Final Exam
Supplementary Content

试卷形式

- 单选题：30分
  - 10道单选题，每题3分
- 简答题：70分
  - 7道大题，每题10分
- 中文
- 闭卷
- 可以带计算器

Topics to Exam

1. ML Basic Concepts/evaluation metrics
2. Bayesian
3. Maximum likelihood estimation
4. Linear Regression/Classification methods
5. Kernel methods
6. Neural networks
7. Decision tree
8. bagging & boosting
9. KNN (no A-KNN)
10. clustering: k-Means
11. Dimensionality Reduction: PCA , LDA
12. RL

ML Basic Concepts/evaluation metrics

- 留出法（hold-out):
  - 直接将数据集划分为两个互斥集合——训练和测试集
  - 训练/测试集划分要尽可能保持数据分布的一致性
  - 一般若干次随机划分、重复实验取平均值
  - 训练/测试样本比例通常为2:1~4:1

ML Basic Concepts/evaluation metrics

- 交叉验证法(cross-validation）：
  - 将数据集分层采样划分为k个大小相似的互斥子集，每次用k-1个子集的并集作为训练集，余下的子集作为测试集，最终返回k个测试结果的均值。
  - 假设数据集D包含m个样本，若令k=m，则得到留一法:
    - 不受随机样本划分方式的影响
    - 结果往往比较准确
    - 当数据集比较大时，计算开销难以忍受，实际中不采用

ML Basic Concepts/evaluation metrics

- 信息检索、Web搜索等场景中经常需要衡量正例被预测出来的比率或者预测出来的正例中正确的比率，此时精确率(Precision)和召回率(Recall)比错误率和精度更适合。
- 统计真实标记和预测结果的组合可以得到“混淆矩阵”

分类结果混淆矩阵

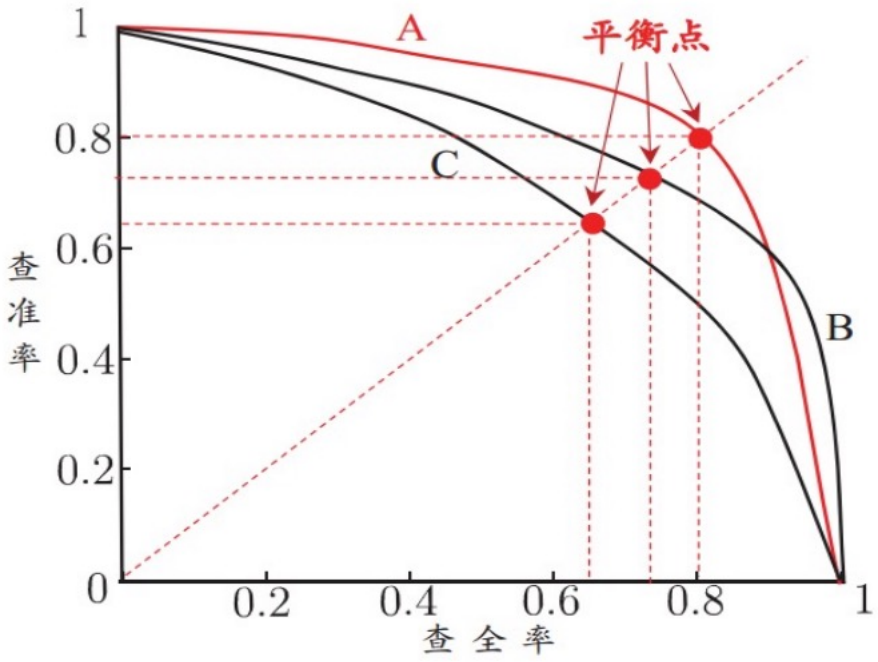
真实情况	预测结果	
	正例	反例
正例	TP (真正例)	FN (假反例)
反例	FP (假正例)	TN (真反例)

精确率：$P = \frac{TP}{TP+FP}$

召回率：$R = \frac{TP}{TP+FN}$

ML Basic Concepts/evaluation metrics

- 根据预测结果按正例可能性大小对样例进行排序，并逐个把样本作为正例进行预测，则可以得到精确率-召回率曲线，简称“P-R曲线”。



P-R曲线与平衡点示意图

平衡点是曲线上在“精确率=召回率”时的取值，可用来用于度量P-R曲线有交叉的分类器性能高低。

ML Basic Concepts/evaluation metrics

比P-R曲线平衡点更常用的是F1度量：

$$F1 = \frac{2PR}{P+R} = \frac{2TP}{\text{样例总数} + TP - TN}$$

比$F1$更一般的形式F_β：

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

$\beta = 1$：标准F1

$\beta > 1$：偏重召回率（逃犯检索系统）

$\beta < 1$：偏重精确率（商品推荐系统）

Bayesian & Maximum likelihood estimation

Bayesian & Maximum likelihood estimation

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

Naïve Bayes Classifier

- Given $\boldsymbol{x} = (x_1, \cdots x_p)^T$
  - Goal is to predict class ω
  - Specifically, we want to find the value of ω that maximizes $P(\omega|\boldsymbol{x}) = P(\omega|x_1, \cdots x_p)$

$$P(\omega|x_1, \cdots x_p) \propto p(x_1, \cdots x_p|\omega)P(\omega)$$

- Independence assumption among features

$$p(x_1, \cdots x_p|\omega) = p(x_1|\omega) \cdots p(x_p|\omega)$$

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Class: $P(\omega_k) = \frac{N_{\omega_k}}{N}$
  - e.g., P(No) = 7/10,
    P(Yes) = 3/10
- For discrete attributes:

$$P(x_i|\omega_k) = \frac{|x_{ik}|}{N_{\omega_k}}$$

  - where $|x_{ik}|$ is number of instances having attribute x_i and belongs to class ω_k
  - Examples:

    P(Status=Married | No) = 4/7
    P(Refund=Yes | Yes)=0

How to Estimate Probabilities from Data?

- For continuous attributes:
  - Discretize the range into bins
    - one ordinal attribute per bin
  - Two-way split: (x < v) or (x > v)
    - choose only one of the two splits as new attribute
  - Probability density estimation:
    - Assume attribute follows a normal distribution
    - Use data to estimate parameters of distribution (e.g., mean and standard deviation)
    - Once probability distribution is known, can use it to estimate the conditional probability $P(x_1|\omega)$

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:

$$P(x_i \mid \omega_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} \exp\left(-\frac{(x_i - \mu_{ij})^2}{2\sigma_{ij}^2}\right)$$

  - One for each (x_i, ω_i) pair

- For (Income, Class=No):
  - If Class=No
    - sample mean = 110
    - sample variance = 2975

$$P(Income = 120 \mid No) = \frac{1}{\sqrt{2\pi}(54.54)} \exp\left(-\frac{(120-110)^2}{2(2975)}\right) = 0.0072$$

Example of Naïve Bayes Classifier

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

naive Bayes Classifier:

P(Refund=Yes|No) = 3/7
P(Refund=No|No) = 4/7
P(Refund=Yes|Yes) = 0
P(Refund=No|Yes) = 1
P(Marital Status=Single|No) = 2/7
P(Marital Status=Divorced|No)=1/7
P(Marital Status=Married|No) = 4/7
P(Marital Status=Single|Yes) = 2/7
P(Marital Status=Divorced|Yes)=1/7
P(Marital Status=Married|Yes) = 0

For taxable income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

- P(X|Class=No) = P(Refund=No|Class=No)
  × P(Married| Class=No)
  × P(Income=120K| Class=No)
  = 4/7 × 4/7 × 0.0072 = 0.0024

- P(X|Class=Yes) = P(Refund=No| Class=Yes)
  × P(Married| Class=Yes)
  × P(Income=120K| Class=Yes)
  = 1 × 0 × 1.2 × 10^{-9} = 0

Since P(X|No)P(No) > P(X|Yes)P(Yes)

Therefore P(No|X) > P(Yes|X)

=> Class = No

Mammals vs. Non-mammals

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

Mammals vs. Non-mammals

Name	Give Birth	Can Fly	Live in Water	Have Legs	Class
human	yes	no	no	yes	mammals
python	no	no	no	no	non-mammals
salmon	no	no	yes	no	non-mammals
whale	yes	no	yes	no	mammals
frog	no	no	sometimes	yes	non-mammals
komodo	no	no	no	yes	non-mammals
bat	yes	yes	no	yes	mammals
pigeon	no	yes	no	yes	non-mammals
cat	yes	no	no	yes	mammals
leopard shark	yes	no	yes	no	non-mammals
turtle	no	no	sometimes	yes	non-mammals
penguin	no	no	sometimes	yes	non-mammals
porcupine	yes	no	no	yes	mammals
eel	no	no	yes	no	non-mammals
salamander	no	no	sometimes	yes	non-mammals
gila monster	no	no	no	yes	non-mammals
platypus	no	no	no	yes	mammals
owl	no	yes	no	yes	non-mammals
dolphin	yes	no	yes	no	mammals
eagle	no	yes	no	yes	non-mammals

Give Birth	Can Fly	Live in Water	Have Legs	Class
yes	no	yes	no	?

A: attributes

M: mammals

N: non-mammals

$$P(M) = \frac{7}{20}$$

$$P(N) = \frac{13}{20}$$

$$P(A|M) = \frac{6}{7} * \frac{6}{7} * \frac{2}{7} * \frac{2}{7}$$

$$P(A|N) = \frac{1}{13} * \frac{10}{13} * \frac{3}{13} * \frac{4}{13}$$

$$P(A|M)P(M) > P(A|N)P(N)$$

$\Rightarrow$ Mammals

Linear Regression&Classification

Linear Regression Model

- Training data: $(\boldsymbol{x}_i, y_i)$
- $f(\boldsymbol{x}) = \sum_{i=1}^{M} w_i x_i = \boldsymbol{w}^T \boldsymbol{x}$
  - $\boldsymbol{w} = [w_1, \cdots, w_M]^T$: unknown parameters or coefficients
  - $\boldsymbol{x}$: Feature vector, the outcome of feature extraction.

- Minimize the **mean-squared error** :

$$J_n = \frac{1}{n} \sum_{i=1}^{n} (y_i - f(\boldsymbol{x}_i))^2$$

- Minimize the **residual sum of squares**

$$RSS(f) = \sum_{i=1}^{n} \left(y_i - f(\boldsymbol{x}_i)\right)^2$$

The MSE Criterion

$$J_n(\boldsymbol{a}) = \sum_{i=1}^{n} (y_i - \boldsymbol{w}^T \boldsymbol{x}_i)^2$$

- MSE Criterion: Minimize the sum of squared differences between $\boldsymbol{w}^T \boldsymbol{x}_i$ and y_i
- Using matrix notation for convenience

$$X = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n], \qquad \boldsymbol{y} = [y_1, \cdots, y_n]^T$$

$$J_n(\boldsymbol{w}) = (\boldsymbol{y} - X^T \boldsymbol{w})^T (\boldsymbol{y} - X^T \boldsymbol{w})$$

- How to optimize it (finds the optimal solution)?

Optimizing the MSE Criterion

- Computing the gradient gives:

$$\nabla J_n = -2X(\boldsymbol{y} - X^T\boldsymbol{w})$$

- Setting the gradient to zero,

$$XX^T\boldsymbol{w} = X\boldsymbol{y}$$

$$\boldsymbol{w} = (XX^T)^{-1}X\boldsymbol{y}$$

- Any problems?
- What is the rank of the matrix XX^T?
- The solution for $\boldsymbol{w}$ can be obtained uniquely if XX^T is non-singular.
- The fitted values at the training inputs are

$$\hat{\boldsymbol{y}} = X^T\boldsymbol{w} = X^T(XX^T)^{-1}X\boldsymbol{y}$$

Ridge Regression

- How to control the size of the coefficients in Regression?

$$\boldsymbol{w}^* = \text{argmin} \sum_{i=1}^{n} \left(y_i - \boldsymbol{x}_i^T \boldsymbol{w}\right)^2 + \lambda \sum_{j=1}^{p} w_j^2$$

local smoothness

weight decay

- Equivalent formulation

$$\boldsymbol{a}^* = \text{argmin} \sum_{i=1}^{n} \left(y_i - \boldsymbol{x}_i^T \boldsymbol{w}\right)^2$$

Subject to $\sum_{j=1}^{p} w_j^2 \leq t$

Lagrange multipliers

Ridge Regression

$$\boldsymbol{w}^* = \text{argmin} \sum_{i=1}^{n} (y_i - \boldsymbol{x}_i^T \boldsymbol{w})^2 + \lambda \sum_{j=1}^{p} w_j^2$$

- Matrix notations:

$$(\boldsymbol{y} - X^T \boldsymbol{w})^T (\boldsymbol{y} - X^T \boldsymbol{w}) + \lambda \boldsymbol{w}^T \boldsymbol{w}$$

- Computing the gradient gives:

$$-2X(\boldsymbol{y} - X^T \boldsymbol{w}) + 2\lambda \boldsymbol{w}$$

- Setting the gradient to zero,

$$(XX^T + \lambda I)\boldsymbol{w} = X\boldsymbol{y}$$

- The unique solution:

$$\boldsymbol{w}^* = (XX^T + \lambda \boldsymbol{I})^{-1} X\boldsymbol{y}$$

$$\boldsymbol{w} *= (XX^T)^{-1} X\boldsymbol{y}$$

LASSO

- Least Absolute Selection and Shrinkage Operator

$$\hat{\boldsymbol{w}} = \operatorname{argmin} \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \boldsymbol{x}_i^T \boldsymbol{w}\right)^2$$

subject to $\sum_{j=1}^{p} |w_j| \leq t$

$$\hat{\boldsymbol{w}} = \operatorname{argmin} \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \boldsymbol{x}_i^T \boldsymbol{w}\right)^2 + \lambda \|\boldsymbol{w}\|_1$$

- Sparse model

A General formulation of Linear classifiers

$$\min_f \left\{ \sum_{i=1}^{n} \ell(f) + \lambda R(f) \right\}$$

Loss function

Regularizer

Square loss: $\ell(f) = (1 - yf)^2$	Ordinary regression
Logistic loss: $\ell(f) = \log\left(1 + e^{-yf}\right)$	Logistic regression
Hinge loss: $\ell(f) = \max[1 - yf, 0]$	SVM

L2-regularizer

L1-regularizer

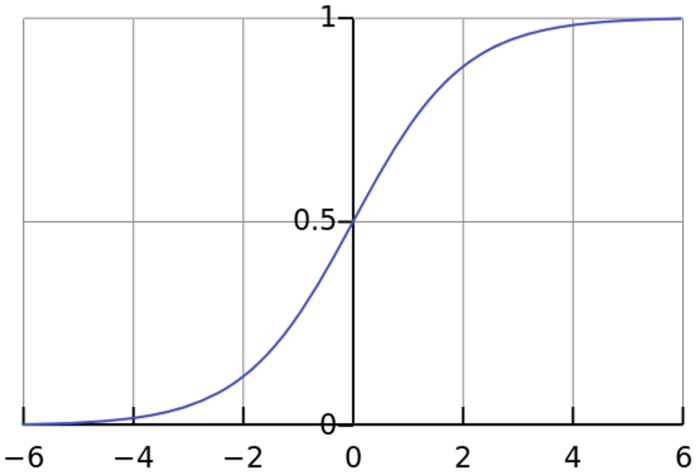
Sigmoid function (logistic function)

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

- It is the cumulative distribution function (CDF) of the standard logistic distribution.
- While the input can have any value from $-\infty$ to $+\infty$, the output takes only values between 0 and 1, and hence is interpretable as probability

$$\sigma: \ R \rightarrow (0,1)$$

- S-shaped



Logistic Regression

- **Logistic Regression (LR)** is a classification model used to describe the relationship between a **categorical** dependent variable and one or several independent variables by estimating **probabilities** using **sigmoid function**.

$$P(y_i = 1|\boldsymbol{x}_i, \boldsymbol{a}) = \sigma(\boldsymbol{a}^T \boldsymbol{x}_i) = \frac{1}{1 + e^{-\boldsymbol{a}^T \boldsymbol{x}_i}}$$

$$P(y_i = -1|\boldsymbol{x}_i, \boldsymbol{a}) = 1 - \sigma(\boldsymbol{a}^T \boldsymbol{x}_i) = 1 - \frac{1}{1 + e^{-\boldsymbol{a}^T \boldsymbol{x}_i}} = \frac{1}{1 + e^{\boldsymbol{a}^T \boldsymbol{x}_i}}$$

$$P(y_i = \pm 1|\boldsymbol{x}_i, \boldsymbol{a}) = \sigma(y_i \boldsymbol{a}^T \boldsymbol{x}_i) = \frac{1}{1 + e^{-y_i \boldsymbol{a}^T \boldsymbol{x}_i}}$$

Maximum Likelihood Estimation for Logistic Regression

$$P(y_i = \pm 1 | \boldsymbol{x}_i, \boldsymbol{a}) = \sigma(y_i \boldsymbol{a}^T \boldsymbol{x}_i) = \frac{1}{1 + e^{-y_i \boldsymbol{a}^T \boldsymbol{x}_i}}$$

$$P(D) = \prod_{i \in I} \sigma(y_i \boldsymbol{a}^T \boldsymbol{x}_i)$$

$$l\big(P(D)\big) = \sum_{i \in I} \log\big(\sigma(y_i \boldsymbol{a}^T \boldsymbol{x}_i)\big) = -\sum_{i \in I} \log\left(1 + e^{-y_i \boldsymbol{a}^T \boldsymbol{x}_i}\right)$$

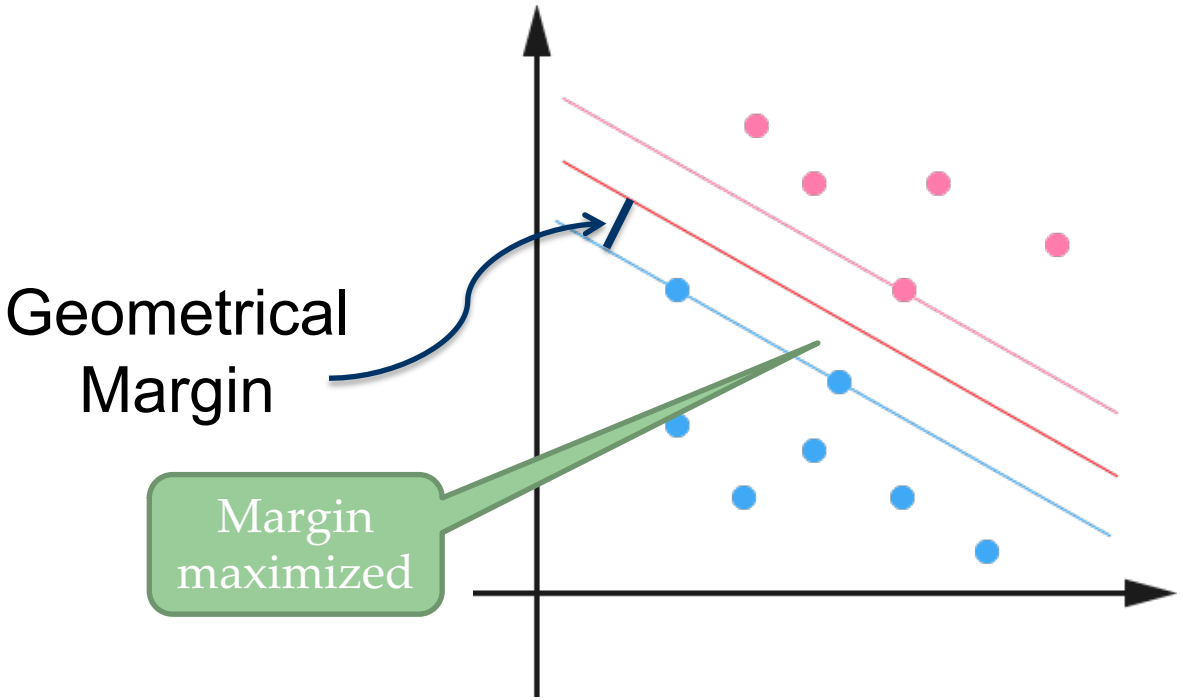
- Logistic Regression:

$$E(\boldsymbol{a}) = \sum_{i \in I} \log\left(1 + e^{-y_i \boldsymbol{a}^T \boldsymbol{x}_i}\right)$$

SVM

- Geometrical margin is a value uniquely determined by the position of the hyperplane
- If we scale $\boldsymbol{w}$, γ will not change as long as the hyperplane is kept fixed

$$\gamma = y\frac{\boldsymbol{w}^T\boldsymbol{x} + b}{\|\boldsymbol{w}\|}$$



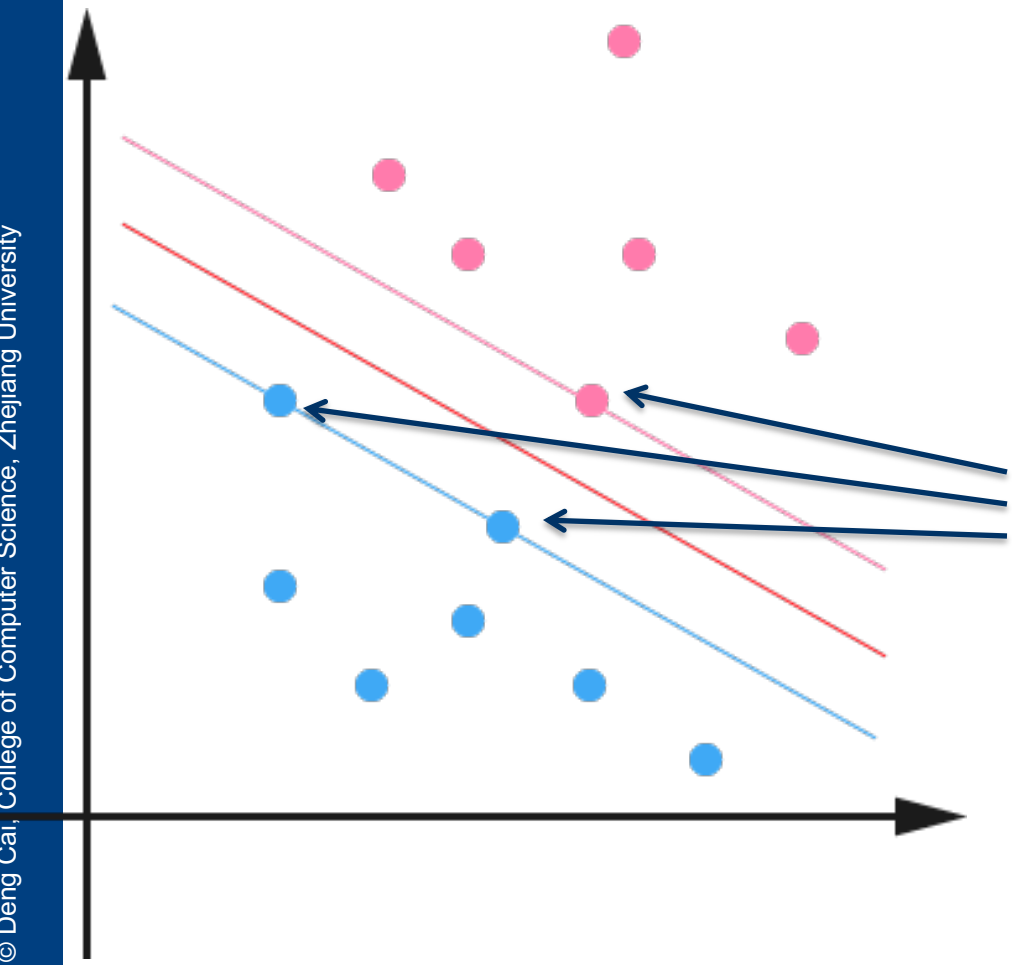
Maximum Margin Classifier

$$\min_{\boldsymbol{w},b} \frac{1}{2} \|\boldsymbol{w}\|^2$$

$$y_i(\boldsymbol{w}^T \boldsymbol{x}_i + b) \geq 1$$

Square and a coefficient $\frac{1}{2}$ are added for the convenience of the derivation of optimization, and the minimizer of $\|\boldsymbol{w}\|$ and $\frac{1}{2}\|\boldsymbol{w}\|^2$ is obviously the same.

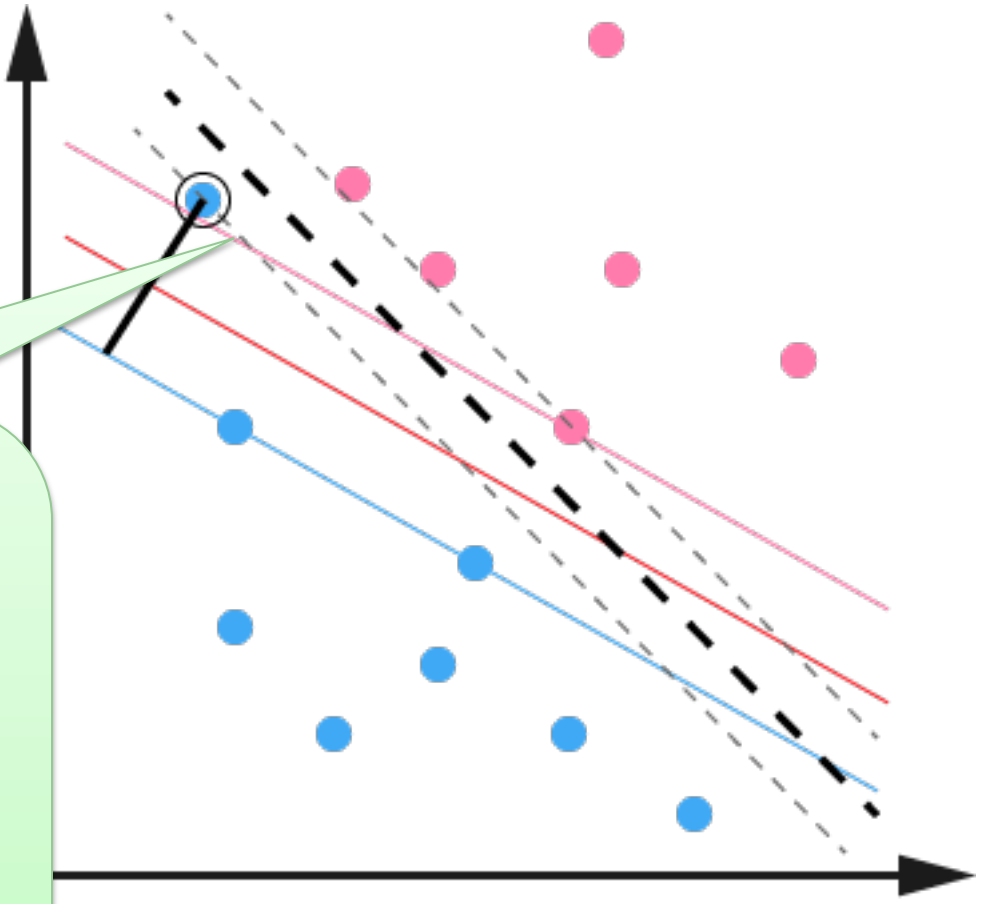
Support Vector Machine



Hyper plane of maximum margin is *support*ed by those points (vectors) on the margin. Those are called Support Vectors. Non-support vectors can move freely without affecting the position of the hyperplane as long as they don't exceed the margin.

Weakness of the Original Model

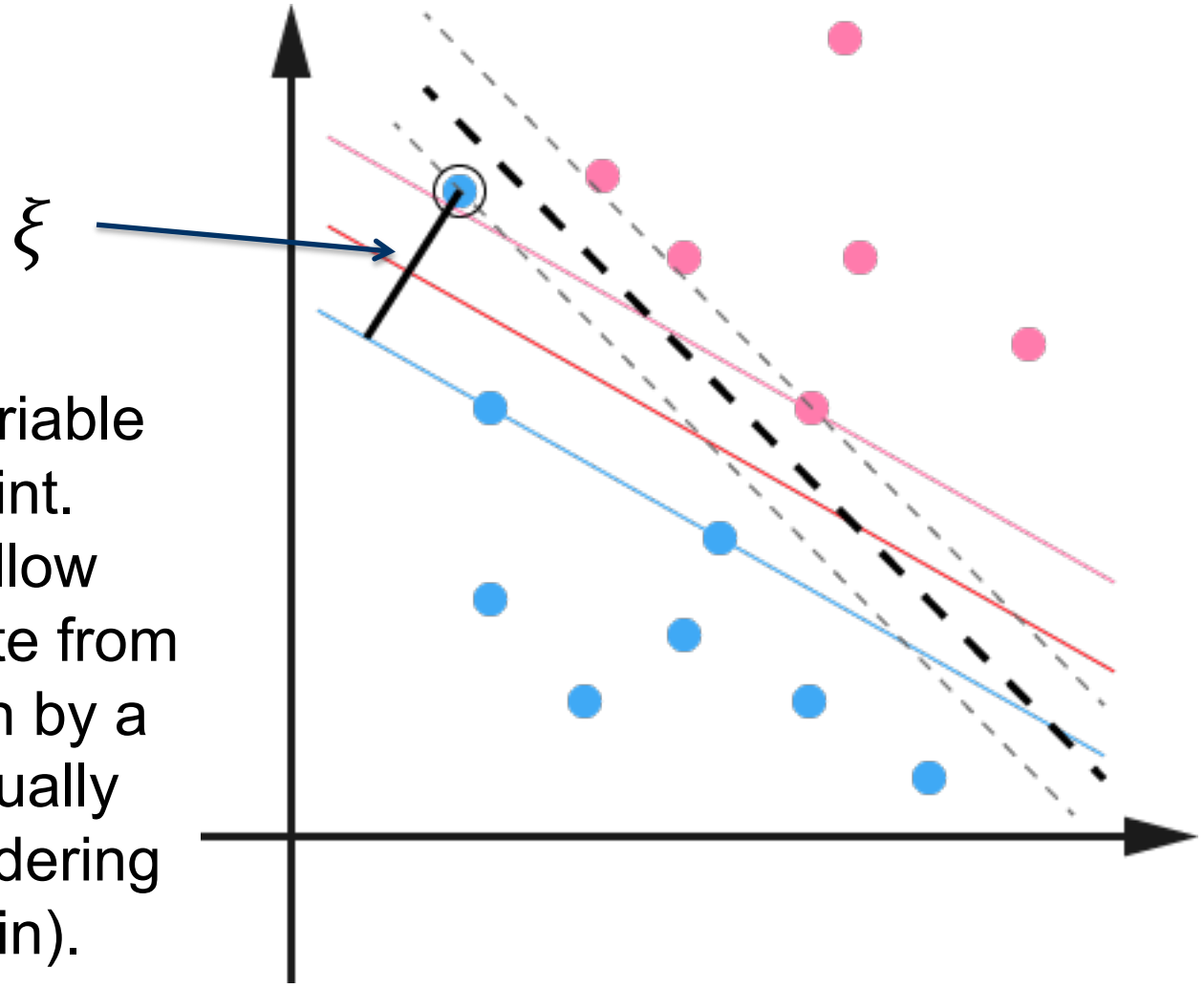
$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2$$
$$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1$$



When an outlier appear, the optimal hyperplane may be pushed far away from its original/correct place. The resultant margin will also be smaller than before.

- Red Solid: the original hyperplane
- Dark dashed: the new hyperplane

Slack Variables



Assign a slack variable ξ to each data point. That means we allow the point to deviate from the correct margin by a distance of ξ (Actually $\|\boldsymbol{w}\|\xi$ when considering geometrical margin).

New Objective Function

Slack variables can't be arbitrarily large, we want to minimize the sum of all slack variables

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$y(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

- Soft-Margin SVM

New Objective Function

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$y(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

- We would pay a cost of the objective function being increased by $C\ \xi_i$. The parameter C controls the relative weighting between the twin goals of making the $\|\boldsymbol{w}\|^2$small (makes the margin large) and of ensuring that most examples have functional margin at least 1.

How to optimize

- SVM optimization problem:

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$
$$y(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1 - \xi_i$$
$$\xi_i \geq 0$$

- Constrained optimization problem
- 考试不需要解优化问题
- KKT不考

Generalized Linear Function & Kernel Methods

Junbo Zhao(赵俊博)

College of Computer Science
Zhejiang University

j.zhao@zju.edu.cn

Generalized Discriminant Function

- A *generalized linear discriminant* function can be written as,

$$g(\mathbf{x}) = \sum_{i=1}^{\hat{d}} a_i y_i(\mathbf{x})$$

Dimensionality of the augmented feature space.

Setting $y_i(x)$to be monomials results in polynomial discriminant functions

Weights in the augmented feature space. Note that the function is linear in *a*.

- Equivalently,

$$g(\mathbf{x}) = \mathbf{a}^t \mathbf{y}$$

$$\mathbf{a} = [a_1, a_2, \ldots, a_{\hat{d}}]^t \qquad \mathbf{y} = [y_1(x), y_2(x), \ldots, y_{\hat{d}}(x)]^t$$

also called the augmented feature vector.

Phi Function

- The discriminant function $g(x)$ is not linear in x, but is linear in y.
- The mapping $y = [y_1(x), y_2(x),..., y_{\hat{d}}(x)]^t$ is taking a d-dimensional vector x and mapping it to a $\hat{d}$ – dimensional space. The mapping y is called the phi-function: $\boldsymbol{x} \rightarrow \boldsymbol{\phi}(\boldsymbol{x})$
- When the input patterns x are non-linearly separable in the input space, mapping them using the *right* phi-function maps them to a space where the patterns are linearly separable.
- Unfortunately, in reality, this is often hard, because of that usually, this mapping of y is not tractable. To have an explicit form of it is not practical. And curse of dimensionality...

Kernel Methods

$$\boldsymbol{x} \rightarrow \phi(\boldsymbol{x}) \qquad \phi(\boldsymbol{x}) = \boldsymbol{x}$$

$$k(\boldsymbol{x}, \boldsymbol{x}') \rightarrow \phi(\boldsymbol{x})^T \phi(\boldsymbol{x}') \qquad k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T \boldsymbol{x}'$$

- Kernel is a symmetric function of its arguments, $k(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}', \boldsymbol{x})$
- Let $k(\boldsymbol{x}, \boldsymbol{x}') \geq 0$ be some measure of similarity between objects $\boldsymbol{x}, \boldsymbol{x}' \in \chi$, where χ is some abstract space; we will call k a kernel function.
- The kernel concept was introduced into the field of pattern recognition by Aizerman et al. (1964)
- Re-introduced into machine learning in the context of large margin classifiers by Boser et al. (1992), SVM
- If we have an algorithm formulated in such a way that the input vector $\boldsymbol{x}$ enters only in the form of scalar products, then we can replace that scalar product with some other choice of kernel, so called the kernel trick.

Dual Representations: Ridge Regression

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N}\{\boldsymbol{w}^T\boldsymbol{x}_i - t_i\}^2 + \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$\boldsymbol{x} \to \phi(\boldsymbol{x})$$

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N}\{\boldsymbol{w}^T\phi(\boldsymbol{x}_i) - t_i\}^2 + \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$\frac{\partial J}{\partial \boldsymbol{w}} = \sum_{i=1}^{N}\{\boldsymbol{w}^T\phi(\boldsymbol{x}_i) - t_i\}\phi(\boldsymbol{x}_i) + \lambda\boldsymbol{w} = 0$$

$$\boldsymbol{w} = -\frac{1}{\lambda}\sum_{i=1}^{N}\{\boldsymbol{w}^T\phi(\boldsymbol{x}_i) - t_i\}\phi(\boldsymbol{x}_i) = \sum_{i=1}^{N} a_i\phi(\boldsymbol{x}_i) \quad \boxed{= \Phi^T\boldsymbol{a}}$$

$$\Phi^T = \{\phi(\boldsymbol{x}_1), \cdots, \phi(\boldsymbol{x}_N)\} \qquad \boldsymbol{a} = (a_1, \cdots, a_N)^T$$

"Dual weight/param"

Dual Representations : Ridge Regression

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N}\{\boldsymbol{w}^T\phi(\boldsymbol{x}_i) - t_i\}^2 + \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$\boldsymbol{b} = [\boldsymbol{w}^T\phi(\boldsymbol{x}_1), \cdots, \boldsymbol{w}^T\phi(\boldsymbol{x}_N)]^T \qquad \boldsymbol{t} = [t_1, \cdots, t_N]^T$$

$$J(\boldsymbol{w}) = \frac{1}{2}(\boldsymbol{b} - \boldsymbol{t})^T(\boldsymbol{b} - \boldsymbol{t}) + \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$J(\boldsymbol{w}) = \frac{1}{2}(\boldsymbol{b}^T\boldsymbol{b} - \boldsymbol{b}^T\boldsymbol{t} - \boldsymbol{t}^T\boldsymbol{b} + \boldsymbol{t}^T\boldsymbol{t}) + \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$\boldsymbol{b}^T = [\boldsymbol{w}^T\phi(\boldsymbol{x}_1), \cdots, \boldsymbol{w}^T\phi(\boldsymbol{x}_N)] \quad = \boldsymbol{w}^T[\phi(\boldsymbol{x}_1), \cdots, \phi(\boldsymbol{x}_N)]$$

$$\boldsymbol{w} = \Phi^T\boldsymbol{a} \qquad \Phi^T = \{\phi(\boldsymbol{x}_1), \cdots, \phi(\boldsymbol{x}_N)\}$$

$$\boldsymbol{b}^T = \boldsymbol{a}^T\Phi\Phi^T$$

$$J(\boldsymbol{a}) = \frac{1}{2}\boldsymbol{a}^T\Phi\Phi^T\Phi\Phi^T\boldsymbol{a} - \boldsymbol{a}^T\Phi\Phi^T\boldsymbol{t} + \frac{1}{2}\boldsymbol{t}^T\boldsymbol{t} + \frac{\lambda}{2}\boldsymbol{a}^T\Phi\Phi^T\boldsymbol{a}$$

Dual Representations : Ridge Regression

$$J(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N}\{\boldsymbol{w}^T\phi(\boldsymbol{x}_i) - t_i\}^2 + \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w}$$

$$J(\boldsymbol{a}) = \frac{1}{2}\boldsymbol{a}^T\Phi\Phi^T\Phi\Phi^T\boldsymbol{a} - \boldsymbol{a}^T\Phi\Phi^T\boldsymbol{t} + \frac{1}{2}\boldsymbol{t}^T\boldsymbol{t} + \frac{\lambda}{2}\boldsymbol{a}^T\Phi\Phi^T\boldsymbol{a}$$

Define Gram matrix $K = \Phi\Phi^T$

$N \times N$ symmetric matrix

$$\Phi^T = \{\phi(\boldsymbol{x}_1), \cdots, \phi(\boldsymbol{x}_N)\}$$

$$K_{nm} = \phi(\boldsymbol{x}_n)^T\phi(\boldsymbol{x}_m) = k(\boldsymbol{x}_n, \boldsymbol{x}_m)$$

$$k(\boldsymbol{x}, \boldsymbol{x}') \rightarrow \boldsymbol{\phi}(\boldsymbol{x})^T\boldsymbol{\phi}(\boldsymbol{x}')$$

Dual Representations : Ridge Regression

$$J(\boldsymbol{a}) = \frac{1}{2}\boldsymbol{a}^T\Phi\Phi^T\Phi\Phi^T\boldsymbol{a} - \boldsymbol{a}^T\Phi\Phi^T\boldsymbol{t} + \frac{1}{2}\boldsymbol{t}^T\boldsymbol{t} + \frac{\lambda}{2}\boldsymbol{a}^T\Phi\Phi^T\boldsymbol{a}$$

Define Gram matrix $K = \Phi\Phi^T$

$$J(\boldsymbol{a}) = \frac{1}{2}\boldsymbol{a}^T KK\boldsymbol{a} - \boldsymbol{a}^T K\boldsymbol{t} + \frac{1}{2}\boldsymbol{t}^T\boldsymbol{t} + \frac{\lambda}{2}\boldsymbol{a}^T K\boldsymbol{a}$$

$$\boldsymbol{a}^* = (K + \lambda\boldsymbol{I})^{-1}\boldsymbol{t}$$

$$y(\boldsymbol{x}) = \boldsymbol{w}^T\phi(\boldsymbol{x}) = \boldsymbol{a}^T\Phi\phi(\boldsymbol{x}) = \{\Phi\phi(\boldsymbol{x})\}^T\boldsymbol{a}$$

$$\boldsymbol{w} = \Phi^T\boldsymbol{a}$$

$$= \{\mathrm{k}(\boldsymbol{x})\}^T(K + \lambda\boldsymbol{I})^{-1}\boldsymbol{t}$$

$\mathrm{k}(\boldsymbol{x}) = \Phi\phi(\boldsymbol{x})$ is a N dimensional vector.

$$\Phi^T = \{\phi(\boldsymbol{x}_1), \cdots, \phi(\boldsymbol{x}_N)\} \qquad k_n(\boldsymbol{x}) = \phi(\boldsymbol{x}_n)^T\phi(\boldsymbol{x}) = k(\boldsymbol{x}_n, \boldsymbol{x})$$

Dual Representations : Ridge Regression

The previous is honestly an easy-to-follow derivation of the du
form of ridge regression. Here is its original definition:

$$C = \frac{1}{2}\sum_i (y_i - \mathbf{w}^T\mathbf{x}_i)^2 + \frac{1}{2}\lambda||\mathbf{w}||^2$$

$$\text{minimize} - \mathbf{w}, \boldsymbol{\xi} \qquad \mathcal{L}_P = \sum_i \xi_i^2$$

$$\text{subject to} \qquad y_i - \mathbf{w}^T\Phi_i = \xi_i \;\; \forall i$$
$$||\mathbf{w}|| \leq B$$

Homework…

Dual Representations : Ridge Regression

- Thus we see that the dual formulation allows the solution to the least-squares problem to be expressed entirely in terms of the kernel function $k(\boldsymbol{x}_n, \boldsymbol{x}_m)$.
- The advantage of the dual formulation, as we shall see, is that it is expressed entirely in terms of the kernel function $k(\boldsymbol{x}_n, \boldsymbol{x}_m)$. We can therefore work directly in terms of kernels and avoid the explicit introduction of the feature vector $\phi(\boldsymbol{x}_n)$, which allows us implicitly to use feature spaces of high, even infinite, dimensionality.
- Remember: inner-product, similarity, kernels are somewhat IDENTICAL.

Maximum Margin Classifier

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2$$

$$y_i(\boldsymbol{w}^T\boldsymbol{x}_i + b) \geq 1$$

$$\boldsymbol{x} \rightarrow \phi(\boldsymbol{x})$$

$$y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b) \geq 1$$

Lagrange multipliers $a_i \geq 0$, $i = 1,2,\cdots,N$

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{a}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{N} a_i\{y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b) - 1\}$$

Setting the derivates of L *w.r.t* $\boldsymbol{w}$ and b equal to zero, we can g

$$\boldsymbol{w} = \sum_{i=1}^{N} a_i\, y_i \phi(\boldsymbol{x}_i) \qquad \sum_{i=1}^{N} a_i\, y_i = 0$$

Maximum Margin Classifier

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{a}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{N} a_i\{y_i(\boldsymbol{w}^T\phi(\boldsymbol{x}_i) + b) - 1\}$$

$$\boldsymbol{w} = \sum_{i=1}^{N} a_i\, y_i\phi(\boldsymbol{x}_i) \qquad \sum_{i=1}^{N} a_i\, y_i = 0$$

$$\frac{1}{2}\|\boldsymbol{w}\|^2 = \frac{1}{2}\sum_{i=1}^{N} a_i\, y_i\phi^T(\boldsymbol{x}_i)\sum_{j=1}^{N} a_j\, y_j\phi(\boldsymbol{x}_j)$$

$$\text{second\_term} = -\sum_{i=1}^{N} a_i y_i \boldsymbol{w}^T\phi(\boldsymbol{x}_i) - \sum_{i=1}^{N} a_i y_i b + \sum_{i=1}^{N} a_i$$

$$= -\sum_{i=1}^{N} a_i y_i \sum_{j=1}^{N} a_j\, y_j\phi^T(\boldsymbol{x}_j)\phi(\boldsymbol{x}_i) + \sum_{i=1}^{N} a_i$$

Maximum Margin Classifier

$$\mathcal{L}(\boldsymbol{a}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_i\, a_j y_i y_j \phi^T(\boldsymbol{x}_i)\phi(\boldsymbol{x}_j) + \sum_{i=1}^{N} a_i$$

$$\mathcal{L}(\boldsymbol{a}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} a_i\, a_j y_i y_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{i=1}^{N} a_i$$

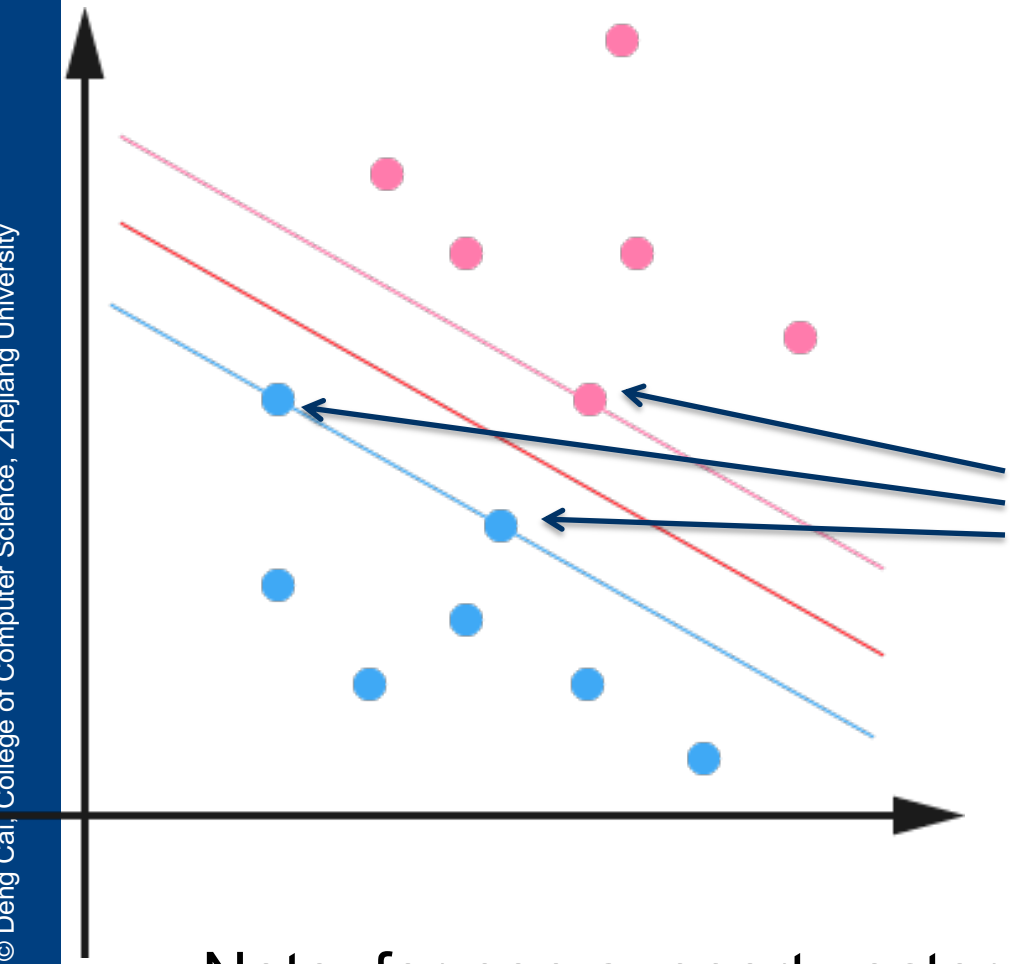
$$a_i \geq 0 \qquad \sum_{i=1}^{N} a_i\, y_i = 0$$

Note that this is the dual representation of the original problem. This problem has *N* variables. If we get ***a***, the hyperplane is:

$$f(\boldsymbol{x}) = \boldsymbol{w}^T\phi(\boldsymbol{x}) + b \quad \boldsymbol{w} = \sum_{i=1}^{N} a_i\, y_i \phi(\boldsymbol{x}_i)$$

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} a_i\, y_i \phi(\boldsymbol{x}_i)^T\phi(\boldsymbol{x}) + b = \sum_{i=1}^{N} a_i\, y_i k(\boldsymbol{x}_i, \boldsymbol{x}) + b$$

Support Vector Machine



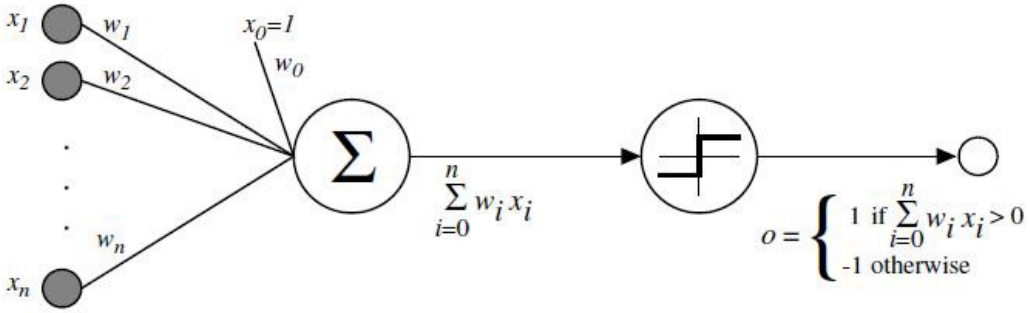
Hyper plane of maximum margin is *support*ed by those points (vectors) on the margin. Those are called Support Vectors. Non-support vectors can move freely without affecting the position of the hyperplane as long as they don't exceed the margin.

Note: for non-support vectors, the corresponding α_i is zero

Common Use Kernels

- Examples
  - Linear kernels $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T \boldsymbol{x}'$
  - Polynomial kernels $k(\boldsymbol{x}, \boldsymbol{x}') = (\boldsymbol{x}^T \boldsymbol{x}' + 1)^d$
  - RBF kernels $k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2\sigma^2}\right)$

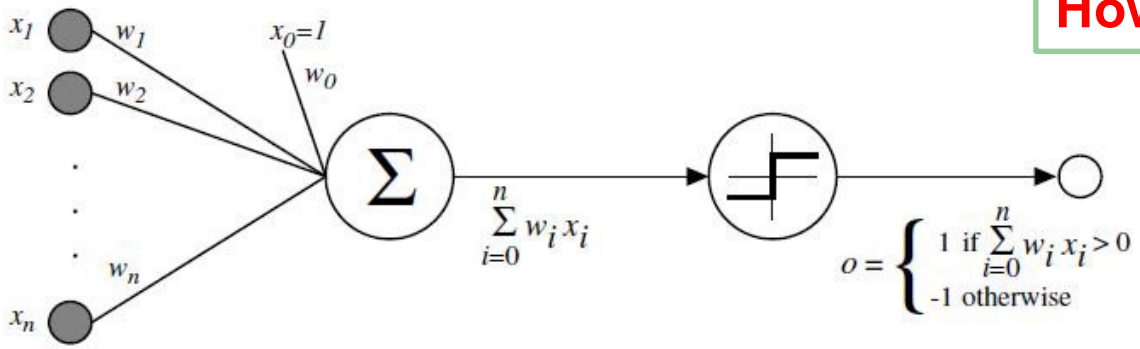
Perceptron



- In 1957, **Frank Rosenblatt** in Cornell University invented the **Perceptron** model. It is:
  - The first self-organizing and self-learning mathematic model.
  - The first algorithm defining Neural Network precisely.
  - The ancestor of many new Neural Network models.

Frank Rosenblatt
(1928-1971)

Perceptron



How to Learn?

- If the model predicts $\boldsymbol{x}$ correct, do nothing
- If the model predicts $\boldsymbol{x}$ wrong, multiply $\boldsymbol{x}$ by its label, and let

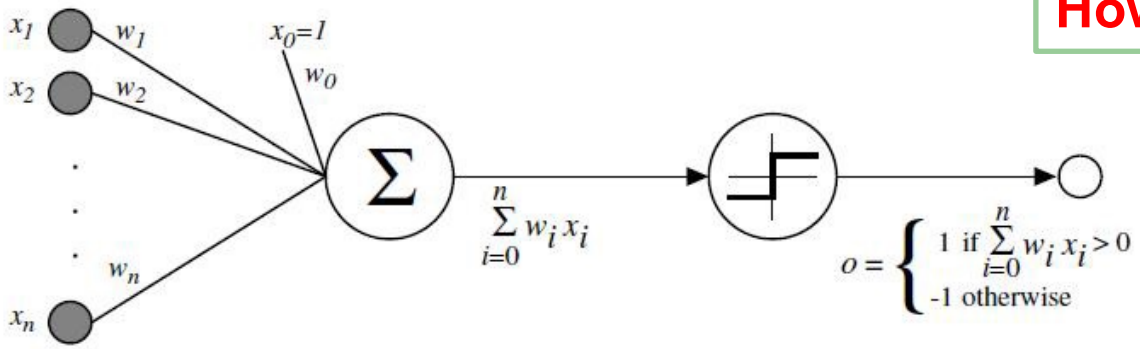
$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} + \boldsymbol{x}y$$

(update only on error. A mistake-driven algorithm)

- Seems correct buy why?
- Hint: If w is a solution, for any $\boldsymbol{x}$ in the training set, we have

$$\boldsymbol{w}^T \boldsymbol{x}y > 0$$

Perceptron

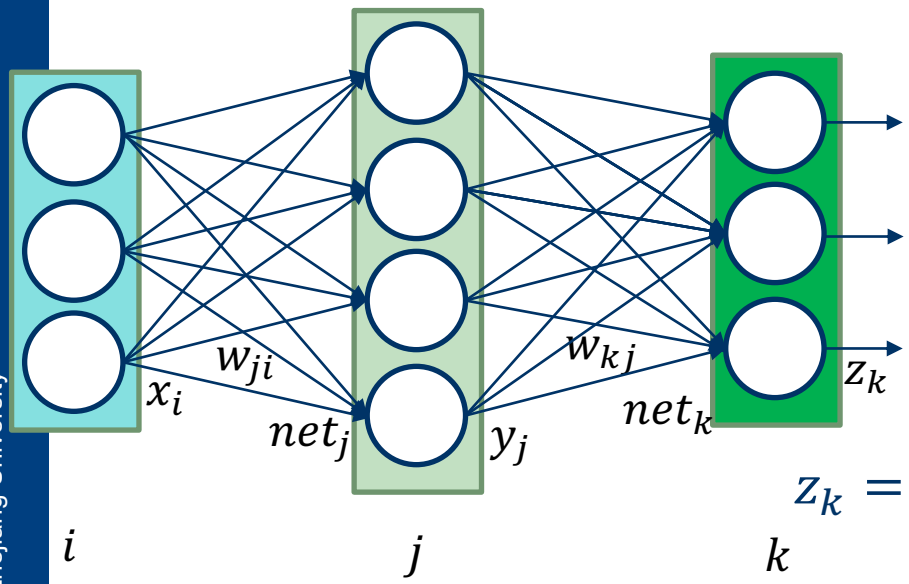


How to Learn?

$$L(\omega, b) = \sum_{x_i \in M} -y_i(\omega x_i + b)$$

$$\min_{w,b} L(\omega, b) = \sum_{x_i \in M} -y_i(\omega x_i + b)$$

Backpropagation Algorithm



$$z_k = f\left(\sum_{j=1}^{n_H} w_{kj}\, f\left(\sum_{i=1}^{d} w_{ji}x_i + w_{j0}\right) + w_{k0}\right)$$

$$k = 1, \cdots, c$$

$$\Delta w_{mn} = -\eta \frac{\partial J}{\partial w_{mn}}$$

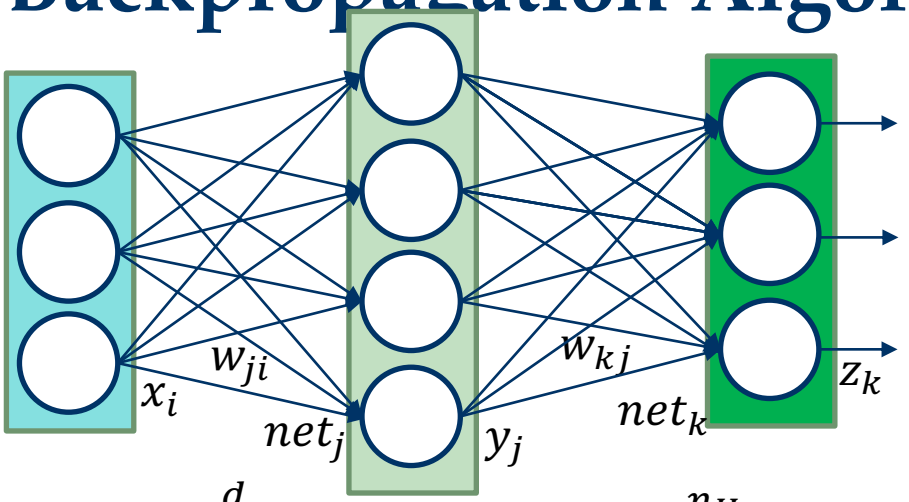
$$\Delta \boldsymbol{w} = -\eta \nabla J = -\eta \frac{\partial J}{\partial \boldsymbol{w}}$$

Where η is the learning rate which indicates the relative size of the change in weights

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \Delta \boldsymbol{w}^{(t)}$$

where t indexes the particular pattern presentation

Backpropagation Algorithm



$$J = \frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2$$

$$net_j = \sum_{j=0}^{d} x_i w_{ji} \qquad net_k = \sum_{j=0}^{n_H} y_j w_{kj}$$

$$\frac{\partial J}{\partial w_{kj}} = \frac{\partial J}{\partial z_k}\cdot\frac{\partial z_k}{\partial net_k}\cdot\frac{\partial net_k}{\partial w_{kj}} = (z_k - t_k)\cdot f'(net_k)\cdot y_j$$

$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y_j}\cdot\frac{\partial y_j}{\partial net_j}\frac{\partial net_j}{\partial w_{ji}}$$

$$= \left(\sum_{k=1}^{c}(z_k - t_k)\cdot f'(net_k)\cdot w_{kj}\right)\cdot f'(net_j)\cdot x_i$$

Activation Function

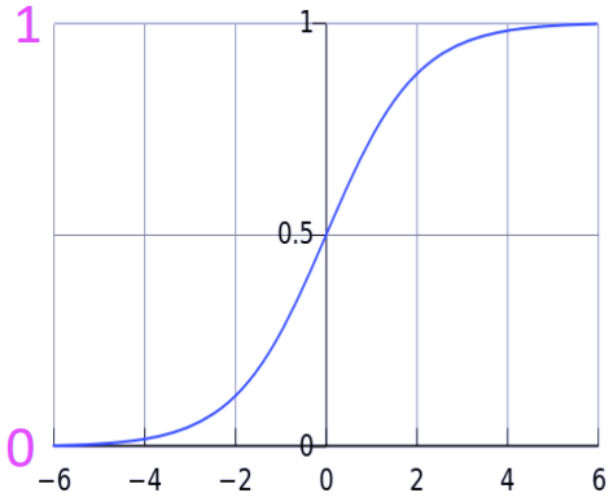
- Activation Function *f*
  - Must be non-linear (otherwise, 3-layer network is just a linear discriminant) and saturate (have max and min value) to keep weights and activation functions bounded
  - Activation function and its derivative must be continuous and smooth; optionally monotonic
  - Choice may depend on the problem. Eg. Gaussian activation if the data comes from a mixture of Gaussians
  - Eg: sigmoid (most popular), polynomial, tanh

- Parameters of activation function (e.g. Sigmoid)
  - Centered at 0, odd function f(-net) = -f(net) (anti-symmetric); leads to faster learning
  - Depend on the range of the input values

Activation Function

- sigmoid function (logistic function)

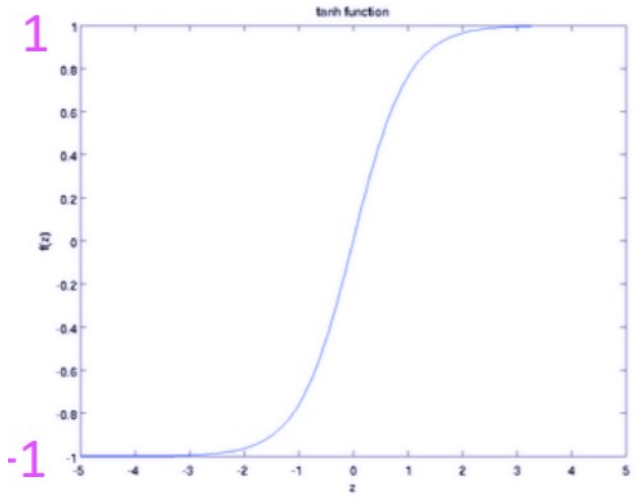
$$\sigma(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{x}}}$$



- tanh function

$$t(\boldsymbol{x}) = \frac{e^{\boldsymbol{x}} - e^{-\boldsymbol{x}}}{e^{\boldsymbol{x}} + e^{-\boldsymbol{x}}}$$

$$t(\boldsymbol{x}) = 2\sigma(2\boldsymbol{x}) - 1$$

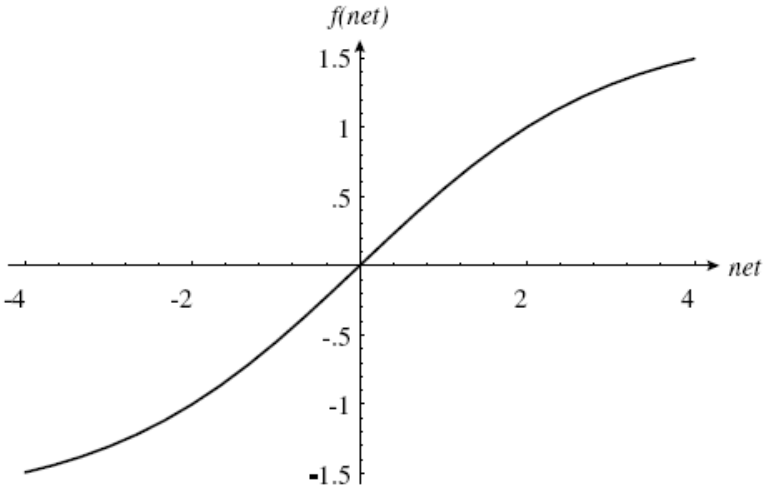


- tanh is just a rescaled and shifted sigmoid.

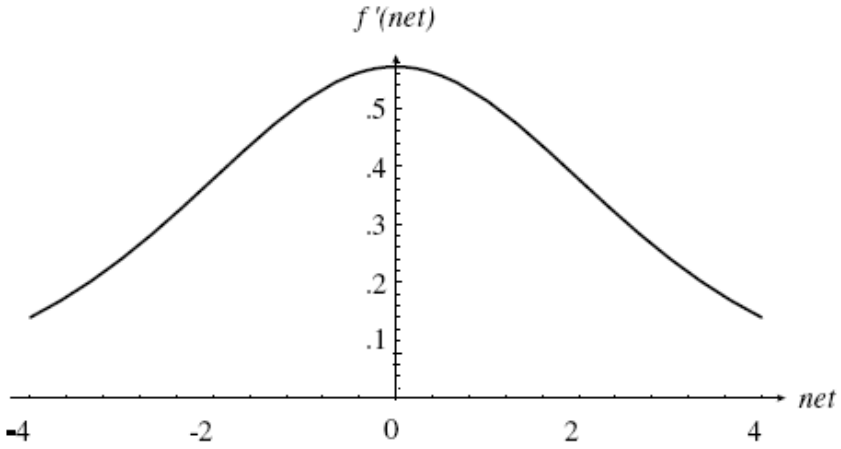
Activation Function

$$f(net) = a \tanh(b\ net)$$

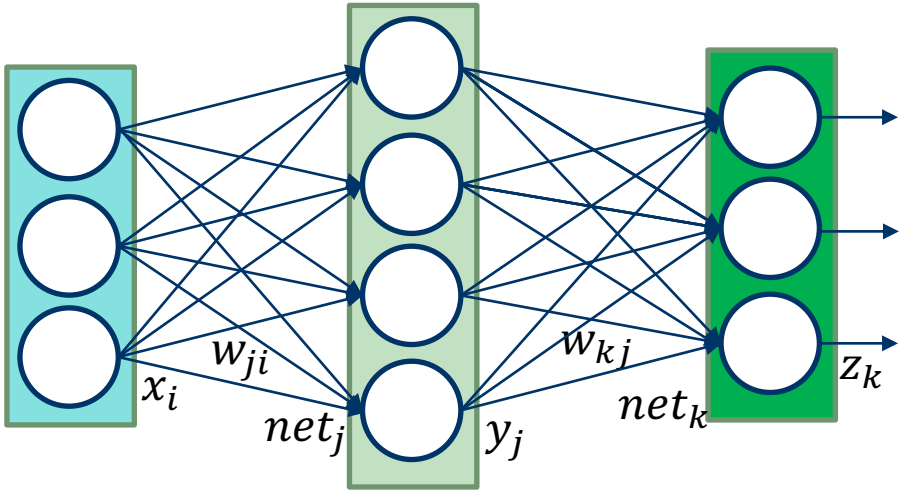
The anti-symmetric sigmoid function:
f(-x) = -f(x).
a = 1.716, b = 2/3.



First order derivative



Backpropagation Algorithm

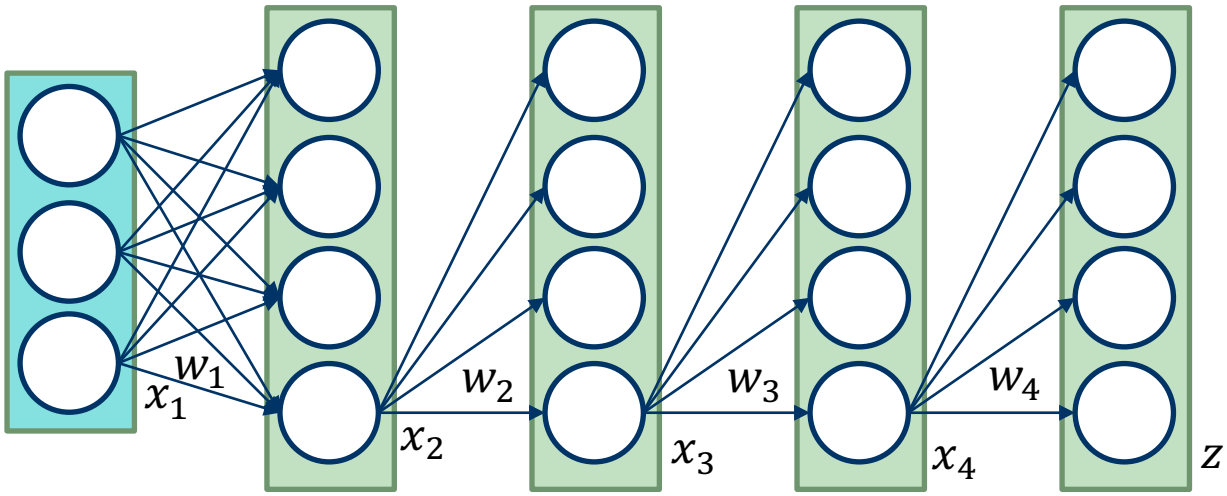


$$\frac{\partial J}{\partial w_{kj}} = \frac{\partial J}{\partial z_k} \cdot \frac{\partial z_k}{\partial net_k} \cdot \frac{\partial net_k}{\partial w_{kj}} = (z_k - t_k) \cdot f'(net_k) \cdot y_j$$

$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y_j} \cdot \frac{\partial y_j}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}}$$

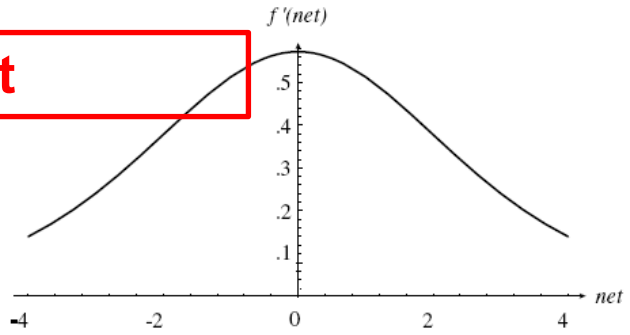
$$= \left(\sum_{k=1}^{c} (z_k - t_k) \cdot f'(net_k) \cdot w_{kj} \right) \cdot f'(net_j) \cdot x_i$$

Backpropagation Algorithm



$$\frac{\partial J}{\partial w_4} = \frac{\partial J}{\partial z} \cdot f'(net_4) \cdot x_4$$

vanishing gradient problem



First order derivative

$$\frac{\partial J}{\partial w_3} = \frac{\partial J}{\partial z} \cdot f'(net_4) \cdot w_4 \cdot f'(net_3) \cdot x_3$$

$$\frac{\partial J}{\partial w_2} = \frac{\partial J}{\partial z} \cdot f'(net_4) \cdot w_4 \cdot f'(net_3) \cdot w_3 f'(net_2) \cdot x_2$$

$$\frac{\partial J}{\partial w_1} = \frac{\partial J}{\partial z} \cdot \underbrace{f'(net_4) \cdot w_4}_{<1} \cdot \underbrace{f'(net_3) \cdot w_3}_{<1} \underbrace{f'(net_2) \cdot w_2}_{<1} \underbrace{f'(net_1)}_{<1} \cdot x_1$$

Decision Tree

Basic Decision Tree Learning Algorithm

- Data is processed in Batch (I.e., all the data is available).
- Recursively build a decision tree top-down.

- DT(Examples, Attributes)

```
If all Examples have same label:
        return a leaf node with Label
Else
        If  Attributes is empty:
           return a leaf with majority Label
        Else
           Pick an attribute A as root
           For each value v of A
           Let Examples(v) be all the examples for which  A=v
           Add a branch out of the root for the test  A=v
              If  Examples(v)  is empty
                    create a leaf node labeled with the majority label in Examples
              Else
                    recursively create subtree by calling DT(Examples(v), Attribute-{A})
```

An Illustrative Example

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

An Illustrative Example (2)

$$Entropy(S) = -\frac{9}{14}\log\left(\frac{9}{14}\right) - \frac{5}{14}\log\left(\frac{5}{14}\right) = 0.94$$

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
3	Overcast	Hot	High	Weak	Yes
4	Rain	Mild	High	Weak	Yes
5	Rain	Cool	Normal	Weak	Yes
6	Rain	Cool	Normal	Strong	No
7	Overcast	Cool	Normal	Strong	Yes
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
10	Rain	Mild	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes
12	Overcast	Mild	High	Strong	Yes
13	Overcast	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

9+,5-

An Illustrative Example (2)

3+,4- 6+,1- 6+2- 3+,3-

E=.985 *E*=.592 *E*=.811 *E*=1.0

*Gain(S,*Humidity*)*= .94 - 7/14 0.985 - 7/14 0.592= 0.151

*Gain(S,*Wind*)*= .94 - 8/14 0.811 - 6/14 1.0 = 0.048

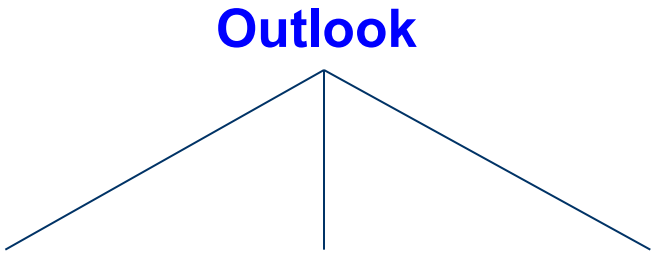
Humidity	Wind	PlayTennis
High	Weak	No
High	Strong	No
High	Weak	Yes
High	Weak	Yes
Normal	Weak	Yes
Normal	Strong	No
Normal	Strong	Yes
High	Weak	No
Normal	Weak	Yes
Normal	Weak	Yes
Normal	Strong	Yes
High	Strong	Yes
Normal	Weak	Yes
High	Strong	No

9+,5-

E=.94

$$Gain(S,a) = Entropy(S) - \sum_{v \in values(a)} \frac{|S_v|}{|S|} Entropy(S_v)$$

An Illustrative Example (3)



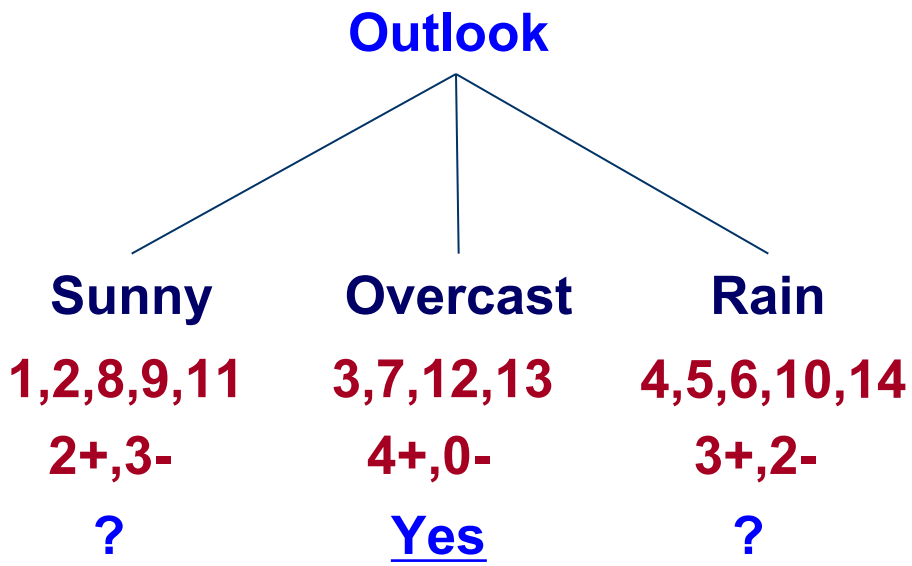
*Gain(S,*Humidity*)*=0.151

*Gain(S,*Wind*)*=0.048

*Gain(S,*Temperature*)*=0.029

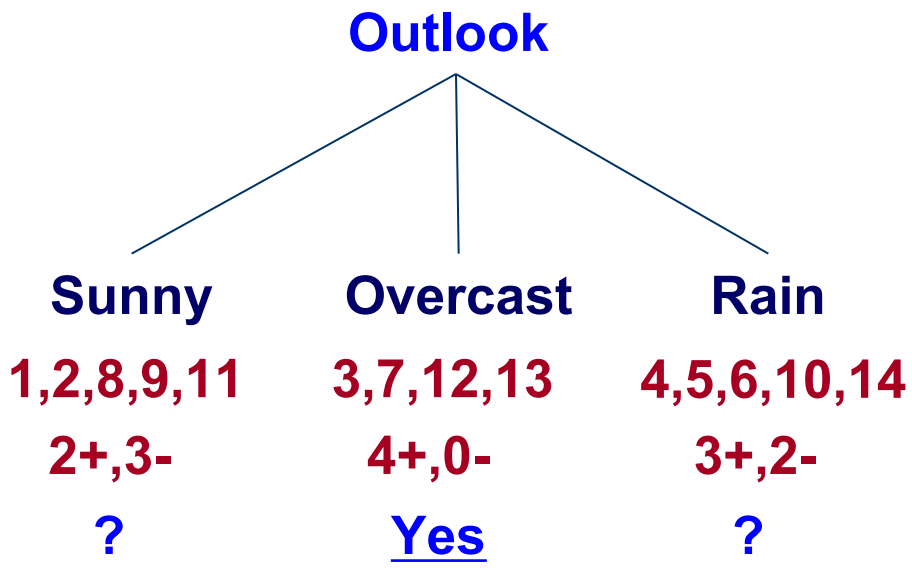
*Gain(S,*Outlook*)*=0.246

An Illustrative Example (3)



Day	Outlook	PlayTennis
1	Sunny	No
2	Sunny	No
3	Overcast	Yes
4	Rain	Yes
5	Rain	Yes
6	Rain	No
7	Overcast	Yes
8	Sunny	No
9	Sunny	Yes
10	Rain	Yes
11	Sunny	Yes
12	Overcast	Yes
13	Overcast	Yes
14	Rain	No

An Illustrative Example (3)



Continue until:

- **Every attribute is included in path, or,**
- **All examples in the leaf have same label**

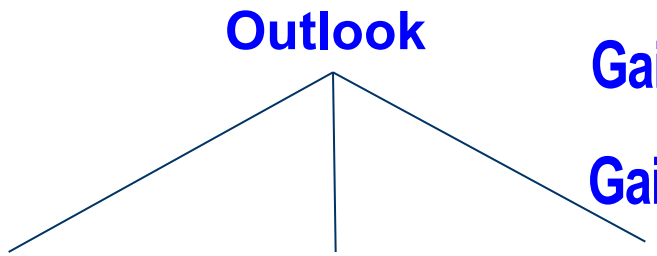
Day	Outlook	PlayTennis
1	Sunny	No
2	Sunny	No
3	Overcast	Yes
4	Rain	Yes
5	Rain	Yes
6	Rain	No
7	Overcast	Yes
8	Sunny	No
9	Sunny	Yes
10	Rain	Yes
11	Sunny	Yes
12	Overcast	Yes
13	Overcast	Yes
14	Rain	No

An Illustrative Example (4)

$Gain(S_{sunny}, Humidity) = .97-(3/5)\ 0-(2/5)\ 0 = .9$

$Gain(S_{sunny}, Temp) = .97-\ 0-(2/5)\ 1 = .57$

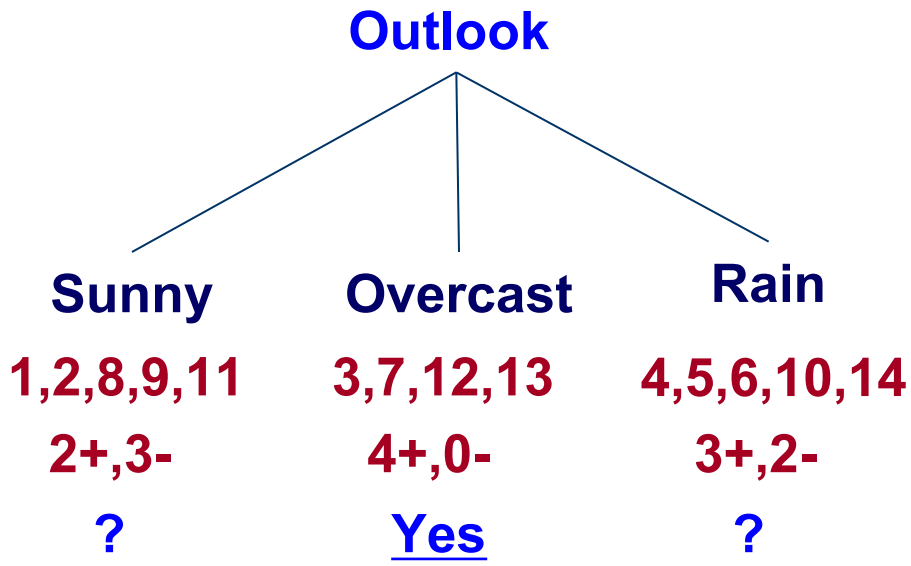
$Gain(S_{sunny}, Wind) = .97-(2/5)\ 1\ -\ (3/5)\ .92 = .0$



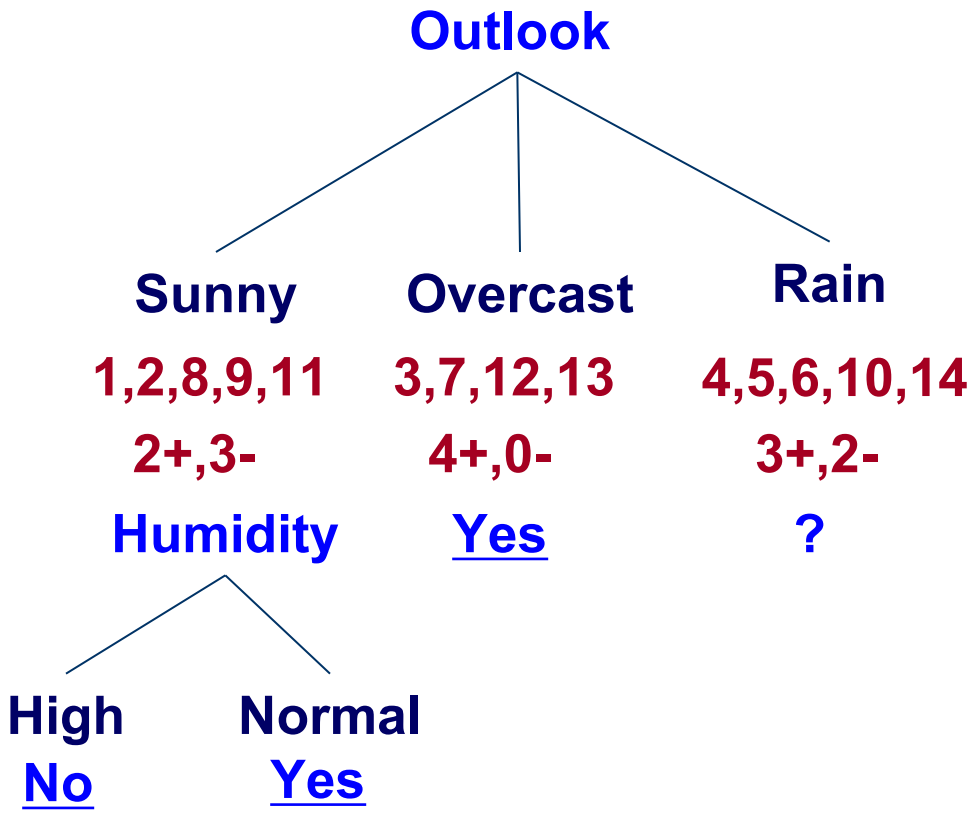
Sunny	Overcast	Rain
1,2,8,9,11	3,7,12,13	4,5,6,10,14
2+,3-	4+,0-	3+,2-
?	Yes	?

Day	Outlook	Temperature	Humidity	Wind	PlayTen
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

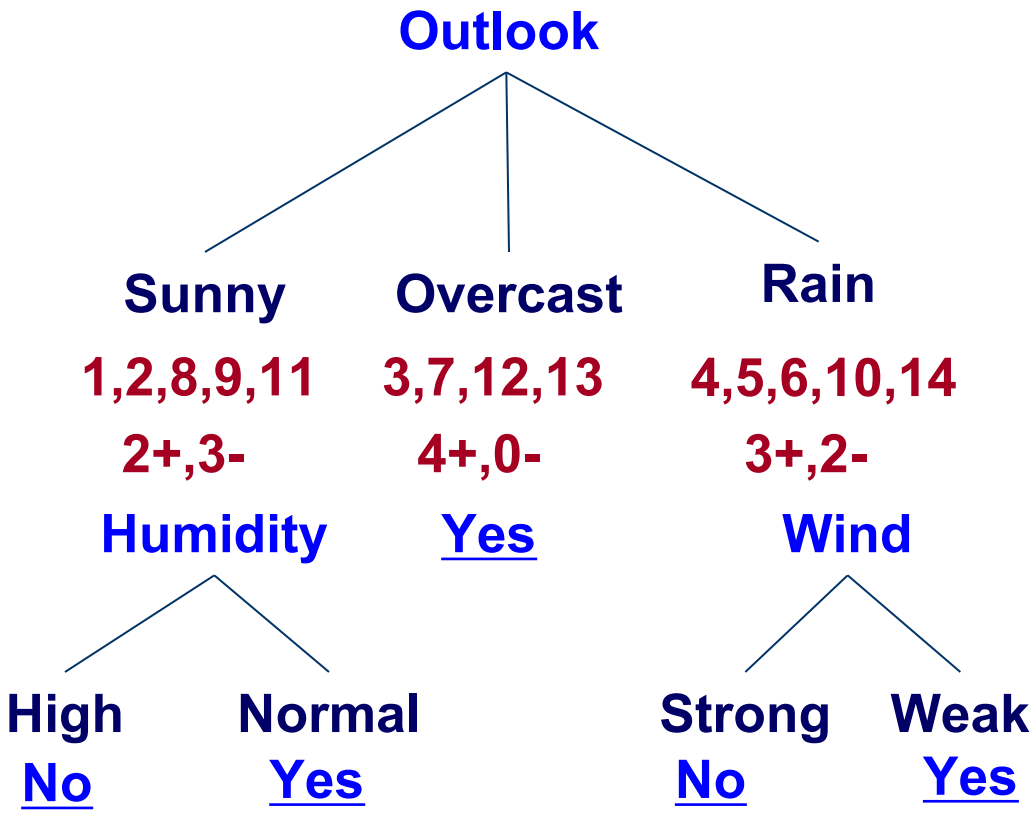
An Illustrative Example (5)



An Illustrative Example (5)



An Illustrative Example (6)



Summary: ID3(Examples, Attributes, Label)

- Let S be the set of Examples
  Label is the target attribute (the prediction)
  Attributes is the set of measured attributes
-
- Create a Root node for tree
- If all examples are labeled the same return a single node tree with Label
- Otherwise Begin
- A = attribute in Attributes that *best* classifies S
- for each possible value v of A
- Add a new tree branch corresponding to A=v
- Let *Sv* be the subset of examples in S with A=v
- if *Sv* is empty: add leaf node with the common value of Label in S
- Else: below this branch add the subtree
- ID3(*Sv*, Attributes - {a}, Label)

End

Return Root

Ensemble Methods

Which Classifier is a good choice for base learner?

- Naïve Bayesian classifier
- Perceptron
- Linear Regression
- Logistic Regression
- SVM
- Neural Network
- k Nearest Neighbor
- Decision Tree

Model with low bias
benefits from bagging

- Decision Tree!
  - Non-linear classifier
  - Easy to use and interpret
  - Can perfectly fit to any training data (overfitting) with high test error. (zero bias, high variance)

Random Forest

- Random Forest, which uses decision tree as basic unit in bagging, is an ensemble model.
- Why Random Forest?
  - Advantages of Decision Tree:
    - Handling missing value
    - Robust to outliers in input space
    - Fast
    - Good Interpretability
  - Limitations of Decision Tree:
    - Low accuracy, low bias with high variance and easy to overfit
  - Ensemble to maintain advantages while increasing accuracy

Random Forest

- The main difference between the method of RF and Bagging is that RF incorporate randomized feature selection at each split step.
- At each split step : Get feature subsets ***f*** from the whole **F**
  - More efficient in building trees
    - each time we only pick the best feature from size(f) rather than size(F).
      - We often let size(f) = $\sqrt{k}$ in classification and k/3 in regression
  - Each tree is not as good as DT, but work better when they are combined.

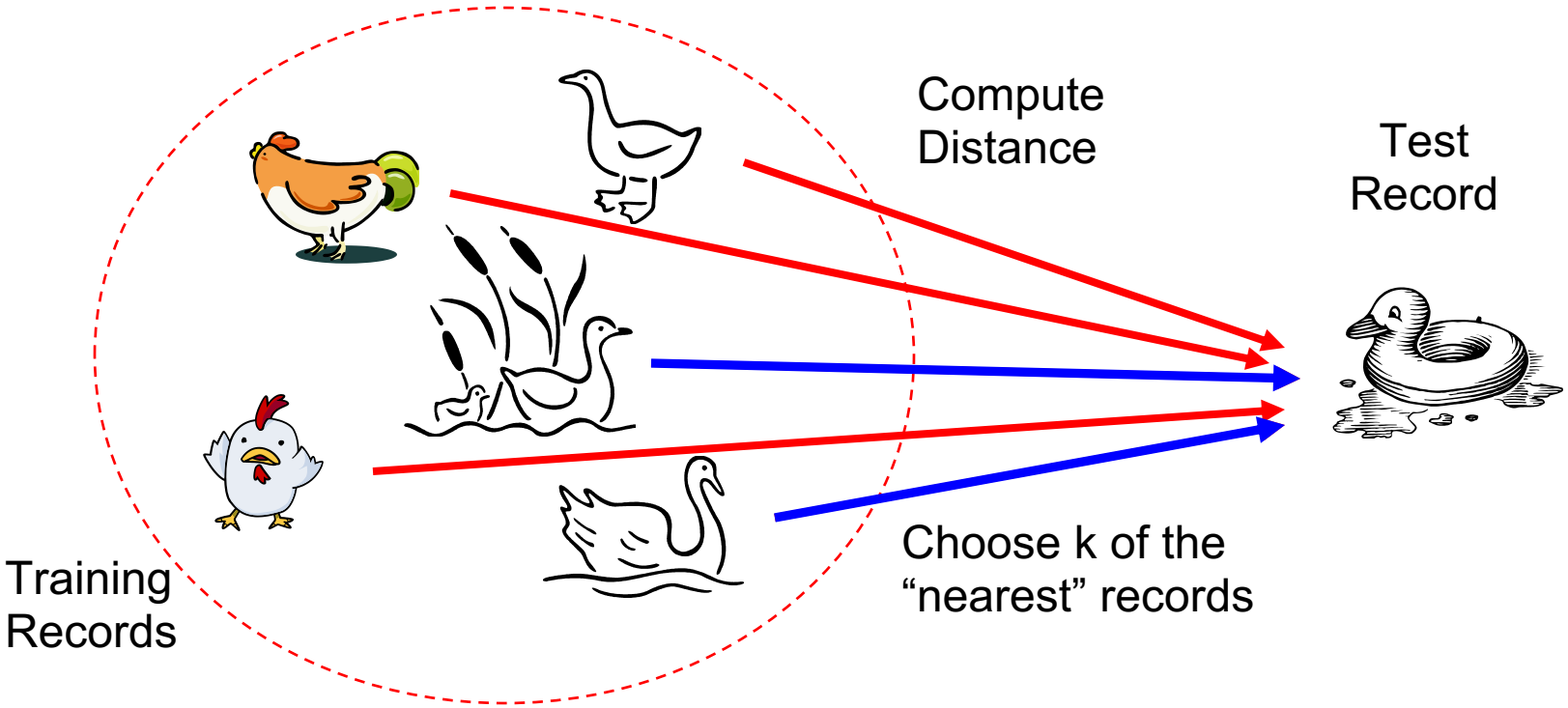
Construct Random Forest

- Let N be the number of trees and K be the feature subset size.
- For each N iterations: (building a tree in forest)
  - 1. Select a new bootstrap sample from original training set
  - 2. Growing a tree…
  - 3. At each internal node, randomly select K features from ALL features and then, determine the best split in ONLY the K features.
  - 4. Do not pruning
  - 5. Until Validation Error never decrease (here, validation(dev set) versus testing, remember?)
- At last, overall prediction as the average(or vote) from N trees.

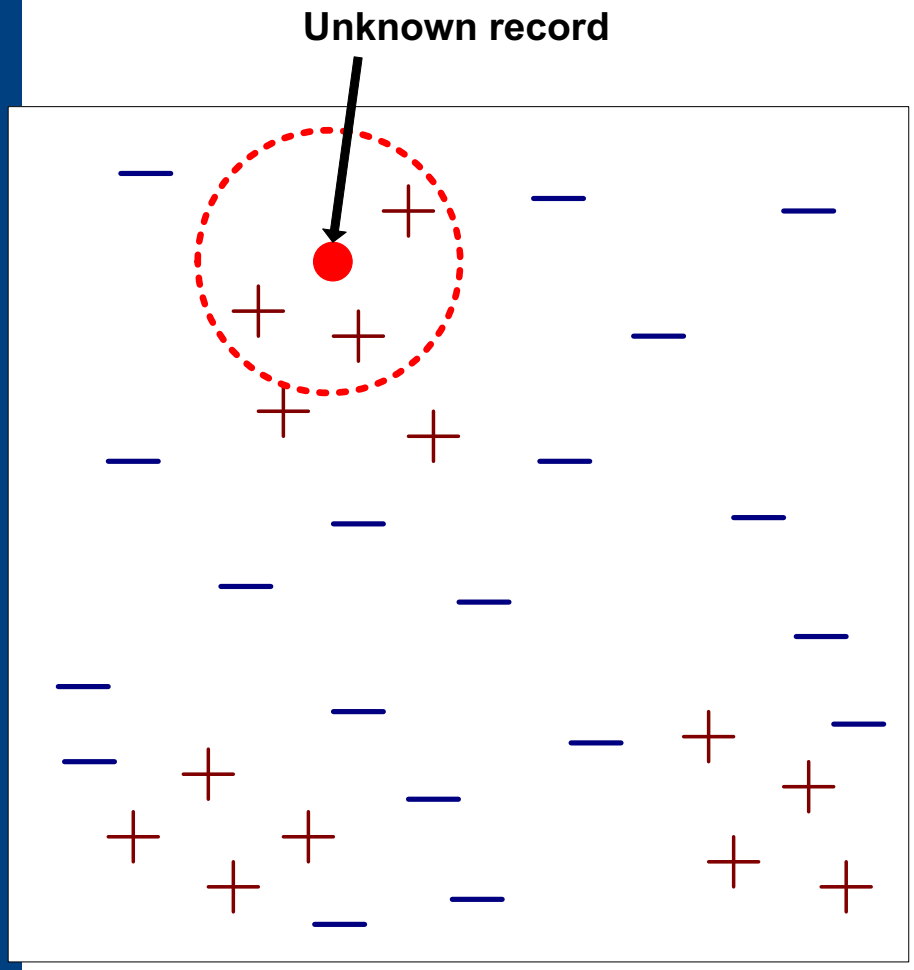
k Nearest Neighbor Classifier

Nearest Neighbor Classifiers

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck



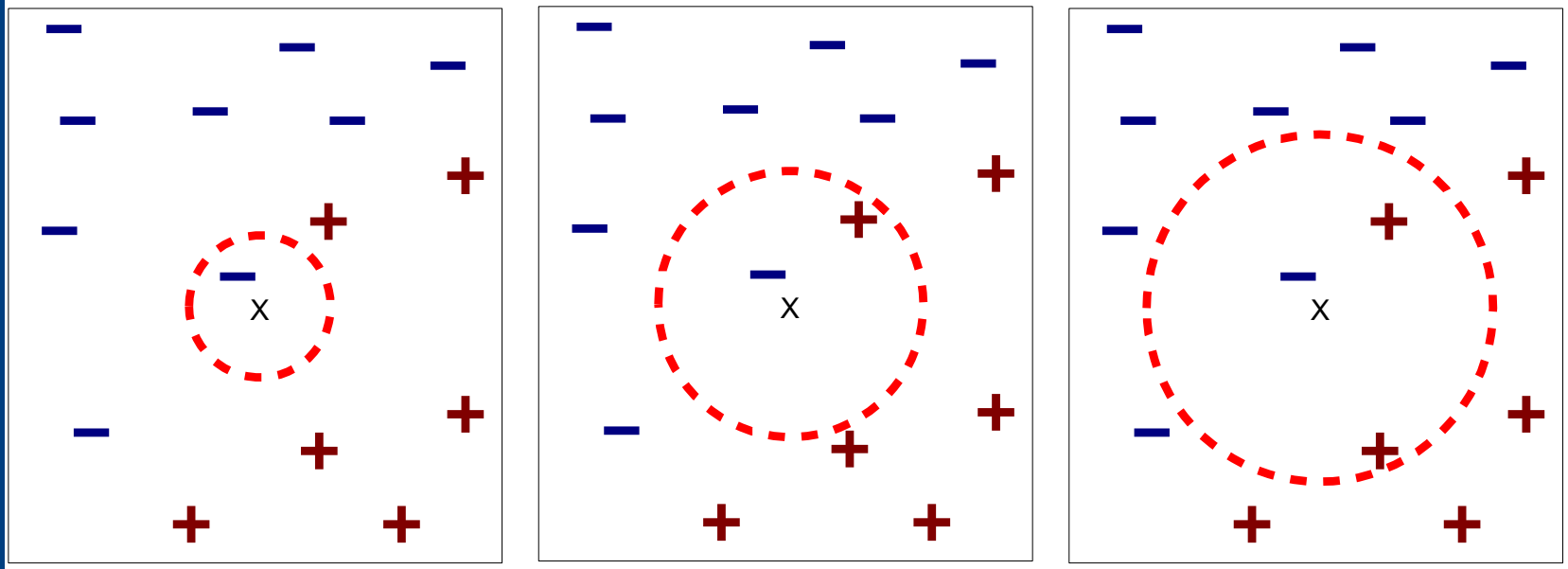
Nearest-Neighbor Classifiers



- Requires three things
  - The set of stored records
  - Distance Metric to compute distance between records
  - The value of *k*, the number of nearest neighbors to retrieve

- To classify an unknown record:
  - Compute distance to other training records
  - Identify *k* nearest neighbors
  - Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

Definition of Nearest Neighbor



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

K-nearest neighbors of a record ***x*** are data points that have the *k* smallest distance to ***x***

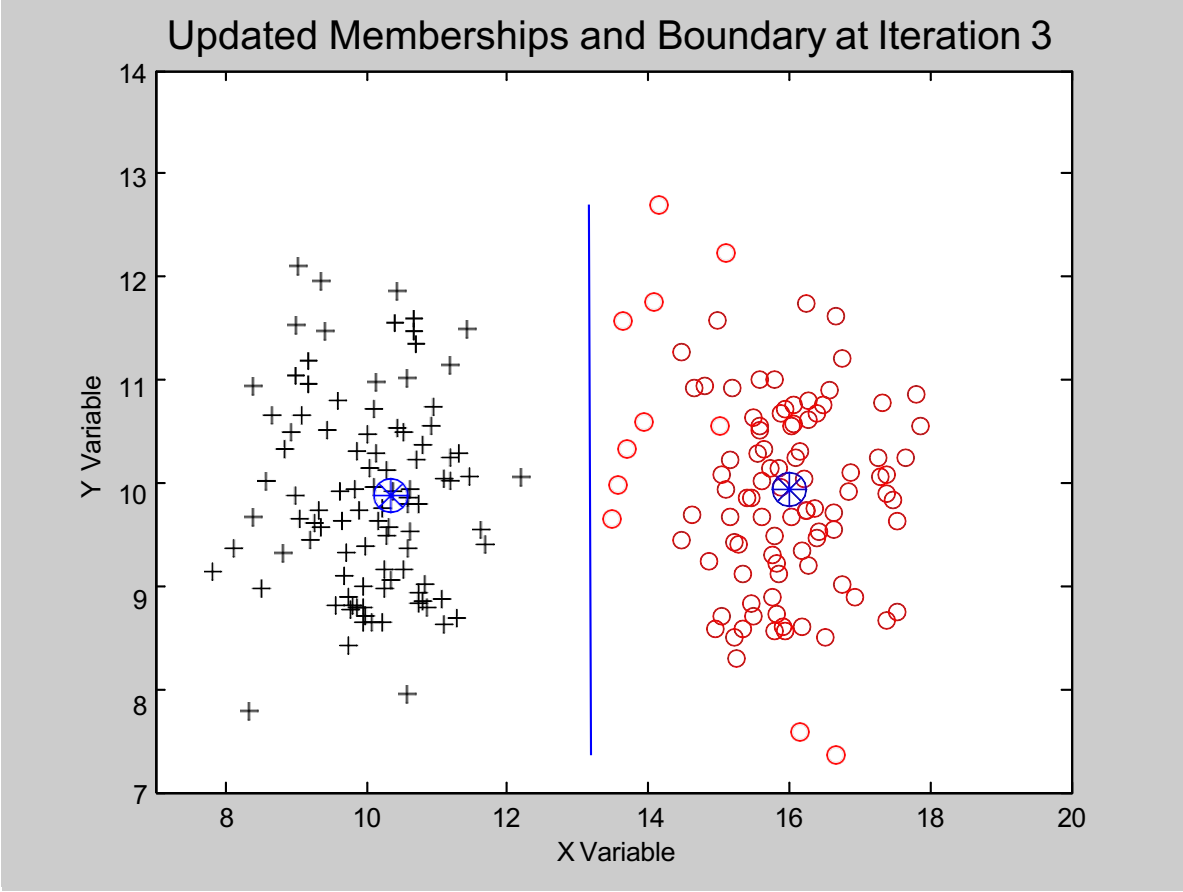
Clustering

The K-Means Algorithm (MacQueen'67)

Given K, the k-means algorithm is performed:

1. Randomly pick K data points as the seed points $\boldsymbol{\mu}_k$
2. Recursively run the following steps until converge
   1. Assign each point to the cluster with the nearest seed point
   $$z_i = \arg\min_k \|\boldsymbol{x}_i - \boldsymbol{\mu}_k\|^2$$
   2. Update the seed point for each cluster
   $$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{i=1}^{N_k} \boldsymbol{x}_i^{(k)}$$

K-Means Example



Dimensionality Reduction

What is Principal Component Analysis?

- Principal component analysis (PCA)
  - Reduce the dimensionality of a data set by finding a new set of variables, smaller than the original set of variables
  - Retains most of the sample's information.
  - Useful for the compression and classification of data.

- By information we mean the variation present in the sample, given by the correlations between the original variables.
  - The new variables, called principal components (PCs), are uncorrelated, and are ordered by the fraction of the total information each retains.

Algebraic Derivation of PCs

- Given a sample of n observations on a vector of p variables

$$\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \in \mathcal{R}^p$$

- Define the first principal component of the sample by the linear transformation

$$z_i^{(1)} = \boldsymbol{a}_1^T \boldsymbol{x}_i, \qquad i = 1, \cdots n$$

is chosen such that $var\left(z^{(1)}\right)$ is maximum.

Algebraic Derivation of PCs

$$var(z^{(1)}) = E\left(\left(z^{(1)} - \bar{z}^{(1)}\right)^2\right)$$

$$= \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{a}_1^T\boldsymbol{x}_i - \boldsymbol{a}_1^T\bar{\boldsymbol{x}})^2$$

$$= \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{a}_1^T(\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T\boldsymbol{a}_1 = \boldsymbol{a}_1^T S\boldsymbol{a}_1$$

Where $S = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T$

is the covariance matrix and $\bar{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i$ is the mean.

Algebraic Derivation of PCs

$$\max_{\boldsymbol{a}_1} \boldsymbol{a}_1^T S \boldsymbol{a}_1$$

$$s.t. \quad \boldsymbol{a}_1^T \boldsymbol{a}_1 = 1$$

Let λ be a Lagrange multiplier

$$L = \boldsymbol{a}_1^T S \boldsymbol{a}_1 - \lambda(\boldsymbol{a}_1^T \boldsymbol{a}_1 - 1)$$

$$\frac{\partial L}{\partial \boldsymbol{a}_1} = 2S\boldsymbol{a}_1 - 2\lambda \boldsymbol{a}_1 = 0$$

$$S\boldsymbol{a}_1 = \lambda \boldsymbol{a}_1$$

therefor, $\boldsymbol{a}_1$ is an eigenvector of S corresponding to the largest eigenvalue $\lambda = \lambda_1$.

Algebraic Derivation of PCs

$$\max_{\boldsymbol{a}_2} \boldsymbol{a}_2^T S \boldsymbol{a}_2$$

$$s.t. \quad \boldsymbol{a}_2^T \boldsymbol{a}_2 = 1, \mathrm{cov}\left(z^{(2)}, z^{(1)}\right) = 0$$

$$\mathrm{cov}\left(z^{(2)}, z^{(1)}\right) = \boldsymbol{a}_2^T S \boldsymbol{a}_1 = \lambda \boldsymbol{a}_2^T \boldsymbol{a}_1 = 0$$

$$S\boldsymbol{a}_2 = \lambda \boldsymbol{a}_2$$

$\boldsymbol{a}_2$ is an eigenvector of S corresponding to the second largest eigenvalue $\lambda = \lambda_2$.

Algebraic Derivation of PCs

- In general:

$$var\left(z^{(k)}\right) = \boldsymbol{a}_k^T S \boldsymbol{a}_k = \lambda_k$$

- The k^{th} largest eigenvalue of S is the variance of k^{th} PC.
- The k^{th} PC $z^{(k)}$ retains the k^{th} greatest fraction of the variation in the sample.

Principle Component Analysis

- Main steps for computing PCs:
  - Form the covariance matrix S.
  - Compute its eigenvectors: $\{\boldsymbol{a}_i\}_{i=1}^{p}$
  - Use the first d eigenvectors $\{\boldsymbol{a}_i\}_{i=1}^{d}$ to form the d PCs.
  - The transformation A is given by

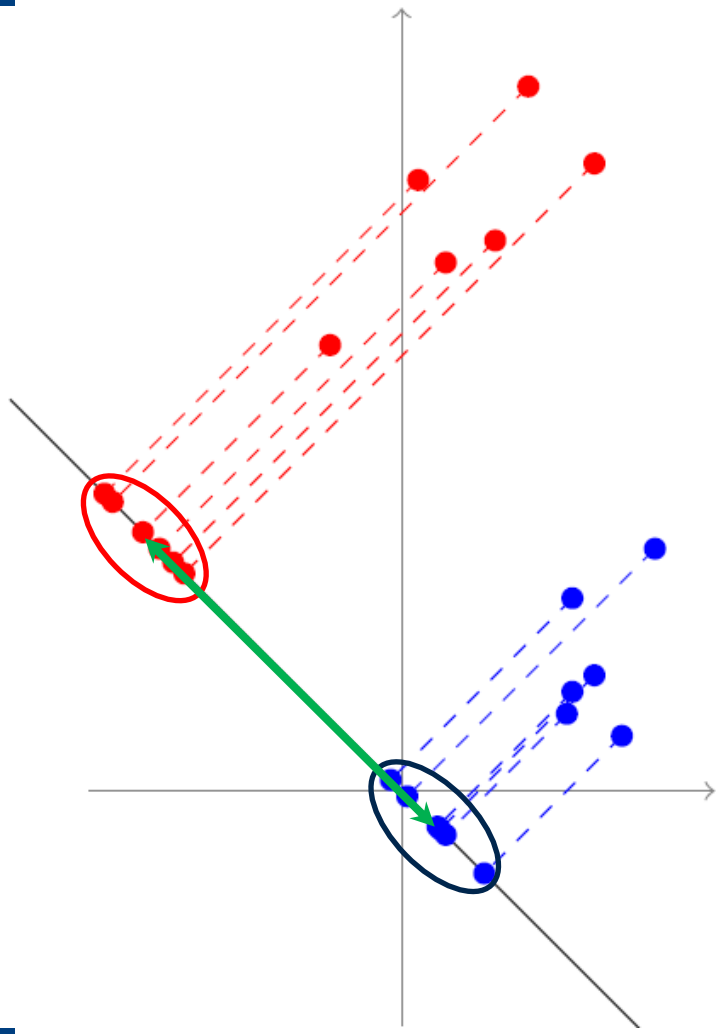
$$A = [\boldsymbol{a}_1, \cdots \boldsymbol{a}_d]$$

- A test point $\boldsymbol{x} \in \mathcal{R}^p \rightarrow A^T\boldsymbol{x} \in \mathcal{R}^d$

Linear Discriminant Analysis

- Perform dimensionality reduction "while preserving as much of the class discriminatory information as possible".
- Seeks to find directions along which the classes are best separated.
- Takes into consideration the scatter within-classes but also the scatter between-classes.

Linear Discriminant Analysis



- **Two Classes ω_1, ω_2**

$$z = \boldsymbol{a}^T \boldsymbol{x}$$

$$\tilde{\mu}_i = \frac{1}{n_i} \sum_{z \in \omega_i} z$$

$$\boldsymbol{\mu}_i = \frac{1}{n_i} \sum_{\boldsymbol{x} \in \omega_i} \boldsymbol{x}, \tilde{\mu}_i = \boldsymbol{a}^T \boldsymbol{\mu}_i$$

$$|\tilde{\mu}_1 - \tilde{\mu}_2| = |\boldsymbol{a}^T (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)|$$

$$\tilde{s}_i^2 = \sum_{z \in \omega_i} (z - \tilde{\mu}_i)^2$$

$$\frac{1}{n} (\tilde{s}_1^2 + \tilde{s}_2^2)$$

$$J(\boldsymbol{a}) = \frac{(\tilde{\mu}_1 - \tilde{\mu}_2)^2}{\tilde{s}_1^2 + \tilde{s}_2^2}$$

Linear Discriminant Analysis

- Main steps:
  - Form the scatter matrices S_B and S_W.
  - Compute the eigenvectors $\{\boldsymbol{a}_i\}_{i=1}^{c-1}$ corresponding to the non-zero eigenvalue of the generalized eigen-problem:

$$S_B\boldsymbol{a} = \lambda S_W\boldsymbol{a} \quad \text{or} \quad S_B\boldsymbol{a} = \lambda S_T\boldsymbol{a}$$

  - The transformation A is given by

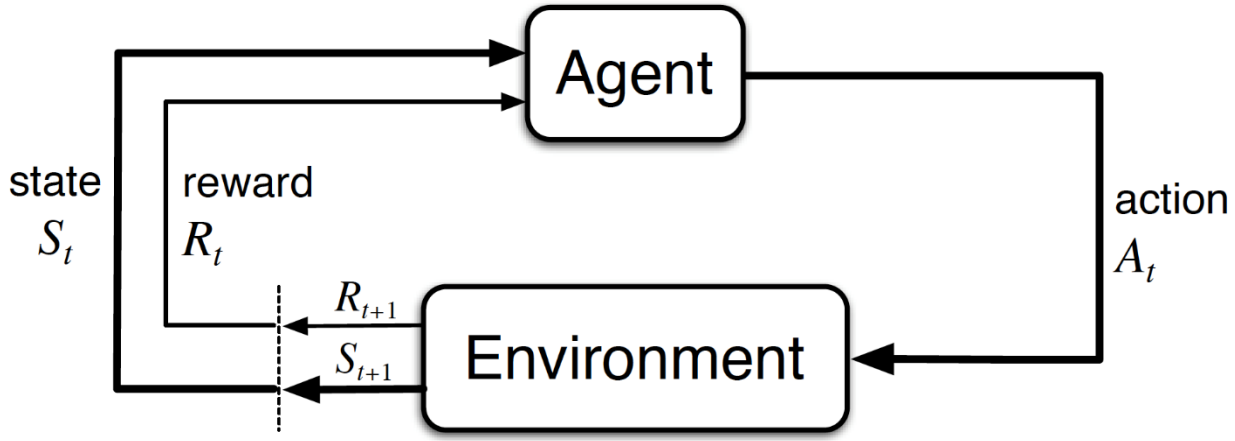
$$A = [\boldsymbol{a}_1, \cdots \boldsymbol{a}_{c-1}]$$

- A test point $\boldsymbol{x} \in \mathcal{R}^p \rightarrow A^T\boldsymbol{x} \in \mathcal{R}^{(c-1)}$

Reinforcement Learning

The reinforcement learning problem

- The agent–environment interaction



  - agent: the learner and decision-maker
  - environment: the thing agent interacts with, comprising everything outside the agent
  - reward: special numerical values that the agent tries to maximize over time
  - state and action: in general, actions can be any decisions we want to learn how to make, and the states can be anything we can know that might be useful in making them

The reinforcement learning problem

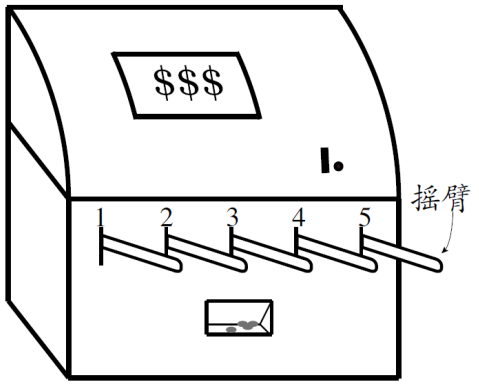
- What is the training data of RL?
  - (S, A, R). (State-Action-Reward)
- What is the goal of RL?
  - Develop an optimal policy (sequence of decision rules) for the learner so as to maximize its long-term reward

Elements of RL

- A policy
  - A map from state space to action space.
  - May be stochastic.
- A reward function
  - It maps each state (or, state-action pair) to a real number, called reward.
- A value function
  - Value of a state (or, state-action pair) is the total expected reward, starting from that state (or, state-action pair).

Sample: K-Armed Bandit

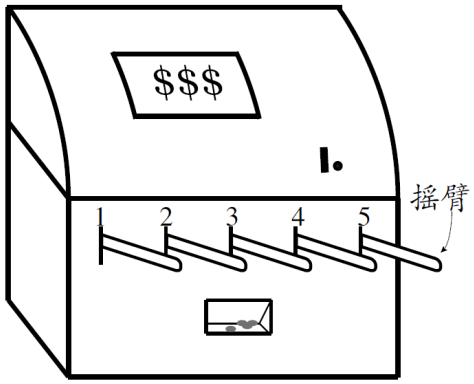
- K-摇臂赌博机 (K-Armed Bandit)
  - 只有一个状态，K个动作
  - 每个摇臂的奖赏服从某个期望未知的分布
  - 执行有限次动作
  - 最大化累积奖赏



Sample: K-Armed Bandit

- 强化学习面临的主要困难：探索-利用窘境 (Exploration-Exploitation dilemma)
  - 探索：估计不同摇臂的优劣 (奖赏期望的大小)
  - 利用：选择当前最优的摇臂

- 在探索与利用之间进行折中
  - ϵ-Greedy



ϵ-greedy

- ϵ-greedy
  - 以ϵ的概率探索：均匀随机选择一个摇臂
  - 以$1-\epsilon$的概率利用：选择当前平均奖赏最高的摇臂
- ϵ-贪心算法
  - 若摇臂k机器通过被尝试了n次，得到的奖赏为$v_1, v_2, \ldots, v_n$则
    - 平均奖赏：$$Q(k) = \frac{1}{n}\sum_{i=1}^{n} v_i \ .$$
    - 增量式更新平均奖赏：$$Q_n(k) = \frac{1}{n}\left((n-1) \times Q_{n-1}(k) + v_n\right)$$
      $$= Q_{n-1}(k) + \frac{1}{n}\left(v_n - Q_{n-1}(k)\right)$$

ϵ-greedy

- ϵ-贪心算法

输入： 摇臂数 K;
奖赏函数 R;
尝试次数 T;
探索概率 ϵ.

过程：

$r = 0$;
$\forall i = 1, 2, \ldots K : \ Q(i) = 0, \ \text{count}(i) = 0$;
for $t = 1, 2, \ldots, T$ **do**
  if rand()$< \epsilon$ **then**
    $k =$ 从 $1, 2, \ldots, K$ 中以均匀分布随机选取
  else
    $k = \arg\max_i Q(i)$
  end if
  $v = R(k)$;
  $r = r + v$;
  $Q(k) = \frac{Q(k) \times \text{count}(k) + v}{\text{count}(k) + 1}$;
  $\text{count}(k) = \text{count}(k) + 1$;
end for

输出： 累积奖赏 r

- 不确定性增加，概率分布较宽，ϵ增加
- 尝试次数增加，ϵ减少：$\epsilon = \frac{1}{\sqrt{t}}$

Summary

Junbo Zhao(赵俊博)

College of Computer Science
Zhejiang University

j.zhao@zju.edu.cn

The key takeaways

- So after ten years, what should you remember about this course?
- Supervised learning: learn a mapping function $f(x) = y$
- Unsupervised learning: only $x$are given, find "interesting patterns"
- Reinforcement learning: learn to maximize expected return, with only reward given by environments, and learn by trial-and-error
- Three components of one algorithm: representation(hypothesis space), metric & loss, optimization
- ML Pipeline
- ML needs data, and needs good data
  - It learns what in the data: Garbage in, Garbage out
- Python programming

The key takeaways

- For each algorithm, what is the most important to learn/remember?
  - Assumption/Motivation of this algorithm
  - Advantages and disadvantages
  - When it works (mostly the assumption holds)
  - When it does not work (mostly the assumption breaks)

The key takeaways

- There takeaways are irrelevant to ML.
- Meta-learning: learn how to learn.
- When you want to learn a course/topic, search good materials and use good materials to learn it.
  - I often google the courses from the top universities, and find courses with videos.
- Learn how to solve problems
- When deriving the math part, remember that motivation can be really important
- Active learning & critical thinking:
  - Think about the motivation when learning new stuff
  - Ask a lot of Questions (WHYs). Sometimes take NOTHING for granted.
- ……
- ML is interesting and useful. Learning is interesting, and can be not that hard.

Where to go: learn ML

- Great ML intro courses from top universities, for topics we do not cover: Stanford CS229 CMU 10-701 ……
- For the most popular deep learning:
- Stanford CS231n：Deep learning for CV.
- Stanford CS224n：Deep learning for NLP.
- Berkeley CS285：Deep RL.
- Berkeley CS294-158：Deep Unsupervised Learning.
- Stanford CS230：Deep learning.
- A lot in the internet. Find the one you are interested yourself.
- When you are more senior, you can pay attention to the instructors of the course. Course from them may have different advantages and disadvantages.

Where to go: industry

- Intern in the top groups in the top companies:
  - Bytedance, Ali, Tencent, Kuaishou ,……
  - Google, Hulu, Microsoft, Apple, ……
- Competitions:
  - Kaggle
  - Tianchi
  - Competitions in top conferences: KDD, CVPR……
  - ……
- They are not your destination. They are approaches to learning.

Where to go: research

- Learn the basic knowledge
  - Take good courses(online or offline)
  - Read good books
- Find a problem you are interested
  - Not solved yet! (You are not satisfied with SOTA)
  - Choose a problem your advisor worked on / are working on
  - Read a good survey (if existing)
  - Organize the papers published in the last CCF A conference based on topics.
    - http://www.ccf.org.cn/sites/ccf/paiming.jsp
  - How to search papers?
    - Search the conference name in google, find the homepage
    - DBLP also list the papers accepted by conferences
    - Search the paper name in google or google scholar

Where to go: research

- Work on this problem
  - Try to know everything about the problem.
  - Why it is unsolved?
  - What are the bottlenecks?
  - What is the key bottleneck?
- Understand (be familiar with) the SOTA
  - With questions
- Use some bad cases to find the problem and try to solve the problem
- Be focus and patient!

Where to go: research

- Find a good advisor
- Read top conference papers (CCF-A)
- Re-implement papers, do experiments
- Come up with ideas
- Verify whether assumption is correct or not with experiments
- Assumption accept or reject?
  - Accept: more experiments to verify it
  - Reject: why? Modify your assumption according to it.
- Document your experiment results.
- When you think you can get conclusion, write papers and submit to top conferences.
  - You can write paper a little bit earlier.

Where to go: philosophy

- The way we introduce algorithm to you is often showing the right way, but not the wrong way.
- The road to truth is difficult, there is no straightforward way to go.
- So regardless of industry or research, remember that failure is normal.
- We are getting closer and closer to the truth, that's the important thing.