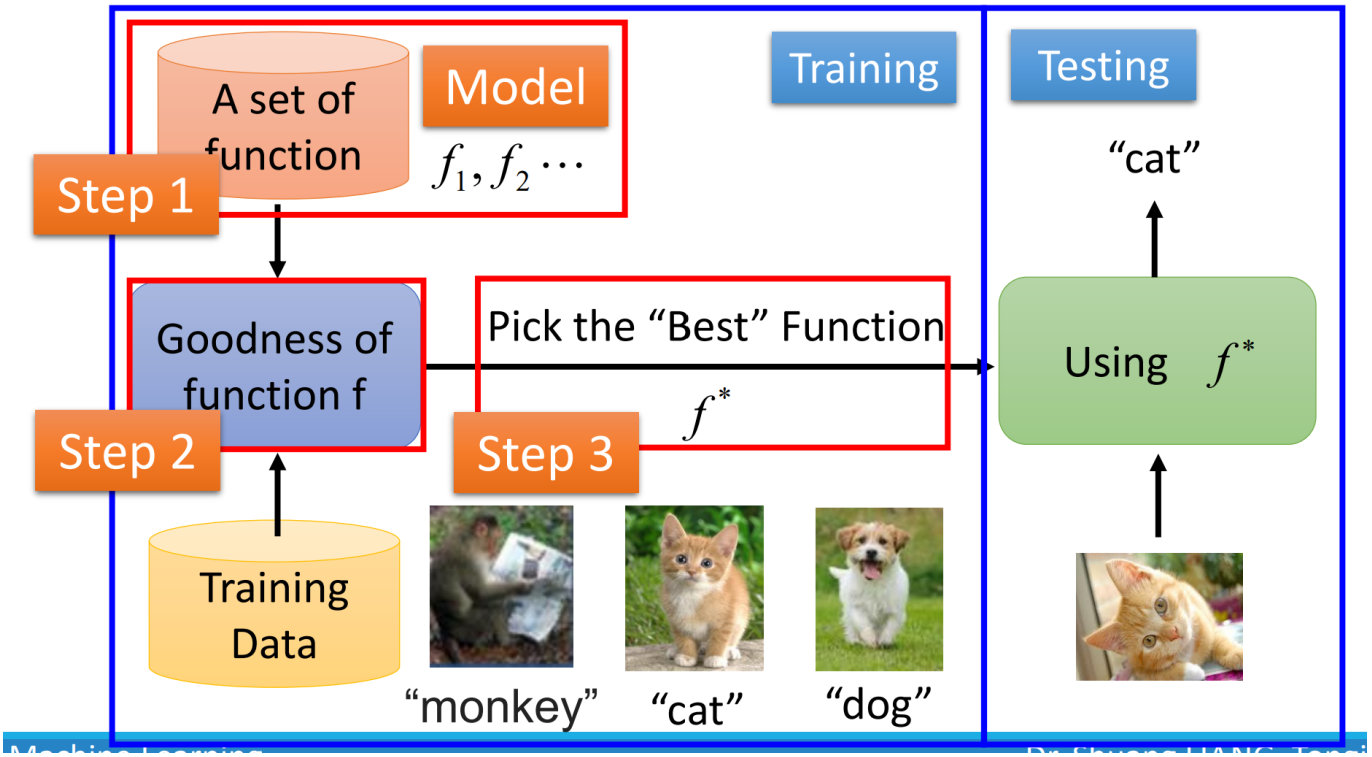
- Hype
  - Cycles of AI popularity
- Data Ethics, Privacy, Fairness
- Lack of Interpretability
  - Example: medical applications of deep learning
- Social Implications of AI
  - Threats from Super-human AI?

Need: Scientific Method, Reproducible Research, Open Source and Open Data

b. framework & map

Framework

Image Recognition:

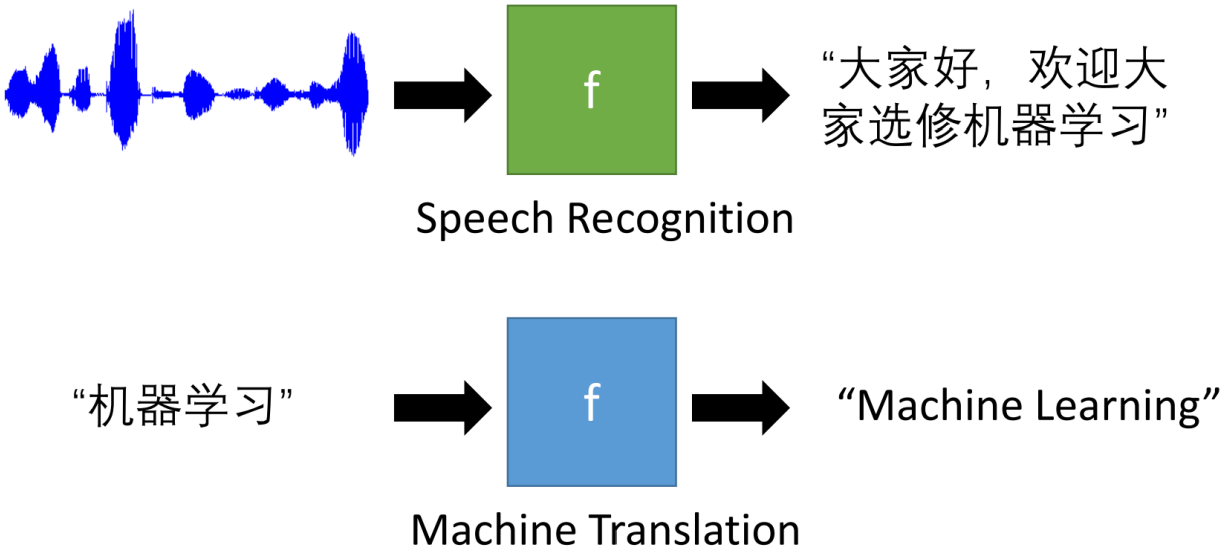


supervised learning(data,label)

1. regression: output scalars
2. classification: output class
   - 2.1 binary classification
   - 2.2 multi-class classification
3. structured learning(beyond classification & regression e.g. ASR, machine translation):

Structured Learning

- Beyond Classification



classification methods

1. linear
2. non-linear
   2.1 deep learning
   2.2 SVM, decision tree, KNN...

non-supervised learning

1. semi-supervised(some labeled,some not)
2. weak-supervised(incompletely labeled)
3. transfer learning(problems alike)
4. unsupervised learning:

Unsupervised Learning

Ref: https://openai.com/blog/generative-models/

Draw something!

Unsupervised Learning

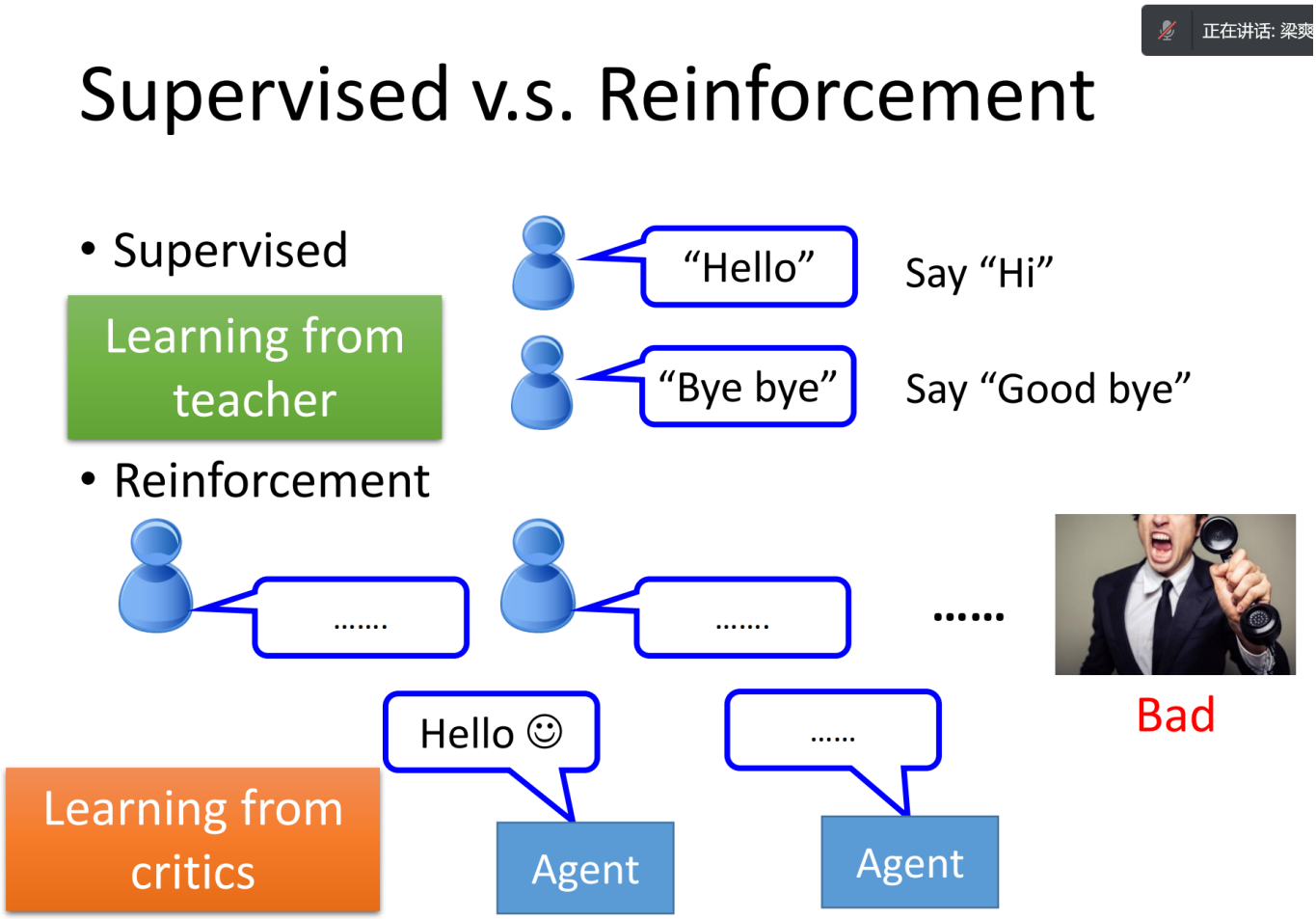
Machine listens to lots of audio book

[Chung & Lee, INTERSPEECH 2016]

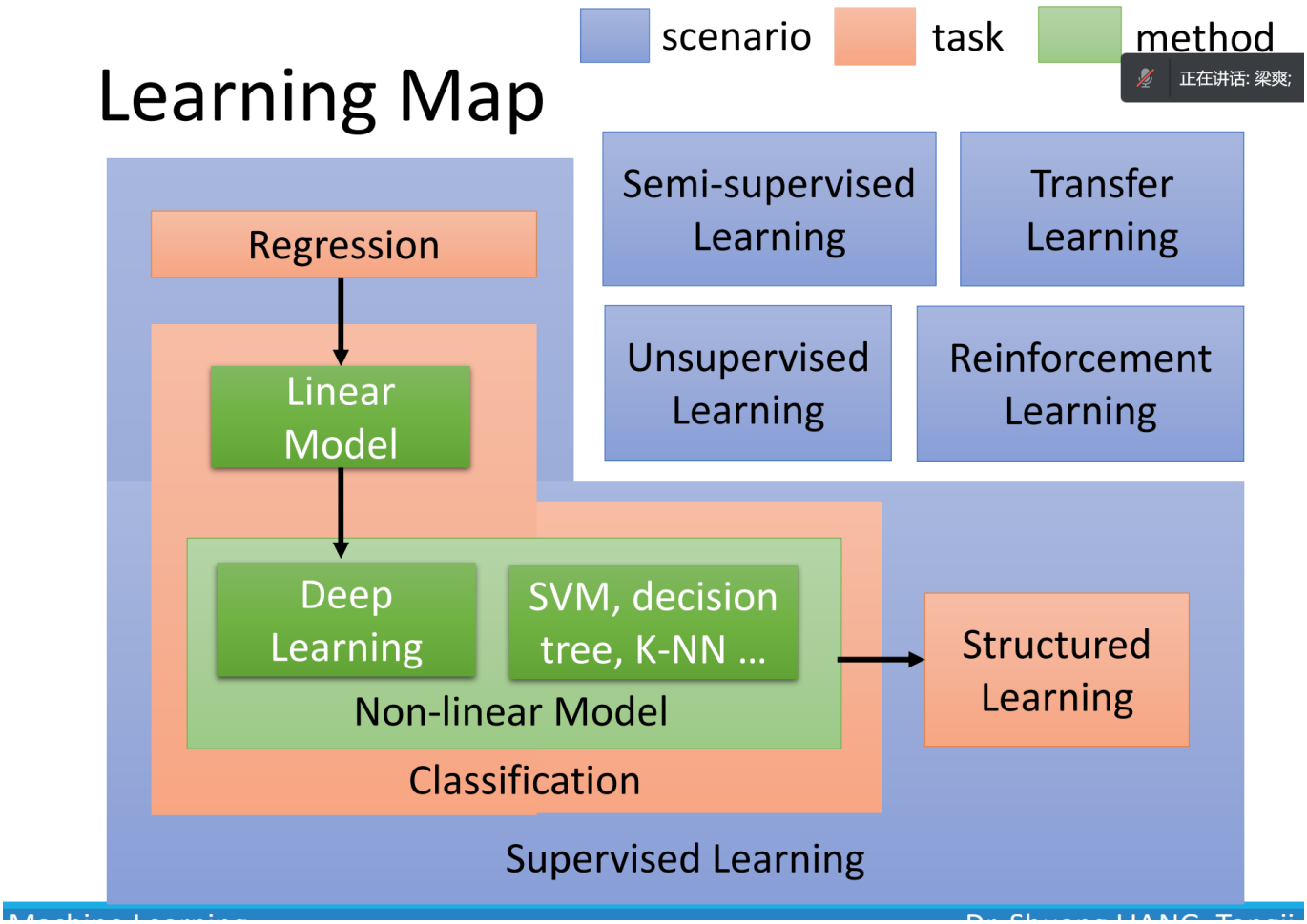
How about machine watch TV?

reinforced learning

learning from critics(feedbacks)



roadmap



2. Mathematical Fundamentals

概率的公理化定义

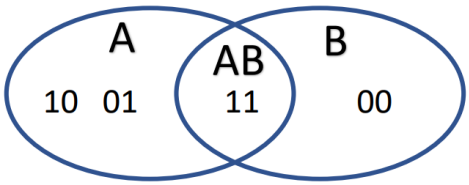
- 设某试验的样本空间为Ω，对其中每个事件A定义一个实数$P(A)$，如果它满足下列三条公理，称$P(A)$为A的概率
  - $0 \leq P(A) \leq 1$
  - $P(\Omega) = 1$
  - 若事件$A_1, A_2, \ldots, A_n$互不相容，则

$$P\left(\sum_{i=1}^{n} A_i\right) = \sum_{i=1}^{n} P(A_i)$$

conditional probability

条件概率

- **例：**将一枚硬币抛掷两次，观察其出现正反面的情况。
- 设事件A表示“至少有一次出现正面”
- 事件B表示“两次都出现同一面”



设1为正，0为反。样本空间$S=\{11,10,01,00\}$
事件$A=\{11,10,01\}$，事件$B=\{11,00\}$

- $P(B|A)=\frac{N(AB)}{N(A)}=\frac{1}{3}$
- 可理解为：B的样本点在A中占的比例
- $P(B|A)=\frac{N(AB)}{N(A)}=\frac{\frac{N(AB)}{N(S)}}{\frac{N(A)}{N(S)}}=\frac{P(AB)}{P(A)}$，由此得到了条件概率的定义

Total probability formula -> prior probability

全概率公式

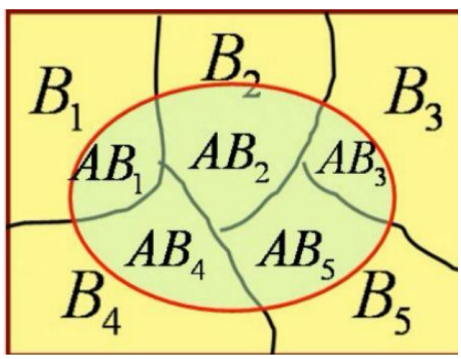
- 根据设试验E的样本空间为S，$(B_1,B_2,\dots,B_n)$为S的一个划分，且$P(B_i)>0\ (i=1,2,\dots,n)$，A为E的一个事件，则

$$P(A)=P(B_1)P(A|B_1)+P(B_2)P(A|B_2)+\cdots+P(B_n)P(A|B_n)=\sum_{i=1}^{n}P(B_i)P(A|B_i)$$

- 证明：

乘法公式

$$P(A)=P(AB_1)+P(AB_2)+\cdots+P(AB_n)$$



- 全概率公式的意义：由原因推结果

reason -> result

Bayes formula-> posterior probability

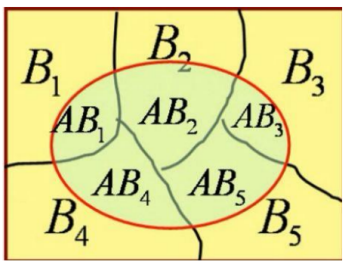
贝叶斯公式

- 乘法公式： $P(AB) = P(A)P(B|A) \quad P(A) > 0$

$$P(AB) = P(B) = P(B)P(A|B) \quad P(B) > 0$$

- 贝叶斯公式： $P(B|A) = \dfrac{P(B)P(A|B)}{P(A)}$
- 进一步： $(B_1, B_2, \dots, B_n)$为样本空间的一个划分，则有

$$P(B_i|A) = \frac{P(B_i)P(A|B_i)}{\sum_{j=1}^{n} P(B_j)P(A|B_j)}$$
$$(i = 1,2, \dots, n)$$



- 贝叶斯公式的意义：**已知结果，寻找原因**

result->reason

prior probability and posterior probability

先验概率与后验概率

- 对以往数据分析表明，机器调整得良好时，产品的合格率为95%，而当机器发生某种故障时，其合格率为50%。设机器调整良好的概率为90%，已知某日生产的第一件产品是合格品，求机器调整良好的概率。
- 用A表示产品合格，B表示机器调整良好。则：

$$P(B) = 0.9 \quad P(B|A) = 0.945$$

- 机器调整良好的概率$P(B) = 0.9$是根据以往的数据分析所得，称为先验概率
- 条件概率是在得到产品合格的信息之后再重新加以修正的概率，称为后验概率

independent event

事件的独立性

- 若事件$A_1, A_2, \dots, A_n$满足条件：

$$P(A_iA_j) = P(A_i)P(A_j) \quad 1 \le i < j \le n$$

则称n个事件$A_1, A_2, \dots, A_n$是两两独立的。

- 若对任意整数$k(2 \le k \le n)$和$1 \le i_1 \le i_2 \le \cdots \le i_k \le n$，恒有：

$$P(A_{i_1}, A_{i_2}, \dots, A_{i_k}) = P(A_{i_1})P(A_{i_2}) \dots P(A_{i_k})$$

则称这n个事件相互独立。

- 相互独立⇒两两独立，两两独立⇏相互独立

random variables

discrete

常用离散型分布

- 0-1分布

$$P\{X=1\}=p, P\{X=0\}=1-p$$

- 二项分布$X\sim B(n,p)$

$$P\{X=k\}=\binom{n}{k}p^k(1-p)^{n-k}=\binom{n}{k}p^kq^{n-k}, k=0,1,2\cdots n$$

- 泊松分布

$$P\{X=k\}=e^{-\lambda}\frac{\lambda^k}{k!}, k=0,1,2\cdots, \lambda>0$$

continuous

常用连续型分布

- 均匀分布

$$f(x)=\begin{cases}\dfrac{1}{b-a} & a\le x\le b\\ 0 & \text{其它}\end{cases}$$

- 指数分布

$$f(x)=\begin{cases}\lambda e^{-\lambda x}, x\ge 0\\ 0, \text{其它}\end{cases}$$

- 高斯分布 $\mathbf{X}\sim N(\mu,\sigma^2)$

$$f(x)=\frac{1}{\sqrt{2\pi}\sigma}\exp\{-\frac{(x-\mu)^2}{2\sigma^2}\} \qquad -\infty<x<\infty$$

numeric features

expectation

variance

随机变量的数字特征

- 方差
- 离散型：

$$D(X) = E[X - E(X)]^2 = \sum_i [x_i - E(X)]^2 p_i$$

- 连续型：

$$D(X) = E[X - E(X)]^2 = \int_{-\infty}^{+\infty} [x - E(X)]^2 f(x) dx$$

- 性质
  - $D(c) = c$
  - $D(cX) = c^2 D(X)$
  - $D(X) = E(x^2) - [E(x)]^2$

extremum

Hessian matrix

Hessian矩阵

- 二元情况下：$H(x_0, y_0) = \begin{bmatrix} f_{xx}(x_0, y_0) & f_{xy}(x_0, y_0) \\ f_{yx}(x_0, y_0) & f_{yy}(x_0, y_0) \end{bmatrix} = \begin{bmatrix} A & B \\ B & C \end{bmatrix}$
- 多元情况下：

$$H(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

- 多元函数极值的判定条件与Hessian矩阵的正定性有关

Lagrange multiplier

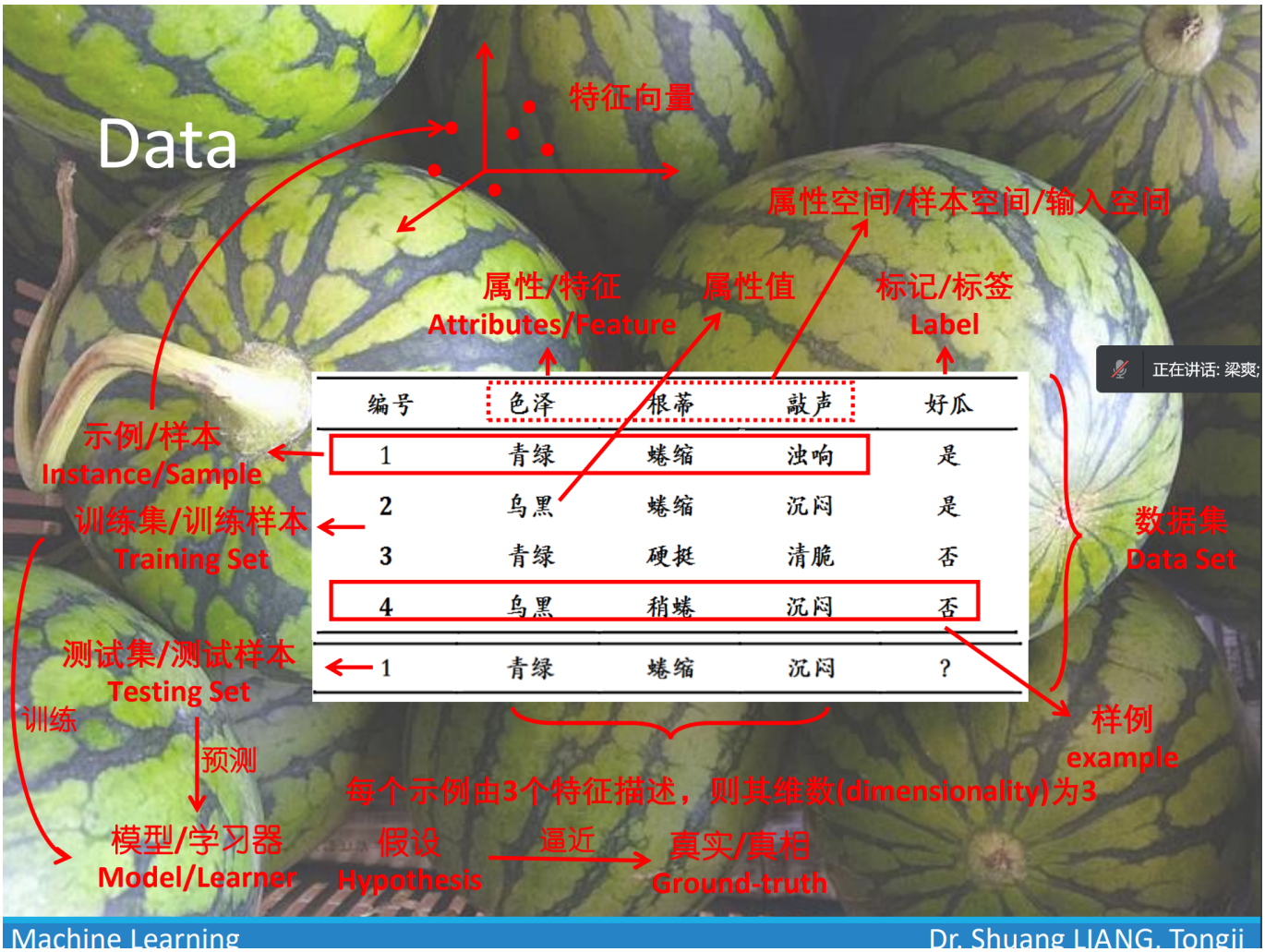
目标：$z=f(x,y)$　　条件：$\varphi(x,y)=0$

1° $L=f(x,y)+\lambda\varphi(x,y)$

2° 令 $\begin{cases} L_x = f_x + \lambda\varphi_x = 0 \\ L_y = f_y + \lambda\varphi_y = 0 \\ L_\lambda = \varphi(x,y) = 0 \end{cases}$ ⇨ $\begin{cases} x = ? \\ y = ? \end{cases}$

3. Model Selection

Terminologies



Note that **sample/instance(样本)** is without labels while **example(样例)** is with labels.

Tasks

Task

- By prediction target
  - Classification: discrete value
    - Binary: Good melon, Bad melon
    - Multiclass: Cucumber; Pumpkin; Watermelon
  - Regression: continuous value
    - Ripeness of melon
  - Clustering: no label
- By label
  - Supervised Learning： Classification, Regression
  - Unsupervised Learning： Clustering
  - Semi-supervised Learning： Combing the two

Accuracy & Error

- We need to evaluate our model
- Error rate
  - $E = \frac{the\ number\ of\ misclassified\ samples\ (a)}{the\ number\ of\ all\ samples\ (m)} = \frac{a}{m}$
- How about accuracy?

$$A = 1 - E = 1 - \frac{a}{m}$$

Accuracy & Error

- Error 误差
  - Error is the difference between the output of a learner and the ground-truth of samples
- Training error/Empirical error 训练/经验误差
  - error on the training set
- Testing error 测试误差
  - error on the testing set
- Generalization error 泛化误差
  - error on all samples except training set

Overfitting

- Small loss on training data, large loss on testing data. Why?

An extreme example

Training data: $\{(\boldsymbol{x^1}, \hat{y}^1), (\boldsymbol{x^2}, \hat{y}^2), \ldots, (\boldsymbol{x^N}, \hat{y}^N)\}$

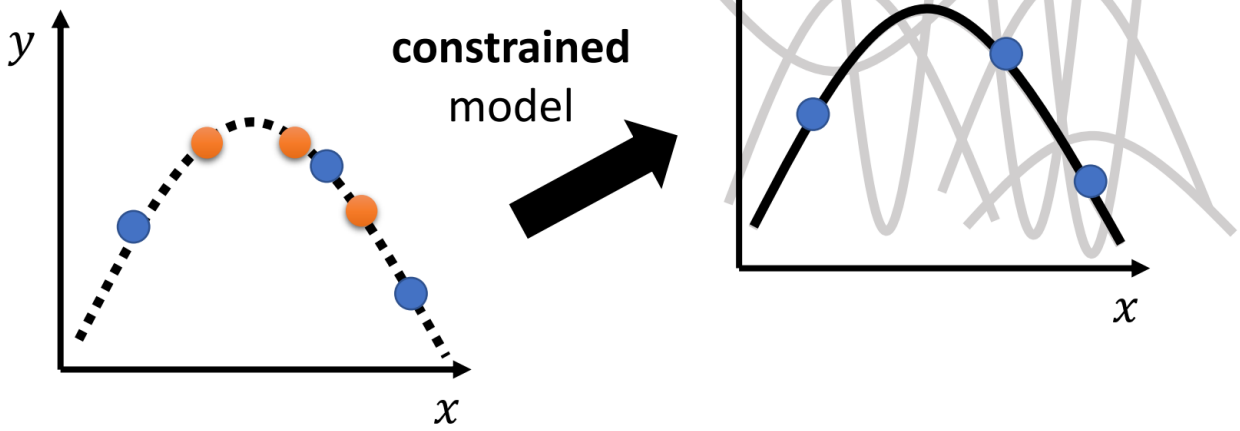
$$f(\boldsymbol{x}) = \begin{cases} \hat{y}^i & \exists \boldsymbol{x^i} = \boldsymbol{x} \\ random & otherwise \end{cases}$$

Less than useless ...

solutions:

1. more training data
2. constrained model:

Overfitting



Some Solutions

- Less parameters, sharing parameters
- Less features
- Early stopping
- Regularization
- Dropout

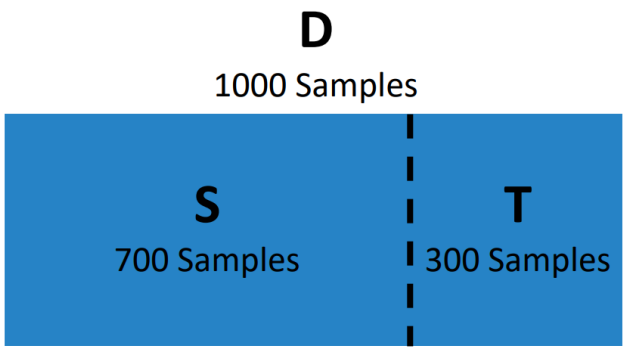
no excessive constraint!

General Performance Evaluation

1. Hold-out

Hold-out 留出法

- Data set D is divided into two mutually exclusive sets
  - a training set S, and a testing set T
- $D=S \cup T$, $S \cap T=\emptyset$
- Example: a binary classification problem



If

training on S, testing on T,
90 misclassification samples

Then

Error rate: (90/300)*100% = 30%
Accuracy: 100% – Error rate = 70%

using **layer sampling(分层采样)** to ensure the consistency of data distribution

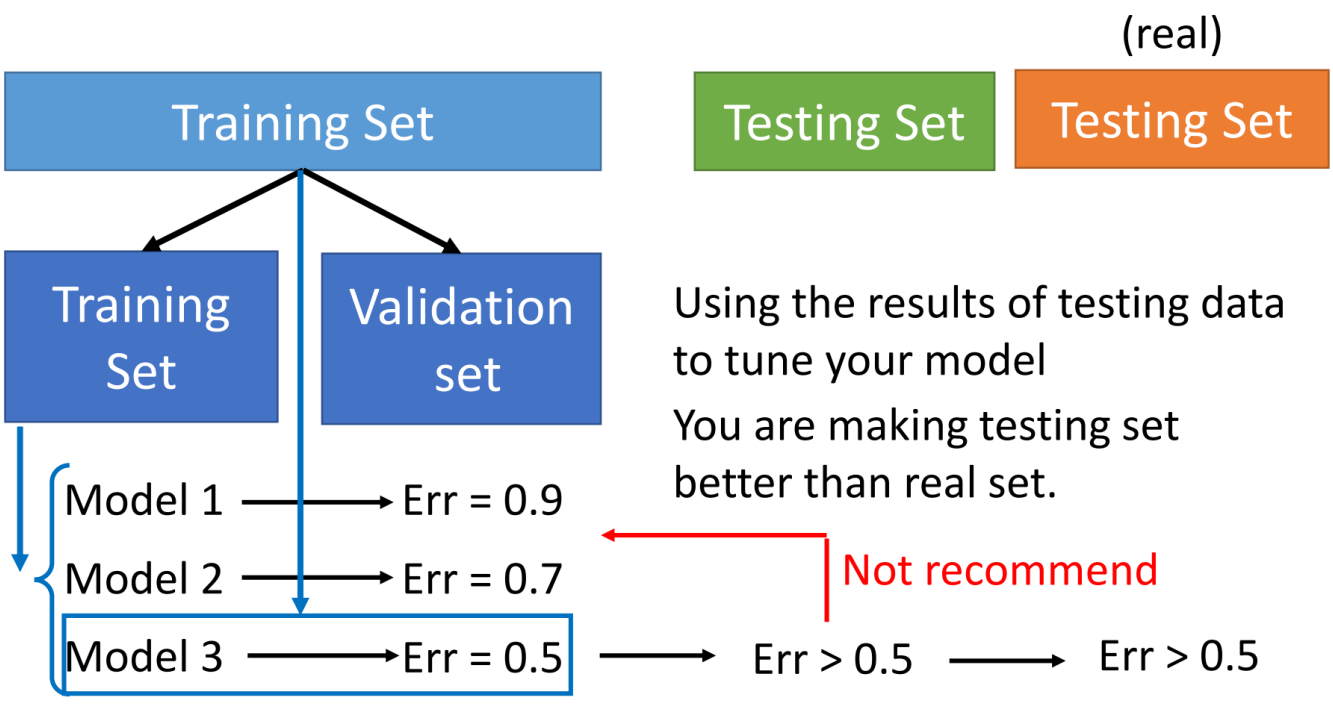
Weakness

Hold-out 留出法

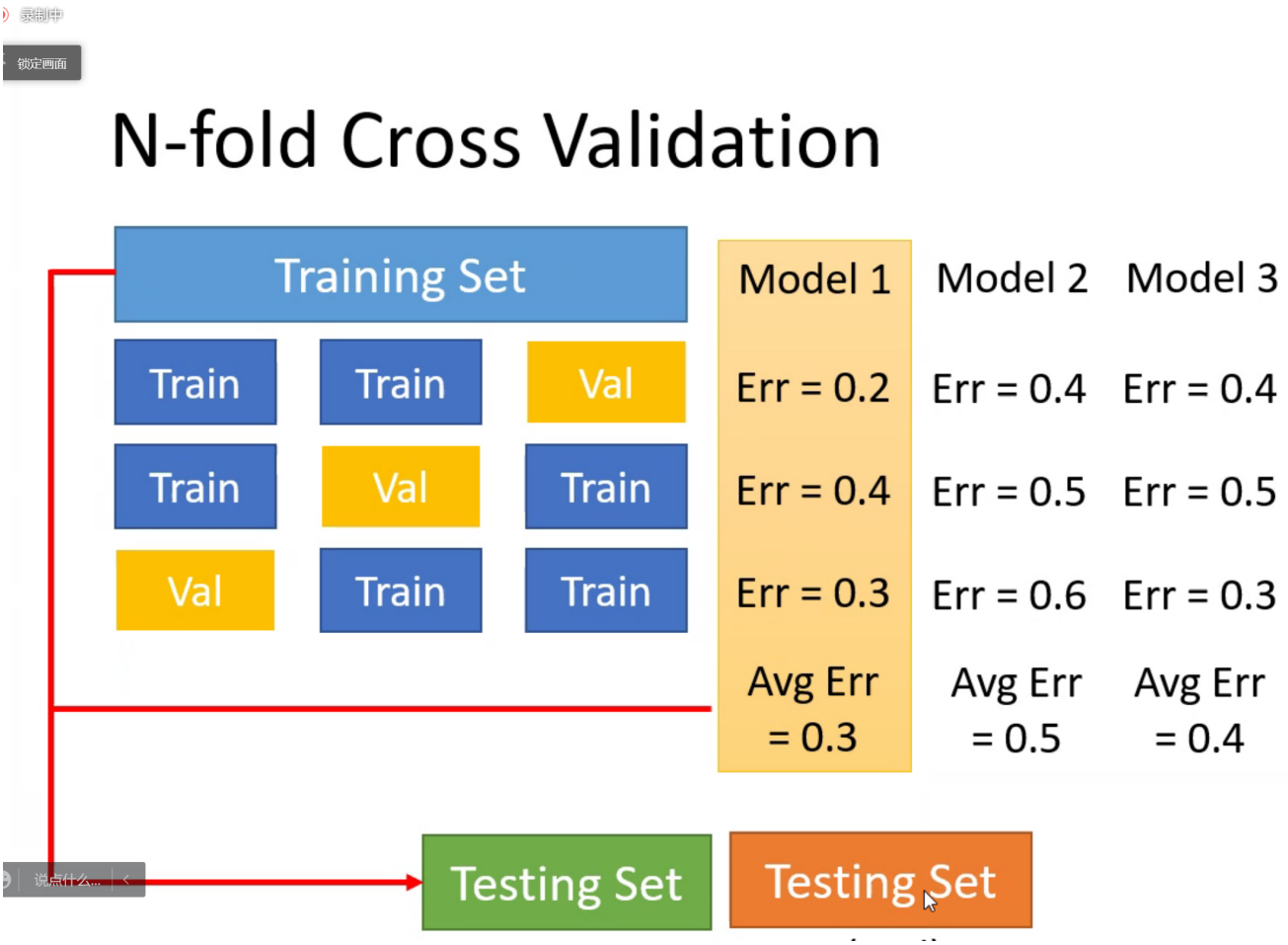
- **Weakness:** We would like to get a model trained with D, but hold-out cannot do this due to the need to divide the dataset
  - S↑,T↓, evaluation results may be unstable
  - S↓,T↑, the model has a large deviation from the model trained by D
- There is no perfect solution.
- Typically about **2/3** - **4/5** of the data is used for training.

2. Cross Validation

Cross Validation 交叉验证法



3. n-fold validation



advantage:

N-fold Cross Validation

- We have used all data for training, and all data for testing, and used each data point the same number of times
- Cross-validation returns an unbiased estimate of the generalization error and its variance

- The value of N is important!
  - What if N=the number of samples (m)?
- **Leave-One-Out** 留一法

4. bootstrapping

Bootstrapping 自助法

- Training set D -> D' (pick a sample from D m times)
- The probability of a sample a not picked is $(1\text{-}1/m)^m$
  - $\lim_{m\to\infty}\left(1-\frac{1}{m}\right)^m=\frac{1}{e}\approx 0.368$
  - D': training set
  - $D \setminus D'$: test set
- Advantages
  - It's useful when dataset is small and training and test set are hard to construct
- Disadvantages
  - The training set D' has different distribution of D, which may introduce bias in evaluation

parameter tuning

Parameter Tuning

- It is *impossible* to exhaust all parameters. We need to select a range and change step for each parameter.
- Example
  - range->[0,0.2] step->0.05.
  - Then we need to evaluate 5 parameters.

- The final model should be trained on dataset D (using all samples) before it can be submitted to users.

performance measure

1. regression:

MSE:

- **Mean Square Error, MSE**
- **均方误差**

$$E(f;D)=\frac{1}{m}\sum_{i=1}^{m}\left(f\left(\boldsymbol{x}_i\right)-y_i\right)^2$$

2. classification:

Error & Accuracy

Measure for Classification

- **Error rate & Accuracy**

Error rate

$$E(f;D)=\frac{1}{m}\sum_{i=1}^{m}\mathbb{I}\left(f\left(\boldsymbol{x}_i\right)\neq y_i\right)$$

Accuracy

$$\begin{aligned}\mathrm{acc}(f;D) &= \frac{1}{m}\sum_{i=1}^{m}\mathbb{I}\left(f\left(\boldsymbol{x}_i\right)=y_i\right)\\ &= 1-E(f;D)\ .\end{aligned}$$

Precision & Recall

Precision and Recall

- Suitable for information retrieval, web search scenarios

A **"confusion matrix"** can be obtained by combining the statistics of the true labels and the predicted results

Ground truth	Predicted Result	
	True	False
True	TP	FN
False	FP	TN

Precision $P = \frac{TP}{TP + FP}$

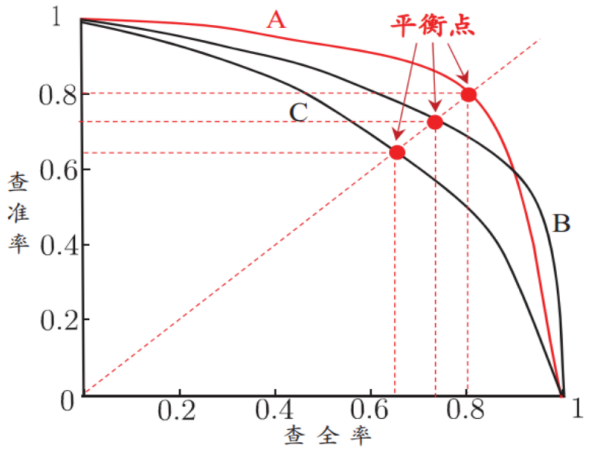
Recall $R = \frac{TP}{TP + FN}$

Recall and precision are contradictory measures. **Why?**

the 2 values are contradictory:

Precision and Recall

- According to the prediction results of the learner, the samples are sorted according to the probability of positive examples. If all samples are predicted as positive examples one by one, then the precision-recall curve, referred to as "P-R curve", can be obtained.



P-R曲线与平衡点示意图

Break-even point

The value of **"precision = recall"** on the curve

Used to measure the performance of classifiers with crossed P-R curves.

Comprehensive consideration of recall and precision

We want P and R both high.

F1 Score

F1 Score

- BEP is too simplistic
- More commonly used than P-R curve break-even point

$$F1 = \frac{2 * P * R}{P + R} = \frac{2 * TP}{the\ number\ of\ samples + TP - TN}$$

- A more general form F_β

$$F_\beta = \frac{(1 + \beta^2) * P * R}{(\beta^2 * P) + R}$$

$\beta > 0$ measures the relative importance of recall to precision

$\beta = 1$: F1 Score
$\beta > 1$: Recall is more important (Information retrieval)
$\beta < 1$: Precision is more important (Recommender system)

F1 score is essentially harmonic mean

F-beta score is essentially weighted harmonic mean

ROC

ROC

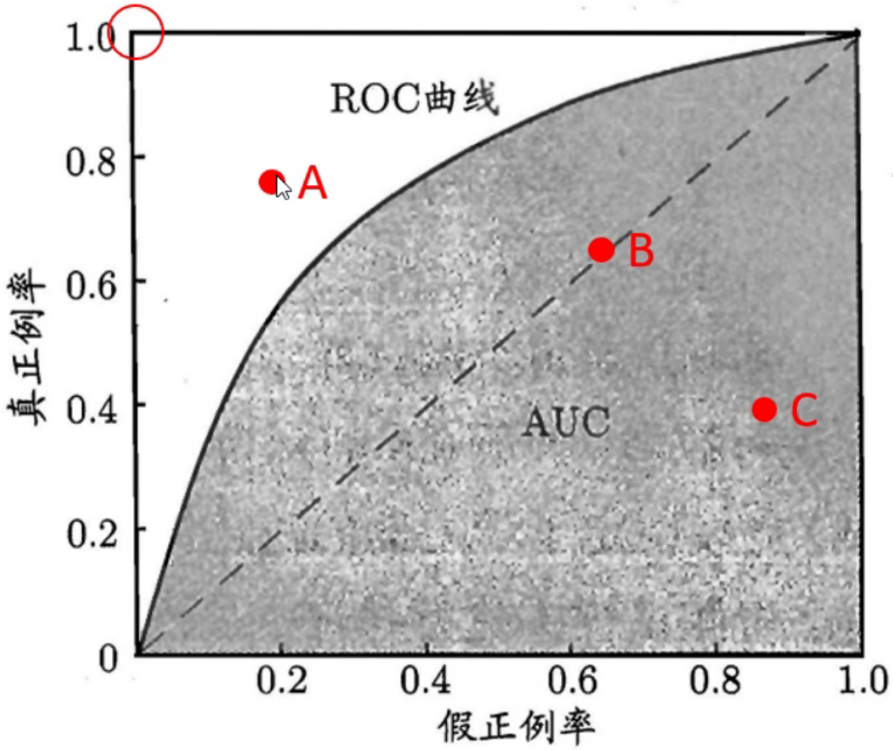
- Receiver Operating Characteristic(受试者工作特征)
- The steps of sorting the samples and predicting the samples as positive examples are consistent with the PR curve
- But the axes change

	Name	Expression
Vertical axis	True Positive Rate/TPR	$\frac{TP}{TP+FN}$
Horizontal axis	False Positive Rate/FPR	$\frac{FP}{TN+FP}$

Ground truth	Predicted Result	
	True	False
True	TP	FN
False	FP	TN

We want TPR high while FPR low

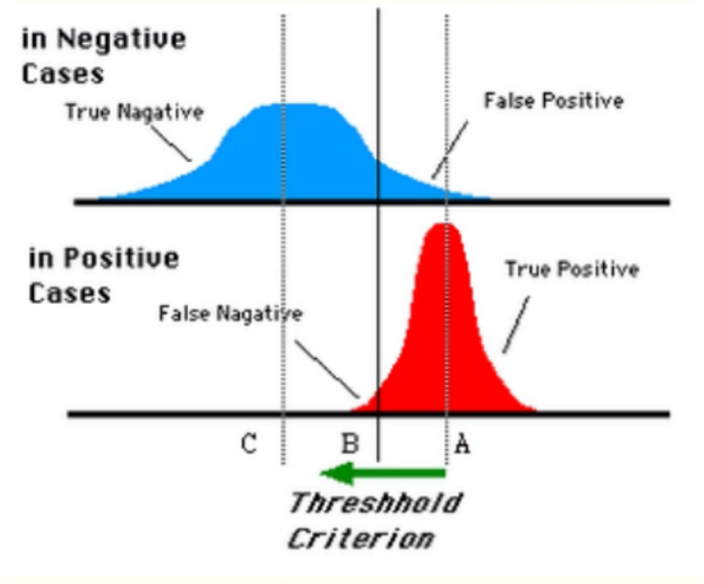
ROC

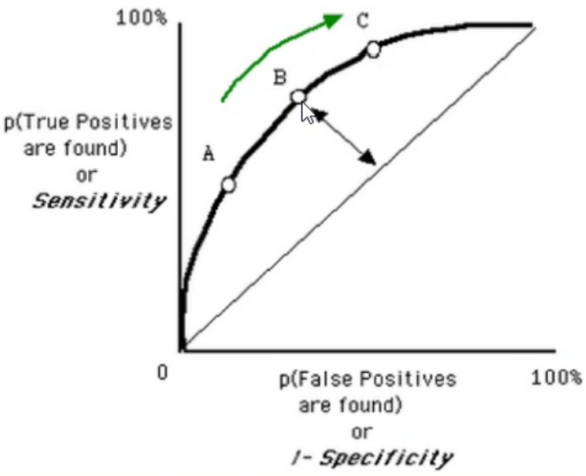


the variant of the ROC curve is the **threshold**:

ROC

- ROC represents the performance of a classifier under different thresholds





the variant of PR curve is **R**(How to artificially control it?)

Cost-sensitive Error Rate

Cost-sensitive Error Rate
代价敏感错误率

- In order to weigh the different losses caused by different types of errors, ***unequal costs*** can be assigned to errors.
- Cost Matrix: $cost_{ij}$ represents the cost of predicting the i-th class sample as the j-th class sample

真实类别	预测类别	
	第 0 类	第 1 类
第 0 类	0	$cost_{01}$
第 1 类	$cost_{10}$	0

- Cost-sensitive Error Rate: Minimize the total cost

$$E(f;D;cost)=\frac{1}{m}\left(\sum_{\boldsymbol{x}_i\in D^+}\mathbb{I}(f(\boldsymbol{x}_i)\neq y_i)\times cost_{01}+\sum_{\boldsymbol{x}_i\in D^-}\mathbb{I}(f(\boldsymbol{x}_i)\neq y_i)\times cost_{10}\right).$$

---

bias and variance

bias

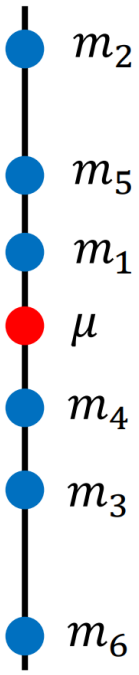
Bias and Variance of Estimator

- Estimate the mean of a variable x
  - assume the mean of x is μ
  - assume the variance of x is σ^2
- Estimator of mean μ
  - Sample N points: $\{x^1, x^2, \ldots, x^N\}$

$$m = \frac{1}{N}\sum_n x^n \neq \mu$$

$$E[m] = E\left[\frac{1}{N}\sum_n x^n\right] = \frac{1}{N}\sum_n E[x^n] = \mu$$

unbiased



variance

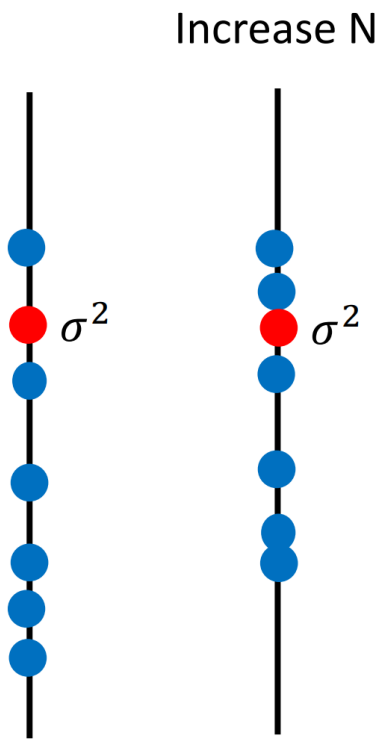
Bias and Variance of Estimator

- Estimate the mean of a variable x
  - assume the mean of x is μ
  - assume the variance of x is σ^2
- Estimator of variance σ^2
  - Sample N points: $\{x^1, x^2, \ldots, x^N\}$

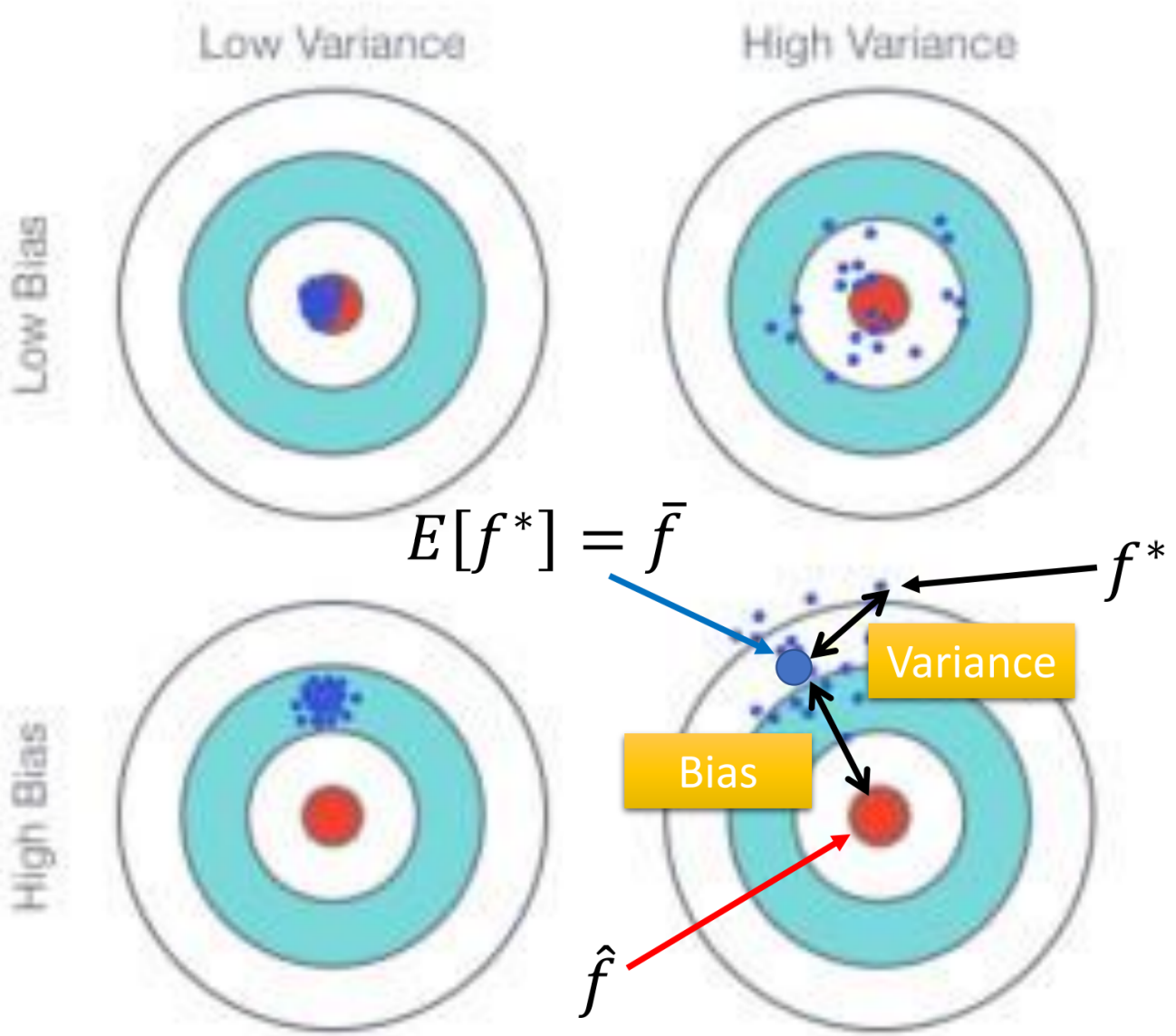
$$m = \frac{1}{N}\sum_n x^n \quad s^2 = \frac{1}{N}\sum_n (x^n - m)^2$$

Biased estimator

$$E[s^2] = \frac{N-1}{N}\sigma^2 \neq \sigma^2$$



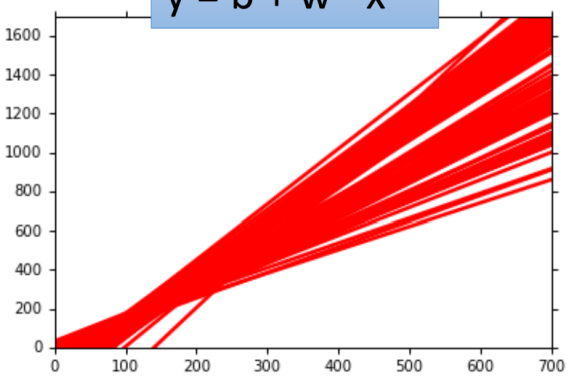
"Visualization" of bias and variance



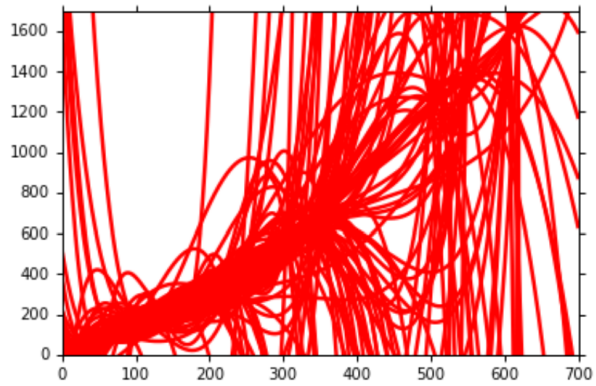
Variance & Bias in many models:

Variance

$y = b + w \cdot x$



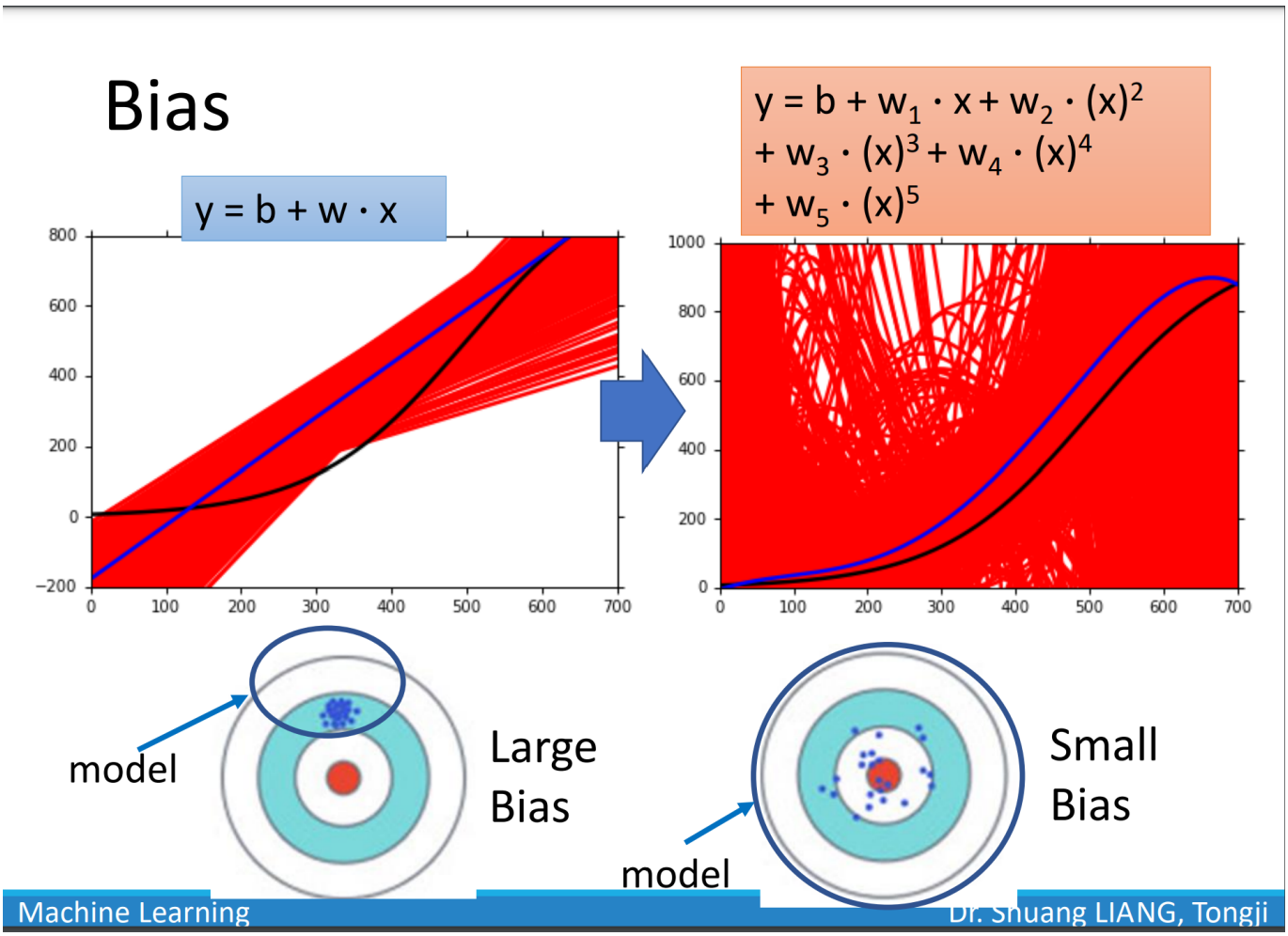
$y = b + w_1 \cdot x + w_2 \cdot (x)^2 + w_3 \cdot (x)^3 + w_4 \cdot (x)^4 + w_5 \cdot (x)^5$



Small Variance

Large Variance

Simpler model is less influenced by the sampled data

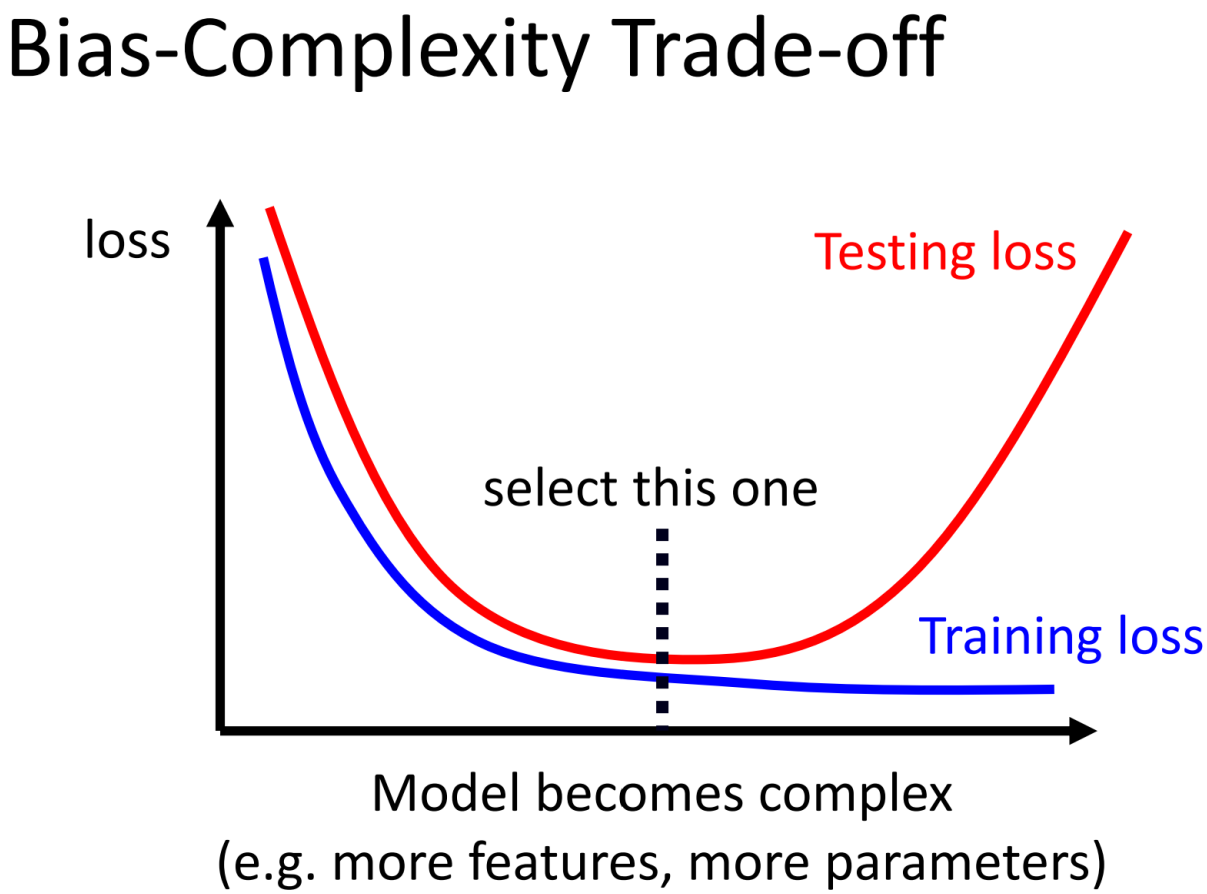


What to do with bias & variance?

Debugging a learning algorithm:
Suppose you have implemented regularized linear regression to predict housing prices. However, when you test your hypothesis in a new set of houses, you find that it makes unacceptably large errors in its prediction. What should you try next?

- Get more training examples → fixes high variance
- Try smaller sets of features → fixes high variance
- Try getting additional features → fixes high bias
- Try adding polynomial features$(x_1^2, x_2^2, x_1x_2, \text{etc})$ → fixes high bias.
- Try decreasing λ → fixes high bias
- Try increasing λ → fixes high variance

tradeoff between bias & variance:



4. Linear Regression & GD

linear regression

Step1: select function model

录制中

锁定画面

Step1: Function with Unknown Parameters

$y = f($)

时间	播放量
2021-10-21	12442
2021-10-22	12465
2021-10-23	12478
2021-10-24	12499
2021-10-25	12547
2021-10-26	12584
2021-10-27	12610
2021-10-28	12648
2021-10-29	12664
2021-10-30	12692
2021-10-31	12716
2021-11-01	12740
2021-11-02	12769
2021-11-03	12790
2021-11-04	12808
2021-11-05	12825
2021-11-06	12863

Model $y = b + wx_1$ based on domain knowledge

feature

说点什么...

y: no. of views on 11/18, x_1: no. of views on 11/17

w and b are unknown parameters (learned from data)

weight **bias**

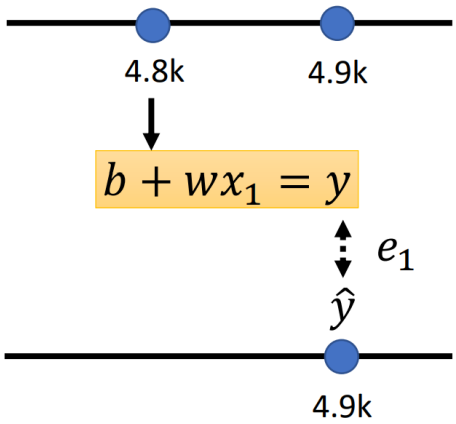
Step 2: define the loss function

- MAE
- MSE

MAE is what I haven't seen but more intuitive

Step2: Define Loss from Training Data

- Loss is a function of parameters $L(b, w)$
- Loss: how good a set of values is.



Loss: $L = \frac{1}{N}\sum_n e_n$

$e = |y - \hat{y}|$ L is mean absolute error (**MAE**)

$e = (y - \hat{y})^2$ L is mean square error (**MSE**)

If y and $\hat{y}$ are both probability distributions ➡ Cross-entropy

the contour plot of the loss function

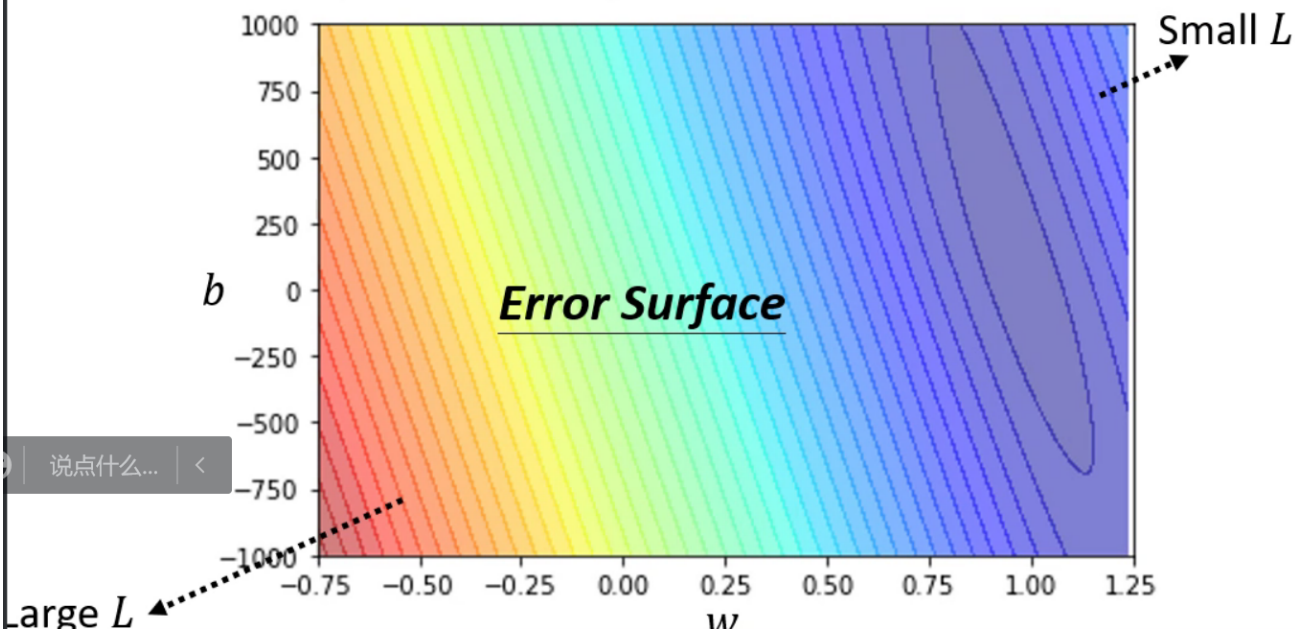
录制中

锁定画面

Step2: Define Loss from Training Data

- Loss is a function of parameters $L(b, w)$
- Loss: how good a set of values is.

Model $y = b + wx_1$

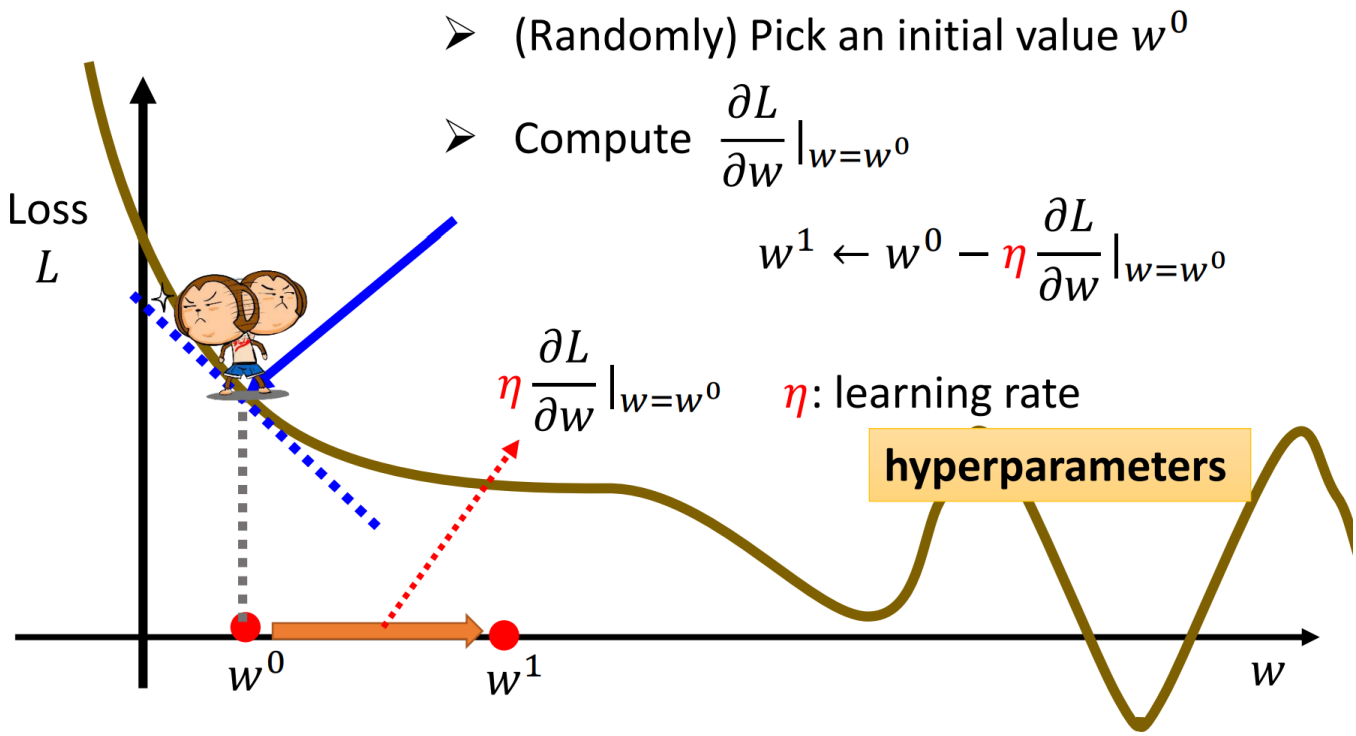


说点什么...

gradient descent

Gradient Descent

$w^* = arg \min_w L$



for more parameters(w and b):

Gradient Descent

$$w^*, b^* = arg \min_{w,b} L$$

正在讲话:

- (Randomly) Pick initial values w^0, b^0
- Compute

$$\frac{\partial L}{\partial w}\Big|_{w=w^0,b=b^0} \qquad w^1 \leftarrow w^0 - \eta \frac{\partial L}{\partial w}\Big|_{w=w^0,b=b^0}$$

$$\frac{\partial L}{\partial b}\Big|_{w=w^0,b=b^0} \qquad b^1 \leftarrow b^0 - \eta \frac{\partial L}{\partial b}\Big|_{w=w^0,b=b^0}$$

Can be done in one line in most deep learning frameworks

- Update w and b interatively

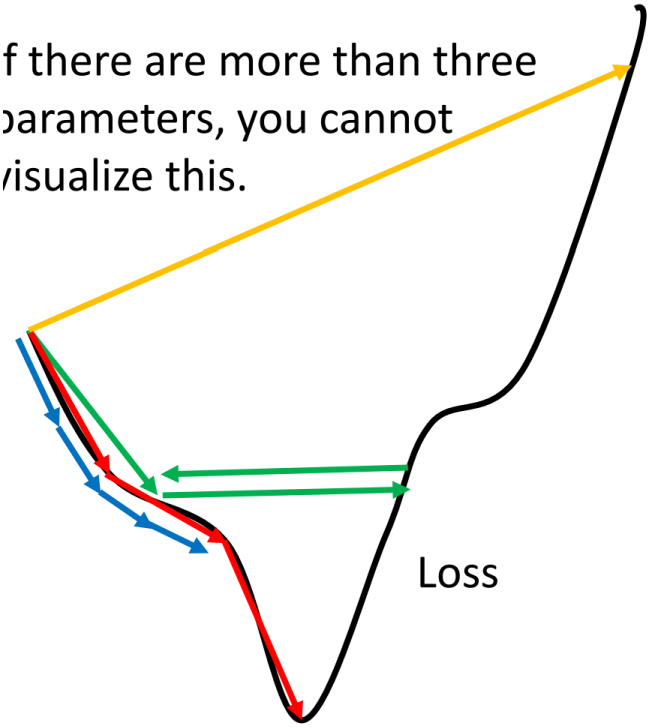
learning rate:

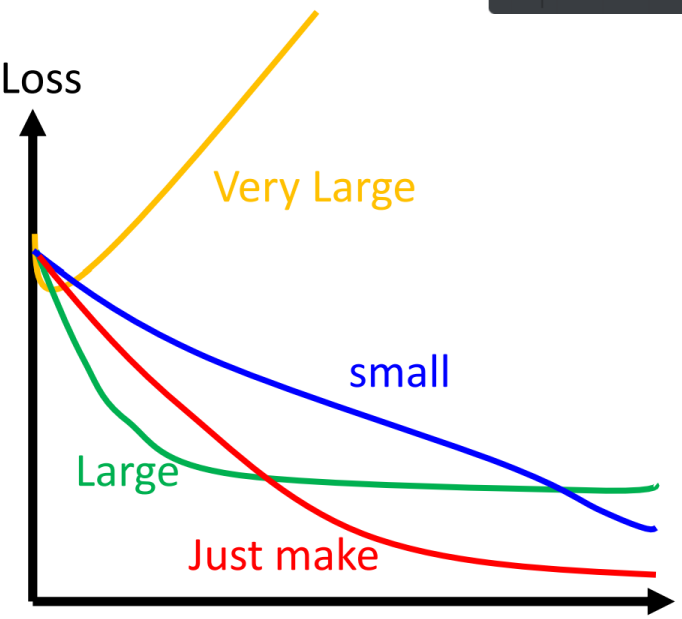
Learning Rate

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

Set the learning rate η carefully

f there are more than three
parameters, you cannot
visualize this.





But you can always visualize this.

adaptive LR

Adaptive LR

Adagrad

$$w^{t+1} \leftarrow w^{t} - \frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^{i})^{2}}} g^{t}$$

- Popular & Simple Idea: Reduce the learning rate by some factor every few epochs.
  - At the beginning, we are far from the destination, so we use larger learning rate
  - After several epochs, we are close to the destination, so we reduce the learning rate
  - E.g. 1/t decay: $\eta^{t} = \eta/\sqrt{t+1}$
- Learning rate cannot be one-size-fits-all
  - Giving different parameters different learning rates

SGD

Stochastic Gradient Descent (SGD)

$$L = \sum_n \left(\hat{y}^n - \left(b + \sum w_i x_i^n \right) \right)^2$$

Loss is the summation over all training examples

◆ ***Gradient Descent*** $\theta^i = \theta^{i-1} - \eta \nabla L\left(\theta^{i-1}\right)$

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

Pick an example x^n

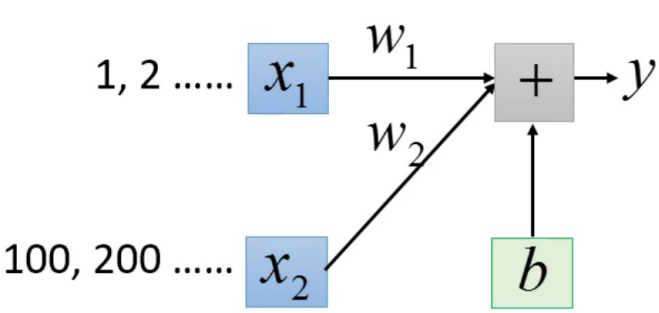
Loss for only one example

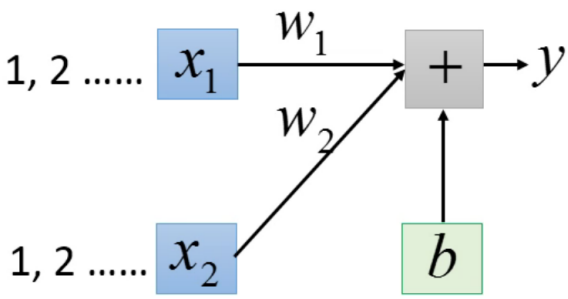
$$L^n = \left(\hat{y}^n - \left(b + \sum w_i x_i^n \right) \right)^2 \qquad \theta^i = \theta^{i-1} - \eta \nabla L^n\left(\theta^{i-1}\right)$$

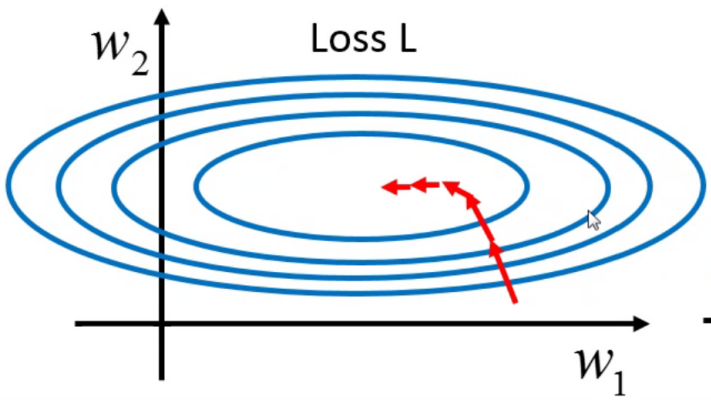
Feature Scaling

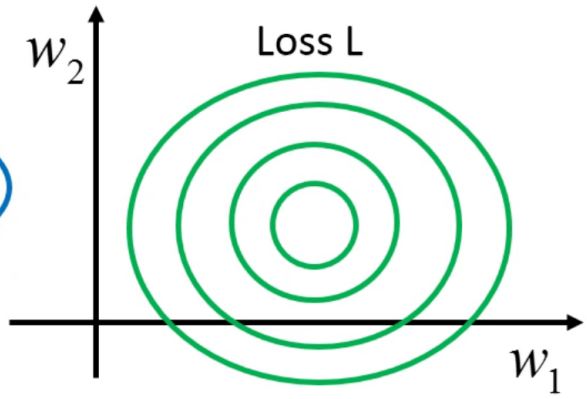
Feature Scaling

$$y = b + w_1 x_1 + w_2 x_2$$

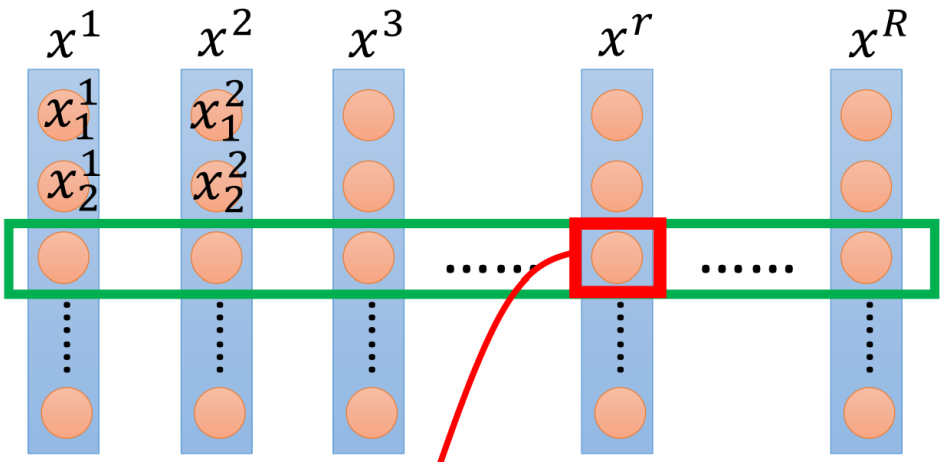








Feature Scaling



For each dimension i:

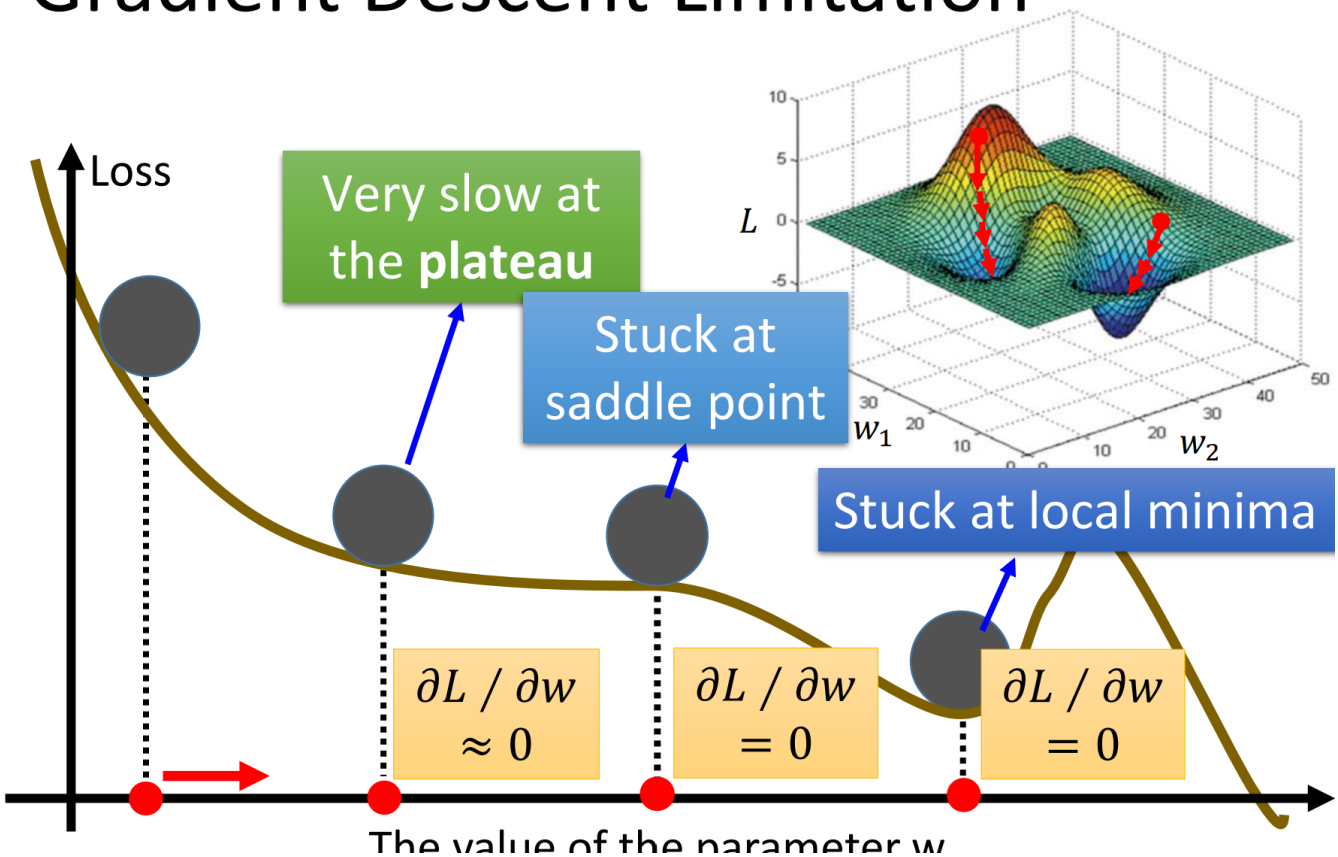
mean: m_i

standard deviation: σ_i

$$x_i^r \leftarrow \frac{x_i^r - m_i}{\sigma_i}$$

The means of all dimensions are 0, and the variances are all 1

Limitations



for 1D function we can compute **the 2-degree derivative** to determine whether it is saddle point

regularization

Ridge Regression

(2-norm regularization)

Ridge Regression

- **Ridge Regression with L2 Regularization**

$$\Omega(w) = \lambda \|w\|_2^2$$

$$\textit{where } \|w\|_2^2 = \Sigma_i w_i^2$$

Here the main effect is that large model weights w_i will be **penalized** (avoided), since we consider them "unlikely", while small ones are ok.

When $L(w)$ is **MSE,** this is called **ridge regression**:

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \sum_{n=1}^{N} \left[y_n - \mathbf{x}_n^\top \mathbf{w} \right]^2 + \lambda \|\mathbf{w}\|_2^2$$

LASSO Regression

(1-norm regularization)

LASSO Regression

- **LASSO Regression with L1 Regularization**

$$\Omega(w) = \lambda \|w\|$$

$$\textit{where } \|w\|_1 = \Sigma_i |w_i|$$

For the L1-regularization the optimum solution is likely going to be **sparse** (only has few non-zero components) compared to the case where we use L2-regularization.

When $L(w)$ is **MSE,** this is called **LASSO regression**:

$$\min_{\mathbf{w}} \quad \frac{1}{2N} \sum_{n=1}^{N} [y_n - \mathbf{x}_n^\top \mathbf{w}]^2 \; + \; \lambda \, \|\mathbf{w}\|_1$$

what L1 and L2 regularization implies:

L1 VS L2

- **Ridge Regression** tends to distribute weights evenly among related features

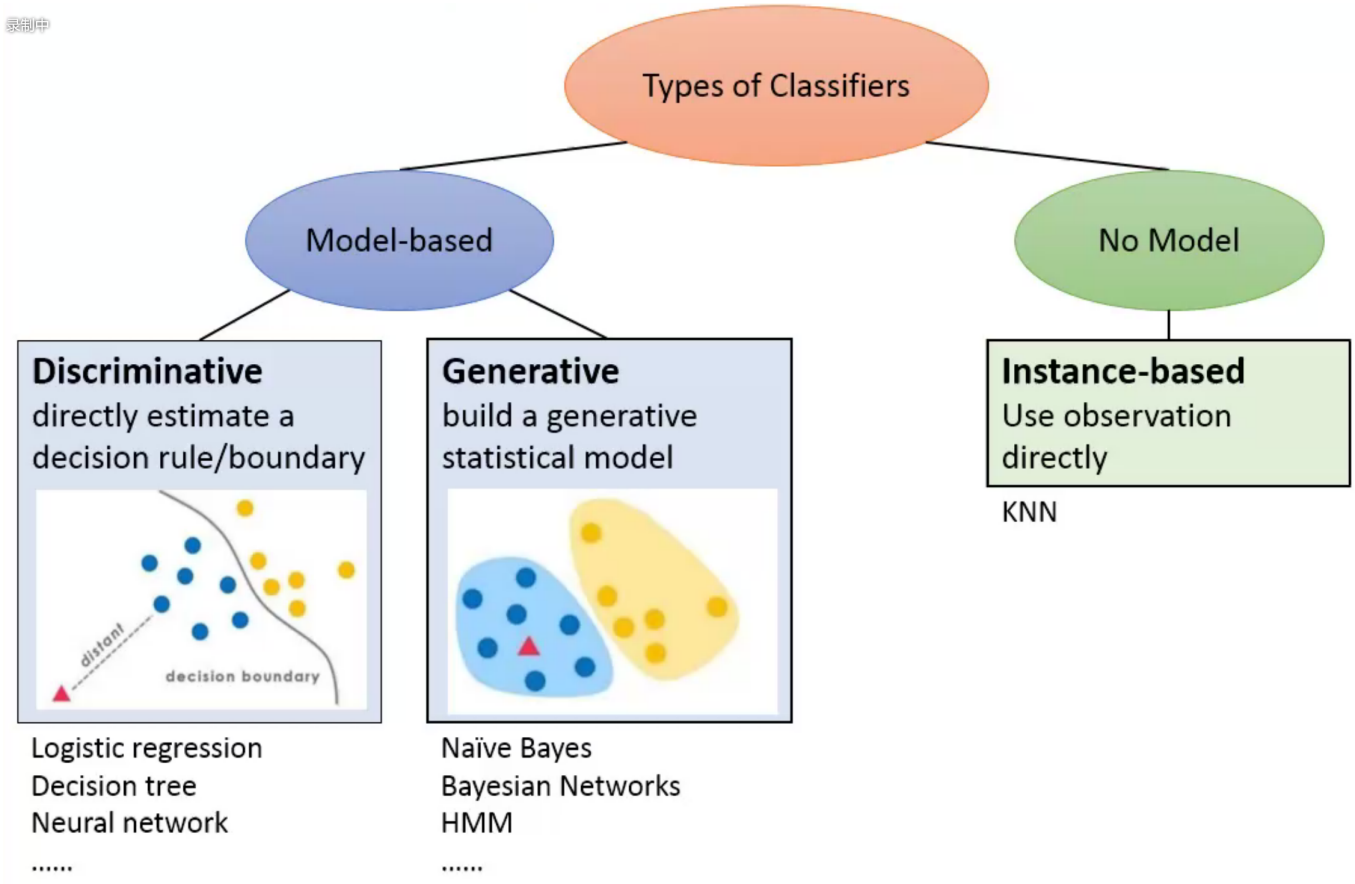
- **Lasso Regression** tends to select one from the relevant features, and the rest of the feature weights decay to zero (*Feature Selection*)

L2: **evenly distribute** weights among features

L1: **select some of** the features

5. Logistic Regression

type of classifiers



Discriminative:

conditional probabilities;

doesn't care about how the data is distributed(easier)

Generative

union probabilities

cares about how the data is distributed(harder)

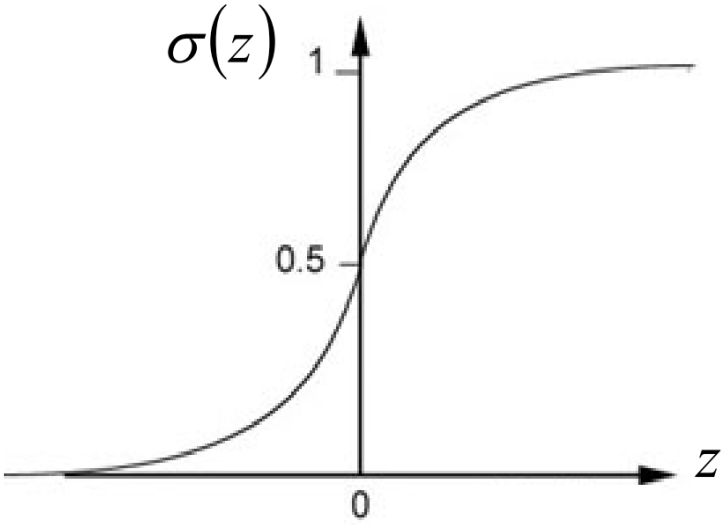
At the same scale, **discriminative approach performs better than generative model**, as at most of the time, it is very difficult to solve the distribution of the data

logistic regression

1. mapping $[-\infty, +\infty]$ to $[0, 1]$

using sigmoid function:

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



to calculate two formulas:

- $1-\sigma(z)$
- $\sigma'(z)$

properties:

- $1-\sigma(z)=\frac{1+e^{-z}-1}{1+e^{-z}}=(1+e^{z})^{-1}=\sigma(-z)$
- $\sigma'(z)=-\frac{-e^{-z}}{(1+e^{-z})^{2}}=\frac{1}{(1+e^{-z})}\frac{1}{(1+e^{z})}=\sigma(z)(1-\sigma(z))$

2. loss function

Step2: Goodness of a function

Training Data

x^1	x^2	x^3	……	x^N
C_1	C_1	C_2		C_1

Assume the data is generated based on $f_{w,b}(x) = P_{w,b}(C_1|x)$

Given a set of w and b, what is its probability of generating the data?

$$L(w,b) = f_{w,b}(x^1)f_{w,b}(x^2)\left(1 - f_{w,b}(x^3)\right)\cdots f_{w,b}(x^N)$$

where $L(w,b)$ is the likelihood of all the data in set C_1

We want to maximize $L(w,b)$

after some mathematical transformation:

1. to maximize $L(w,b)$ is equivalent to minimize $-\ln L(w,b)$ as logarithm converts multiplication to add, making the algorithm faster
2. We apply a trick below to make derivation easier as there are just 2 possible prediction values in bi-classification problem:

$$-lnL(w,b)$$

$$= -lnf_{w,b}(x^1) \Rightarrow -[\,1\; lnf(x^1) + 0\; ln(1-f(x^1))]$$

$$-lnf_{w,b}(x^2) \Rightarrow -[\,1\; lnf(x^2) + 0\; ln(1-f(x^2))]$$

$$-ln\left(1 - f_{w,b}(x^3)\right) \Rightarrow -[\,0\; lnf(x^3) + 1\; ln(1-f(x^3))]$$

$$\vdots$$

3. we get the **cross entropy function** to be minimized:

$$L(w,b) = f_{w,b}(x^1) f_{w,b}(x^2) \left(1 - f_{w,b}(x^3)\right) \cdots f_{w,b}(x^N)$$

$$-lnL(w,b) = -\left[lnf_{w,b}(x^1) + lnf_{w,b}(x^2) + ln\left(1 - f_{w,b}(x^3)\right)\right] \cdots$$

$\hat{y}^n$: 1 for class 1, 0 for class 2

$$= \sum_n -\left[\hat{y}^n lnf_{w,b}(x^n) + (1 - \hat{y}^n) ln\left(1 - f_{w,b}(x^n)\right)\right]$$

Cross entropy between two Bernoulli distribution

$$H(p,q) = -\sum_x p(x) ln\left(q(x)\right)$$

Distribution p:

$p(x = 1) = \hat{y}^n$

$p(x = 0) = 1 - \hat{y}^n$

cross entropy

Distribution q:

$q(x = 1) = f(x^n)$

$q(x = 0) = 1 - f(x^n)$

3. optimization(GD)

gradient of cross entropy*(to be filled)*:

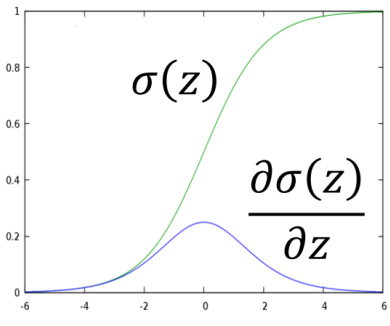
To apply GD for the cross entropy we have to compute the gradient: $\frac{\partial L}{\partial w}$

for the first part:

$$\frac{-lnL(w,b)}{\partial w_i} = \sum_n -\left[\hat{y}^n \frac{lnf_{w,b}(x^n)}{\partial w_i} + (1 - \hat{y}^n) \frac{ln\left(1 - f_{w,b}(x^n)\right)}{\partial w_i}\right]$$

$$\frac{\partial lnf_{w,b}(x)}{\partial w_i} = \frac{\partial lnf_{w,b}(x)}{\partial z} \frac{\partial z}{\partial w_i} \qquad \frac{\partial z}{\partial w_i} = x_i$$

$$\frac{\partial ln\sigma(z)}{\partial z} = \frac{1}{\sigma(z)} \frac{\partial \sigma(z)}{\partial z} = \frac{1}{\sigma(z)} \sigma(z)\left(1 - \sigma(z)\right)$$



---

for the second part:

$$\frac{-lnL(w,b)}{\partial w_i} = \sum_n -\left[\hat{y}^n \frac{lnf_{w,b}(x^n)}{\partial w_i} + (1-\hat{y}^n)\frac{ln\left(1-f_{w,b}(x^n)\right)}{\partial w_i}\right]$$

$$\frac{\partial ln\left(1-f_{w,b}(x)\right)}{\partial w_i} = \frac{\partial ln\left(1-f_{w,b}(x)\right)}{\partial z}\frac{\partial z}{\partial w_i} \qquad \frac{\partial z}{\partial w_i} = x_i$$

$$\frac{\partial ln\left(1-\sigma(z)\right)}{\partial z} = -\frac{1}{1-\sigma(z)}\frac{\partial \sigma(z)}{\partial z} = -\frac{1}{1-\sigma(z)}\sigma(z)\left(1-\sigma(z)\right)$$

putting them together:

- Loss: Cross-Entropy $\left(1-f_{w,b}(x^n)\right)x_i^n \qquad -f_{w,b}(x^n)x_i^n$

$$\frac{-lnL(w,b)}{\partial w_i} = \sum_n -\left[\hat{y}^n \frac{lnf_{w,b}(x^n)}{\partial w_i} + (1-\hat{y}^n)\frac{ln\left(1-f_{w,b}(x^n)\right)}{\partial w_i}\right]$$

$$= \sum_n -\left[\hat{y}^n \left(1-f_{w,b}(x^n)\right)x_i^n - (1-\hat{y}^n)f_{w,b}(x^n)x_i^n\right]$$

$$= \sum_n -\left[\hat{y}^n - \hat{y}^n f_{w,b}(x^n) - f_{w,b}(x^n) + \hat{y}^n f_{w,b}(x^n)\right]x_i^n$$

$$= \sum_n -\left(\hat{y}^n - f_{w,b}(x^n)\right)x_i^n$$

Larger difference, larger update

$$w_i \leftarrow w_i - \eta \sum_n -\left(\hat{y}^n - f_{w,b}(x^n)\right)x_i^n$$

The same form as Linear Regression!

4. why not MSE?

Step3: Find the best function

- Loss: Square Error

Step 1: $f_{w,b}(x) = \sigma\left(\sum_i w_i x_i + b\right)$

Step 2: Training data: $(x^n, \hat{y}^n)$, $\hat{y}^n$: 1 for class 1, 0 for class 2

$$L(f) = \frac{1}{2}\sum_n \left(f_{w,b}(x^n) - \hat{y}^n\right)^2$$

Step 3:

$$\frac{\partial\, (f_{w,b}(x)-\hat{y})^2}{\partial w_i} = 2\left(f_{w,b}(x) - \hat{y}\right) \frac{\partial f_{w,b}(x)}{\partial z} \frac{\partial z}{\partial w_i}$$

$$= 2\left(f_{w,b}(x) - \hat{y}\right) f_{w,b}(x)\left(1 - f_{w,b}(x)\right) x_i$$

$\hat{y}^n = 1$ If $f_{w,b}(x^n) = 1$ (close to target) ➡ $\partial L/\partial w_i = 0$

If $f_{w,b}(x^n) = 0$ (far from target) ➡ $\partial L/\partial w_i = 0$!!!

---

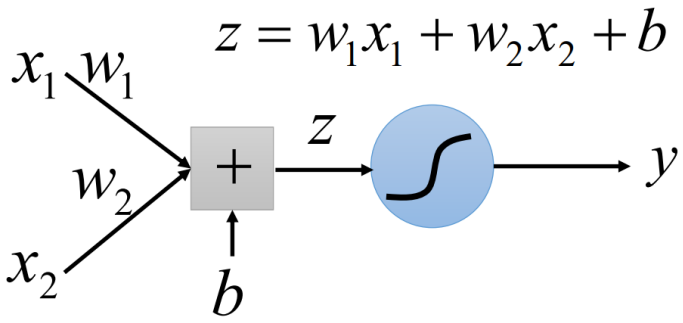
if the estimation is **far away from the target, the gradient of MSE is still 0**, which is we don't want to see.

Logistic regression vs. linear regression

	Logistic Regression	***Linear Regression***
Step 1:	$f_{w,b}(x) = \sigma\left(\sum_i w_i x_i + b\right)$ Output: between 0 and 1	$f_{w,b}(x) = \sum_i w_i x_i + b$ Output: any value
Step 2:	Training data: $(x^n, \hat{y}^n)$ $\hat{y}^n$: 1 for class 1, 0 for class 2 $L(f) = \sum_n l(f(x^n), \hat{y}^n)$	Training data: $(x^n, \hat{y}^n)$ $\hat{y}^n$: a real number $L(f) = \frac{1}{2}\sum_n (f(x^n) - \hat{y}^n)^2$
Step 3:	Logistic regression: $w_i \leftarrow w_i - \eta \sum_n -\left(\hat{y}^n - f_{w,b}(x^n)\right) x_i^n$ Linear regression: $w_i \leftarrow w_i - \eta \sum_n -\left(\hat{y}^n - f_{w,b}(x^n)\right) x_i^n$	

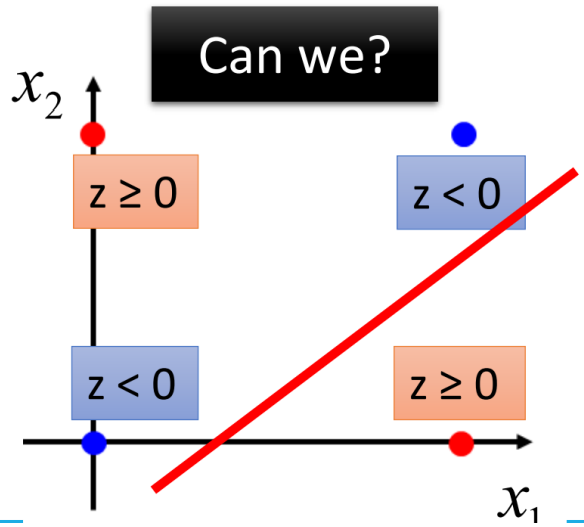
limitations

1. cannot solve XOR problem(leave it for NN)

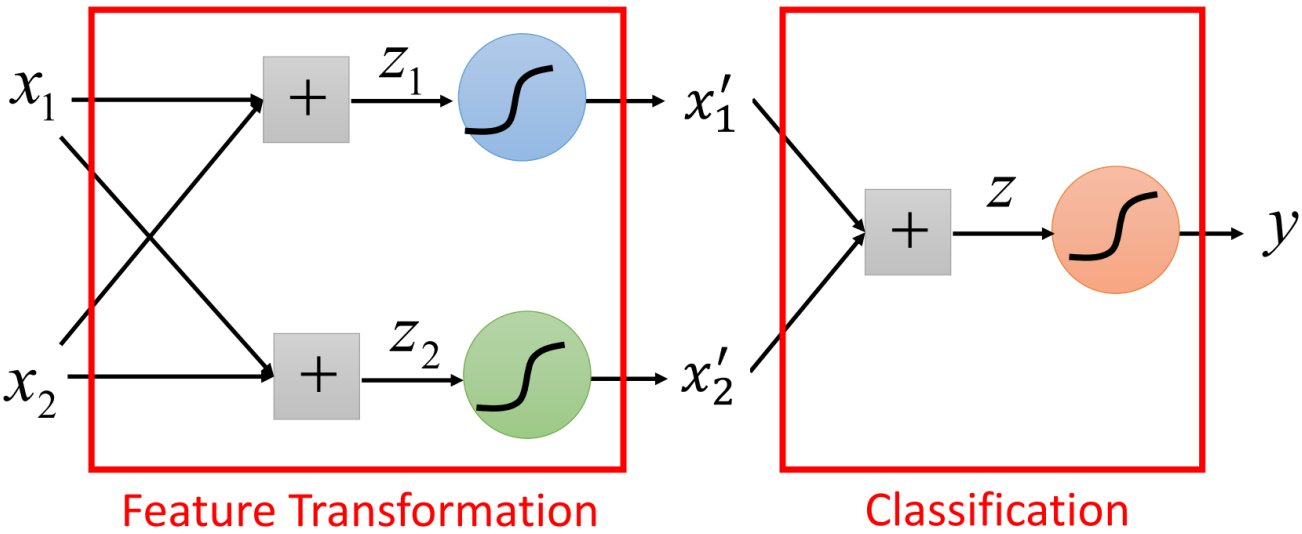


$$\begin{cases} Class1 & y \geq 0.5 \quad (z \geq 0) \\ Class2 & y < 0.5 \quad (z < 0) \end{cases}$$

Input Feature		Label
x_1	x_2	
0	0	Class 2
0	1	Class 1
1	0	Class 1
1	1	Class 2



motivation of NN: cascading sigmoid functions to solve XOR problem

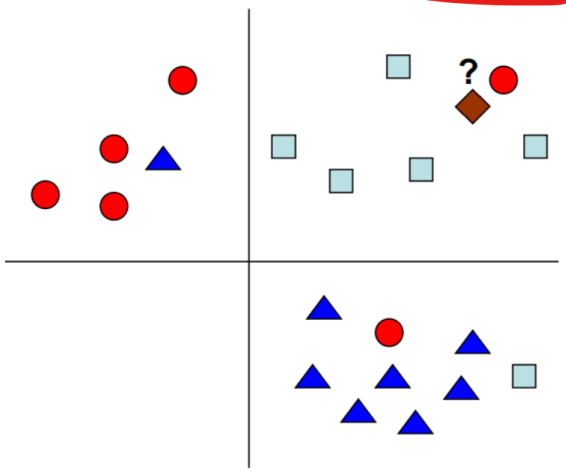


6. KNN

K nearest neighbors definition

KNN

- A simple, yet surprisingly efficient algorithm
- Requires the definition of a **distance function** or similarity measures between samples
- Select the class based on the **majority vote** in the k closest points



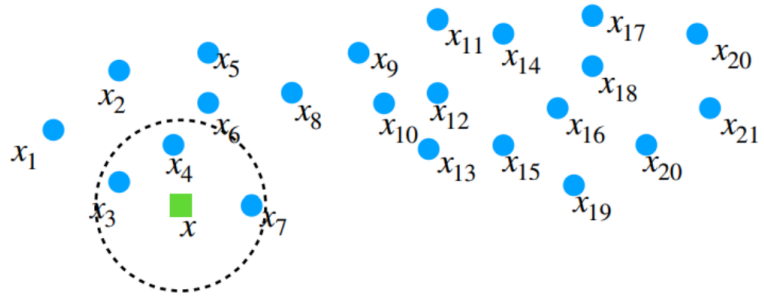
steps

1. find K nearest neighbors

Step1: Find nearest neighbors

$$nbh_{S_{train},k}: \mathcal{X} \to \mathcal{X}^k$$

$$x \longmapsto \{k \text{ elements of } S_{train} \text{ which are the closest to } x\}$$



How to define the distance?

S_{train}
Testing point

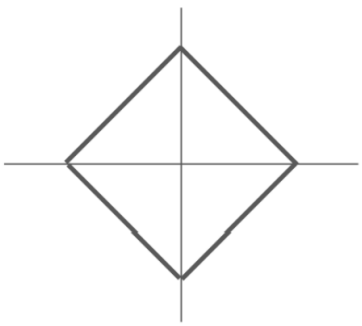
$$nbh_{S_{train},3}(x) = \{x_3, x_4, x_7\}$$

definition of the distance

- **Distance Metric**

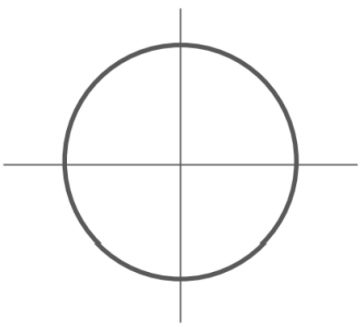
L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$



L2 (Euclidean) distance

$$d_1(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$

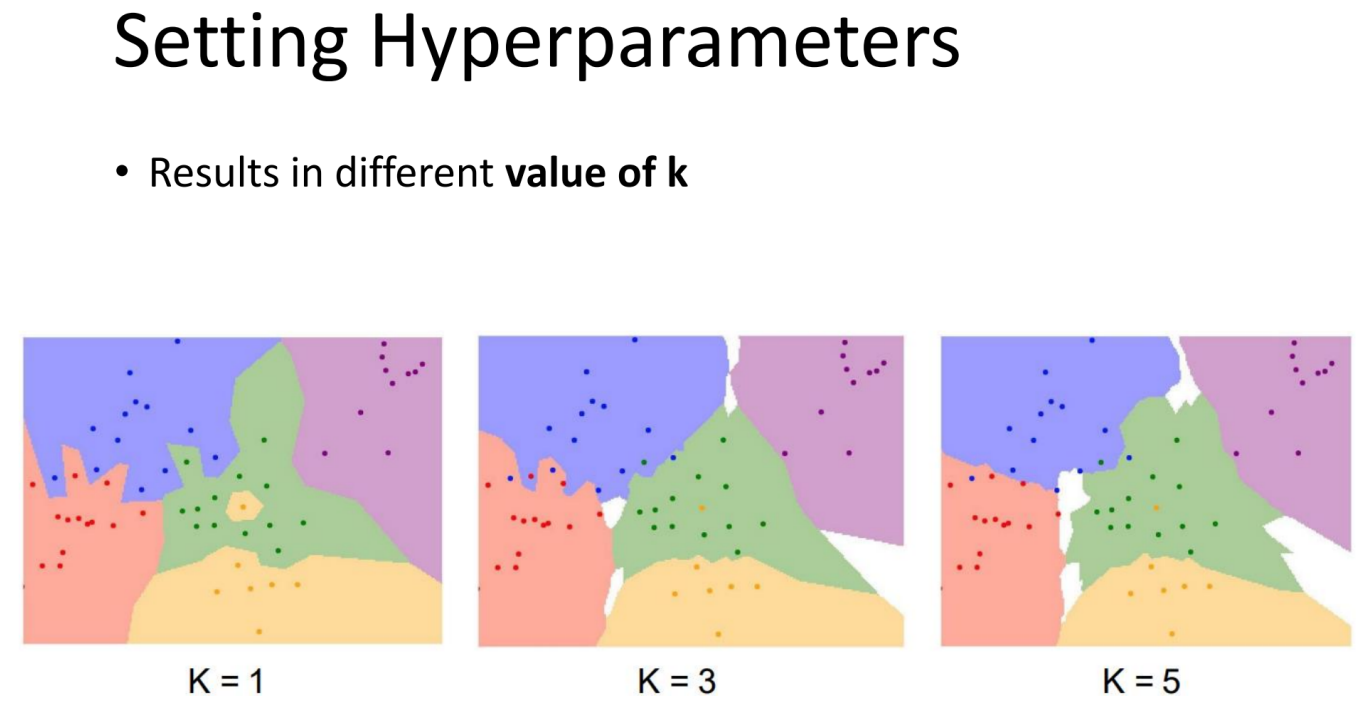


2. choose the class

image-20220506193628587

hyper-parameter fine-tuning

1. K



2. distance definition

Setting Hyperparameters

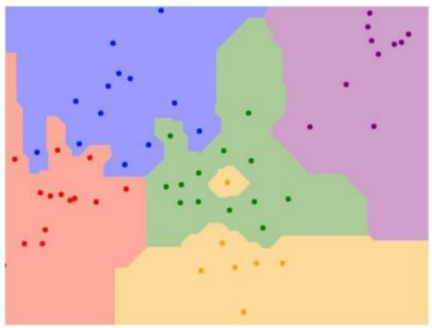
- Results in different **distance metrics**

L1 (Manhattan) distance

$$d_1(I_1, I_2) = \sum_p |I_1^p - I_2^p|$$

L2 (Euclidean) distance

$$d_1(I_1, I_2) = \sqrt{\sum_p (I_1^p - I_2^p)^2}$$



K = 1

K = 1

3. How to choose the hyper-parameters?

K-fold cross validation

Setting Hyperparameters

Your Dataset

Idea #4: **Cross-Validation**: Split data into **folds**,
try each fold as validation and average the results

fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test

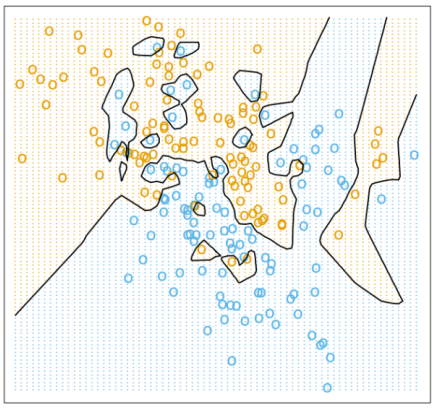
Useful for small datasets, but not used too frequently in deep learning

analysis of KNN

bias & variance

Bias-Variance for KNN

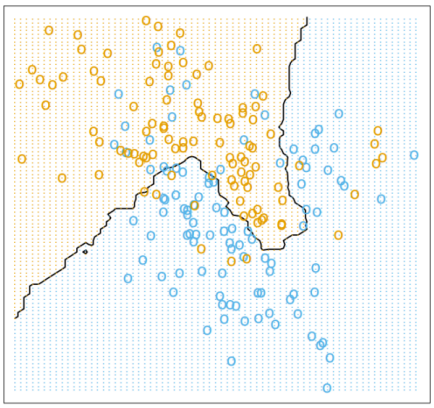
K=1



Small k

Small bias
Very complex decision boundary
Large variance
Overfitting

K=15



Large k

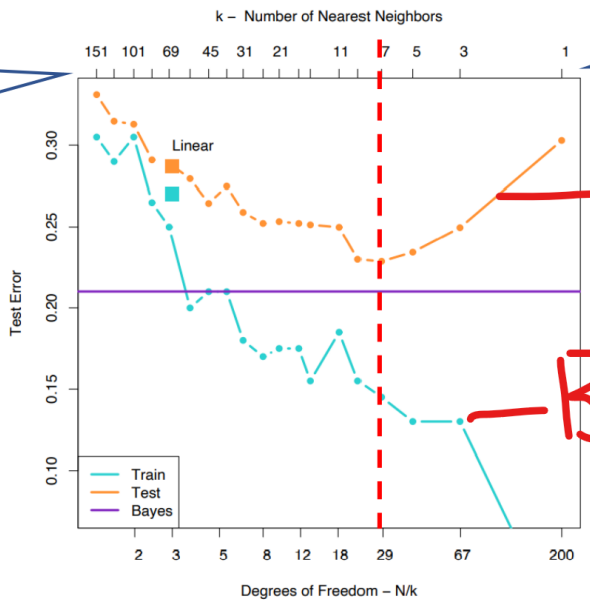
Large bias
extreme case: k=n, constant prediction
Small variance

Bias-Variance for KNN

Complexity increases when k decreases

Large k
Large bias
Small variance

Small k
Small bias
Large variance



Var

Bias

Good k
Small bias
complex enough
decision boundary
Small variance
no overfitting

Complexity

Complexity of KNN

Q: With N examples, how fast are **training** and **prediction**?

	Training	Prediction
Complexity	**O(1)**	**O(N)**
Action	**Simply remembers all the training data** **No explicit training process** ***"Lazy Learning"***	**For each test sample:** **Find closest training sample** **Predict label of nearest sample**

limitation

Can we use KNN on images?

Can we use KNN on images?

- Very slow at test time
- Distance metrics on pixels are not informative

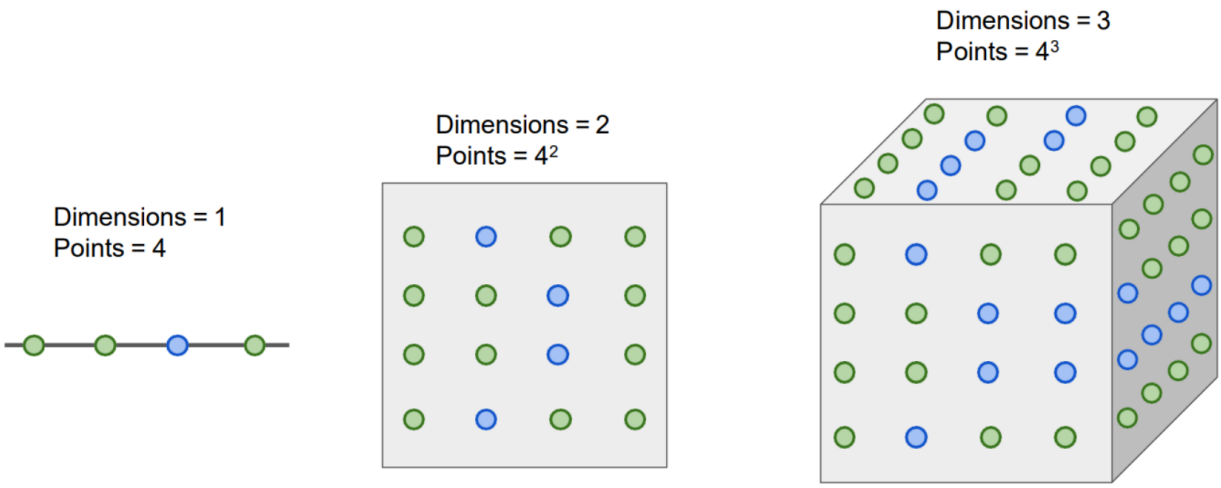
Never

(all 3 images **have same L2 distance** to the one on the left)

Can we use KNN on images?

- *Curse of dimensionality*
- In high-dimensional situations, the data samples are sparse and the distance calculation is difficult

Never



When to use KNN?

1. spatial correlation

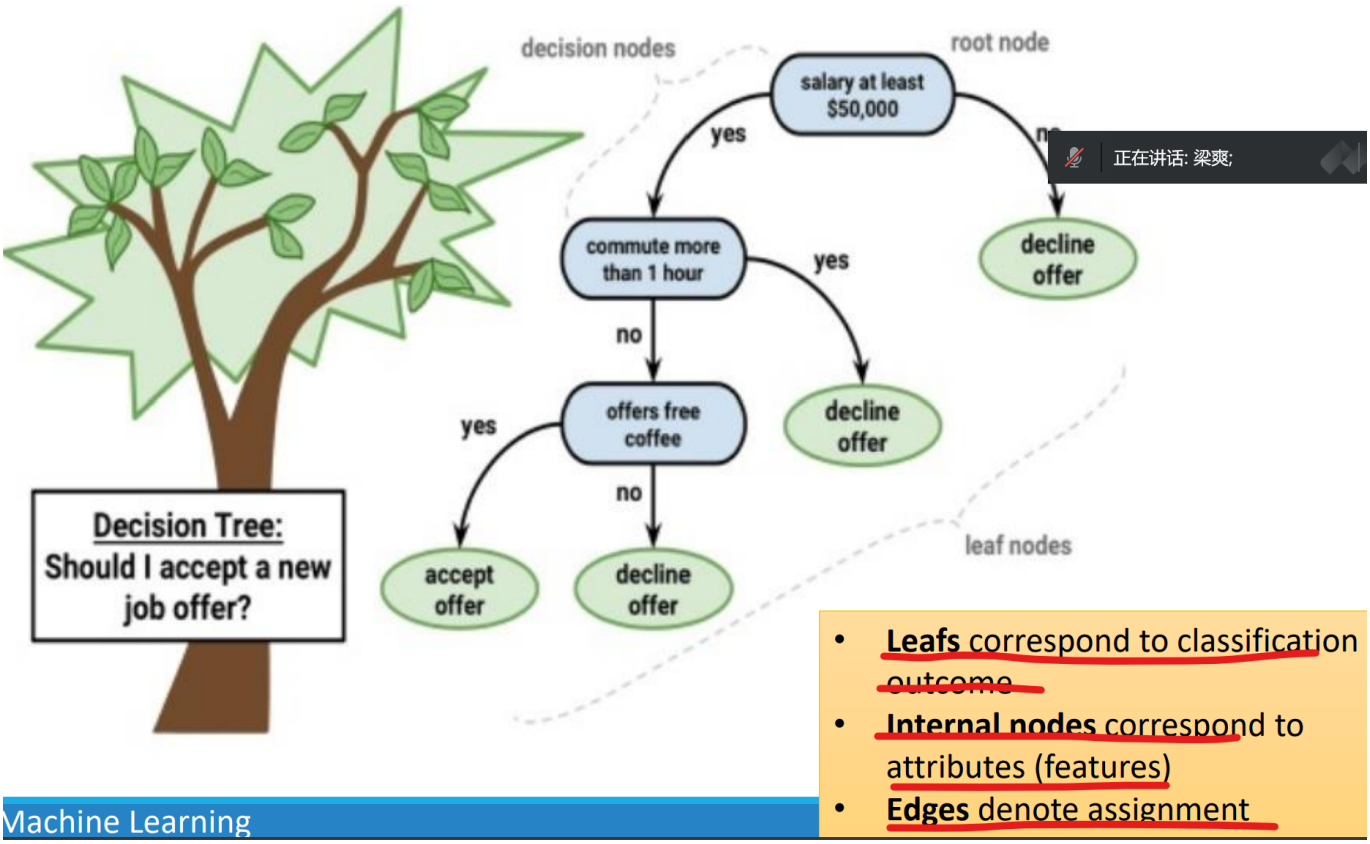
Recommender system

2. low-dimension data

Text mining

7. Decision Tree

basic concept



building a decision tree

Pseudo Code for Building A Decision Tree

Algorithm 1 决策树学习基本算法

输入:

- 训练集 $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\}$;
- 属性集 $A = \{a_1, \ldots, a_d\}$.

过程: 函数 TreeGenerate(D, A)

```
1: 生成结点 node;
2: if D 中样本全属于同一类别 C then
3:     将 node 标记为 C 类叶结点; return
4: end if
5: if A = ∅ OR D 中样本在 A 上取值相同 then
6:     将 node 标记叶结点，其类别标记为 D 中样本数最多的类; return
7: end if
8: 从 A 中选择最优划分属性 a_*;
9: for a_* 的每一个值 a_*^v do
10:    为 node 生成每一个分枝; 令 D_v 表示 D 中在 a_* 上取值为 a_*^v 的样本子集;
11:    if D_v 为空 then
12:        将分枝结点标记为叶结点, 其类别标记为 D 中样本最多的类; return
13:    else
14:        以 TreeGenerate(D_v, A − {a_*}) 为分枝结点
15:    end if
16: end for
```

输出: 以 node 为根结点的一棵决策树

The key step

（1）当前结点包含的样本全部属于同一类别

（2）当前属性集为空，或所有样本在所有属性上取值相同

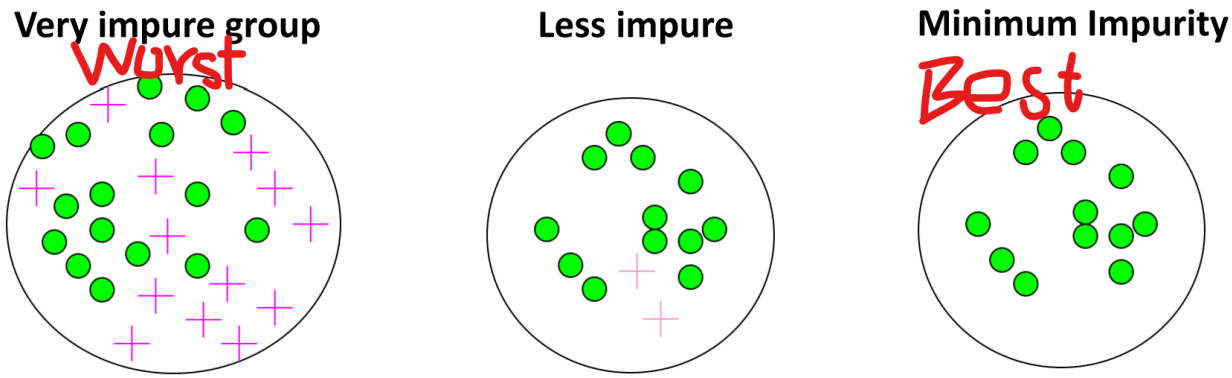
（3）当前结点包含的样本集合为空

the key step

define a metric to measure the **purity** of an attribute

- The key to decision tree learning is **how to identify the best attribute**.
- **Our Goal**: The samples contained in the branch nodes of the decision tree belong to the same class as much as possible, that is, the ***"purity"*** of the nodes is getting higher and higher

- How to measure that how much purity a certain split brings?
- We can measure the level of ***impurity*** in a group of examples



*Entropy: a metric to measure **impurity** & **uncertainty***

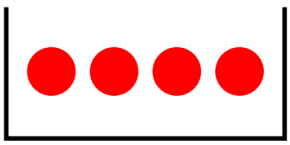
1. **entropy is the measurement of uncertainty**
2. **information is the amount of decrement of uncertainty**

example of entropy:

Entropy

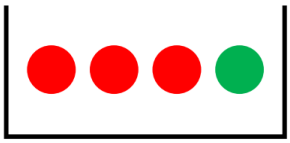
$$Ent(X) = \sum_{c} -p(X = c) log_2 p(X = c)$$

"+" for red, "-" for green



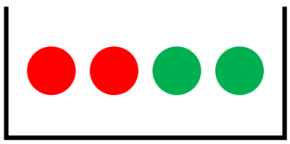
p(+)=1, p(-)=0

$$Ent(4+, 0-) = -(1 \log_2 1 + 0 \log_2 0) = 0$$



p(+)=3/4, p(-)=1/4

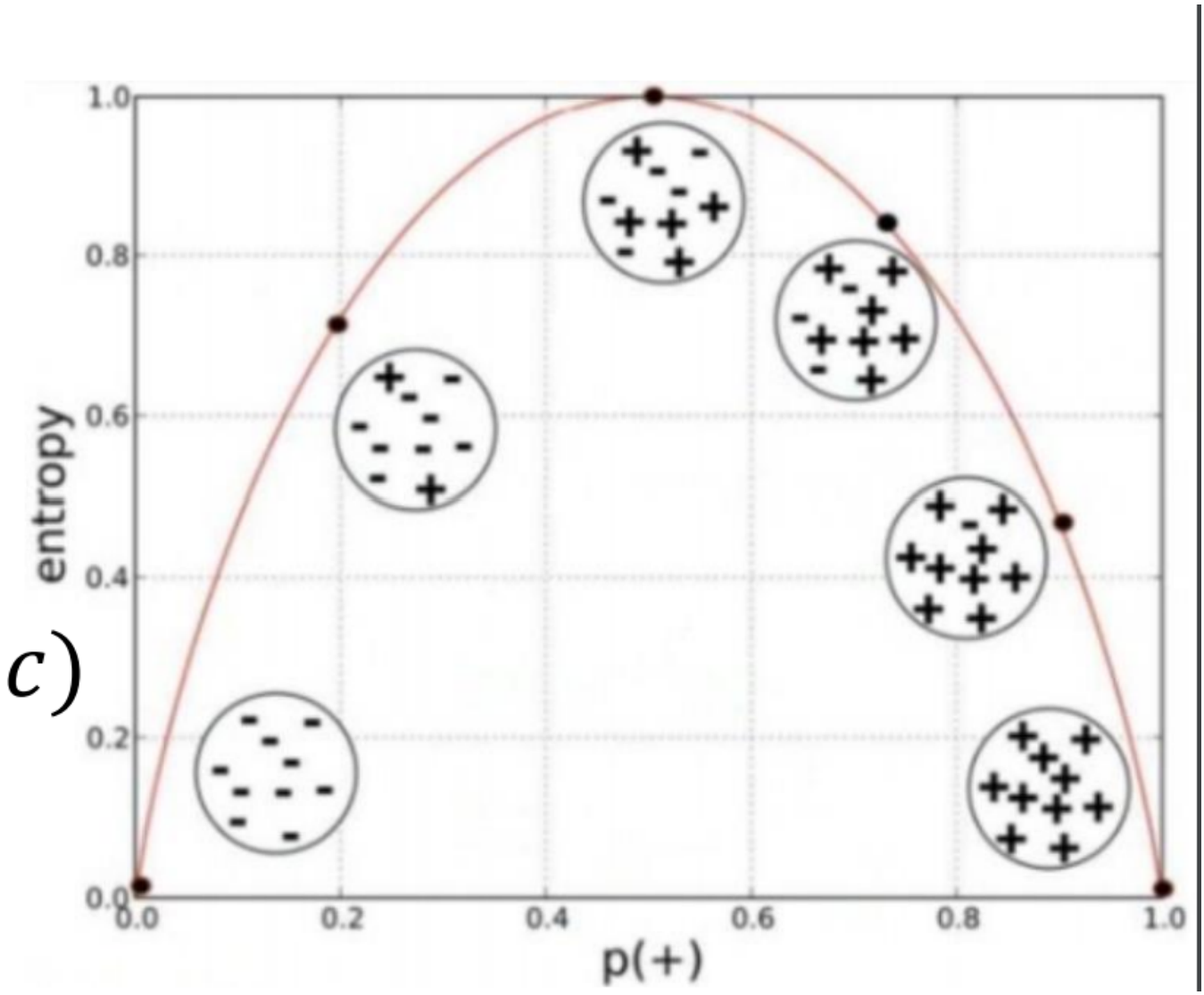
$$Ent(3+, 1-) = -\left(\frac{3}{4}\log_2\frac{3}{4} + \frac{1}{4}\log_2\frac{1}{4}\right) = 0.811$$



p(+)=1/2, p(-)=1/2

$$Ent(2+, 2-) = -\left(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}\right) = 1$$

the varying of entropy



information gain (used by ID3)

Information Gain

- Discrete attribute a has v possible values
- Dividing with a will generate v branch nodes
- The v -th branch node contains all samples in D whose value is a^v on attribute a, denoted as D^v
- Then the "information gain" obtained by dividing the sample set D with attribute a can be calculated:

$$\mathrm{Gain}(D,a) = \mathrm{Ent}(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|}\mathrm{Ent}(D^v)$$

$\mathrm{Ent}(D)$ is the original entropy

$\sum_{v=1}^{V} \frac{|D^v|}{|D|}\mathrm{Ent}(D^v)$ is the weighed sum of all subsets

We are finding the attribute that maximizes the information gain

case study *(to be filled)*

limitation of information gain

Information Gain

- If 编号 is also used as a candidate division attribute, its information gain is generally much greater than other attributes.
- Obviously, such a decision tree does not have the ability to generalize and cannot make effective predictions on new samples

Information gain has a preference for attributes with a larger number of possible values

other rules to calculate information gain

1. gain ratio (used by C4.5)

Other rules

- **Gain Ratio**

$$\text{Gain\_ratio}(D,a) = \frac{\text{Gain}(D,a)}{\text{IV}(a)}$$

Where $$\text{IV}(a) = -\sum_{v=1}^{V} \frac{|D^v|}{|D|} \log_2 \frac{|D^v|}{|D|}$$

- $IV(a)$ is called the *intrinsic value* (固有值) of attribute a. The more possible values of attribute a (that is, the larger V), the larger the value of $IV(a)$ is.

2. Gini index (used by CART)

- **Gini index**
- We also use Gini value to measure the ***"purity"*** of the sample set

$$\mathrm{Gini}(D) = \sum_{k=1}^{|\mathcal{Y}|} \sum_{k' \neq k} p_k p_{k'} \quad = 1 - \sum_{k=1}^{|\mathcal{Y}|} p_k^2$$

Reflects the probability that two samples are randomly drawn from D with inconsistent class labels

- $Gini(D) \downarrow$, *Purity* $\uparrow$
- $Gini\_index$ of attribute a is defined as:

$$\mathrm{Gini\_index}(D, a) = \sum_{v=1}^{V} \frac{|D^v|}{|D|} \mathrm{Gini}(D^v)$$

where p_k and p'_k are probabilities

pruning of a tree

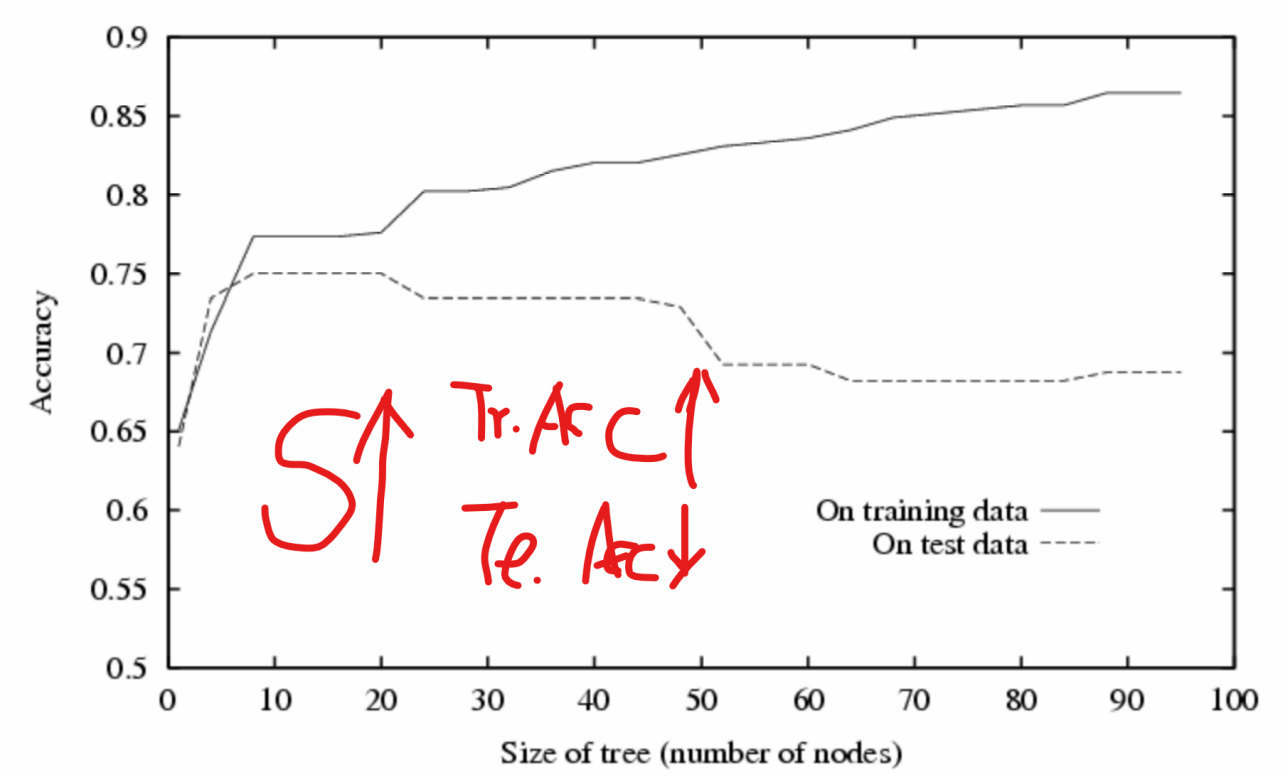
1. overfitting of a tree

Overfitting in decision trees

- In order to classify the training samples as correctly as possible, the node split process will be repeated, sometimes resulting in ***too many branches in the decision tree***.
- At this time, the training samples may be learned "too well", so that some features of the training set itself are regarded as the general nature of all data, resulting in ***overfitting***.

Overfitting in decision trees

正在讲话: 梁爽;



2. pruning

a. pre-pruning(预剪枝)

Prepruning

- Estimate the generalization performance before dividing the node, and stop if the division cannot bring about an improvement.

✓Reduce risk of overfitting

✓Significantly reduces training time and test time overhead

×Underfitting risk: some branches that may bring performance improvements in the future are prohibited from expanding

case study *(to be filled)*

b. post-pruning(后剪枝)

all of the non-leaf nodes can be pruned(including the root)

Postpruning

- First generate a complete decision tree, and then examine non-leaf nodes in a bottom-up manner. Replacing the subtree corresponding to the node with a leaf node when the generalization performance can be improved.

✓Low underfitting risk

✓The generalization performance is often better than that of pre-pruned decision trees

✕High training time

case study *(to be filled)*

DT for continuous values

continuous value discretization

example: bi-partition

- **Step 1**: Sort the *n* values of continuous attributes *a* on the sample set
- **Step 2**: The median point of adjacent values is used as a candidate division point to obtain a set of candidate division points

$$T_a = \left\{ \frac{a^i + a^{i+1}}{2} \mid 1 \leq i \leq n-1 \right\}$$

- **Step 3**: Identify the best attribute according to the method of discrete attributes (e.g. information gain)

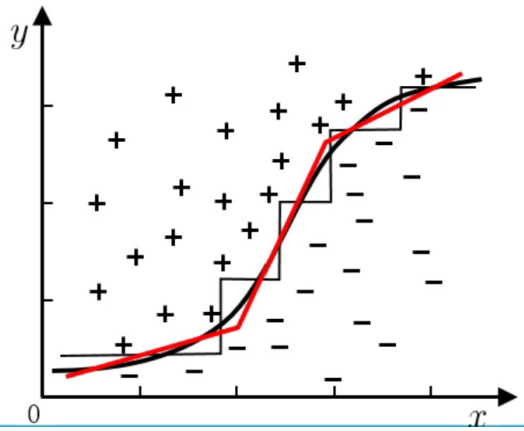
case study *(to be filled)*

multivariant DT

the efficiency of univariant DT is low on datasets like this:

Multivariate decision tree

- We can use oblique boundaries to simplify the model
- Non-leaf nodes are no longer only for a certain attribute, but a linear combination of attributes
- Learning: build a suitable classifier for each non-leaf node (instead of find the best attribute)

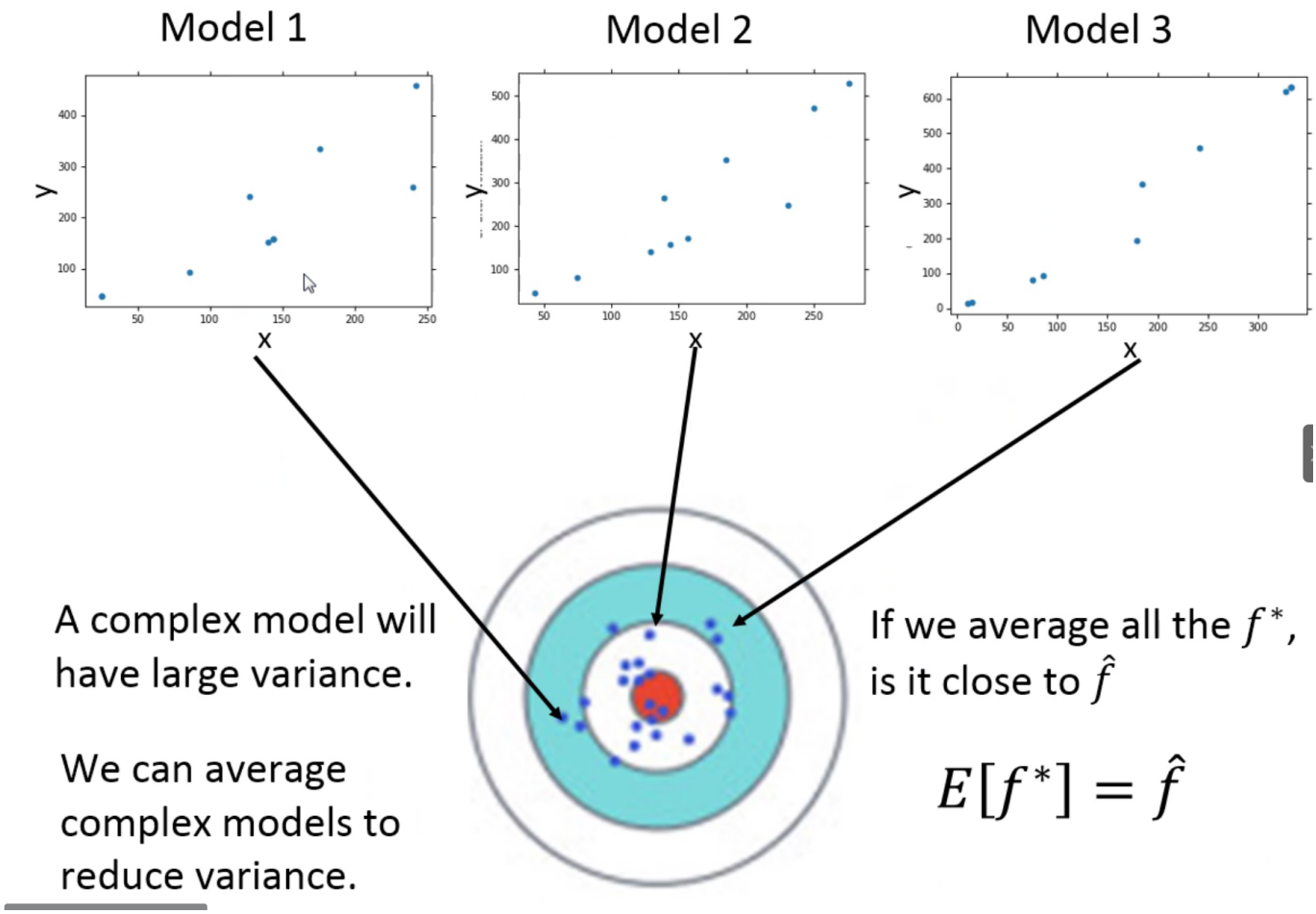


we can use a smooth curve as the decision boundary

case study *(to be filled)*

random forest

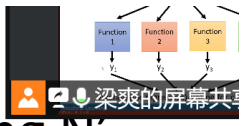
To reduce the variance of DT, we can **average different models**:

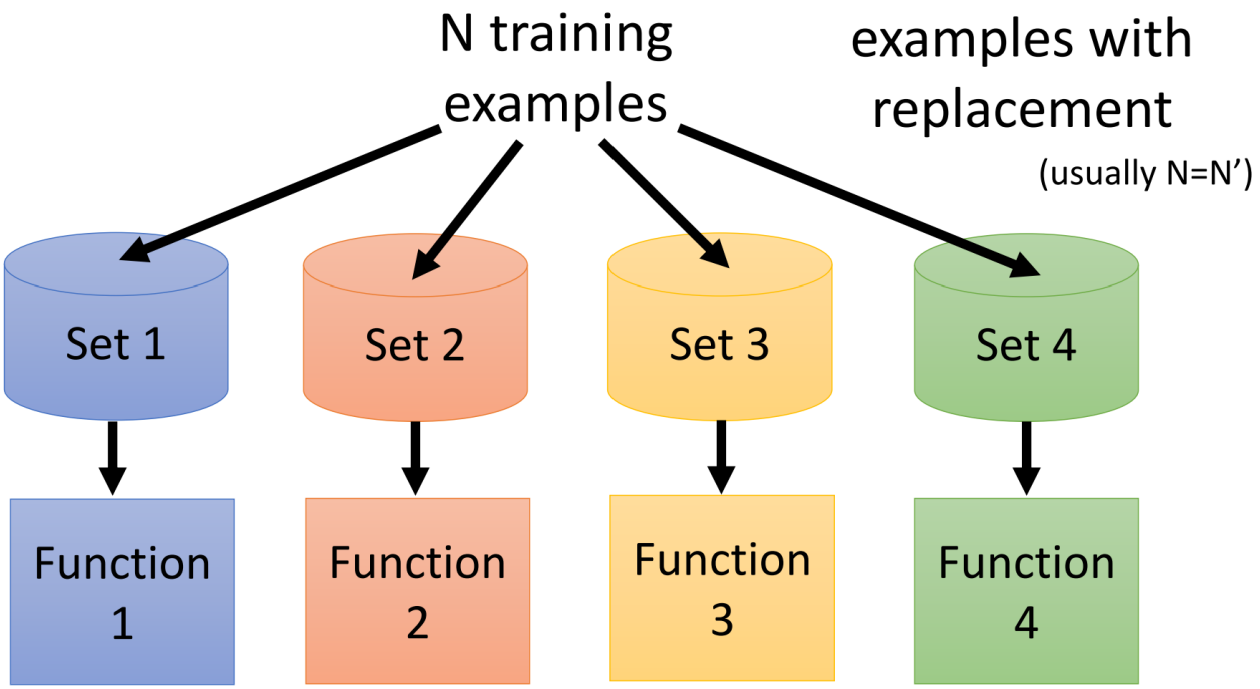


bagging: used for models that is complex and easy to overfit

to train different models, we can use bootstrapping sampling:

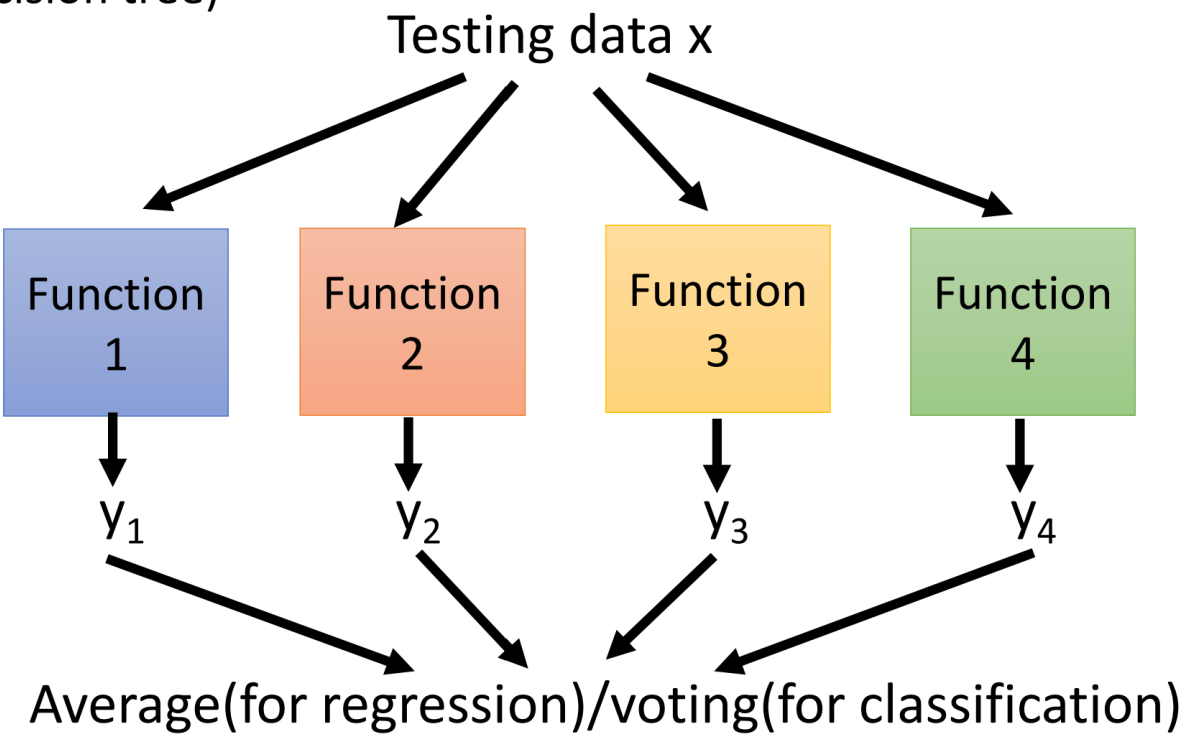
Bagging


梁爽的屏幕共享


N training examples
Sampling N' examples with replacement
(usually N=N')
Set 1
Set 2
Set 3
Set 4
Function 1
Function 2
Function 3
Function 4

Bagging

- Helpful when your model is ***complex, easy to overfit*** (e.g. decision tree)



bagging for random forest

Random Forest

train	f_1	f_2	f_3	f_4
x^1	O	X	O	X
x^2	O	X	X	O
x^3	X	O	O	X
x^4	X	O	X	O

- Decision tree:
  - Easy to achieve 0% error rate on training data
    - If each training example has its own leaf ……
- Random forest: ***Bagging of decision tree***
  - Resampling training data is not sufficient
  - Randomly restrict the features/questions used in each split
- Out-of-bag validation for bagging
  - Using RF = f_2+f_4 to test x^1
  - Using RF = f_2+f_3 to test x^2
  - Using RF = f_1+f_4 to test x^3
  - Using RF = f_1+f_3 to test x^4

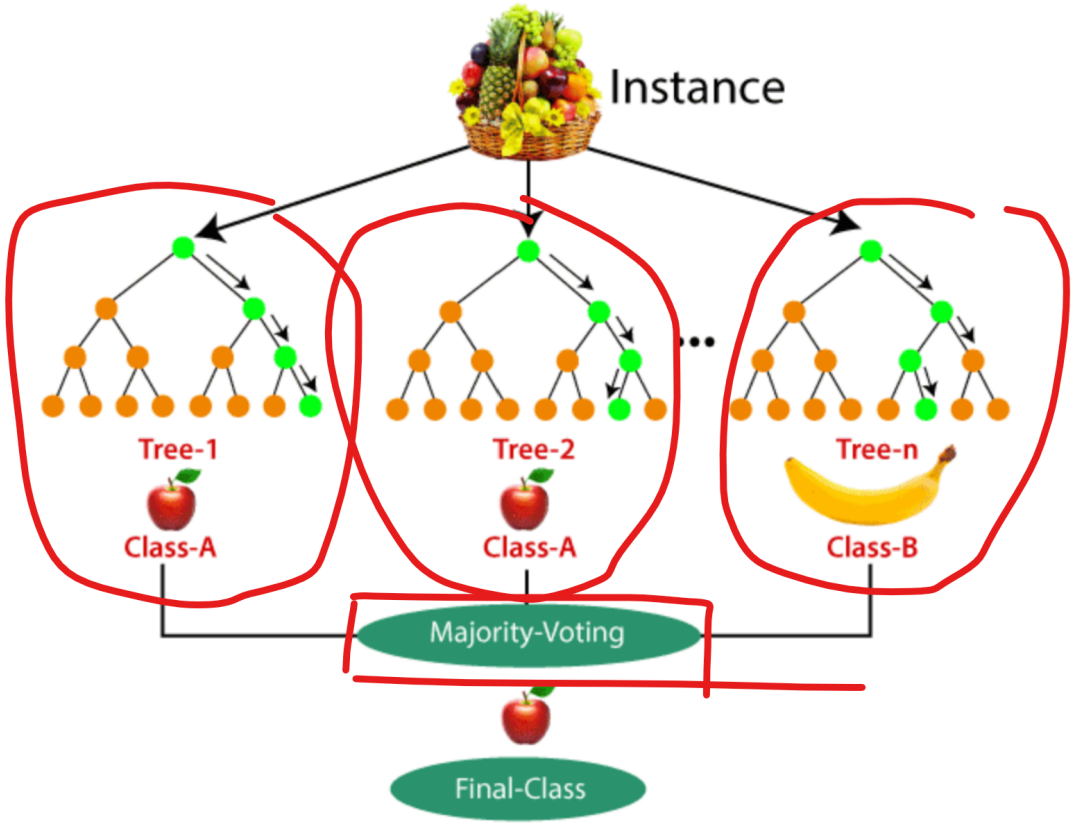
Out-of-bag (OOB) error

Good error estimation of testing set

out-of-bag's bag is not bagging's bag

To predict, use majority voting:

Random Forest

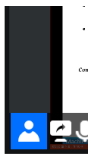


8. Bayes Classifier

Bayes decision theory

basic idea:

Bayesian decision theory



- Suppose there are N possible class labels

$$y = \{c_1, c_2, \dots, c_N\}$$

- Then the expected loss of classifying sample x as c_i is

$$R(c_i|x) = \sum_{j=1}^{N} \lambda_{ij} P(c_j|x)$$

Conditional Risk

λ_{ij} : Loss for misclassifying a true sample labeled c_j as c_i

$P(c_j|x)$: Posterior probability

- Our task is to find a decision rule $h: X \to Y$ that minimizes the overall risk

$$R(h) = E_x[R(h(x)|x)]$$

- For each sample x, if h can minimize the conditional risk $R(h(x)|x)$, then the overall risk $R(h)$ will also be minimized

The loss function:

- To minimize the overall risk, we can simply select the class that minimizes the conditional risk $R(c|x)$ on each sample

$$h^*(x) = argmin\, R(c|x) \quad (c \in y)$$

- $h^*(x)$: Bayes optimal classifier
- $R(h^*)$: Bayes risk
- $1 - R(h^*)$: The best performance the classifier can achieve

λ_{ij} is essentially the weight for each misclassified sample.

for a specific case:

- We need to minimize *the classification error rate*, then
- Misclassification loss

1. $$\lambda_{ij} = \begin{cases} 0, & if \ \ i = j \\ 1, & otherwise \end{cases}$$

- Conditional risk

2. $$R(c|x) = 1 - \ P(c|x)$$

- Bayes optimal classifier

3. $$h^*(x) = argmax \ P(c|x) \ \ (c \in y)$$

For each sample, select the class that **maximizes the posterior probability**

what it really does:

Explaining Machine Learning from a **Probabilistic View**

What machine learning wants to achieve is to ***estimate*** the posterior probability as accurately as possible based on limited training samples

Generative models:

Generative Models

$$P(c|x) = \frac{P(x,c)}{P(x)}$$

- According to Bayes' theorem $P(c|x)$ can be written as

$$P(c|x) = \frac{P(c)P(x|c)}{P(x)}$$

Class-conditional probability of sample x relative to class c

Prior Probability
can be estimated by the frequency of occurrence of various samples

"Evidence" factor, independent of the class

$P(c)$ is the frequency for class c in the training set.

$P(x|c)$ is the class conditional probability, which we will do a good research

Naïve Bayes Classifier

Assumption: attributes are independent from each other

difficult to estimate directly from limited training samples

- **Attribute conditional independence assumption**: For known classes, all attributes are assumed to be independent of each other
- Based on this assumption, we can get:

$$P(c|x) = \frac{P(c)P(x|c)}{P(x)} = \frac{P(c)}{P(x)}\prod_{i=1}^{d} P(x_i|c)$$

and one more step, we can get the optimization object of NBC:

$$P(c|x) = \frac{P(c)P(x|c)}{P(x)} = \frac{P(c)}{P(x)} \prod_{i=1}^{d} P(x_i|c)$$

- $P(x)$ is the same for all classes, so according to the Bayesian decision rule (on page 11):

$$h_{nb}(x) = argmax\ P(c) \prod_{i=1}^{d} P(x_i|c) \quad (c \in y)$$

- This is the expression for the **Naive Bayes classifier**

Training the NBC:

1. estimate $P(c)$

It is just the frequency.

2. estimate $P(x_i|c)$

- For ***discrete attributes***, Let D_{c,x_i} denote the set of samples in D_c with the value x_i on the i-th attribute, then

$$P(x_i|c) = \frac{|D_{c,x_i}|}{|D_c|}$$

- For ***continuous attributes***, assume that $p(x_i|c) \sim N(\mu_{c,i}, \sigma_{c,i}^2)$, where $\mu_{c,i}$ and $\sigma_{c,i}^2$ are the mean and variance of the value of the c-class sample on the i-th attribute, respectively. Then

$$p(x_i|c) = \frac{1}{\sqrt{2\pi}\sigma_{c,i}} \exp\left(-\frac{(x_i - \mu_{c,i})^2}{2\sigma_{c,i}^2}\right)$$

Case Study(to be filled)

What if an attribute does not appear?

- What if an attribute value does not appear at the same time with a class in the training set?
- For example, for a test sample with attribute value “敲声=清脆",

$$P_{清脆|是} = P(敲声 = 清脆 \mid 好瓜 = 是) = \frac{0}{8} = 0$$

- Then we will find that $P(c)\prod_{i=1}^{d} P(x_i|c) = 0$!
- The classification result will be “好瓜=否” even if it is obviously like a good melon in other attributes.

Unreasonable!

We can do Laplacian correction

- Laplacian correction

$$\hat{P}(c) = \frac{|D_c| + 1}{|D| + N}$$

$$\hat{P}(x_i|c) = \frac{|D_{c,x_i}| + 1}{|D_c| + N_i}$$

- N: The number of possible classes in the training set D
- N_i: The number of possible values of the i-th attribute
- Example: For $P(c)$ in the watermelon case,

$$\hat{P}(好瓜 = 是) = \frac{8+1}{17+2} \approx 0.474\ , \quad \hat{P}(好瓜 = 否) = \frac{9+1}{17+2} \approx 0.526\ .$$

Bayesian Network

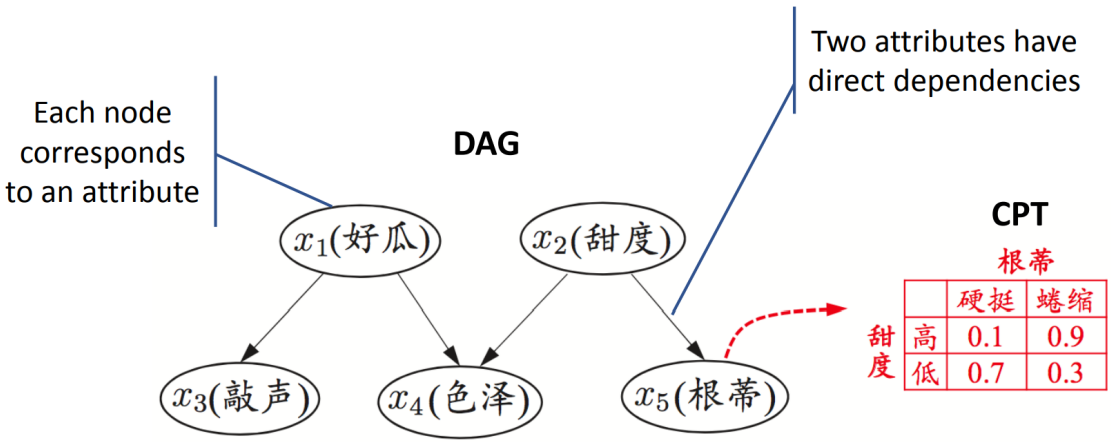
basic idea: not independent

Bayesian Network

- A Bayesian network B consists of two parts, the structure G and the parameter Θ, i.e. $B =< G, \Theta >$
- The network structure G is a DAG
- The parameter Θ quantitatively describes the direct dependencies between attributes
- Assuming that the parent node set of attribute x_i in G is π_i, then Θ contains the CPT of each attribute: $\theta_{x_i|\pi_i} = P_B(x_i|\pi_i)$

example of the network:

A watermelon case



- What we can learn from the DAG
  - “色泽”直接依赖于“好瓜”和“甜度”
  - “根蒂”直接依赖于“甜度”
- What we can learn from the CPT
  - P(根蒂=硬挺|甜度=高)=0.1

Structure:

Structure

- The Bayesian network structure effectively expresses the conditional independence between attributes
- Given a set of parent nodes, the Bayesian network assumes that each attribute is independent of its non-descendant attributes, then $B =< G, \Theta >$ defines the joint probability distribution of attributes $x_1, x_2, \dots, x_d$ as:

$$P_B(x_1, x_2, \dots, x_d) = \prod_{i=1}^{d} P_B(x_i|\pi_i) = \prod_{i=1}^{d} \theta_{x_i|\pi_i}$$

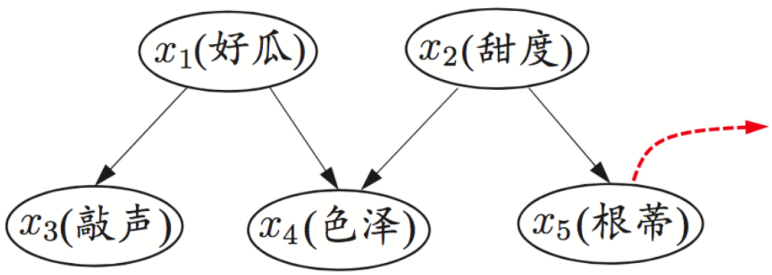
example:

Structure

- Example: The joint probability distribution of the watermelon case is defined as

$$P(x_1, x_2, x_3, x_4, x_5) = P(x_1)\, P(x_2) P(x_3|x_1) P(x_4|x_1, x_2) P(x_5|x_2)$$

- x_3 and x_4 are independent when given x_1 => $x_3 \perp x_4 \,|x_1$
- x_4 and x_5 are independent when given x_2 => $x_4 \perp x_5 \,|x_2$

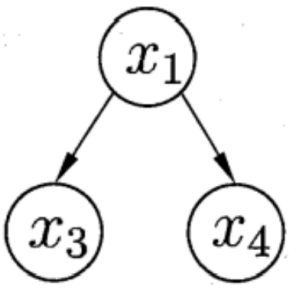


根蒂

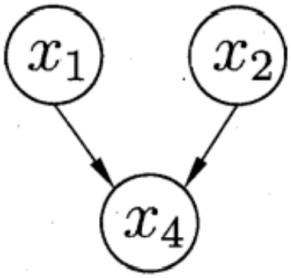
甜度	硬挺	蜷缩
高	0.1	0.9
低	0.7	0.3

Typical structures:

Typical dependencies

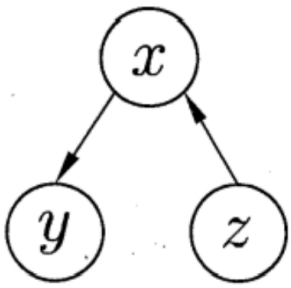


Common parent
同父结构
Given x_1, x_3 and x_4 are independent



V structure
V型结构
Given x_4, x_1 and x_2 are not independent.
But when x_4 is unknown, x_1 and x_2 are independent.

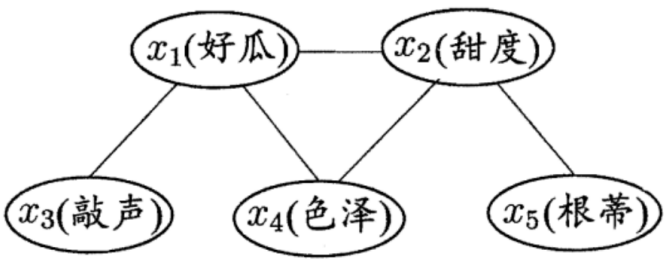
Marginal independence, $x_1 \perp\!\!\!\perp x_2$



Sequential structure
顺序结构
Given x, y and z are independent

moral graph(to be filled)

- Based on the moral graph, the conditional independence between variables can be found intuitively and quickly
- Assuming that there are variables x, y and variable set $\boldsymbol{z} = \{z_i\}$ in the moral graph. If x, y can be separated by $\boldsymbol{z}$ on the graph, then $x \perp y \mid \boldsymbol{z}$
- Try to find as many conditional independence relationships as possible in the moral graph of the watermelon case



Example:

$x_3 \perp x_4 \mid x_1$
$x_4 \perp x_5 \mid x_2$
$x_3 \perp x_2 \mid x_1$
$x_3 \perp x_5 \mid x_1$
$x_3 \perp x_5 \mid x_2$

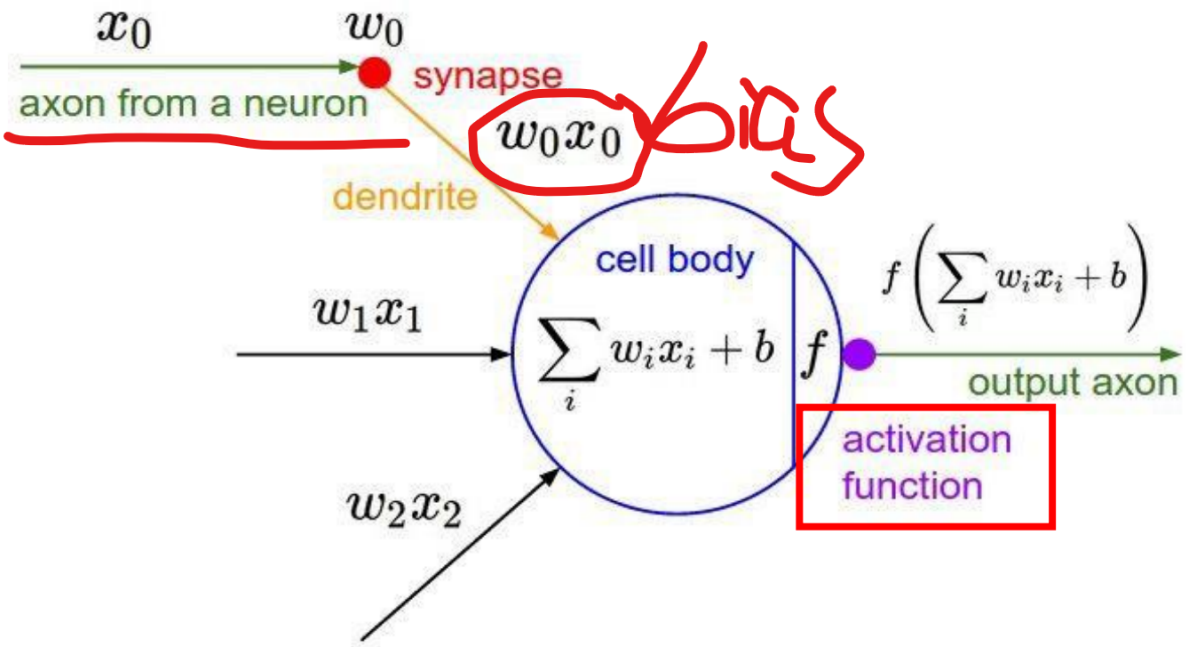
General Steps

1. learning
2. inference(NP hard)

9. Neural Network & CNN

a. NN

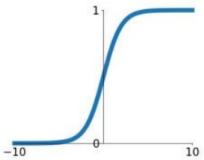
artificial neuron



activation function

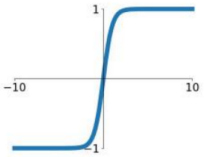
Sigmoid

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



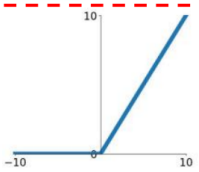
tanh

$$\tanh(x)$$



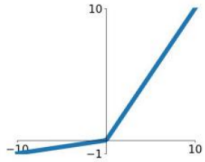
ReLU

$$\max(0, x)$$



Leaky ReLU

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

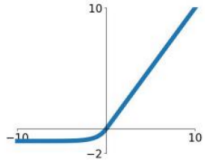


Maxout

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

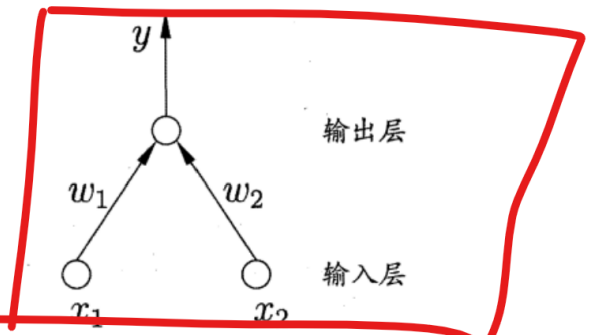
ELU

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



perceptron

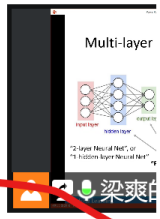
- It consists of two layers of neurons, the **input layer** and the **output layer**.

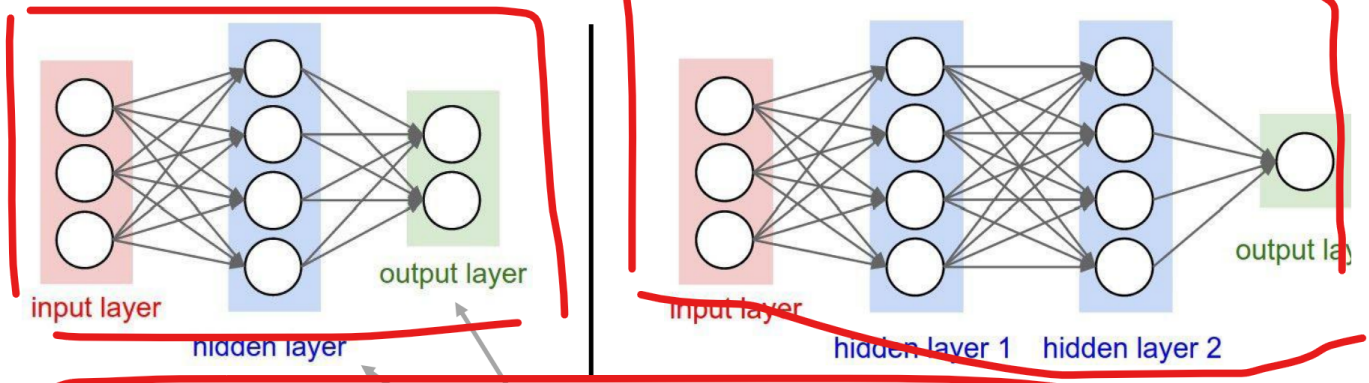


- Can realize logical AND, OR, NOT operation
- Only the neurons in the output layer perform activation

NN structure

Multi-layer Network





2-layer Neural Net", or
"1-hidden-layer Neural Net"

"3-layer Neural Net", or
"2-hidden-layer Neural Net"

"Fully-connected" layers

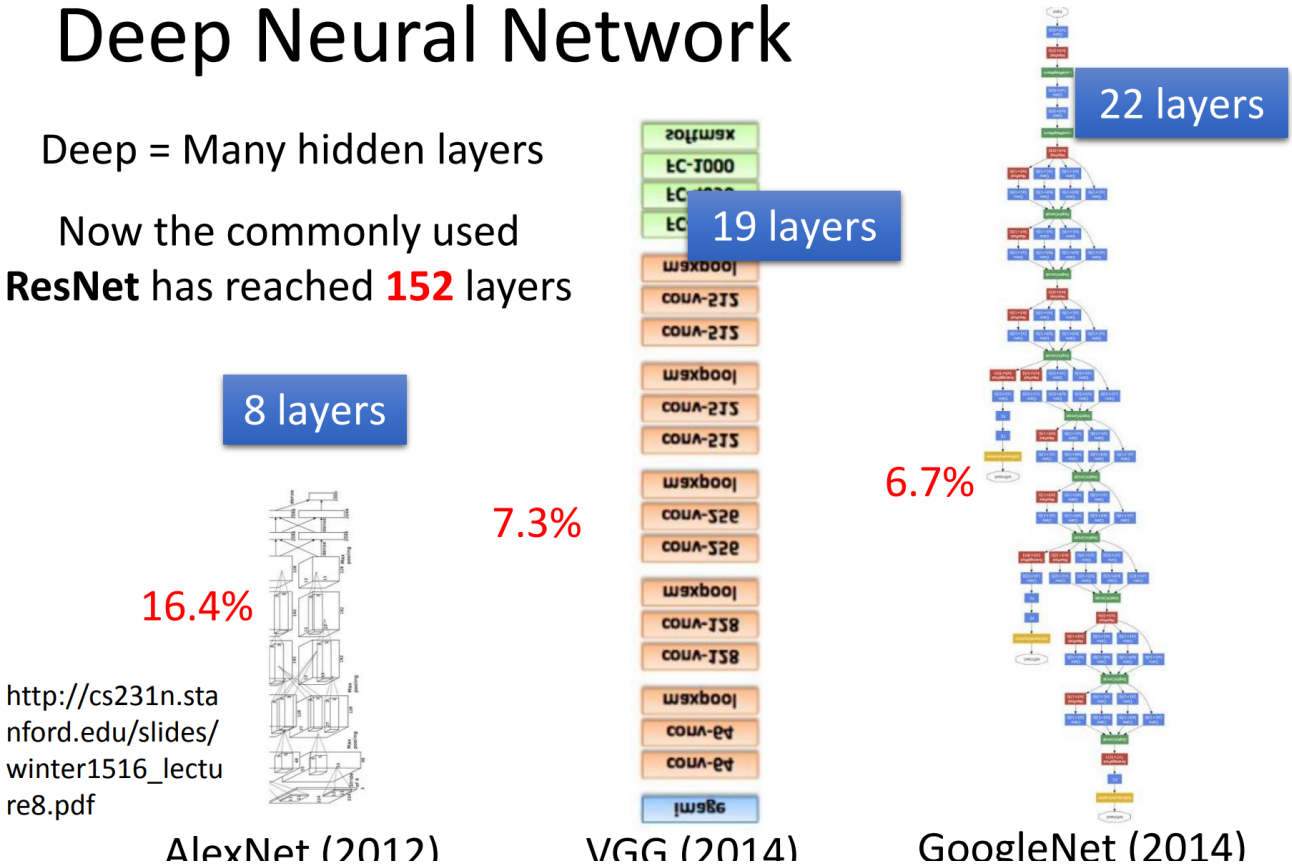
Hidden layer and output layer neurons are ***functional neurons*** with activation functions

Deep NN

Deep Neural Network

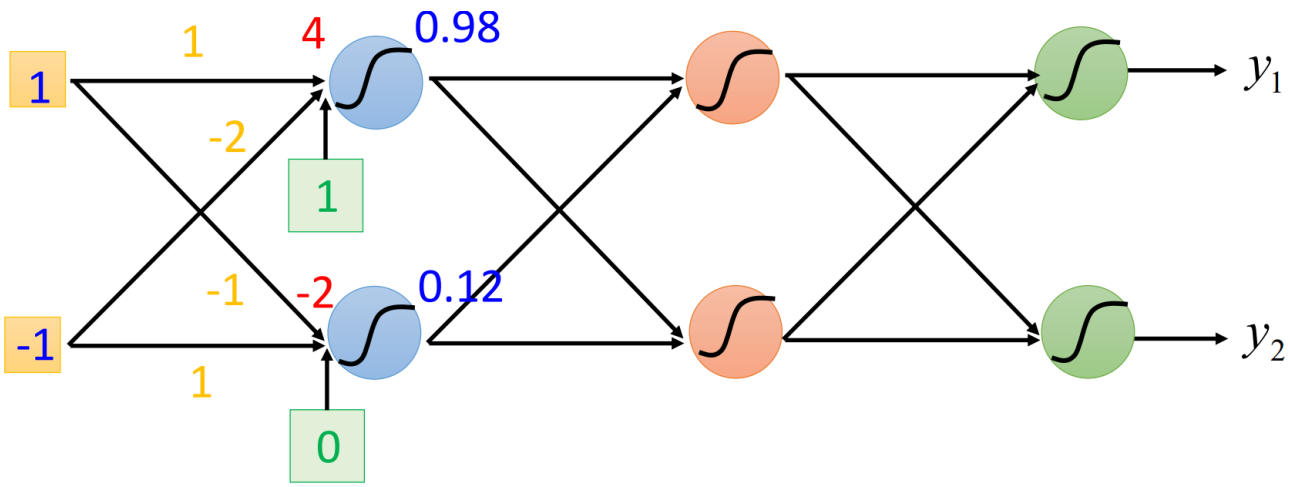
Deep = Many hidden layers

Now the commonly used
ResNet has reached **152** layers



How does this work

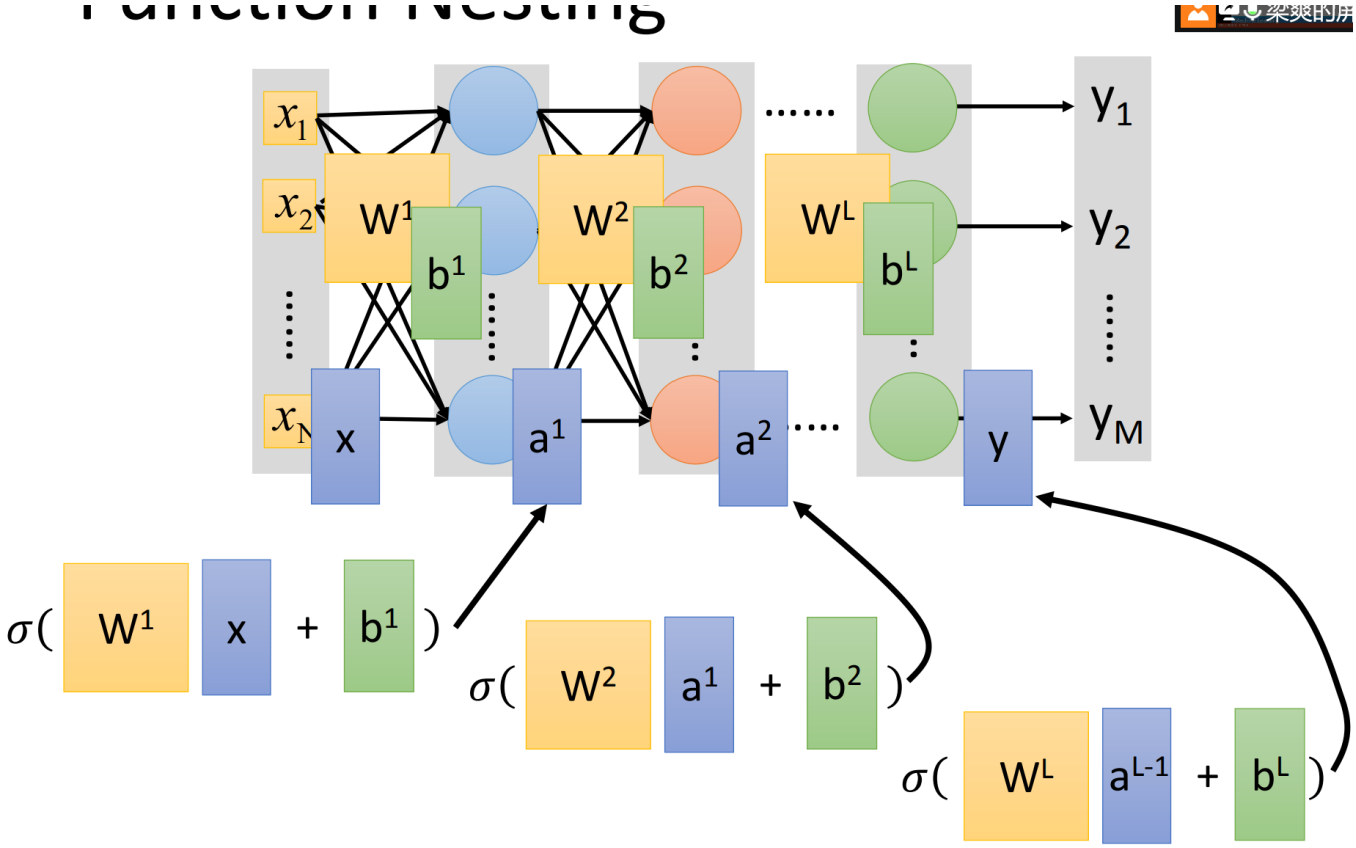
a basic structure:



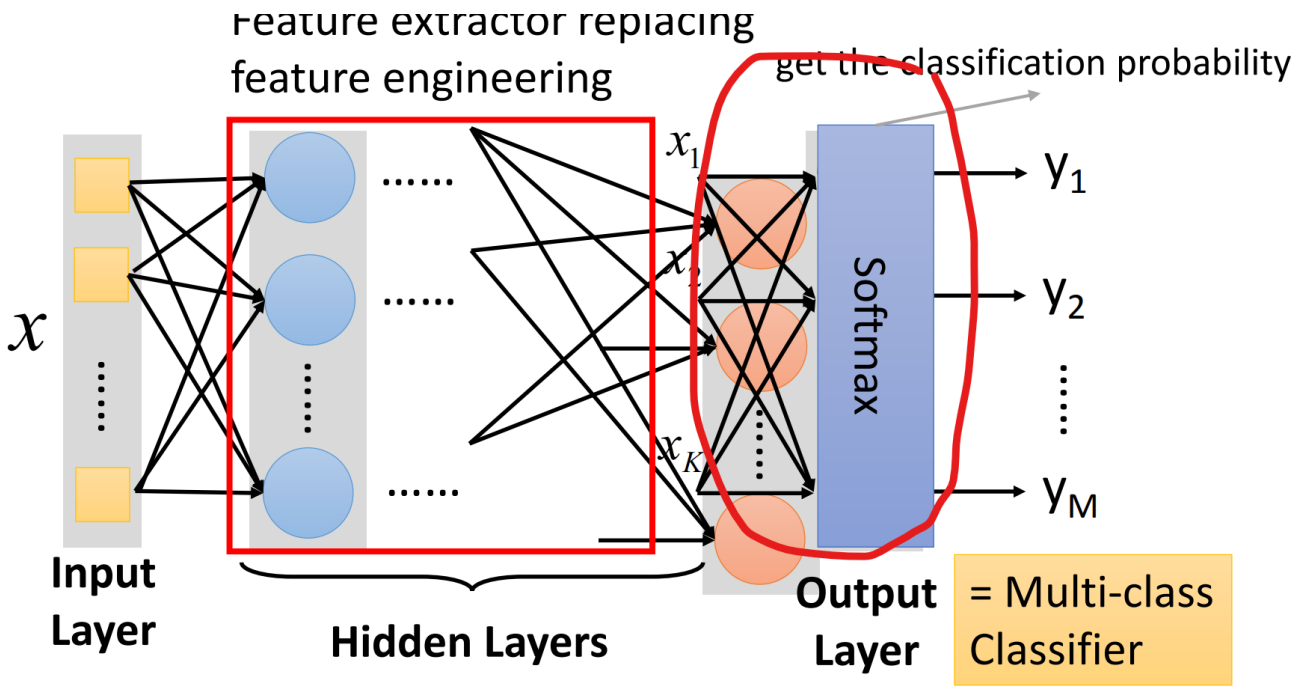
$$\sigma\left(\begin{bmatrix}1 & -2\\ -1 & 1\end{bmatrix}\begin{bmatrix}1\\ -1\end{bmatrix}+\begin{bmatrix}1\\ 0\end{bmatrix}\right)=\begin{bmatrix}0.98\\ 0.12\end{bmatrix}$$

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

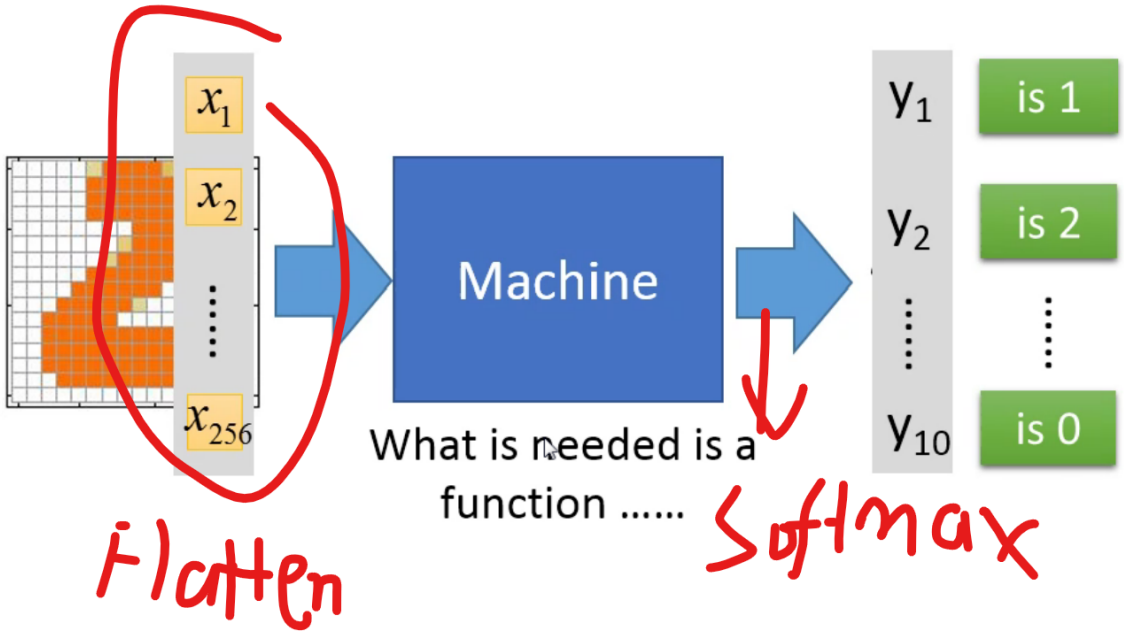
Generally:



As a classifier:



case study: handwritten digits recognition



Backpropagation

- If we use gradient descent directly

  Network parameters $\theta = \{w_1, w_2, \cdots, b_1, b_2, \cdots\}$

  Starting Parameters $\theta^0 \longrightarrow \theta^1 \longrightarrow \theta^2 \longrightarrow \cdots\cdots$

$$\nabla L(\theta) = \begin{bmatrix} \partial L(\theta)/\partial w_1 \\ \partial L(\theta)/\partial w_2 \\ \vdots \\ \partial L(\theta)/\partial b_1 \\ \partial L(\theta)/\partial b_2 \\ \vdots \end{bmatrix}$$

$$Compute\ \nabla L(\theta^0) \qquad \theta^1 = \theta^0 - \eta \nabla L(\theta^0)$$

$$Compute\ \nabla L(\theta^1) \qquad \theta^2 = \theta^1 - \eta \nabla L(\theta^1)$$

Millions of parameters .....

To compute the gradients efficiently, we use ***backpropagation***.

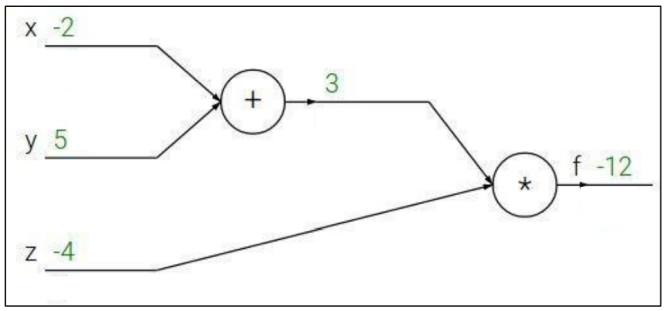
computation diagram

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

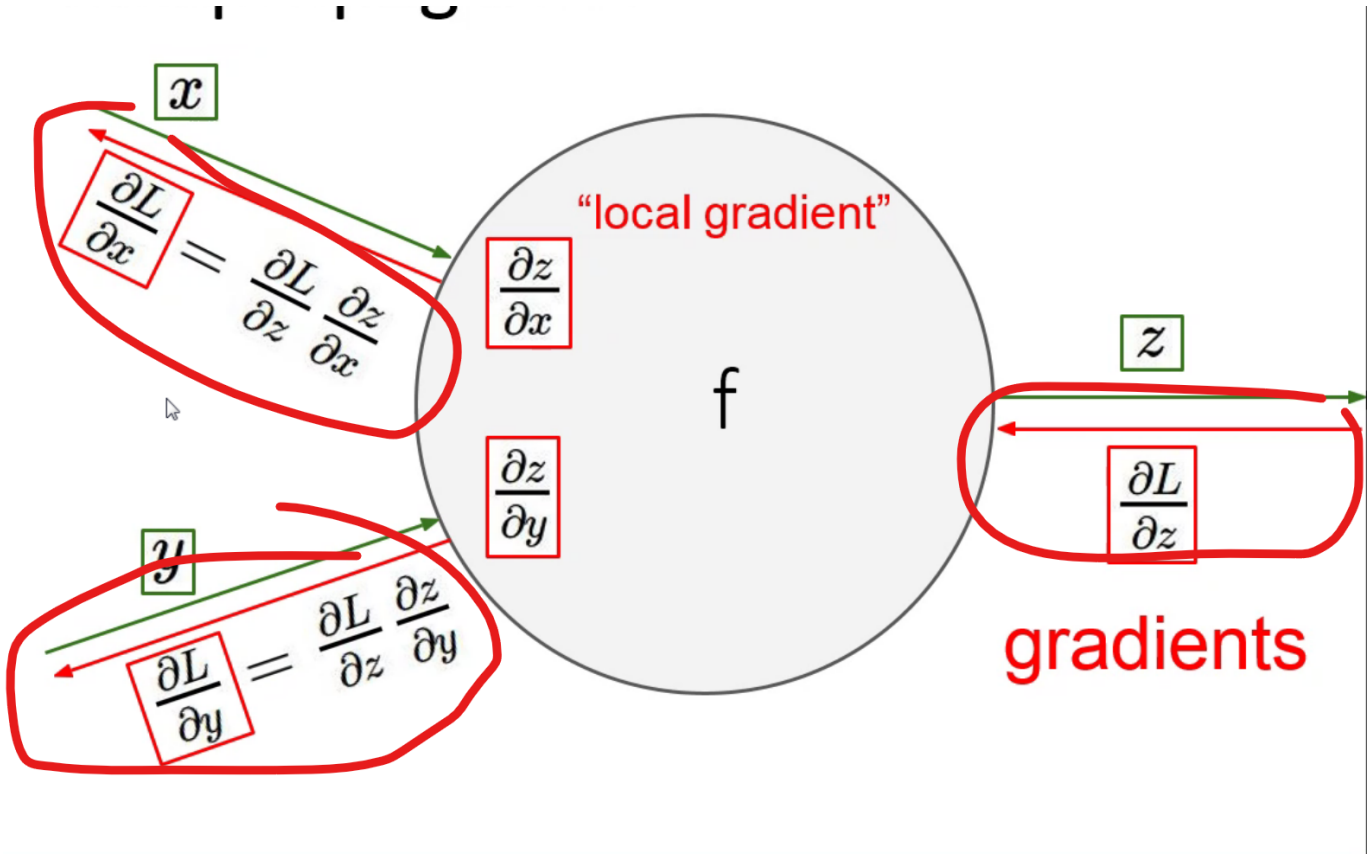
$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

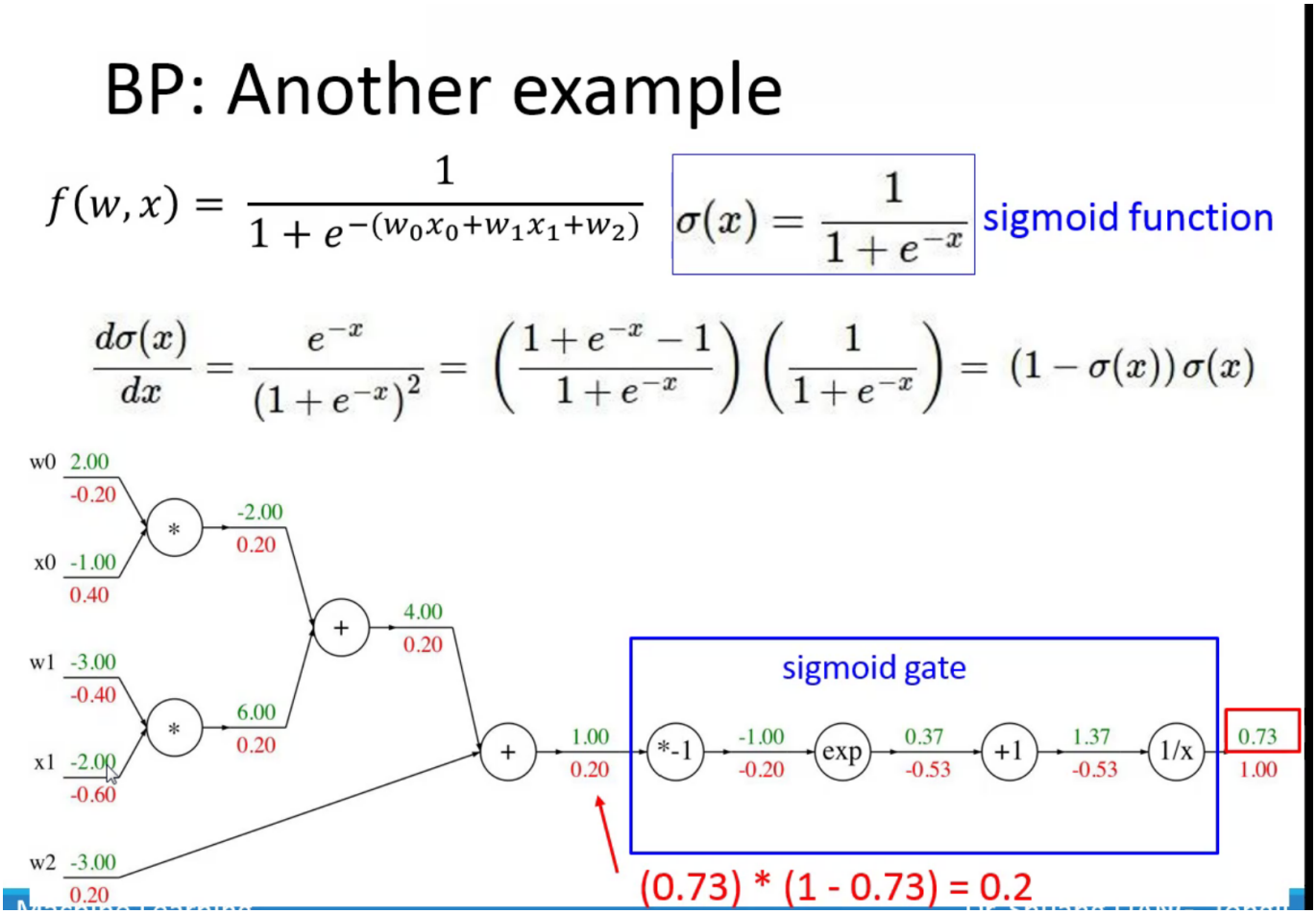


Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Chain



a complicated example



The steps of NN training is:

forward calculate the values

backwards calculate the derivative using **chain principle**.

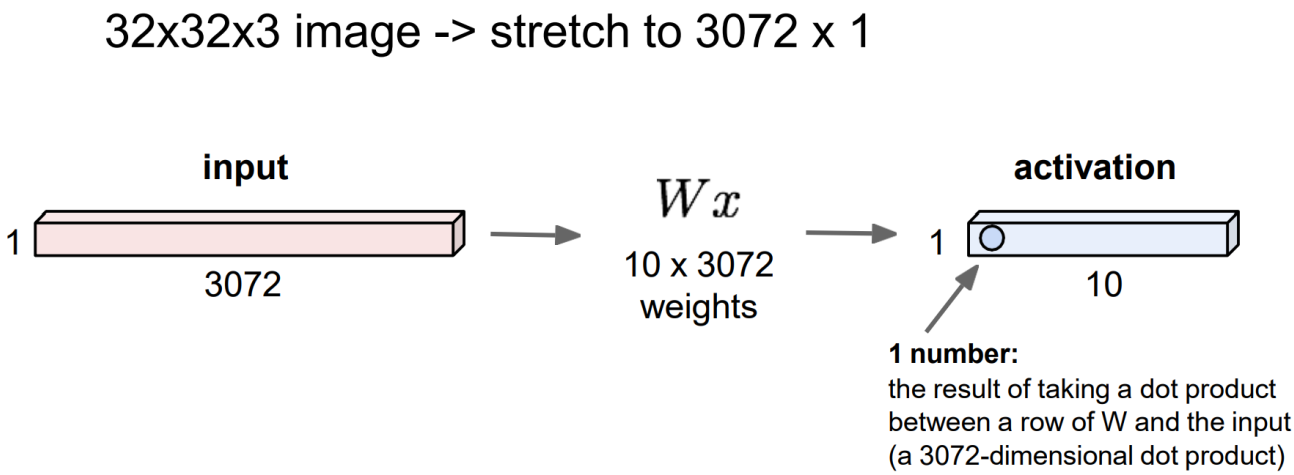
- Why shouldn't we use linear function as activation function of neural network?

The model may be non-linear and **linear functions cannot fit non-linear function**.

b. CNN

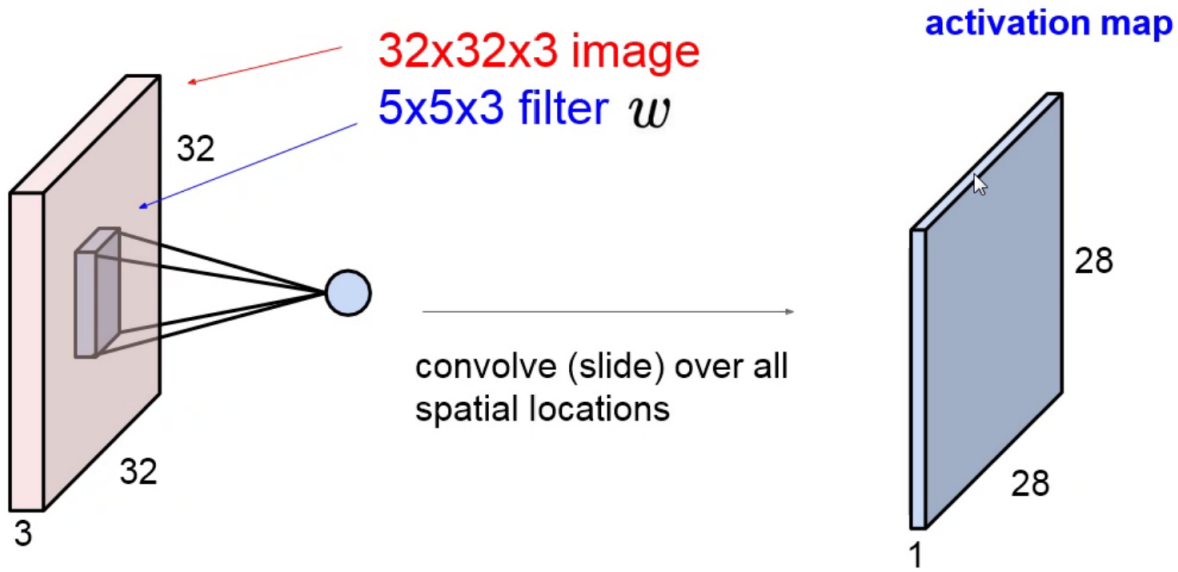
Layers

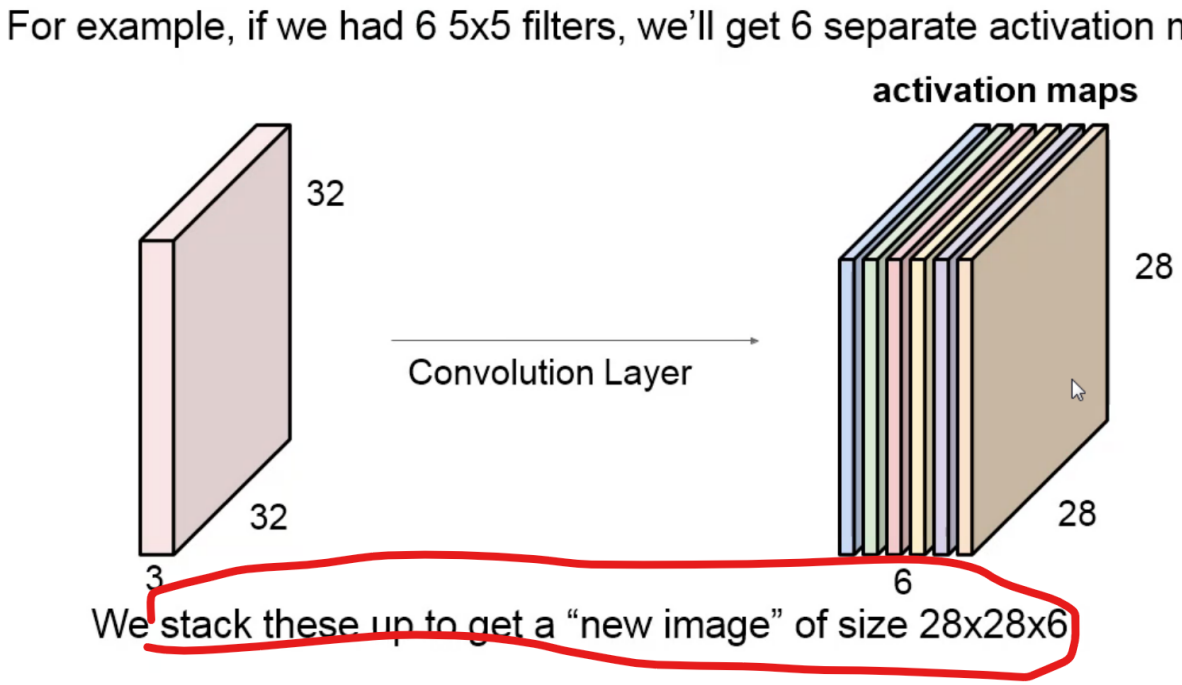
1. FCN(Fully Connected Layer)



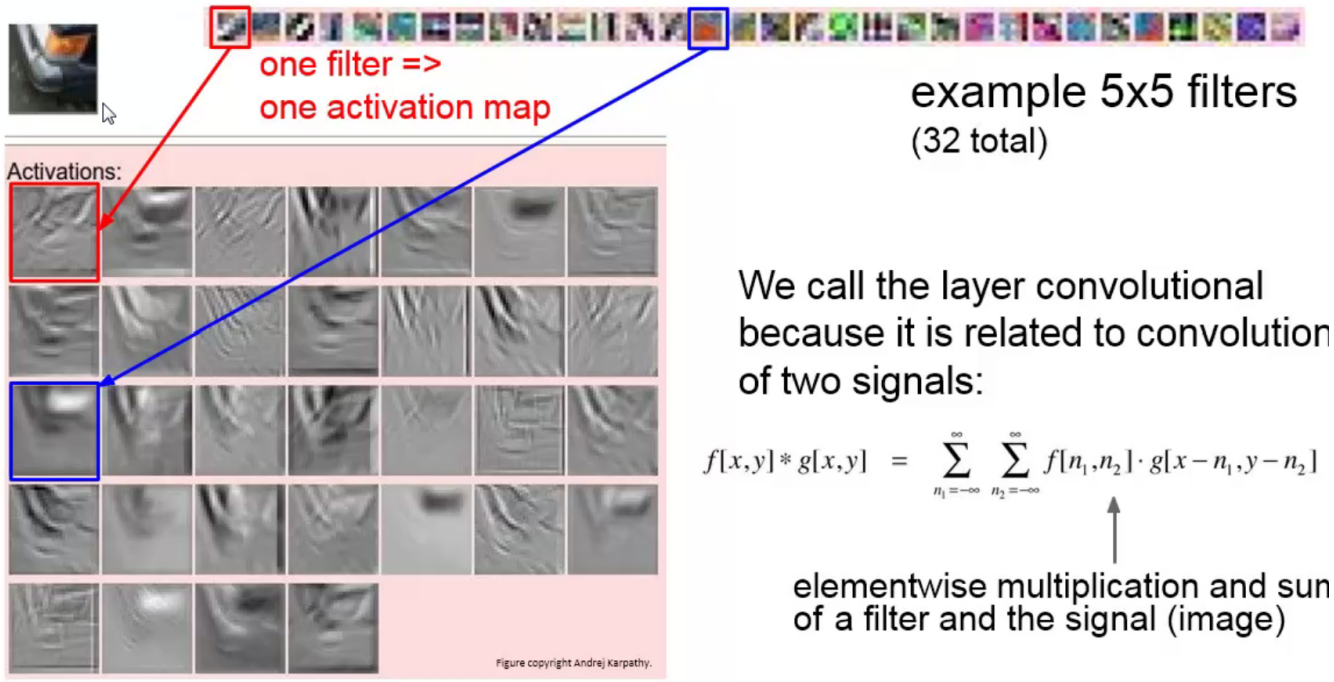
just like the tradition NN

2. Convolution Layer



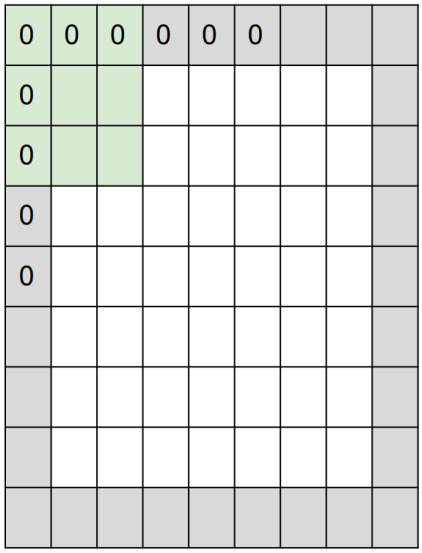


an example of kernel:



The more similar, the more white, the bigger the conv value is

padding



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

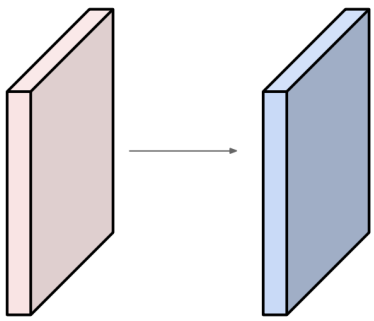
7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

an example:

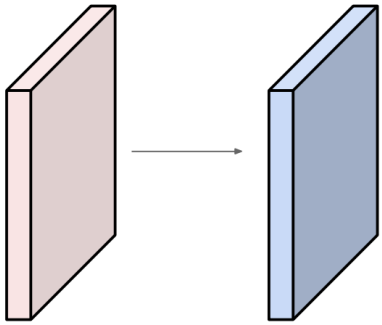
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size:
(32+2*2-5)/1+1 = 32 spatially, so
32x32x10



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?
each filter has 5*5*3 + 1 = 76 params
=> 76*10 = **760** (+1 for bias)

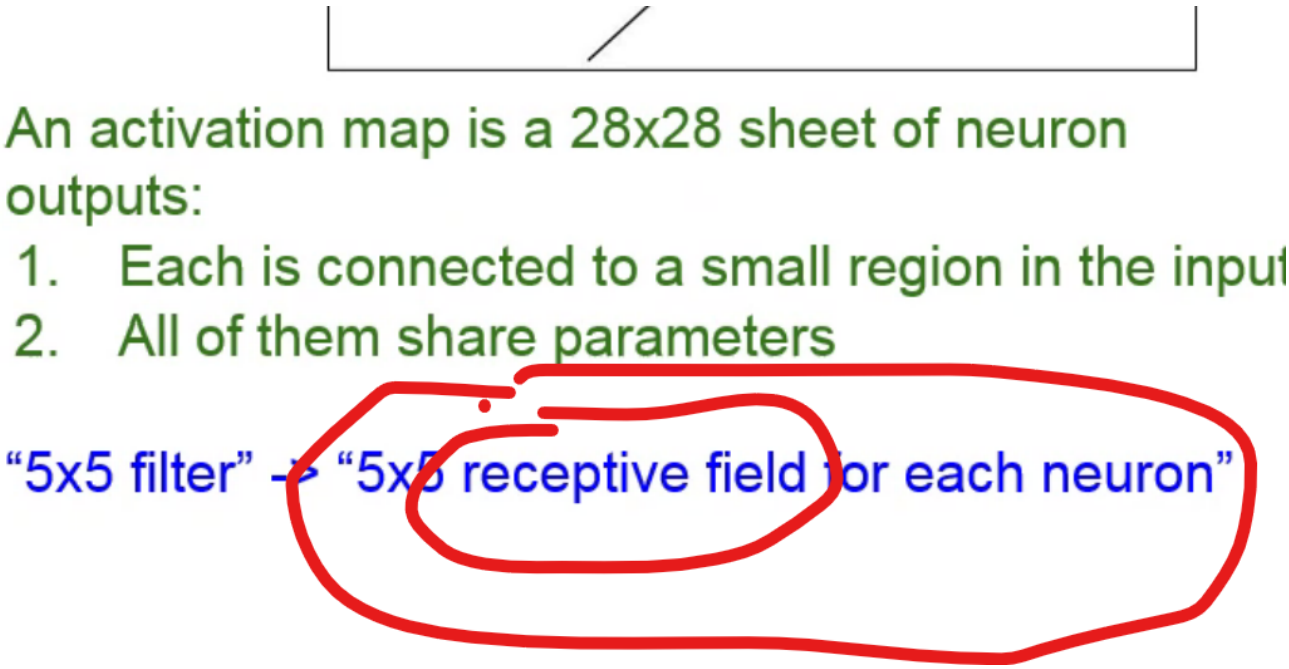


summary

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires 4 hyperparameters:
  - Numbers of filters K
  - Their spatial extent F
  - The stride S
  - The amount of zero padding P

Common settings:
K = (powers of 2, e.g. 32, 64, 128, 512)
- F = 3, S = 1, P = 1
- F = 5, S = 1, P = 2
- F = 5, S = 2, P = ? (whatever fits)
- F = 1, S = 1, P = 0

- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = \frac{W_1 - F + 2P}{S} + 1$
  - $H_2 = \frac{H_1 - F + 2P}{S} + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$



3. Pooling Layer

Common settings:

F = 2, S = 2
F = 3, S = 2

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires 2 hyperparameters:
  - Their spatial extent F
  - The stride S
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = \frac{W_1 - F}{S} + 1$
  - $H_2 = \frac{H_1 - F}{S} + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

Questions

- Why do we introduce a max pooling layer instead of using a convolutional layer with a larger stride?
- What are the advantages of convolutional layers over fully connected layers for image classification?

1. Pooling is FASTER
2. a) reduce the number of parameters; b) easy to learn features, flattening images maybe hard to learn features