# Reason for Classification

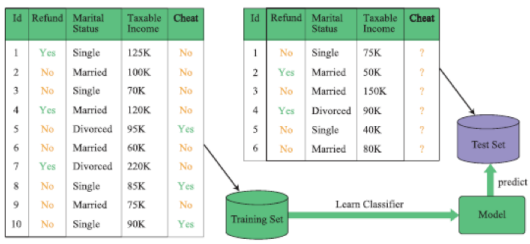
- Knowing the classes of the data, we could easily manage the data and react to the possible outcomes
- Predict whether users would default in the future based on their basic information and historical transaction records
- Predict whether a tumor is benign or malignant based on their physical and geometrical features
- Predict the users' interests in the new products based on their historical purchasing records and behaviorial preferences
- Separate spams and advertisements from emails

# What is Classification

- Supervised learning : predict label $y$ from features $\mathbf{x}$
- Training stage : Given a data set $D = \{(\mathbf{x}, y)\}$, including both features and labels, split $D = D_{train} \bigcup D_{test}$, find a classifier (function $y = f(\mathbf{x})$) that best relates $y_{train}$ with $\mathbf{x}_{train}$, then evaluate how close $f(\mathbf{x}_{test})$ is to $y_{test}$
- Predicting stage : apply the predictor to the unlabeled data $\mathbf{x}_{pred}$ (only features) to find the proper labels $y_{pred} = f(\mathbf{x}_{pred})$

有监督学习：对特征x都有预测标签y
训练：train + test

预测：$y_{pred} = f(x_{pred})$



| Id | Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|---|
| 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 |

| Id | Refund | Marital Status | Taxable Income | Cheat |
|---|---|---|---|---|
| 1 | No | Single | 75K | ? |
| 2 | Yes | Married | 50K | ? |
| 3 | No | Married | 150K | ? |
| 4 | Yes | Divorced | 90K | ? |
| 5 | No | Single | 40K | ? |
| 6 | No | Married | 80K | ? |



# Classification Methods

- Different assumptions on $f$ lead to different models
- Basic classification models
  - k-nearest neighbor (kNN) K近邻
  - Decision trees 决策树
  - Naive Bayes 朴素贝叶斯
  - Support vector machines (SVM) 支持向量机
  - Logistic regression 逻辑回归
  - Linear discriminant analysis (LDA) 线性判别分析
  - Artificial neural network (ANN) 人工神经网络
  - ...
- Ensemble learning : Random forest and Adaboost

集成学习：随机森林、Adaboost.

# K-nearest Neighbour

- k-nearest neighbor (kNN) is the simplest supervised learning method, especially useful when prior knowledge on the data is very limited
- Do training and test simultaneously
- When classifying a test sample $x$, scan the training set and find the closest $k$ samples $D_k = \{x_1, \ldots, x_k\}$ to the test sample; make vote based on the labels of the samples in $D_k$; the majority vote is the label of the test sample
- Low bias, high variance
- Advantages : not sensitive to outliers, easy to implement and parallelize, good for large training set
- Drawbacks : need to tune $k$, take large storage, computationally intensive

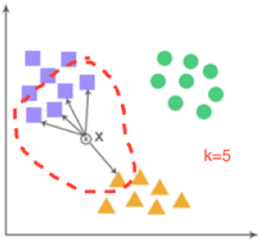
K近邻是最简单的监督学习
数据分布只有很少 / 没有任何先验知识.
训练、测试同时
多数投票
低bias，高variance
对异常值不敏感
大存储、计算密度高

k和距离的选取是关键.

# Algorithm

- Input : training set $D_{train} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, a test sample $x$ without label $y$, $k$ and distance metric $d(x, y)$
- Output : predicted label $y_{pred}$ for $x$

1. Compute $d(x, x_j)$ for each $(x_j, y_j) \in D_{train}$
2. Sort the distances in an ascending order, choose the first $k$ samples $(x_{(1)}, y_{(1)}), \ldots, (x_{(k)}, y_{(k)})$
3. Make majority vote $y_{pred} = \mathrm{Mode}(y_{(1)}, \ldots, y_{(k)})$




最近的k个投票.

# Distance Matrics

- Minkowski distance : $d_h(\mathbf{x}_1, \mathbf{x}_2) = \sqrt[h]{\sum_{i=1}^{d} (x_{1i} - x_{2i})^h}$; $h = 2$, Euclidean distance; $h = 1$, Manhattan distance
- Mahalanobis distance : $d(\mathbf{x}_1, \mathbf{x}_2) = \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^T \hat{\Sigma}^{-1} (\mathbf{x}_1 - \mathbf{x}_2)}$, where $\hat{\Sigma}$ is the covariance matrix of sample set; introduce correlations, could be applied to the non-scaling data
- Hamming distance : $Hamming(\mathbf{x}_1, \mathbf{x}_2) = d - \sum_{i=1}^{d} I(x_{1i} = x_{2i})$; used to compare two strings, e.g., $Hamming('toned', 'roses') = 3$, $Hamming('101110', '101101') = 2$

汉明距离：字符串中差异字符的个数.

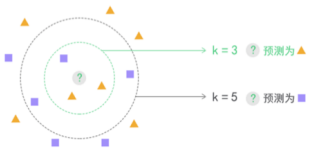
# Similarity and Divergence

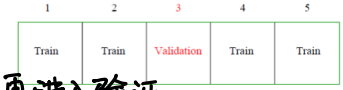
- Cosine similarity : $\cos(\mathbf{x}_1, \mathbf{x}_2) = \frac{\mathbf{x}_1^T \mathbf{x}_2}{|\mathbf{x}_1||\mathbf{x}_2|} = \frac{\sum_{i=1}^{d} x_{1i} x_{2i}}{\sqrt{\sum_{i=1}^{d} x_{1i}^2} \sqrt{\sum_{i=1}^{d} x_{2i}^2}}$; its range is $[-1, 1]$; the greater the cosine similarity, the more similar (closer) the two samples; insensitive to absolute value, popular in measuring user rankings; it is related to Pearson correlation coefficient
- Jaccard similarity for sets $A$ and $B$ : $Jaccard(A, B) = \frac{|A \bigcap B|}{|A \bigcup B|}$, used in comparing texts
- Kullback-Leibler (KL) divergence : $d_{KL}(P \| Q) = \mathrm{E}_P[\log \frac{P(x)}{Q(x)}]$ measures the distance between two probability distributions $P$ and $Q$; in discrete case, $d_{KL}(p \| q) = \sum_{i=1}^{m} p_i \log \frac{p_i}{q_i}$

KL散度：用于衡量两个概率分布的距离.

# Tuning k

- Different values of $k = 3$ and $k = 5$ leads to different classification results
- $M$-fold Cross-validation (CV) to tune $k$ : partition the dataset into M parts ($M = 5$ or 10), let $\kappa$ : $\{1, \ldots, N\} \to \{1, \ldots, M\}$ be randomized partition index map, The CV estimate of prediction error is $CV(\hat{f}, k) = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}^{-\kappa(i)}(x_i, k))$




M折交叉验证：

每个样本点都被选为验证集

最小CV的k即所求.

→ 取k得模型$\hat{f}$，再进入验证

-k(i)意思是从训练集中去掉第i个作为验证集

# Bayes Classifier (Oracle Classifier) 贝叶斯分类器

- Assume $Y \in \mathcal{Y} = \{1, 2, \ldots, C\}$, the classifier $f : \mathcal{X} \to \mathcal{Y}$ is a piecewise constant function
- For 0-1 loss $L(y, f)$, the learning problem is to minimize

$$\begin{aligned}\mathcal{E}(f) &= \mathrm{E}_{\mathrm{P}(X,Y)} L(Y, f(X)) = 1 - \mathrm{P}(Y = f(X)) \\ &= 1 - \int_{\mathcal{X}} \mathrm{P}(Y = f(X) | X = x) p_X(x) \mathrm{d}x\end{aligned}$$

- Bayes rule : $f^*(x) = \arg\max_c \mathrm{P}(Y = c | X = x)$, "the most probable label under the conditional probability on $x$"
- Bayes error rate : $\inf_f \mathcal{E}(f) = \mathcal{E}(f^*) = 1 - \mathrm{P}(Y = f^*(X))$
- Bayes decision boundary : the boundary separating the $K$ partition domains in $\mathcal{X}$ on each of which $f^*(x) \in \mathcal{Y}$ is constant. For binary classification, it is the level set on which $\mathrm{P}(Y = 1 | X = x) = \mathrm{P}(Y = 0 | X = x) = 0.5$.

需要将贝叶斯误差最小化.

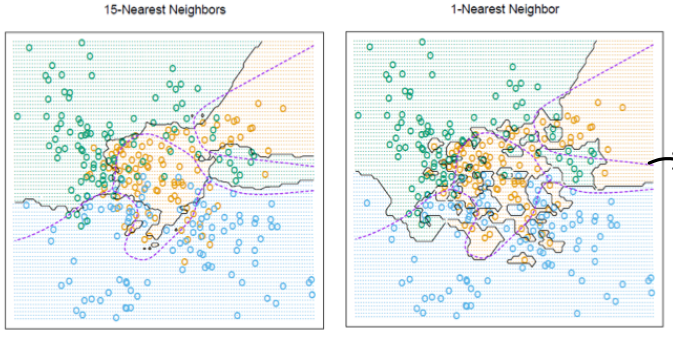
贝叶斯规则：$f^*(x) = \arg\max_c P(Y=c|X=x)$
能使误差最小.

贝叶斯错误率：$\inf_f \mathcal{E}(f) = \mathcal{E}(f^*) = 1 - P(Y=f^*(X))$
即贝叶斯误差的上界.

贝叶斯决策边界：将 $\mathcal{X}$ 分成 K 类的边界，在边界上有 $f^*(x) \in \mathcal{Y}$ 是一个常数.

# Decision Boundary

- The decision boundary of 15NN is smoother than that of 1NN




界限即是决策边界
训练样本是有噪音的，k太小容易将噪音点当作真实点.
→ 紫色虚线是贝叶斯边界.

# Analysis

- Time complexity : $O(mndK)$ where $n$ is the number of training samples, $m$ is the number of test samples, $d$ is the dimension, and $K$ is the number of nearest neighbors
- KD tree for indexing : $K$-dimensional binary search tree
- 1NN error rate is twice the Bayes error rate :
  - Bayes error $= 1 - p_{c^*}(x)$ where $c^* = \arg\max_c p_c(x)$
  - Assume the samples are i.i.d., for any test sample $x$ and small $\delta$, there is always a training sample $z \in B(x, \delta)$ (the label of $x$ is the same as that of $z$), then 1NN error is

$$\begin{aligned}\epsilon = \sum_{c=1}^{C} p_c(x)(1 - p_c(z)) &\xrightarrow{\delta \to 0} 1 - \sum_{c=1}^{C} p_c^2(x) \\ &\leqslant 1 - p_{c^*}^2(x) \\ &\leqslant 2(1 - p_{c^*}(x))\end{aligned}$$

(Remark : In fact, $\epsilon \leqslant 2(1 - p_{c^*}(x)) - \frac{C}{C-1}(1 - p_{c^*}(x))^2$)

时间复杂度 $O(mndk)$

可以用KD树优化（K维的二叉搜索树）

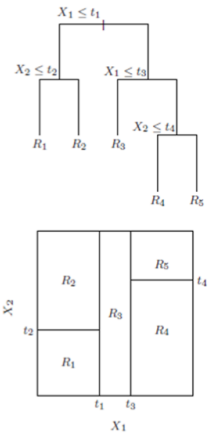
Bayes error ≤ 1NN error ≤ 2× Bayes error

- Use kNN to diagnose breast cancer (cookdata)
- Data scaling : 0-1 scaling or z-score scaling
- from sklearn.neighbors import KNeighborsClassifier
- KNeighborsClassifier(n_neighbors = 10, metric = 'minkowski', p=2)

- radius（半径）
- texture（质地）
- perimeter（周长）
- area（面积）
- smoothness（光滑度）
- compactness（致密性=$perimeter^2/area - 1.0$）
- concavity（凹度）
- concave points (凹点)
- symmetry（对称性）
- fractal dimension（分形维数）

# Decision Tree

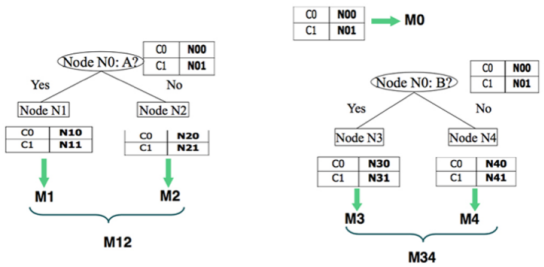
- Tree structure : internal nodes indicate features, while leaf nodes represent classes
- Start from root, choose a suitable feature $x_i$ and its split point $c_i$ at each internal node, split the node to two child nodes depending on whether $x_i \leqslant c_i$, until the child nodes are pure
- Equivalent to rectangular partition of the region




节点分割是贪心策略.

# How to choose features & split points

- Impurity : choose the feature and split point so that after each slit the impurity should decrease the most
- Impurity(M0)-Impurity(M12) > Impurity(M0)-Impurity(M34), choose A as split node ; otherwise choose B



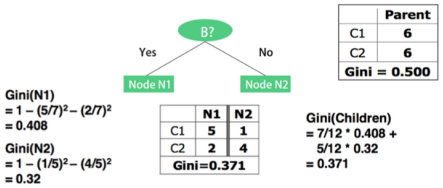

在每次分裂节点后不纯度减少最多.

# Impurity Measures 不纯度度量

## GINI Index 基尼系数

- Gini index of node $t$ : $Gini(t) = 1 - \sum_{c=1}^{C}(p(c|t))^2$ where $p(c|t)$ is the proportion of class-$c$ data in node $t$
- Maximum at $1 - \frac{1}{C}$, when $p(c|t) = \frac{1}{C}$ → 最不纯的
- Minimum at 0, when $p(c|t) = 1$ for some $c$ → 最纯的
- Gini index of a split : $Gini_{split} = \sum_{k=1}^{K} \frac{n_k}{n} Gini(k)$ where $n_k$ is the number of samples in the child node $k$, $n = \sum_{k=1}^{K} n_k$
- Choose the split so that $Gini(t) - Gini_{split}$ is maximized

每次节点分裂选使 $Gini(t) - Gini_{split}$ 最大化的.
即 $Gini_{split}$ 最小的



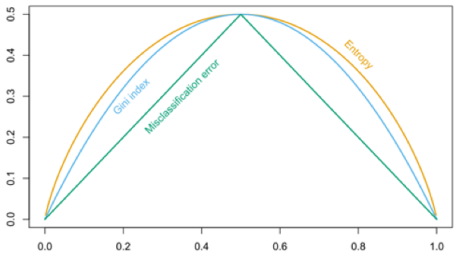

# Information Gain 信息增益

- Entropy at $t$ : $H(t) = -\sum_{c=1}^{C} p(c|t) \log_2 p(c|t)$
- Maximum at $\log_2 C$, when $p(c|t) = \frac{1}{C}$
- Minimum at 0, when $p(c|t) = 1$ for some $c$
- Information gain : $InfoGain_{split} = H(t) - \sum_{k=1}^{K} \frac{n_k}{n} H(k)$ where $n_k$ is the number of samples in the child node $k$, $n = \sum_{k=1}^{K} n_k$
- Choose the split so that $InfoGain_{split}$ is maximized (ID3 algorithm)
- Drawback : easy to generate too many child nodes and overfit
- Introduce information gain ratio : 信息增益率 $SplitINFO = -\sum_{k=1}^{K} \frac{n_k}{n} \log_2 \frac{n_k}{n}$, $InfoGainRatio = \frac{InfoGain_{split}}{SplitINFO}$ (C4.5 algorithm)

ID3算法用信息增益来分裂节点
容易产生过多子节点, 过拟合.
C4.3算法用信息增益率对ID3有所改良.

# Misclassification Error 错分错误

- Misclassification error at $t$ : $Error(t) = 1 - \max_c p(c|t)$ ; use majority vote
- Maximum at $1 - \frac{1}{C}$, when $p(c|t) = \frac{1}{C}$
- Minimum at 0, when $p(c|t) = 1$ for some $c$
- For two-class classification, $Gini(p) = 2p(1-p)$, $H(p) = -\frac{1}{2}p\log_2 p - (1-p)\log_2(1-p)$ (up to a factor $\frac{1}{2}$), $Error(p) = 1 - \max(p, 1-p)$




用多数投票.

# Comparing Three Impurity Measure

- Information gain and Gini index are more sensitive to changes in the node probabilities than the misclassification error
- Consider a two-class problem with 400 observations in each class, (400, 400) ; two possible splits, A : (300, 100) + (100, 300), and B : (200, 400) + (200, 0) ; B should be preferred
  - $Gini(A) = \frac{1}{2}Gini(A1) + \frac{1}{2}Gini(A2) = 2 \times \frac{1}{2}(2 \times \frac{3}{4} \times \frac{1}{4}) = \frac{3}{8}$, $Gini(B) = \frac{3}{4}Gini(A1) + \frac{1}{4}Gini(A2) = \frac{3}{4}(2 \times \frac{1}{3} \times \frac{2}{3})) = \frac{1}{3}$
  - $H(A) = 2 \times \frac{1}{2}(-\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{4}\log_2\frac{1}{4})) = 0.81$, $H(B) = \frac{3}{4}(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3})) = 0.69$
  - $Error(A) = 2 \times \frac{1}{2}(1 - \max(\frac{3}{4}, \frac{1}{4})) = \frac{1}{4}$, $Error(B) = \frac{3}{4}(1 - \max(\frac{1}{3}, \frac{2}{3})) = \frac{1}{4}$
- Gini index and information gain should be used when growing the tree
- In pruning, all three can be used (typically misclassification error)

信息增益和Gini指数对节点概率变化更敏感

Gini和信息增益都表明B较A好.
但错分率却表明两者一样

Gini指数和信息增益常用于生成树
错分率常用于剪枝.

# Algorithms

- Iterative Dichotomiser 3 (ID3) : by Ross Quinlan (1986), based on Occam's Razor rule (be simple) ; information gain, choose feature values by enumeration
- C4.5 and C5.0 : by R. Quinlan (1993), use information gain ratio instead, choose split thresholds for continuous features
- Classification and Regression Tree (CART) : by Leo Breiman etc. (1984) ; for classification, use Gini index ; for regression, use mean square error ; binary split

| 算法 | 属性类型 | 不纯度度量 | 分割的子节点数量 | 目标属性类型 |
|---|---|---|---|---|
| ID3 | 离散型 | 信息增益 | $k \geq 2$ | 离散型 |
| C4.5 | 离散型、连续型 | 信息增益率 | $k \geq 2$ | 离散型 |
| C5.0 | 离散型、连续型 | 信息增益率 | $k \geq 2$ | 离散型 |
| CART | 离散型、连续型 | GINI指数 | $k = 2$ | 离散型、连续型 |

## ID3 Algorithm

- Input : training set $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, $Y = \{y_1 \ldots, y_n\}$, set of features $F = \{$column variables of $X = (\mathbf{x}_1 \ldots \mathbf{x}_n)^T\}$
- Output : decision tree $T$

1. Create a root node
2. Check $Y$ : if all are positive, then return a single node tree $T$ with label "+" ; if all are negative, then return a single node tree $T$ with label "-"
3. Check $F$ : if empty, then return a single node tree $T$ with label as majority vote of $Y$
4. For each feature in $F$, compute information gain, choose the feature $A \in F$ which maximizes information gain as root
5. For $A = i$, let $D(i) = \{(\mathbf{x}_j, y_j) \in D | x_{jA} = i\}$ :
   - 5.1 If $D(i) = \emptyset$, then create a leaf node and make majority vote of $D$ as the label
   - 5.2 Else, let $D = D(i)$, go back to step 1 iteratively

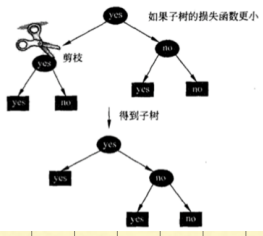
# Tree Pruning 树剪枝

- Too complex tree structure easily leads to overfitting
- Prepruning : set threshold $\delta$ for impurity decrease in splitting a node; if $\Delta Impurity_{split} > \delta$, do slitting, otherwise stop
- Postpruning : based on cost function

$$Cost_\alpha(T) = \underbrace{\sum_{t=1}^{|T|} n_t Impurity(t)}_{\text{data fidelity}} + \underbrace{\alpha |T|}_{\text{model complexity}}$$

节点个数

- Input : a complete tree $T$, $\alpha$
- Output : postpruning tree $T_\alpha$
  1. Compute $Impurity(t)$ for $\forall t$
  2. Iteratively merge child nodes bottom-up : $T_A$ and $T_B$ are the trees before and after merging, do merging if $Cost_\alpha(T_A) \geqslant Cost_\alpha(T_B)$




防止模型过于复杂导致过拟合.
前剪枝：在分裂节点时设限制δ，当$\Delta Impurity_{split} > \delta$时分裂；否则停止生成树.

后剪枝：先生成树，再回溯用.

剪枝后的cost要减少.

# Pros and Cons

- Advantages
  - Easy to interpret and visualize : widely used in finance, medical health, biology, etc.
  - Easy to deal with missing values (treat as new data type)
  - Could be extended to regression : decision tree is a rectangular partition of the domain, the predictor can be written as

$$f(x) = \sum_{m=1}^{M} c_m \mathrm{I}(\mathbf{x} \in R_m) \text{; for regression problems}$$

$$c_m = \bar{y}_m = \frac{1}{n_m} \sum_{i=1}^{n} y_i \mathrm{I}(\mathbf{x}_i \in R_m) \text{ where } n_m = \sum_{i=1}^{n} \mathrm{I}(\mathbf{x}_i \in R_m)$$

- Drawbacks :
  - Easy to be trapped at local minimum because of greedy algorithm
  - Simple decision boundary : parallel lines to the axes

可以解决缺失值
也可以做回归

容易到局部最小值（由于贪心算法）
只有简单的平行于坐标轴的边界

# Naive Bayes 朴素贝叶斯

- Based on Bayes Theorem and conditional independency assumption on features
- Widely used in text analysis, spam filtering, recommender systems, and medical diagnosis
- Bayes Theorem : let $X$ and $Y$ be a pair of random variables having joint probability $P(X = x, Y = y)$; by definition, the condition probability of $Y$ given $X$ is $P(Y|X) = \frac{P(X,Y)}{P(X)}$; then by symmetry, $P(X|Y) = \frac{P(X,Y)}{P(Y)}$; upon eliminating $P(X, Y)$

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

- $P(Y)$ is prior prob. distribution, $P(X|Y)$ is likelihood function, $P(X)$ is evidence, $P(Y|X)$ is posterior prob. distribution
- The core problem of machine learning is to estimate $P(Y|X)$ (or its moments $\mathrm{E}[Y|X] = \arg\min_f \mathrm{E}[\|Y - f(X)\|^2]$)
- Let $X = \{X_1, \ldots, X_d\}$, for fixed sample $X = x$, $P(X = x)$ is independent of $Y$, by Bayes Theorem

$$P(Y|X = x) \propto P(X = x|Y)P(Y)$$

不用算

- Assume conditional independency of $X_1, \ldots, X_d$ given $Y = c$ :

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

- Naive Bayes model :

$$\hat{y} = \arg\max_c P(Y = c) \prod_{i=1}^{d} P(X_i = x_i|Y = c)$$

基于贝叶斯定理和条件独立性假设.

对Y取最大，与X无关.
maximize (P(Y|X)) = maximize(P(X|Y)P(Y))
因为X个的联合概率分布未知，需要多条件.

在朴素贝叶斯里有的一个假设：给定Y=c，$X_i$之间相互独立
在文本分析中较常用，因为每个样本通常独立.

贝叶斯模型

# Maximum Likelihood Estimate (MLE)

- Estimate $P(Y=c)$ and $P(X_i=x_i|Y=c)$ from the dataset $D=\{(\mathbf{x}_1,y_1),\ldots,(\mathbf{x}_n,y_n)\}$
- MLE for $P(Y=c)$ : $P(Y=c)=\frac{\sum_{i=1}^{n}\mathrm{I}(y_i=c)}{n}$
- When $X_i$ is discrete variable with range $\{v_1,\ldots,v_K\}$, MLE for $P(X_i=v_k|Y=c)=\frac{\sum_{i=1}^{n}\mathrm{I}(x_i=v_k,y_i=c)}{\sum_{i=1}^{n}\mathrm{I}(y_i=c)}$
- When $X_i$ is continuous variable
  1. Do discretization, and go back to the above formula
  2. Assume $X_i$ follows some distribution (e.g., $N(\mu,\sigma^2)$) :

$$P(X_i=x|Y=c)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

  Then use MLE to estimate $\mu$ and $\sigma^2$

如伯努利分布 $\{y_i\}_{i=1}^{n} \overset{i.i.d}{\sim} \mathrm{Bernulle}(\lambda)$, $P(Y=1)=\lambda$, $P(Y=0)=1-\lambda$.

$\Rightarrow P(Y=y)=\lambda^y(1-\lambda)^{1-y}$ $(y=0 \text{ or } 1)$

$P(y_1,\ldots,y_n;\lambda)=\prod_{i=1}^{n}\lambda^{y_i}(1-\lambda)^{1-y_i}=\lambda^{\sum_{i=1}^{n}y_i}(1-\lambda)^{\sum_{i=1}^{n}(1-y_i)}$

$L(\lambda;y_1,\ldots y_n) \longrightarrow \max L \Rightarrow \hat{\lambda}=\ldots$ (采样后 $y_i$ 已知)

$l(\lambda)=\log L=\sum_{i=1}^{n}y_i\log\lambda+\sum_{i=1}^{n}(1-y_i)\log(1-\lambda)$

$\frac{dl}{d\lambda}=\frac{1}{\lambda}\sum_{i=1}^{n}y_i-\frac{1}{1-\lambda}(n-\sum_{i=1}^{n}y_i)=0 \Rightarrow \lambda=\frac{\sum_{i=1}^{n}y_i}{n}$

对连续型数据：① 离散化
② 假设 $X_i$ 有分布.

# Pros and Cons

- Where it is good
  - Spam filter : compute the posterior prob. distribution of frequently used words (convert to vector by word2vec)
  - Stable : for outliers and miss values
  - Robust : for uncorrelated features; $P(X_i|Y)$ is independent of $Y$ and thus has no effect on posterior probability
  - May outperform far more sophisticated alternatives even if conditional independency assumption is not satisfied
- Disadvantage
  - However, when conditional independency assumption is violated, performance of Naive Bayes can be poorer
  - Depends heavily on how well the parameter estimates are

垃圾邮件-文本处理
作了条件独立性假设

若不独立，则效果不好
极其依赖于估计的参数的好坏

# Model Assessment
# Confusion Matrix 混淆矩阵

- For two-class classification :
  - True Positive (TP) : both true label and predicted label are positive
  - True Negative (TN) : both true label and predicted label are negative
  - False Positive (FP) : true label is negative, but predicted label is positive
  - False Negative (FN) : true label is positive, but predicted label is negative

| 真实标签 | 预测结果 | |
|---|---|---|
| | 1（正例） | 0（反例） |
| 1（正例） | TP（真正例） | FN（假反例） |
| 0（反例） | FP（假正例） | TN（真反例） |

- $Accuracy=\frac{TP+TN}{TN+FN+FP+TP}$; not a good index when samples are imbalanced
- $Precision=\frac{TP}{TP+FP}$
- $Recall=\frac{TP}{TP+FN}$; important in medical diagnosis (sensitivity)
- $F$ score : $F_\beta=\frac{(1+\beta^2)Precision\times Recall}{\beta^2\times Precision+Recall}$; $\beta=1$, $F_1$ score
- $Specificity=\frac{TN}{TN+FP}$; recall for negative samples

$$F_\beta=\left(\frac{1}{\beta^2}+1\right)\frac{1}{\frac{1}{precision}+\frac{1}{recall}}$$

正确率：预测正确的样本比例
样本类别不均衡时不是一个很好的指标.

精度：真正例占所有预测为正样本的比例.（猜对多少）

召回率：正样本中被正确预测的比例.（命中多少）
又称灵敏度/命中率）

F值：对 precision 和 recall 作了加权的调和平均.
$\beta$ 越大越关心 recall.

特异度：负样本中被正确预测的比例.（负样本的召回率）

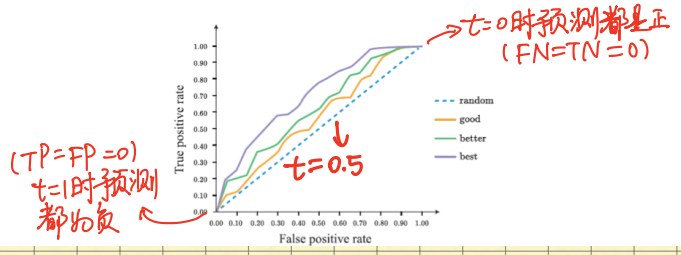
# Receiver Operating Characteristic (ROC) and Area Under ROC Curve (AUC)

用于解决样本在不同类别上的不均衡分布.

- Aim to solve class distribution imbalance problem
- Set different threshold $t$ for continuous predicted values (probability), e.g., if $P(Y=1|X=x_i) > t$, then $\hat{y}_i = 1$
- Compute TPR ($= \frac{TP}{TP+FN}$, or recall) vs. FPR($= \frac{FP}{FP+TN}$) for different $t$ and plot ROC curve
- The higher the ROC, the better the performance
- AUC : area under ROC, the larger the better, the more robust of the method for the change of $t$; very good if $> 0.75$

FPR: False Positive Rate 假正率
TPR: True Positive Rate 真正率

ROC曲线下区域面积(AUC)越大越好
一般来说, AUC > 0.75就认为比较理想.
AUC < 0.5不可取




# Cohen's Kappa Coefficient

- $\kappa = \frac{p_o - p_e}{1-p_e} = 1 - \frac{1-p_0}{1-p_e}$ measures the agreement between two raters
- $p_o$ is the accuracy (or the relative observed agreement)
- $p_e$ is the hypothetical probability of chance agreement, $p_e = \sum_{c=1}^{C} \frac{n_c^{pred}}{N} \frac{n_c^{true}}{N}$, where $n_c^{pred}$ is the number of samples predicted in class $c$, $n_c^{true}$ is the true number of samples in class $c$, $N$ is the total number of samples
- Eg : $p_o = \frac{20+15}{50} = 0.7$, $p_e = \frac{25}{50} \times \frac{20}{50} + \frac{25}{50} \times \frac{30}{50} = 0.5$, $\kappa = 0.4$

| | | Predicted Label | | |
|---|---|---|---|---|
| | | 1 | 0 | Total |
| True Label | 1 | 20 TP | 10 FN | 30 C |
| | 0 | 5 FP | 15 TN | 20 D |
| | Total | 25 A | 25 B | 50 N |

## the Values of Kappa Coefficient

- $\kappa \in [-1, 1]$
- $\kappa = 1$ : perfect agreement between two raters
- $\kappa = -1$ : completely disagreement
- $\kappa = 0$ : no agreement among the raters other than what would be expected by chance
- $\kappa < 0$ : worse than random
- $\kappa > 0$ : the result is meaningful, agree more as $\kappa$ gets larger
- $\kappa \geqslant 0.75$ : good performance
- $\kappa < 0.4$ : bad performance

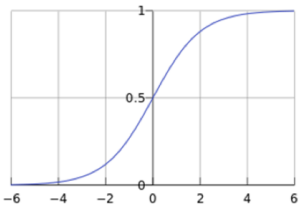
# Multiple Class Problem

- ROC and AUC are not well-defined
- Confusion matrix : $C \times C$, each entry means the number of samples in the intersection of the predicted class $i$ and the true class $j$
- Positive sample is the sample belonging to the class $i$, negative sample is the sample not belonging to the class $i$, so every sample could be positive or negative
- Convert to multiple 0-1 classification problems
- Precision and recall are the averages of that in the each 0-1 classification problem
- $F1$ score is still defined as the harmonic average of precision and recall

在多分类里没有良定义ROC和AUC.
将多分类看成多个二分类问题:
将每个类别都当作正样本,把此类当作负样本. 转化为一个多次的二分类问题. 计算每个二分类子问题的precision和recall, 再取平均得到多分类总问题的平均precision和recall.
F1值也即precision和recall的调和平均.

# Logistic Regression

- Not regression, but a classification method
- Connection with linear regression : $y = w_0 + w_1 x + \epsilon$, $y$ is binary (0 or 1) ; then $E(y|x) = P(y = 1|x) = w_0 + w_1 x$ ; but $w_0 + w_1 x$ may not be a probability
- Find a function to map it back to $[0, 1]$ : Sigmoid function $g(z) = \frac{1}{1+e^{-z}}$ with $z = w_0 + w_1 x_1 + \ldots + w_d x_d$




- Equivalently, $\log \frac{P(y=1|x)}{1-P(y=1|x)} = w_0 + w_1 x_1 + \ldots + w_d x_d$, logit transform $logit(z) = \log \frac{z}{1-z}$

不是回归，是分类方法！只是用了回归思想
逻辑回归可看作是一种没有隐藏层的神经网络模型。

线性回归有函数 $y = \vec{w}^T \vec{x} + \epsilon$.
将该函数通过 sigmoid 函数映射到 $[0,1]$.
并且计算概率：若大于某值则属于某类.
通过 logit 变换能将结果映射回去

# MLE for Logistic Regression

- The prob. distribution for two-class logistic regression model is

$$Pr(y = 1|\mathbf{X} = \mathbf{x}) = \frac{\exp(\mathbf{w}^T\mathbf{x})}{1 + \exp(\mathbf{w}^T\mathbf{x})},$$
$$Pr(y = 0|\mathbf{X} = \mathbf{x}) = \frac{1}{1 + \exp(\mathbf{w}^T\mathbf{x})}.$$

- Let $P(y = k|\mathbf{X} = \mathbf{x}) = p_k(\mathbf{x}; \mathbf{w})$, $k = 0$ or 1. The likelihood function is defined by $L(\mathbf{w}) = \prod_{i=1}^{n} p_{y_i}(\mathbf{x}_i; \mathbf{w})$
- MLE estimate of $\mathbf{w}$ : $\hat{\mathbf{w}} = \arg\max_{\mathbf{w}} L(\mathbf{w})$
- Solve $\nabla_{\mathbf{w}} \log L(\mathbf{w}) = 0$ by Newton-Raphson method

# Linear Discriminant Analysis (LDA) 线性判别分析

- Bayes Classifier amounts to know the class posteriors $\mathrm{P}(Y|\mathbf{X})$ for optimal classification : $k^* = \arg\max_k \mathrm{P}(Y = k|\mathbf{X})$
- Let $\pi_k = \mathrm{P}(Y = k)$ be the prior probability, $f_k(\mathbf{x}) = \mathrm{P}(\mathbf{X} = \mathbf{x}|Y = k)$ be the density function of samples in each class $Y = k$
- By Bayes theorem, $\mathrm{P}(Y|\mathbf{X} = \mathbf{x}) \propto f_k(\mathbf{x})\pi_k$ (Recall naive Bayes)
- Assume $f_k(x)$ is multivariate Gaussian : $f_k(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}|\mathbf{\Sigma}_k|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mu_k)^T\mathbf{\Sigma}_k^{-1}(\mathbf{x}-\mu_k)}$, with a common covariance matrix $\mathbf{\Sigma}_k = \mathbf{\Sigma}$, sufficient to look at the log-ratio

$$\log \frac{\mathrm{P}(Y = k|\mathbf{X} = \mathbf{x})}{\mathrm{P}(Y = l|\mathbf{X} = \mathbf{x})} = \log \frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k + \mu_l)^T\mathbf{\Sigma}^{-1}(\mu_k - \mu_l) + \mathbf{x}^T\mathbf{\Sigma}^{-1}(\mu_k - \mu_l)$$

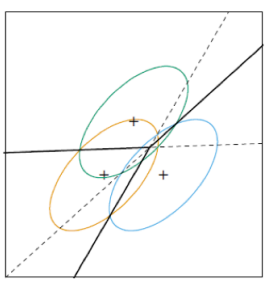
for the decision boundary between class $k$ and $l$

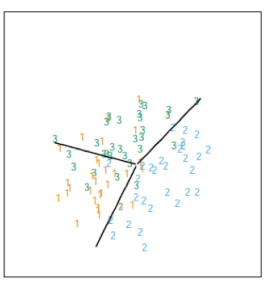
Σ是协方差矩阵

假设了多元高斯分布是等方差的。若没有假设，则算出的log是二次函数，此时叫QDA (Quadratic).
当该log>0时将样本分去类k中，否则分去类l中.

# Discriminant Rule

- Linear discriminant functions : $\delta_k(\mathbf{x}) = \mathbf{x}^T\mathbf{\Sigma}^{-1}\mu_k - \frac{1}{2}\mu_k^T\mathbf{\Sigma}^{-1}\mu_k + \log \pi_k$
- Then $\log \frac{\mathrm{P}(Y=k|\mathbf{X}=\mathbf{x})}{\mathrm{P}(Y=l|\mathbf{X}=\mathbf{x})} = \delta_k(\mathbf{x}) - \delta_l(\mathbf{x})$
- Decision rule : $k^* = \arg\max_k \delta_k(\mathbf{x})$
- Sample estimate of unknowns : $\hat{\pi}_k = N_k/N$, where $N = \sum_{k=1}^{K} N_k$, $\hat{\mu}_k = \frac{1}{N_k}\sum_{y_i=k} \mathbf{x}_i$, $\hat{\Sigma} = \frac{1}{N-K}\sum_{k=1}^{K}\sum_{y_i=k}(\mathbf{x}_i - \hat{\mu}_k)(\mathbf{x}_i - \hat{\mu}_k)^T$





判别规则：$k^* = \arg\max_k \delta_k(x)$
$\hat{\mu}_k$ 是类k的样本均值
$\hat{\pi}_k$ 是类k在总样本中的频率
$\hat{\Sigma}$ 是不同类的协方差矩阵求和后作平均

# Two-class LDA

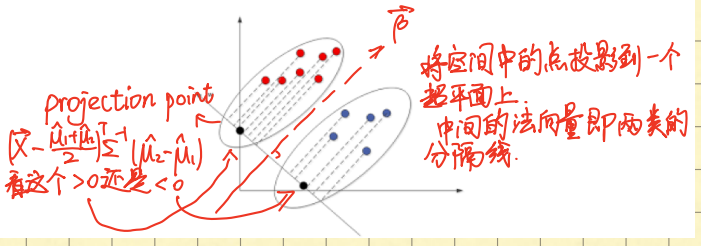
- LDA rule classifies to class 2 if

$$(\mathbf{x} - \frac{\hat{\mu}_1 + \hat{\mu}_2}{2})^T \mathbf{\Sigma}^{-1}(\hat{\mu}_2 - \hat{\mu}_1) + \log \frac{\hat{\pi}_2}{\hat{\pi}_1} > 0$$

修正分类边界

投影

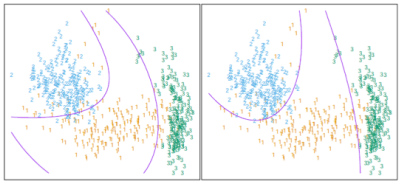
- Discriminant direction : $\beta = \mathbf{\Sigma}^{-1}(\hat{\mu}_2 - \hat{\mu}_1)$
- Bayes misclassfication rate $= 1 - \Phi(\beta^T(\mu_2 - \mu_1)/(\beta^T\mathbf{\Sigma}\beta)^{\frac{1}{2}})$, where $\Phi(x)$ is the Gaussian distribution function




LDA能将原来高维的数据降到一维，再做分类（或是其他操作）（线性降维）

# Other Variant

- Quadratic discriminant analysis (QDA) : $\delta_k(\mathbf{x}) = -\frac{1}{2}\log|\mathbf{\Sigma}_k| - \frac{1}{2}(\mathbf{x} - \mu_k)^T\mathbf{\Sigma}_k^{-1}(\mathbf{x} - \mu_k) + \log \pi_k$
- Regularized discriminant analysis : $\hat{\mathbf{\Sigma}}_k(\alpha) = \alpha\hat{\mathbf{\Sigma}}_k + (1 - \alpha)\hat{\mathbf{\Sigma}}$
- Computations for LDA :
  1. Sphere the data with respect to $\hat{\mathbf{\Sigma}} = \mathbf{UDU}^T$ : $\mathbf{X}^* = \mathbf{D}^{-\frac{1}{2}}\mathbf{U}^T\mathbf{X}$. Then the common covariance estimate of $\mathbf{X}^*$ is $\mathbf{I}_p$
  2. Classsify to the closest class centroid in the transformed space, taking into account of the class prior probabilities $\pi_k$'s
- Reduced-Rank LDA : see dimensionality reduction

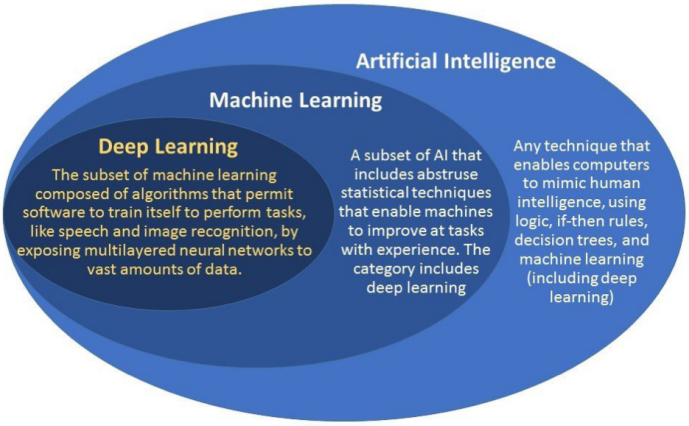


若每个类别的方差不等时，Discriminant direction会是二次的

RDA是LDA和QDA的加权平均，通过调整$\alpha$能平衡拟合效果.

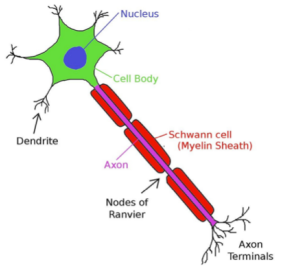
LDA可通过特征值分解计算.（当$\Sigma$比较大时）

# What is Deep Learning



# Deep Learning

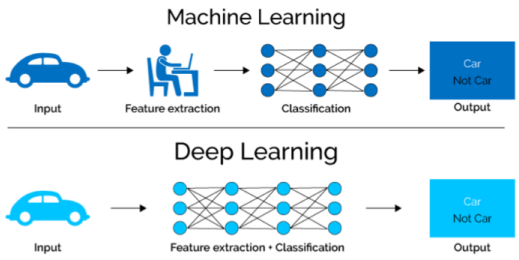
- Deep learning is a sub field of Machine Learning that very closely tries to mimic human brain's working using neurons.
- These techniques focus on building Artificial Neural Networks (ANN) using several hidden layers.
- There are a variety of deep learning networks such as Multilayer Perceptron (MLP), Autoencoders (AE), Convolution Neural Network (CNN), Recurrent Neural Network (RNN).




用二进制信号模拟神经元行为
把连续信号变成二值(sigmoid函数)
ANN = 线性变换 + Sigmoid 激活

# ML vs DL

- In Machine Learning, the features need to be identified by an domain expert.
- In Deep Learning, the features are learned by the neural network.

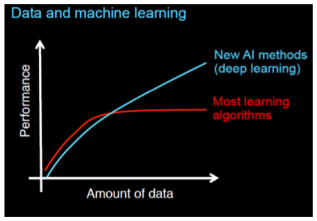



机器学习需要专家来提取特征.
深度学习是由神经网络来提取.

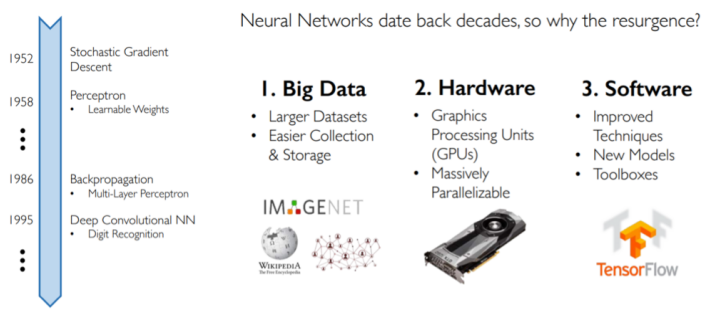
信号在神经元中传播和激活相当于特征提取.

# Why Deep Learning is Growing?

- Processing power needed for Deep learning is readily becoming available using GPUs, Distributed Computing and powerful CPUs.
- Moreover, as the data amount grows, Deep Learning models seem to outperform Machine Learning models.
- Explosion of features and datasets.
- Focus on customization and real time decisioning.

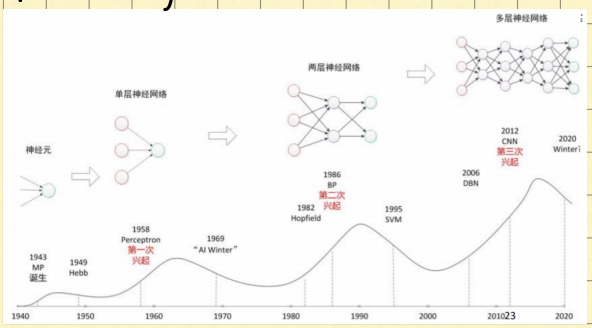



很好的计算资源(GPU、CPU、分布式)



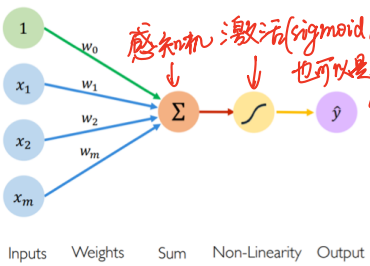

① 数据量
② 硬件提升
③ 软件要多.

# History



# The Perceptron: Forward Propagation

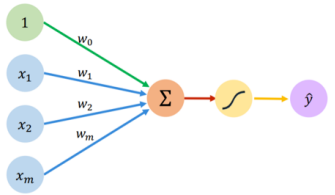
感知机：前向传播




$$\hat{y} = g\left(w_0 + \sum_{i=1}^{m} x_i w_i\right),$$

- $\hat{y}$ is the Output,
- $g$ is a Non-linear activation function,
- $w_0$ is the Bias.

Inputs Weights Sum Non-Linearity Output



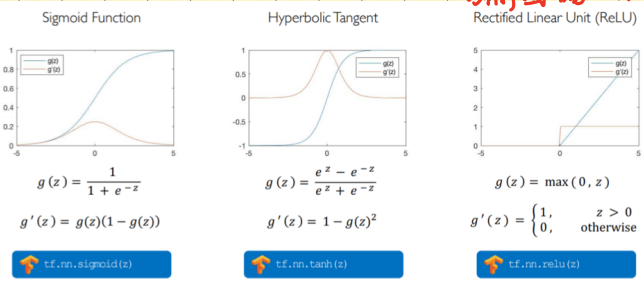

$$\hat{y} = g\left(w_0 + \mathbf{X}^T\mathbf{W}\right), \text{where}$$

$$\mathbf{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \text{ and } \mathbf{W} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}.$$

Inputs Weights Sum Non-Linearity Output

# Common Activation Functions

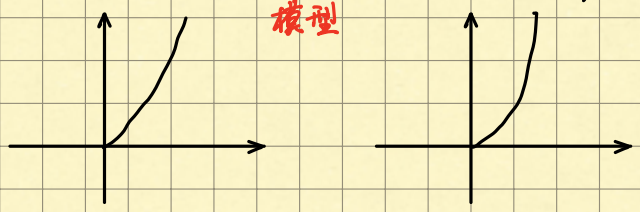
多用于图像识别



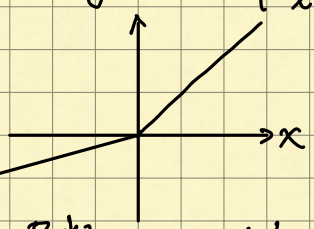

Note all activation functions are nonlinear.

激活函数是模拟数据生成过程

$ReQU = \max\{0, x^2\}$ 多用于物理模型 $ReCU = \max\{0, x^3\}$

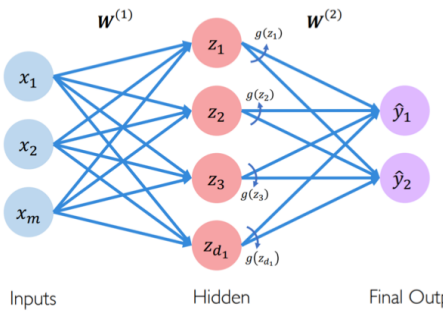


leakey $ReLU = \begin{cases} -\epsilon x, & x<0 \\ x, & x>0 \end{cases}$




层数看中间有多少层神经元.

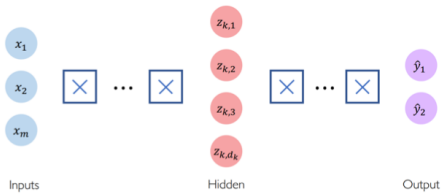
# Single (hidden) Layer Neural Network




$$z_i = w_{0,i}^{(1)} + \sum_{j=1}^{m} x_j w_{j,i}^{(1)}, \qquad \hat{y}_i = g\left(w_{0,i}^{(2)} + \sum_{j=1}^{d_1} g(z_j) w_{j,i}^{(2)}\right).$$

# Deep Neural Network

多层即将单层重复多次.




$$z_{k,i} = w_{0,i}^{(k)} + \sum_{j=1}^{d_{k-1}} g\left(z_{k-1,j}\right) w_{j,i}^{(k)}.$$

Theorem (Universal approximation theorem (Cybenko 1980, 1989))

1. *Any function can be approximated by a three-layer neural network within sufficiently high accuracy.*
2. *Any bounded continuous function can be approximated by a two-layer neural network within sufficiently high accuracy.*

该定理存在性证明而非构造性证明

不知道怎么构造、逼近函数精度、多少不知.

三层NN能逼近任意函数

两层NN能逼近任意有界连续函数

# Loss Optimization

We want to find the network weights that achieve the lowest loss

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\left(f\left(x^{(i)}; \mathbf{W}\right), y^{(i)}\right),$$

where $\mathcal{L}\left(f\left(x^{(i)}; \mathbf{W}\right), y^{(i)}\right)$ is the loss function we defined according to the specific problem to measure the differences between output state $f\left(x^{(i)}; \mathbf{W}\right)$ and reference state $y^{(i)}$. It also can be written as

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}} C(\mathbf{W}).$$

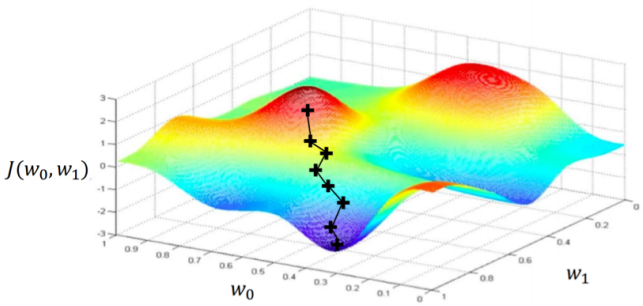
Remember

$$\mathbf{W} = \left\{\mathbf{W}^{(0)}, \mathbf{W}^{(1)}, \cdots\right\}.$$

对人求导

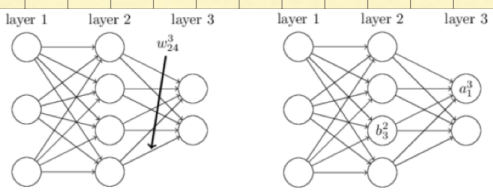
# Grandient Decent

We can use Gradient Decent algorithm to find the optimal parameter $\mathbf{W}$.




Note that we should calculate $\frac{\partial C}{\partial \mathbf{W}}$ to update $\mathbf{W}$.

# Notations




第l层的权重
第l层的第j个神经元和第(l-1)层的第k个神经元的连接.

- $w_{jk}^l$ is the weight for the connection from the $k^{th}$ neuron in the $(l-1)^{th}$ layer to the $j^{th}$ neuron in the $l^{th}$ layer.
- for brevity $b_j^l = w_{j0}^l$ is the bias of the $j^{th}$ neuron in the $l^{th}$ layer.
- $a_j^l$ for the activation of the $j^{th}$ neuron in the $l^{th}$ layer $z_j^l$.

z是激活前, a是激活后

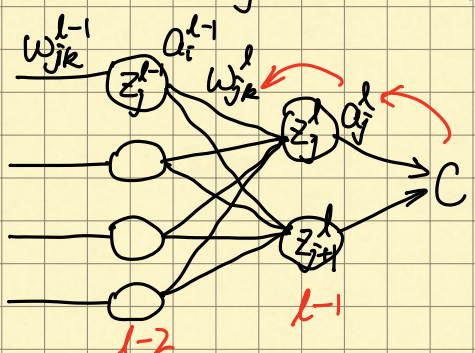
$$a_j^l = g(z_j^l) = g\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right)$$

链式法则

目标函数.

残差 $\delta_j^l = \frac{\partial C}{\partial z_j^l}$

每层的神经元个数是超参数.




对 $l-1$ 层: $\frac{\partial C}{\partial w_{jk}^{l-1}} = \sum_{j=1}^{n} \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial w_{jk}^{l-1}} = \sum_j \frac{\partial C}{\partial a_j^l} \frac{\partial a_j^l}{\partial z_j^l} a_k^{l-1}$ → $\delta_j^l$

每一层: $z_j^l = \sum_k w_{jk}^l a_{jk}^{l-1}$

$\delta_j^l \to \sum w_{ji}^l \sigma'(z_i^{l-1}) \delta_j^l \to \delta_i^{l-1} \to \delta_m^{l-2}$

$\frac{\partial C}{\partial z_j^l} = \frac{\partial C}{\partial a_j^l} \frac{\partial a_j^l}{\partial z_j^l}$ ↓ $\frac{\partial z_j^l}{\partial w_{jk}^l} = a_k^{l-1}$

对 $l-2$ 层: $\frac{\partial C}{\partial w_{im}^{l-1}} = \sum \frac{\partial C}{\partial z_i^{l-1}} \frac{\partial z_i^{l-1}}{\partial w_{im}^{l-1}} = \sum \frac{\partial C}{\partial a_i^{l-1}} \frac{\partial a_i^{l-1}}{\partial z_i^{l-1}} a_m^{l-2} = \sum \frac{\partial C}{\partial z_j^l} \frac{\partial z_j^l}{\partial a_i^{l-1}} \sigma'(z_i^{l-1}) a_m^{l-2}$

$= \sum \delta_j^l w_{ji}^l \sigma'(z_i^{l-1}) a_m^{l-2}$ → $\delta_i^{l-1}$

# 4 fondamental equations

这里$\sigma'$即是上面的激活函数$g$.

We first define the error $\delta_j^l$ of neuron $j$ in layer by

$$\delta_j^l \equiv \frac{\partial C}{\partial z_j^l},$$

and we give the four fundamental equations of back propagation :

$$\delta^L = \nabla_a C \odot \sigma'\left(z^L\right) \qquad (BP1)$$

$$\delta^l = \left(\left(w^{l+1}\right)^T \delta^{l+1}\right) \odot \sigma'\left(z^l\right) \qquad (BP2)$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \qquad (BP3)$$

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l \qquad (BP4)$$

## An equation for the error in the output layer (BP1)

目标函数对最后一层求导即为残差，链式法则.
(C) (激活后的)

The components of $\delta^L$ are given by

$$\delta^L = \nabla_a C \odot \sigma'\left(z^L\right) \qquad (BP1)$$

Démonstration.

$$\begin{aligned}\delta_j^L &= \frac{\partial C}{\partial z_j^L} \\ &= \sum_k \frac{\partial C}{\partial a_k^L}\frac{\partial a_k^L}{\partial z_j^L} \\ &= \frac{\partial C}{\partial a_j^L}\frac{\partial a_j^L}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L}\frac{\partial \sigma\left(z_j^L\right)}{\partial z_j^L} \\ &= \frac{\partial C}{\partial a_j^L}\sigma'\left(z_j^L\right)\end{aligned}$$

□

## An equation for the error in the hidden layer (BP2)

$$\delta^l = \left(\left(w^{l+1}\right)^T \delta^{l+1}\right) \odot \sigma'\left(z^l\right) \qquad (BP2)$$

Démonstration.

$$\begin{cases} \delta_j^l = \frac{\partial C}{\partial z_j^l} = \sum_k \frac{\partial C}{\partial z_k^{l+1}}\frac{\partial z_k^{l+1}}{\partial z_j^l} = \sum_k \delta_k^{l+1}\frac{\partial z_k^{l+1}}{\partial z_j^l} \\ z_k^{l+1} = \left(\sum_i w_{ki}^{l+1}a_i^l\right) + b_k^{l+1} = \left(\sum_i w_{ki}^{l+1}\sigma\left(z_i^l\right)\right) + b_k^{l+1} \end{cases}$$

$$\Rightarrow \begin{cases} \delta_j^l = \sum_k \delta_k^{l+1}\frac{\partial z_k^{l+1}}{\partial z_j^l} \\ \frac{\partial z_k^{l+1}}{\partial z_j^l} = w_{kj}^{l+1}\sigma'\left(z_j^l\right) \end{cases} \Rightarrow \delta_j^l = \sum_k \delta_k^{l+1} w_{kj}^{l+1}\sigma'\left(z_j^l\right)$$

□

## The change of the cost with respect to any bias (BP3)

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \qquad (BP3)$$

Démonstration.

$$\begin{cases} \frac{\partial C}{\partial b_j^l} = \sum_k \frac{\partial C}{\partial z_k^l}\frac{\partial z_k^l}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial b_j^l} \\ z_j^l = \left(\sum_k w_{jk}^l d_k^{l-1}\right) + b_j^l \Rightarrow \frac{\partial z_k^l}{\partial b_j^l} = 1 \end{cases} \Rightarrow \frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l}\cdot 1 = \delta_j^l.$$

□

## The change of the cost with respect to any weight (BP4)

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l \qquad (BP4)$$

Démonstration.

$$\begin{cases} \frac{\partial C}{\partial w_{jk}^l} = \sum_i \frac{\partial C}{\partial z_i^l}\frac{\partial z_i^l}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{jk}^l} \\ z_j^l = \left(\sum_m w_{jm}^l a_m^{l-1}\right) + b_j^l \Rightarrow \frac{\partial z_j^l}{\partial w_{jk}^l} = a_k^{l-1} \end{cases}$$

$$\Rightarrow \frac{\partial C}{\partial w_{jk}^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{jk}^l} = \delta_j^l a_k^{l-1}$$

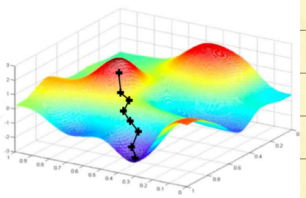
# Back Propagation procedure 反向传播进程

- 1. Input $x$ : Set the corresponding activation $a^1$ for the input layer.
- 2. Feedforward : For each $l = 2, 3, \ldots, L$ compute $z^l = w^l a^{l-1} + b^l$ and $a^l = \sigma\left(z^l\right)$.
- 3. Output error $\delta^L$ : Compute the vector $\delta^L = \nabla_a C \odot \sigma'\left(z^L\right)$.
- 4. Backpropagate the error : For each $l = L-1, L-2, \ldots, 2$ compute $\delta^l = \left(\left(w^{l+1}\right)^T \delta^{l+1}\right) \odot \sigma'\left(z^l\right)$.
- 5. Output : The gradient of the cost function is given by $\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l$ and $\frac{\partial C}{\partial b_j^l} = \delta_j^l$.

# Gradient Descent

**Algorithm**

1. Initialize weights randomly $\sim \mathcal{N}(0, \sigma^2)$
2. Loop until convergence:
3. Compute gradient, $\frac{\partial J(W)}{\partial W}$ ⇗即上面的C
4. Update weights, $W \leftarrow W - \eta \frac{\partial J(W)}{\partial W}$
5. Return weights



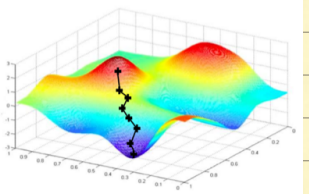
J可能不是凸函数. 梯度下降可能会到局部最优, 需优化算法. (因为中间传播非线性)

对每个样本都要算

# Stochastic Gradient Descent

**Algorithm**

1. Initialize weights randomly $\sim \mathcal{N}(0, \sigma^2)$
2. Loop until convergence:
3. Pick single data point $i$
4. Compute gradient, $\frac{\partial J_i(W)}{\partial W}$
5. Update weights, $W \leftarrow W - \eta \frac{\partial J(W)}{\partial W}$
6. Return weights

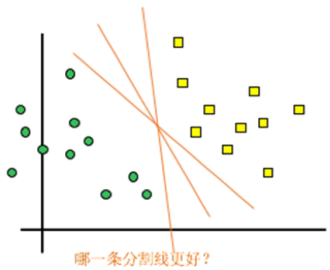


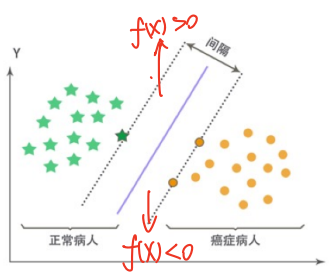
更快

- Mini-batches lead to fast training!
- Can parallelize computation + achieve significant speed increases on GPUs.

# Support Vector Machine (SVM)

- Use hyperplane to separate data : maximize margin
- Can deal with low-dimensional data that are not linearly separated by using kernel functions
- Decision boundary only depends on some samples (support vectors)




用最合适的直线使两部分数据分开, 间隔越大越好

# Linear SVM

- Training data : $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, $y_i \in \{-1, +1\}$
- Hyperplane : $S = \mathbf{w}^T\mathbf{x} + b$; decision function : $f(\mathbf{x}) = \operatorname{sign}(\mathbf{w}^T\mathbf{x} + b) = \begin{cases} -1, & \mathbf{w}^T\mathbf{x}+b<0 \\ 0, & \mathbf{w}^T\mathbf{x}+b=0 \\ 1, & \mathbf{w}^T\mathbf{x}+b>0 \end{cases}$

  $\mathbf{w}^T\mathbf{x}+b$ 与 $f(\mathbf{x})$ 同号

$$\left.\begin{aligned} f(\mathbf{x}_i) > 0 \Leftrightarrow y_i = 1 \\ f(\mathbf{x}_i) < 0 \Leftrightarrow y_i = -1 \end{aligned}\right\} \Rightarrow y_i f(\mathbf{x}_i) > 0$$

- Geometric margin between a point and hyperplane : $r_i = \frac{y_i(\mathbf{w}^T\mathbf{x}_i+b)}{\|\mathbf{w}\|_2}$ → 乘上 $y_i$ 是为了保证是正数，等价于 $|\mathbf{w}^T\mathbf{x}_i+b|$
- Margin between dataset and hyperplane : $\min_i r_i$
- Maximize margin : $\max_{\mathbf{w},b} \min_i \frac{y_i(\mathbf{w}^T\mathbf{x}_i+b)}{\|\mathbf{w}\|_2}$

最大化间隔　每个点到分割线的距离

由 f 来决定决策边界

目标要使得 $y_i$ 与 $f(\vec{x}_i)$ 同号，即 $y_i f(\vec{x}_i) > 0$

用 sign 函数来分类，$y_i f(\vec{x}_i)$ 来看分类是否正确，间隔不仅表示是否正确，还能表示确信度。

几何间隔：$r_i = y_i \frac{\mathbf{w}^T\mathbf{x}_i+b}{\|\mathbf{w}\|_2}$，

函数间隔 $\hat{r}_i = y_i(\mathbf{w}^T\mathbf{x}_i+b)$

关系：$r_i = \frac{\hat{r}_i}{\|\mathbf{w}\|_2}$

# Formulation as Constrained Optimization

- Without loss of generality, let $\min_i y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$ (multiply $\mathbf{w}$ and $b$ by the same proper constant)　最小等于 1，即原本 ≥1
- Maximize margin is equivalent to

$$\max_{\mathbf{w},b} \frac{1}{\|\mathbf{w}\|_2}, \quad \text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geqslant 1, i = 1, \ldots, n$$

- Further reduce to　最大化原数 = 最小化倒数

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2, \quad \text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geqslant 1, i = 1, \ldots, n$$

- This is primal problem : quadratical programming with linear constraints, computational complexity is $O(p^3)$ where $p$ is dimension

在分类时，需要优化　最大化几何间隔

$$\max_{w,b} \gamma \quad \text{s.t. } y_i \frac{\mathbf{w}^T\mathbf{x}_i+b}{\|\mathbf{w}\|_2} \geq \gamma, \forall i$$

→ 对每个点总大于它们之中最小的几何间隔

即 $\max_{w,b} \frac{\hat{\gamma}}{\|\mathbf{w}\|_2}$ $\text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i+b) \geq \hat{\gamma}, \forall i$

$\hat{\gamma}$ 的取值不影响最优化问题的解，两边同除 $\hat{\gamma}$ 即可。（相当于 w、b 乘上了一个系数）

也就是另外一方面说，超平面取 $\mathbf{w}^T\mathbf{x}+b=0$ 时，对左右两边的分类点有 $y_i(\mathbf{w}^T\mathbf{x}_i+b) \geq 1$（若不是 1，w、b 乘上一个系数即可）

凸二次规划问题：目标函数为二次函数且约束函数为仿射函数。

# Method of Lagrange Multipliers

- Introduce $\alpha_i \geqslant 0$ as Lagrange multiplier of constraint $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geqslant 1$
- Lagrange function : $\alpha_i$ 一定 ≥0　可正可负

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|_2^2 - \sum_{i=1}^{n} \alpha_i[y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1]$$

非负

- Since　约束 ≥0，要保证约束有效，等价于原问题

max 时用到 α 的符号

$$\max_{\alpha} L(\mathbf{w}, b, \alpha) = \begin{cases} \frac{1}{2}\|\mathbf{w}\|_2^2, & y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 \geqslant 0, \forall i \\ +\infty, & y_i(\mathbf{w}^T\mathbf{x}_i + b) - 1 < 0, \exists i \end{cases}$$

不参与后面的极小化

- Primal problem is equivalent to the minimax problem

$$\min_{\mathbf{w},b} \max_{\alpha} L(\mathbf{w}, b, \alpha) \Leftrightarrow \min_{w,b} \frac{1}{2}\|\mathbf{w}\|_2^2 \text{ s.t. } y_i(\mathbf{w}^T\mathbf{x}_i+b)-1 \geq 0$$

引入拉格朗日乘子 $\alpha_i \geq 0$.

在不等式约束下，拉格朗日乘子是有符号的。

# Dual Problem

- When slater condition is satisfied, $\min\max \Leftrightarrow \max\min$
- Dual problem : $\max_{\alpha} \min_{\mathbf{w},b} L(\mathbf{w}, b, \alpha)$
- Solve for inner minimization problem :

$$\nabla_{\mathbf{w}} L = 0 \Longrightarrow \mathbf{w}^* = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial L}{\partial b} = 0 \Longrightarrow \sum_i \alpha_i y_i = 0$$

→临界点条件

- Plug into $L$ : $L(\mathbf{w}^*, b^*, \alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_i y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$
- Dual optimization : 只依赖于两两样本的内积

$$\min_{\alpha} \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j) - \sum_i \alpha_i,$$

$$\text{s.t. } \alpha_i \geqslant 0, i = 1, \ldots, n, \sum_i \alpha_i y_i = 0$$

slater condition: ①函数严格凸
②最优解在可行域边界上.

# KKT conditions

- Three more conditions from the equivalence of primal and minimax problems

$$\begin{cases} \alpha_i^* \geqslant 0, \\ y_i((\mathbf{w}^*)^T \mathbf{x}_i + b^*) - 1 \geqslant 0, \\ \alpha_i^* [y_i((\mathbf{w}^*)^T \mathbf{x}_i + b^*) - 1] = 0. \end{cases}$$

还有 $\begin{cases} w^* = \sum_i \alpha_i y_i x_i \\ \sum_i \alpha_i y_i = 0 \end{cases}$

- These together with two zero derivative conditions form KKT conditions
- $\alpha_i > 0 \Rightarrow y_i(\mathbf{w}^T \mathbf{x}_i + b^*) = 1$
- Index set of support vectors $S = \{i | \alpha_i > 0\}$
- $b = y_s - \mathbf{w}^T \mathbf{x}_s = y_s - \sum_{i \in S} \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_s$
- More stable solution : $b = \frac{1}{|S|} \sum_{s \in S} \left( y_s - \sum_{i \in S} \alpha_i y_i \mathbf{x}_i^T \mathbf{x}_s \right)$

这5个条件即为KKT条件.

支持向量机只对边界上的点敏感

边界上的点就是支持向量.

# Sequential Minimal Optimization (SMO) Algorithm

- Invented by John C. Platt (1998)
- Coordinately optimize dual problem, select two variables and fix others, then dual problem reduces to one variable quadratic programming with positivity constraint
  1. Initially, choose $\alpha_i$ and $\alpha_j$
  2. Fix other variables, solve for $\alpha_i$ and $\alpha_j$
  3. Update $\alpha_i$ and $\alpha_j$, redo step 1 iteratively
  4. Stop until convergence
- How to choose $\alpha_i$ and $\alpha_j$ ? choose the pair far from KKT conditions the most
- Computational complexity $O(n^3)$
- Easy to generalize to high dimensional problem with kernel functions

优化离零比较远的$\alpha_i$与$\alpha_j$，将离零较近的$\alpha_i$与$\alpha_j$固定住(认为已经较优)

# Soft Margin

- When data are not linear separable, introduce slack variables (tolerance control of fault) $\xi_i \geqslant 0$
- Relax constraint to $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geqslant 1 - \xi_i$
- Primal problem :

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{s.t. } y_i(\mathbf{w}^T\mathbf{x}_i + b) \geqslant 1 - \xi_i, \xi_i \geqslant 0, i = 1, \ldots, n$$

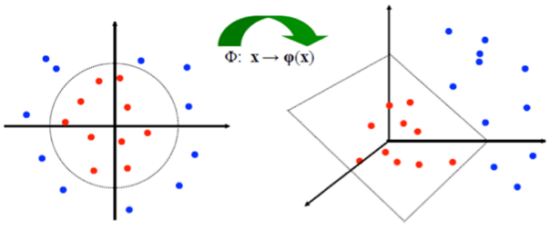
- Similar derivation to dual problem :

$$\min_{\alpha} \frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j(\mathbf{x}_i^T\mathbf{x}_j) - \sum_i \alpha_i,$$
$$\text{s.t. } 0 \leqslant \alpha_i \leqslant C, i = 1, \ldots, n, \sum_i \alpha_i y_i = 0$$

$\xi_i$ 能松弛分类边界，即调整边界之间的距离。

# Nonlinear SVM

- Nonlinear decision boundary could be mapped to linear boundary in high-dimensional space
- Modify objective function in dual problem : $\frac{1}{2}\sum_i\sum_j \alpha_i\alpha_j y_i y_j(\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)) - \sum_i \alpha_i$
- Kernel function as inner product : $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$




对非线性/高维的数据，可以将 SVM 中的 $\vec{x_i}$ 换成 $\phi(\vec{x_i})$

只要知道内积关系就可以做
决策函数只依赖于 kernel

# Kernel Methods 核方法

- Reduce effect of curse of dimensionality
- Different kernels lead to different decision boundaries
- Popular kernels :

| Kernel | Definition | Parameters |
|---|---|---|
| Polynomial | $(\mathbf{x}_1^T\mathbf{x}_2 + 1)^d$ | $d$ is positive integer |
| Gaussian | $e^{-\frac{\|\mathbf{x}_1-\mathbf{x}_2\|^2}{2\delta^2}}$ | $\delta > 0$ |
| Laplacian | $e^{-\frac{\|\mathbf{x}_1-\mathbf{x}_2\|}{\delta^2}}$ | $\delta > 0$ |
| Fisher | $\tanh(\beta\mathbf{x}_1^T\mathbf{x}_2 + \theta)$ | $\beta > 0, \theta < 0$ |

Kernel 一定是对称 & 半正定的
核函数减少了维度（通过将特征映射到高维空间）
不同的核函数有不同的决策边界.

高斯核函数也称为高斯径向基函数

# Pros and Cons

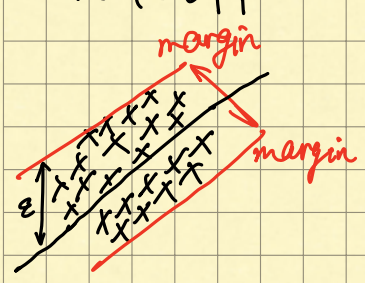
- Where it is good
  - Applications in pattern recognition : text classification, face recognition
  - Easy to deal with high-dimensional data with kernels
  - Robust (only depends on support vectors), and easy to generalize to new dataset
- Disadvantage
  - Poor for ultra high dimensional data
  - Low computational efficiency for nonlinear SVM when sample size is large
  - Poor interpretability without probability

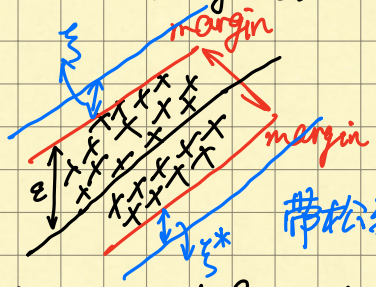
用 kernel 能到高维

对超高维难以选择 kernel.
SVM 缺乏概率曲线，解释性差.

SVM 既可做分类也可做回归.

# SVR (Support Vector Regression) 支持向量回归.




带松弛的 ξ

maximize the margins messured from the farthest points to the line 尽量使边界间距最大.

$$\max_{w,b} \frac{\max_i |y_i - wx_i - b|}{\sqrt{w^2+1}} \Leftrightarrow \min_{w,b} \|w\|_2^2 \quad \text{s.t. } |y_i - wx_i - b| \le \varepsilon, \forall i$$

→最大量 ε. 在带状区域内直线斜率最小.

带有松弛的 SVR: $\min_{w,b} \|w\|_2^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)$ s.t. $y_i - wx_i - b \le \varepsilon + \xi_i$ ① $wx_i + b - y_i \le \varepsilon + \xi_i^*$ ② for $\xi_i, \xi_i^* \ge 0$.

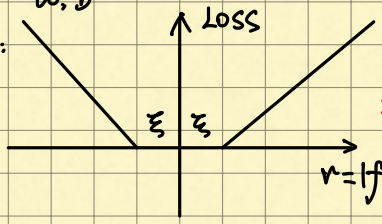
$$\Updownarrow$$

$$\min_{w,b} \frac{\lambda}{2}\|w\|_2^2 + \sum_{i=1}^{n} \max(\underbrace{|\underbrace{w^Tx_i + b}_{f} - y_i|}_{\xi_i} - \varepsilon, 0) \quad ①$$

residual

$$\left(\min \frac{\lambda}{2}\|w\|_2^2 + \sum_i \text{HingeLoss}(y_i f(x_i) - 1) \right.$$
$$\left. \| \quad \max(y_i f(x_i) - 1, 0) \right)$$

loss 函数图像:




在带宽内不惩罚
带宽外的点做惩罚

①② 都可以解.

Dual Problem: linear SVR: $y = \underbrace{\sum_{i=1}^{n}(\alpha_i - \alpha_i^*)x^Tx}_{w} + b$ ②

$\alpha_i$ and $\alpha_i^*$ are Lagrange multiplier of the two constraints ① and ②