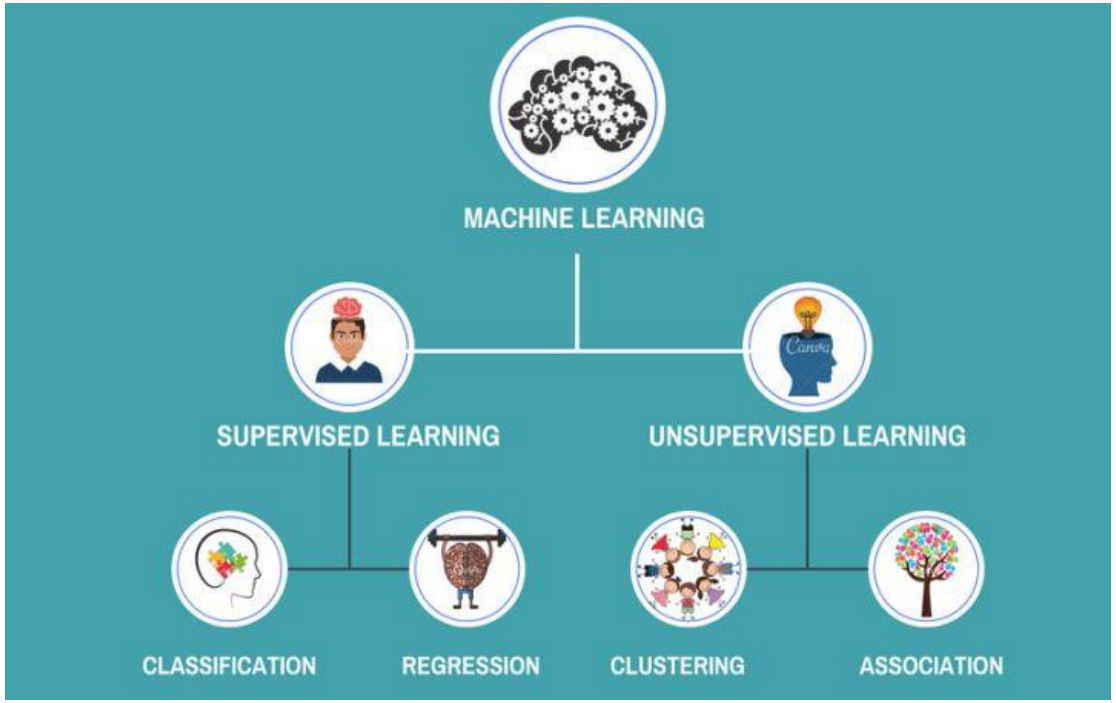
数学建模复习

机器学习

机器学习算法是在没有人为干涉的情况下，从大量的数据和历史经验中学习数据的结构并提升对某一目标的估计的算法。机器学习是实现人工智能的一种途径，它和数据挖掘有一定的相似性，也是一门多领域交叉学科，涉及概率论、统计学、逼近论、凸分析、计算复杂性理论等多门学科。机器学习可以分为监督学习和无监督学习两个大类。



监督学习

监督学习是指利用一组已知类别的样本，如防垃圾邮件系统中标注为“垃圾邮件”“非垃圾邮件”的样本，手写数字识别中标注为“1“，”2“，”3“，”4“等的样本，来调整模型的参数，使其达到所要求性能的过程。

监督学习算法解决的问题大致可以分为两类：

分类问题：预测某一样本所属的类别（离散的）。比如给定一个人（从数据的角度来说，是给出一个人的数据结构，包括：身高，年龄，体重等信息），然后判断其性别，或者是否健康。常见的分类算法有k邻近算法，决策树(ID3，C4.5，CART)，朴素贝叶斯，逻辑回归（Logistic Regression），支持向量机。

回归问题：预测某一样本的所对应的实数输出（连续的）。比如预测某一地区居民的平均身高。常用算法比如线性回归。

无监督学习

无监督学习是在未加标签的数据中找到隐藏结构。

无监督学习算法解决的问题大致分为三类：

关联分析：发现不同事物之间同时出现的概率。在购物篮分析中被广泛地应用。如果发现买面包的客户有百分之八十的概率买鸡蛋，那么商家就会把鸡蛋和面包放在相邻的货架上。例如Apriori算法，FP-growth算法。

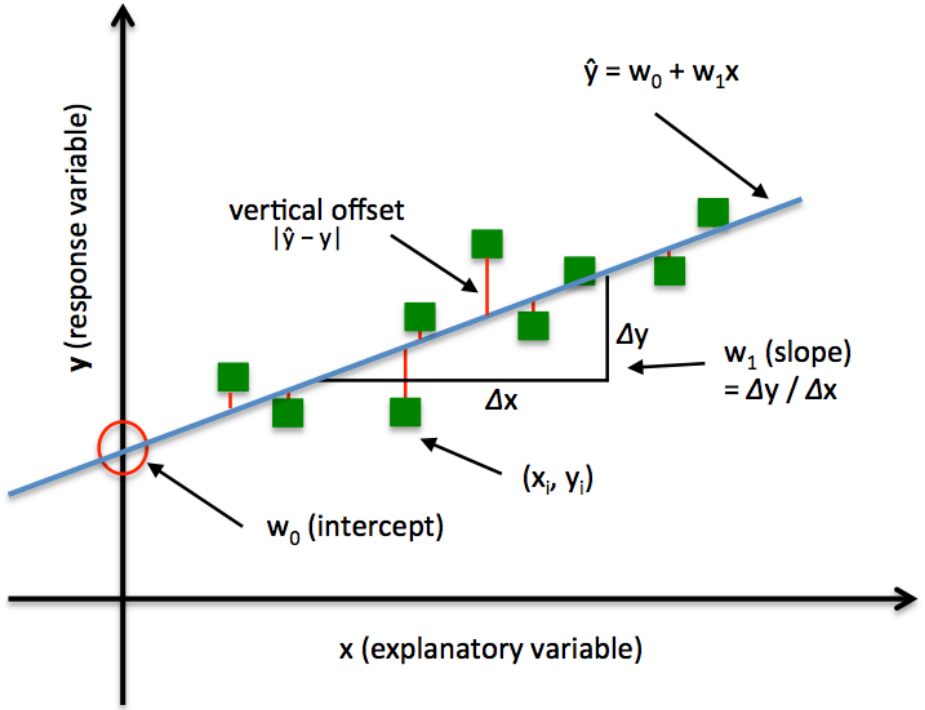
聚类问题：将相似的样本划分为一个簇（cluster）。与分类问题不同，聚类问题预先并不知道类别，自然训练数据也没有类别的标签。例如k-Means算法。

维度约减：指减少数据的维度同时保证不丢失有意义的信息。利用特征提取方法和特征选择方法，可以达到维度约减的效果。特征选择是指选择原始变量的子集。特征提取是将数据从高纬度转换到低纬度。主成分分析算法就是特征提取的方法。

线性回归

回归是监督学习的一个重要问题，回归用于预测输入变量和输出变量之间的关系。回归模型是表示输入变量到输出变量之间映射的函数。回归问题的学习等价于函数拟合：使用一条函数曲线使其很好的拟合已知函数且很好的预测未知数据。

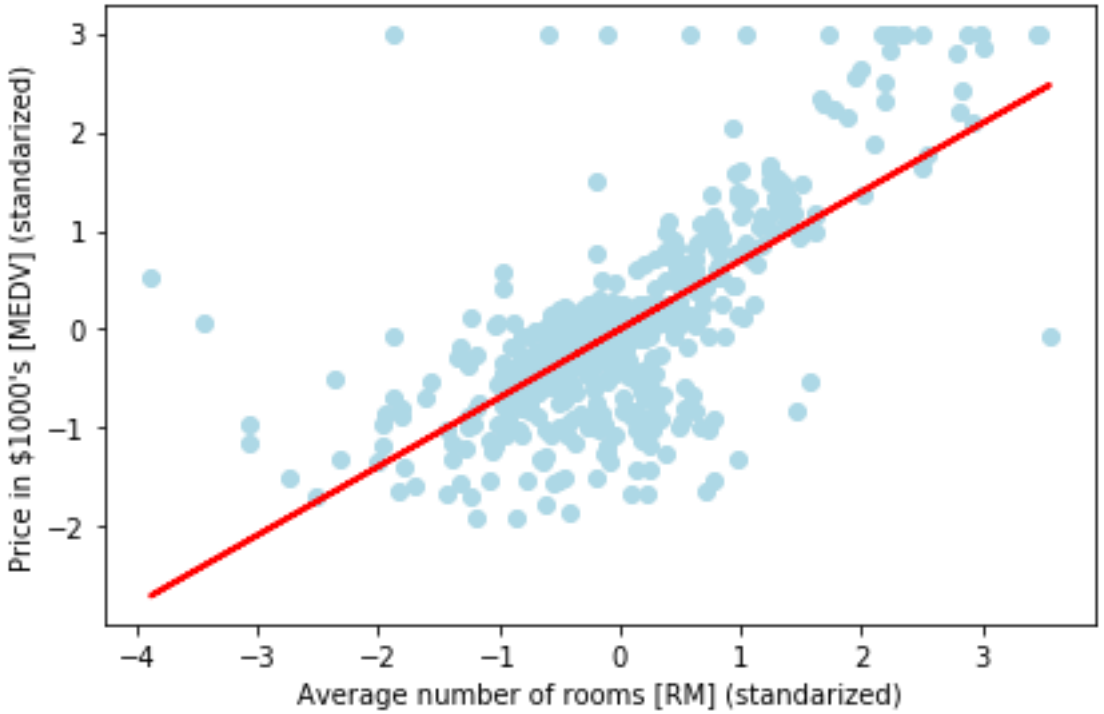
回归问题分为模型的学习和预测两个过程。基于给定的训练数据集构建一个模型，根据新的输入数据预测相应的输出。



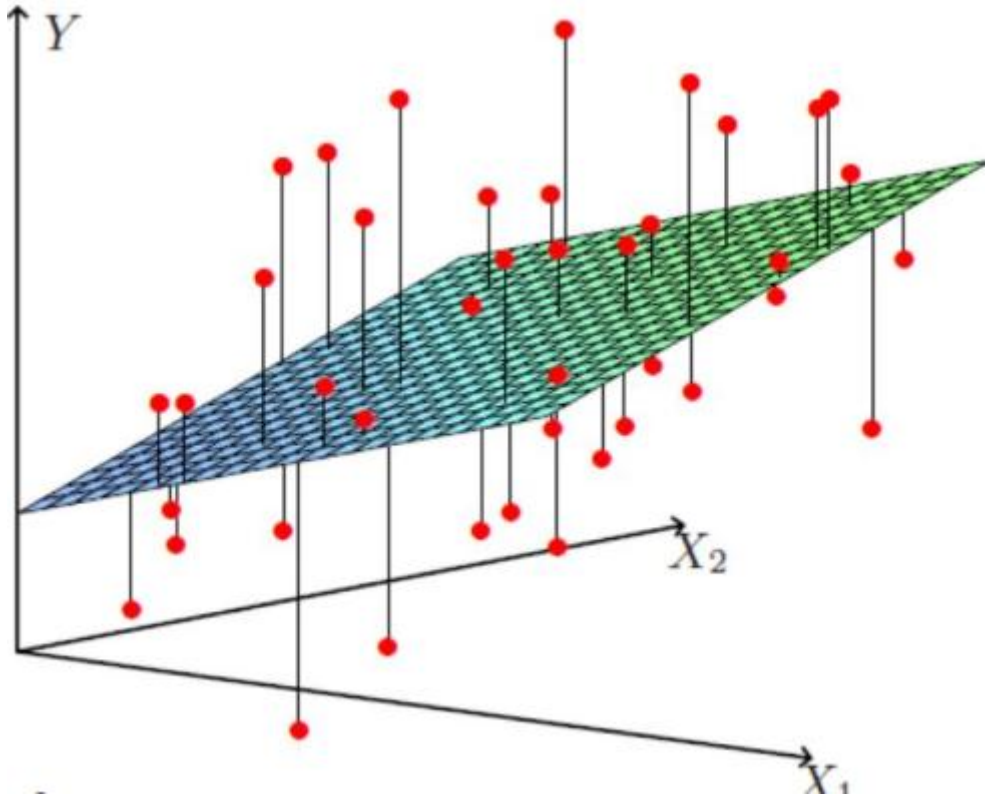
一元回归和多元回归

线性回归问题按照输入变量的个数可以分为一元回归和多元回归。

$$h_\theta(x) = \theta_0 + \theta_1 x$$



$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$



解析解

目标函数 $J(\theta)=\frac{1}{2}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right)-y^{(i)}\right)^2=\frac{1}{2}(X\theta-y)^T(X\theta-y)$

求导寻找驻点

$$\nabla_\theta J(\theta)=\nabla_\theta\left(\frac{1}{2}(X\theta-y)^T(X\theta-y)\right)=\nabla_\theta\left(\frac{1}{2}\left(\theta^T X^T-y^T\right)(X\theta-y)\right)$$

$$=\nabla_\theta\left(\frac{1}{2}\left(\theta^T X^T X\theta-\theta^T X^T y-y^T X\theta+y^T y\right)\right)$$

$$=\frac{1}{2}\left(2X^T X\theta-X^T y-\left(y^T X\right)^T\right)=X^T X\theta-X^T y\xrightarrow{\text{求驻点}}0$$

得到参数 θ 的解析式

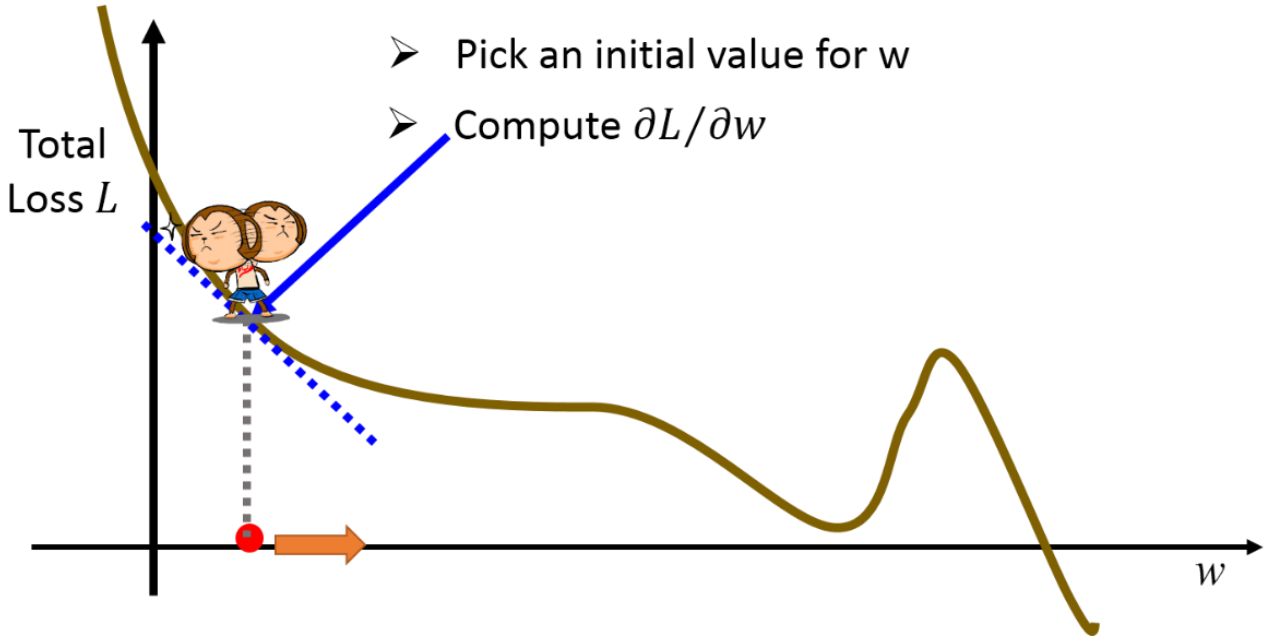
$$\theta=\left(X^T X\right)^{-1}X^T y$$

梯度下降法

思想
通过搜索方向和步长来对参数进行更新。其中搜索方向是目标函数在当前位置的负梯度方向。因为这个方向是最快的下降方向。步长是沿着这个搜索方向下降的大小。

沿着负梯度方向迭代，更新后的θ使得$J(\theta)$更小.

$$\theta = \theta - \alpha \cdot \frac{\partial J(\theta)}{\partial \theta}$$



梯度下降法

梯度方向：

$$
\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} \left(h_\theta(x) - y\right)^2 \\
&= 2 \cdot \frac{1}{2} \left(h_\theta(x) - y\right) \cdot \frac{\partial}{\partial \theta_j} (h_\theta(x) - y) \\
&= \left(h_\theta(x) - y\right) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{n} \theta_i x_i - y \right) \\
&= \left(h_\theta(x) - y\right) x_j
\end{aligned}
$$

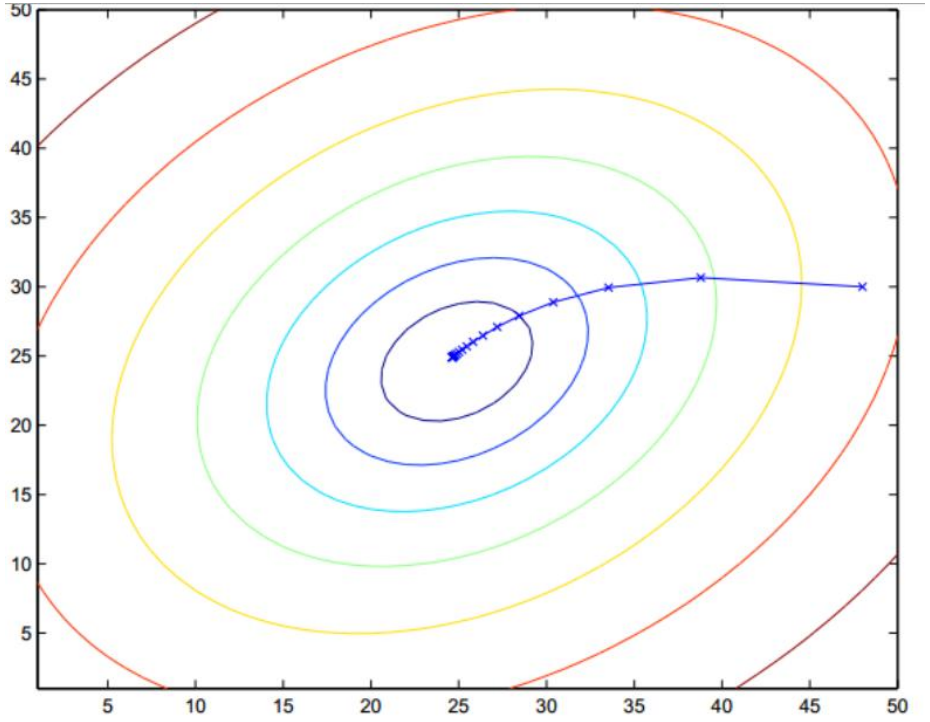
批量梯度下降算法

每次用所以样本进行更新

Repeat until convergence {

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$$

}



1. 最小化所有样本的损失函数，最终得到全局最优解。
2. 由于每次更新参数需要重新训练一次全部的样本，代价比较大，适用于小规模样本训练的情况。

随机梯度下降算法

每次用一个样本进行更新

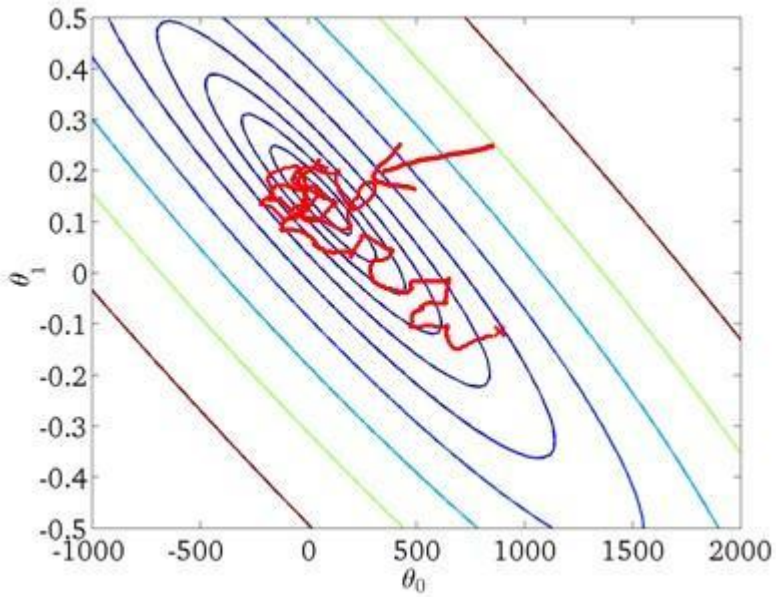
Loop {

for i=1 to m, {

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)}$$

}

}



1. 是最优化每个样本的损失函数。每一次迭代得到的损失函数不是每次向着全局最优的方向，但是大体是向着全局最优，最终的结果往往是在最优解的附近。
2. 当目标函数是凸函数的时候，结果一定是全局最优解。
3. 适合大规模样本训练的情况。

数据标准化

数据标准化是机器学习/数据挖掘的一项基础工作，是数据预处理的重要一步。样本各个特征往往具有不同的分布/取值范围，通过标准化将各个维度的特征值映射到相同区间，使得各特征值具有相同量纲，处于同一数量级。

常用的规范化有：

- 最小-最大标准化也称为归一化，是对原始数据的线性变换，将数据值映射到[0, 1]之间。

$$x^* = \frac{x - min}{max - min}$$

- z-score标准化也称标准差标准化，经过处理的数据的均值为0，标准差为1。

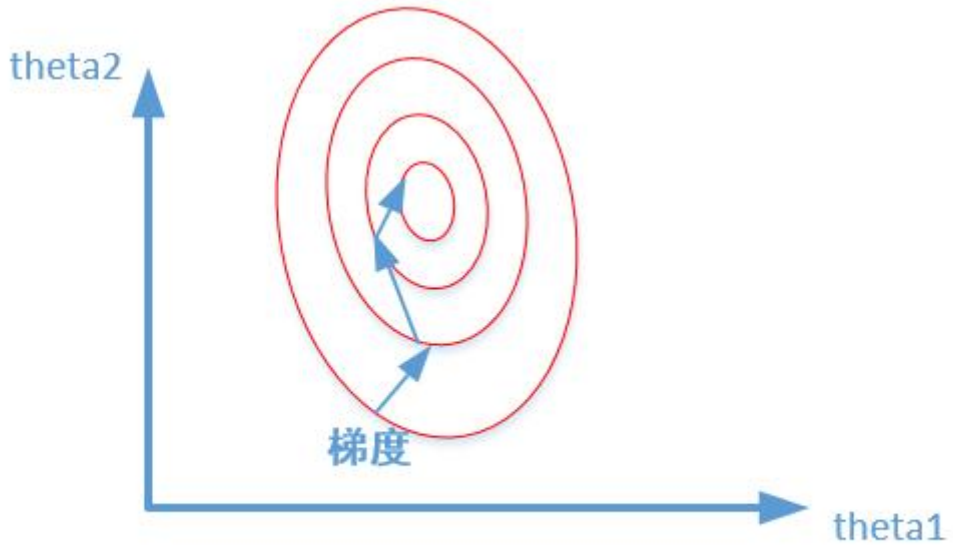
$$x^* = \frac{x - \bar{x}}{\sigma}$$

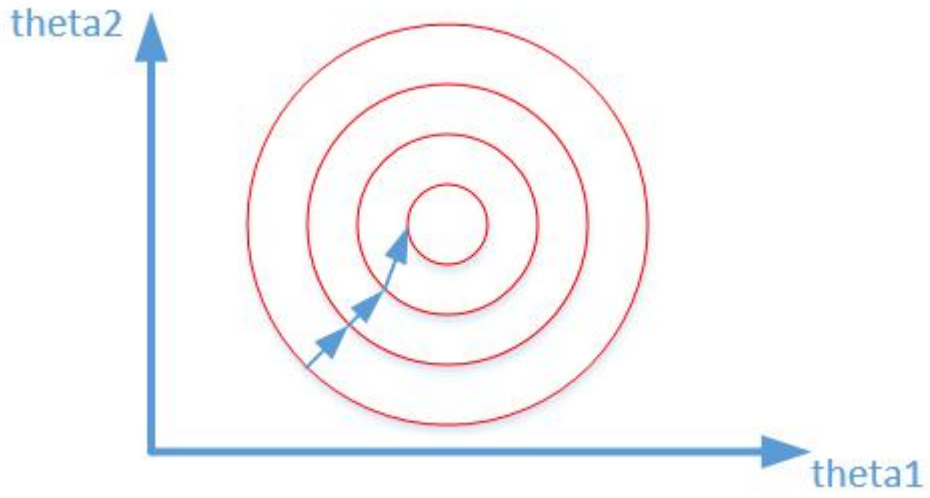
什么时候需要标准化

- 聚类时，因为聚类操作依赖于对类间距离和类内距离之间的衡量。如果一个变量的衡量标准高于其它变量，那么任何衡量标准都将受到该变量的过度影响。
- 主成分分析PCA降维操作之前。在PCA之前，对变量进行标准化至关重要。 这是因为PCA给那些方差较高的变量比那些方差小的变量赋予更多的权重。而标准化原始数据会产生相同的方差，因此高权重不会分配给具有较高方差的变量。
- KNN操作，原因类似于聚类。由于KNN需要用欧式距离去度量。标准化会让变量之间起着相同的作用。
- 在SVM中，使用所有跟距离计算相关的的kernel都需要对数据进行标准化。
- 在选择岭回归和Lasso时候，标准化是必须的。原因是正则化是有偏估计，会对权重进行惩罚。在量纲不同的情况，正则化会带来更大的偏差。

标准化的好处

数据标准化后，最优解的寻优过程明显会变得平缓，更容易正确的收敛到最优解。





回归模型评估指标

- SSE(和方差、误差平方和)：The sum of squares due to error
- MSE(均方差、方差)：Mean squared error 就是 SSE 的平均值
- RMSE(均方根、标准差)：Root mean squared error
- R-square(确定系数)：Coefficient of determination，代表着有多少百分比的数据被模型解释，越高代表模型拟合越好
- SSR：sum of square of the regression，预测数据与原始数据均值之差的平方和
- SST：total sum of square，即原始数据和均值之差的平方和

$$SSE = \sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})^2$$

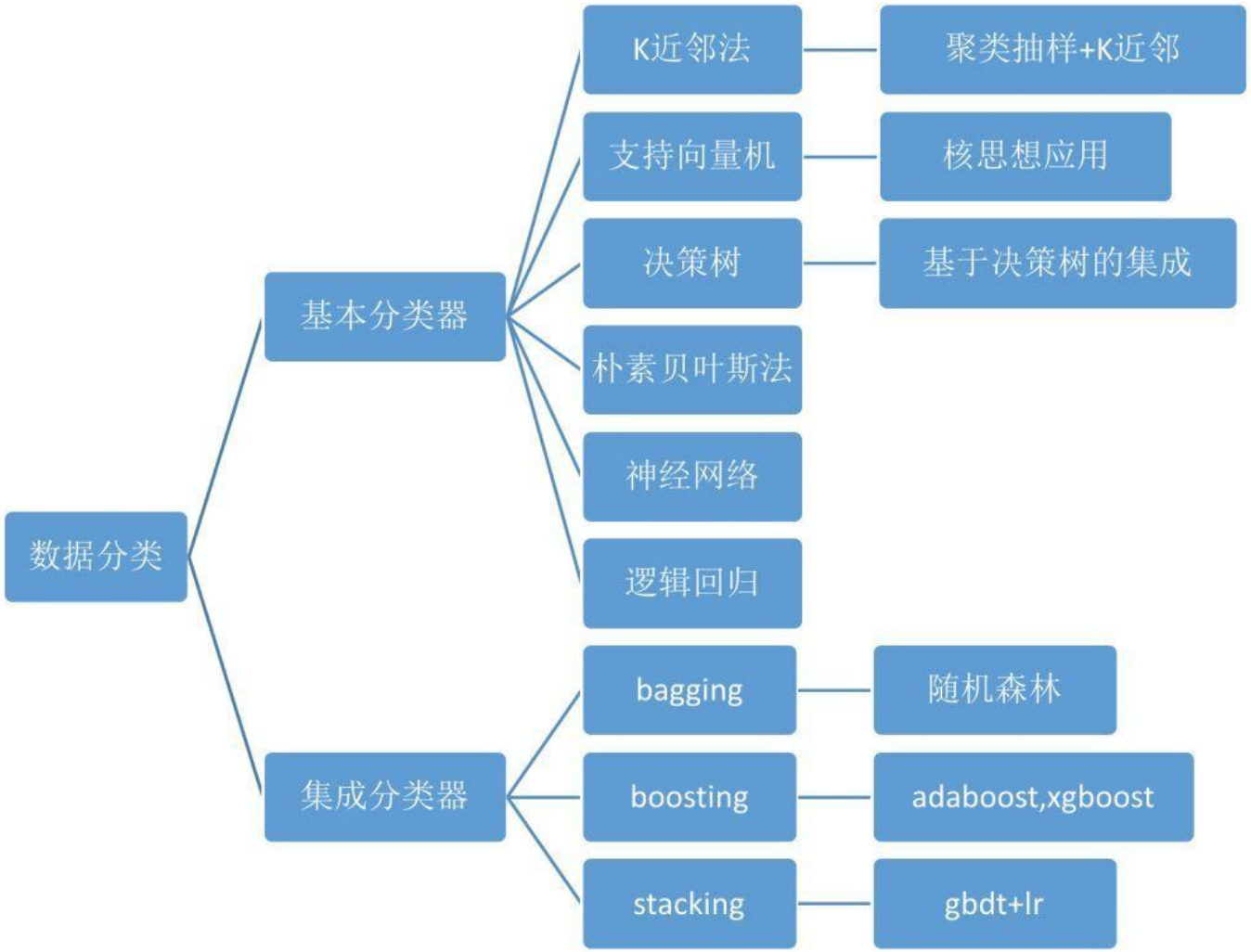
$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y^{(i)} - \hat{y}^{(i)})$$

$$R^2 = 1 - \frac{SSE}{SST} = 1 - \frac{MSE}{Var(y)}$$

$$SSR = \sum_{i=1}^{n}(\hat{y}^{(i)} - \bar{y}^{(i)})^2$$

$$SST = \sum_{i=1}^{n}(y^{(i)} - \bar{y}^{(i)})^2$$

分类算法


数据分类
基本分类器
K近邻法
聚类抽样+K近邻
支持向量机
核思想应用
决策树
基于决策树的集成
朴素贝叶斯法
神经网络
逻辑回归
集成分类器
bagging
随机森林
boosting
adaboost,xgboost
stacking
gbdt+lr

线性可分与线性不可分

- 对于一个数据集

$$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$$

- 其中

$$x_i \in X \subseteq \mathbb{R}^n; y_i \epsilon Y = \{-1, 1\}$$

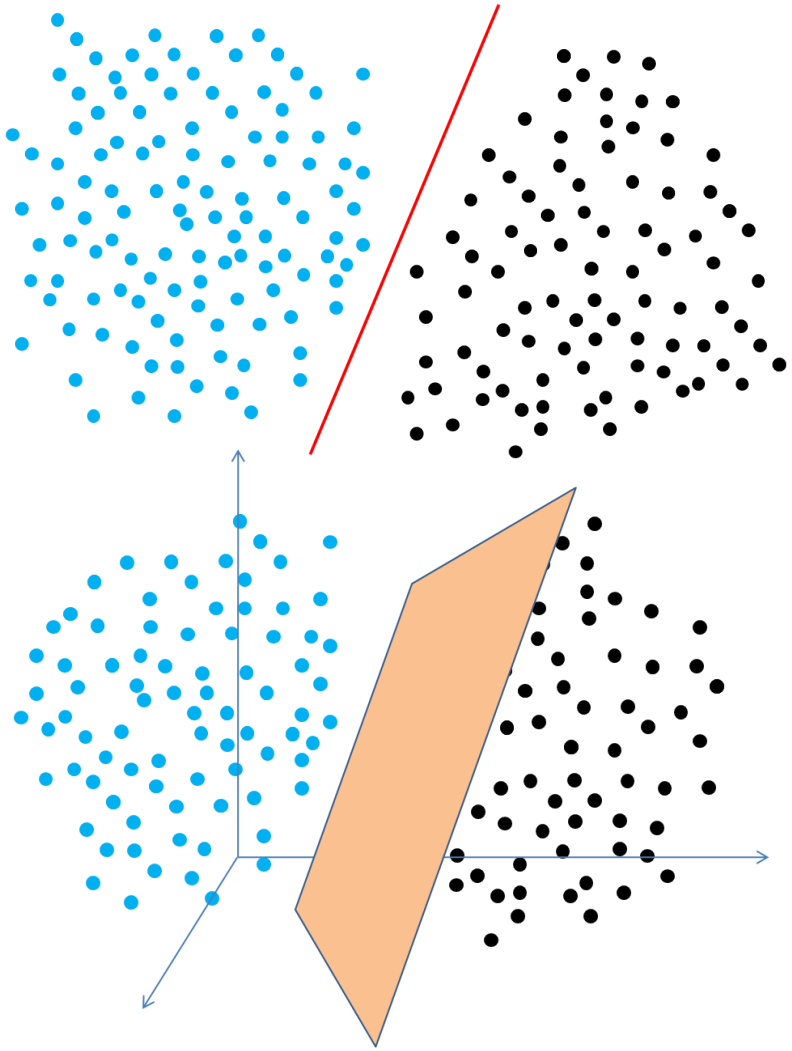
- 如果存在一个超平面Π能够将D中的正负样本点精确地划分到超平面两侧
- 亦即

$$\exists \Pi: w \cdot x + b = 0$$

- 使得

$$w \cdot x + b < 0 (\forall y_i = -1)$$
$$w \cdot x + b > 0 (\forall y_i = +1)$$

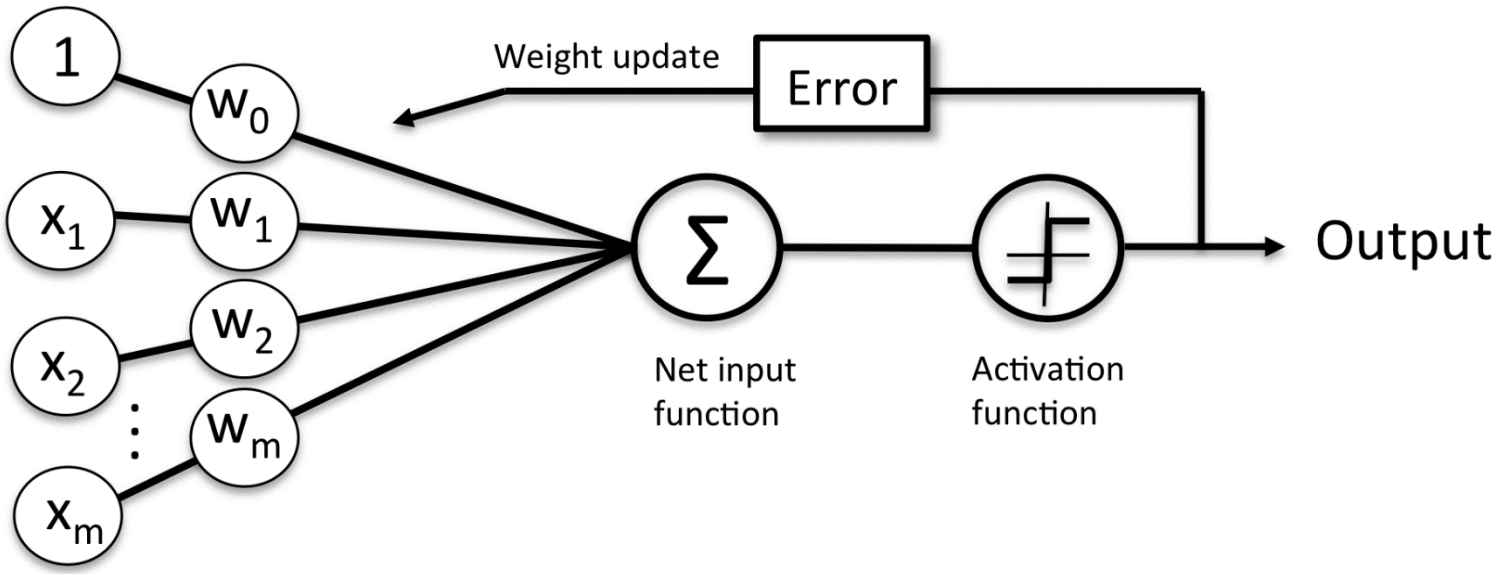
- 那么就称数据集D是线性可分的，否则，就称D线性不可分。



感知机

感知器（Perceptron），是神经网络中的一个概念，在1950s由Frank Rosenblatt第一次引入。感知器是形式最简单的前馈神经网络，是一种二元线性分类器, 把输入x（实数值向量）映射到输出值$f(x)$上（一个二元的值）。

$$f(x) = \begin{cases} +1 & \text{if } w \cdot x + b > 0 \\ -1 & \text{else} \end{cases}$$



学习算法

我们首先定义一些变量：

- x_j 表示n维输入向量中的第j项
- w_j 表示权重向量的第j项
- $f(x)$ 表示神经元接受输入 x 产生的输出
- 更进一步，为了简便我们可以令w^0 等于 b，x^0 等于 1。

感知器的学习通过对**所有训练实例**进行**多次迭代更新**的方式来建模。

令 $D_m = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$ 表示一个有 m 个训练实例的训练集。

我们损失函数的优化目标，就是期望使误分类的所有样本，到超平面的距离之和最小。所以损失函数定义如下：

$$L(w, b) = -\frac{1}{||w||} \sum_{x^{(i)} \in M} y^{(i)}(w \cdot x^{(i)} + b)$$

其中M集合是误分类点的集合。

采用随机梯度下降法，每次迭代权重向量以如下方式**更新**：对于 $D_m = \{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$ 中的每个 (x, y) 对，

$$w_j := w_j + \eta(y - f(x))x_j \quad (j = 1, \ldots, n)$$

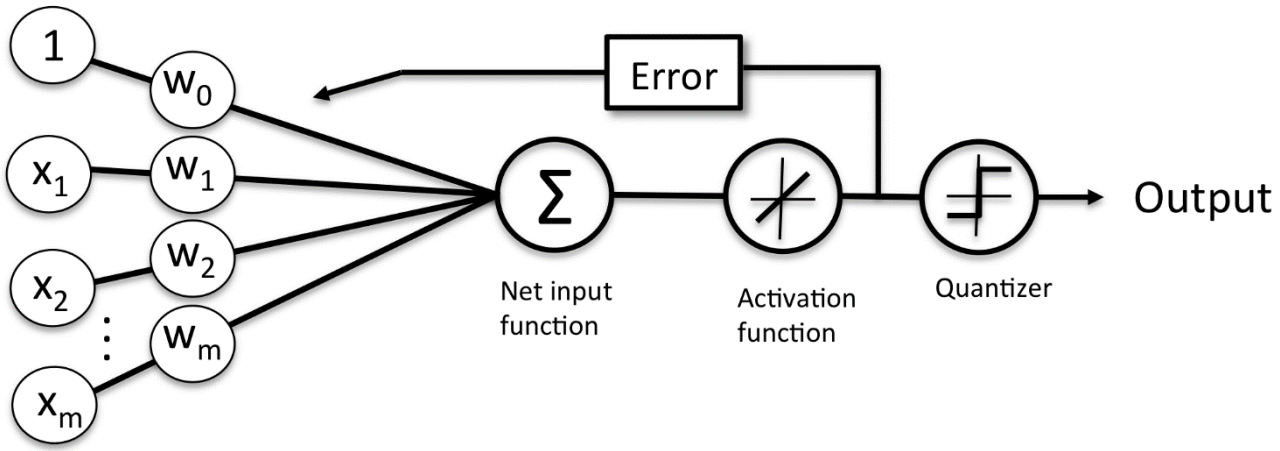
其中η是学习率，在0到1之间取值。

自适应线性神经网络Adaptive linear neurons(Adaline)

ADAptive LInear NEuron classifier 也是一个单层神经网络. 它的重点是定义及最优化 损失函数, 对于理解更高层次更难的机器学习分类模型是非常好的入门.

它与感知器不同的地方在于**更新 weights** 时是用的 linear activation function, 而不是unit step function.

Adaline 中这个 linear activation function 输出等于输入$\phi(w^T\ x) = w^T\ x$。



定义损失函数为 SSE: Sum of Squared Errors

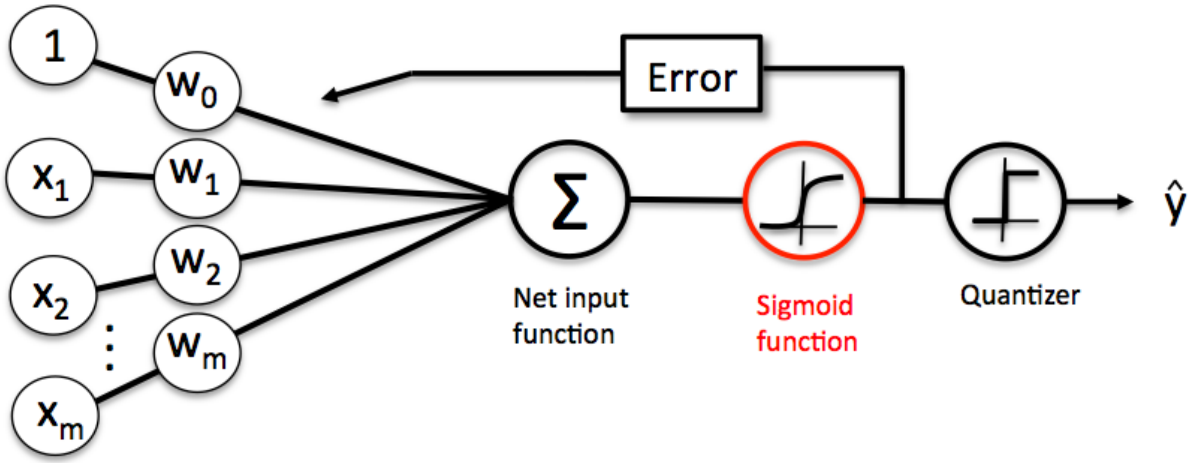
$$J(w) = \frac{1}{2}\sum_i (y^{(i)} - \phi(z^{(i)}))^2$$

这个损失函数是可导的, 并且是凸的, 可以使用梯度下降算法进行最优化.

量化器采用符号函数进行预测

$$f(x) = \begin{cases} +1 & \text{if } w \cdot x + b \geq 0 \\ -1 & \text{else} \end{cases}$$

logistic 回归

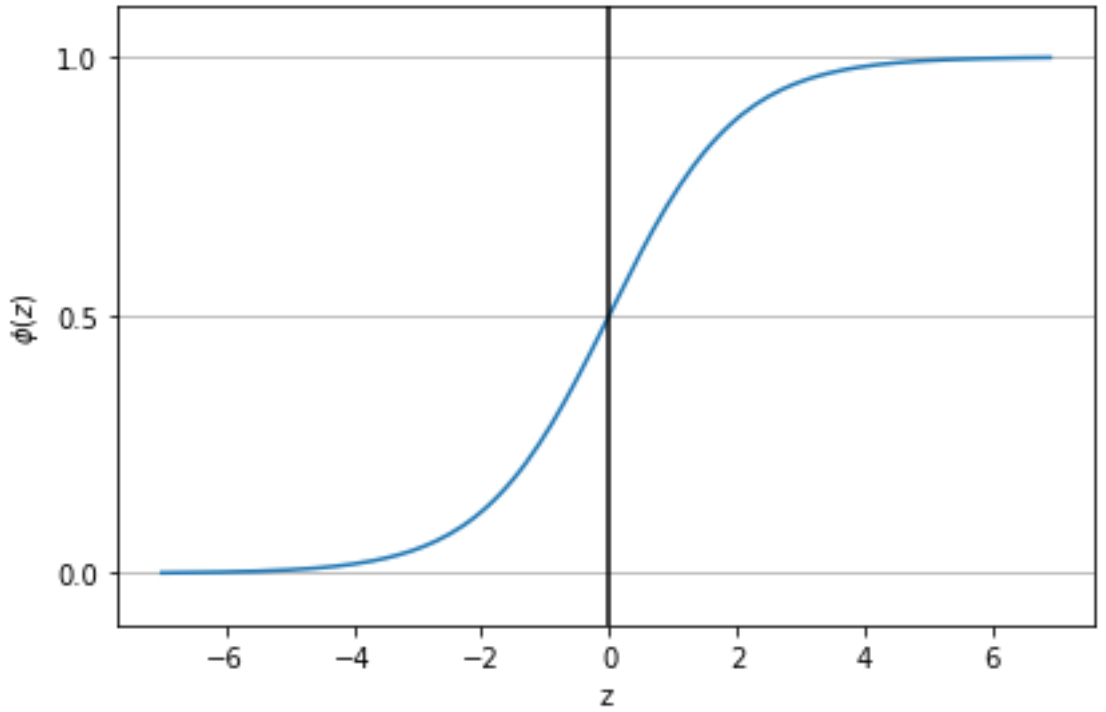


激活函数用**sigmoid function**

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

sigmoid function的输出是样本属于class 1的概率，因此样本属于class 0的概率为$1 - h_\theta(x)$。

sigmoid function



定义损失函数

使用log-likelihood function重新定义损失函数

若想让预测出的结果全部正确的概率最大,根据最大似然估计,就是所有样本预测正确的概率相乘得到的P(总体正确)最大,

$$L(\theta) = \Pi_{i=1}^{m} h_\theta(x)^{y^{(i)}} (1 - h_\theta(x))^{1-y^{(i)}}$$

两边同时取log让其变成连加

$$l(\theta) = \sum_{i=1}^{m} y^{(i)} log\left(h_\theta(x^{(i)}) \right) + \left(1 - y^{(i)}\right) log\left(1 - h_\theta(x^{(i)}) \right).$$

因为在函数最优化的时候习惯让一个函数越小越好,所以我们在前边加一个负号，然后取平均得到损失函数

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} -y^{(i)} log\left(h_\theta(x^{(i)}) \right) - \left(1 - y^{(i)}\right) log\left(1 - h_\theta(x^{(i)}) \right).$$

更新参数

使用梯度下降法更新参数

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)\right)$$

$$\frac{\partial J(\theta)}{\partial \theta_j} = -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\cdot\frac{1}{h_\theta(x^{(i)})} - (1-y^{(i)})\cdot\frac{1}{1-h_\theta(x^{(i)})}\right)\cdot\frac{\partial h_\theta\left(x^{(i)}\right)}{\partial \theta_j}$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(\frac{y^{(i)}}{h_\theta(x^{(i)})} - \frac{1-y^{(i)}}{1-h_\theta(x^{(i)})}\right)\cdot h_\theta\left(x^{(i)}\right)\left(1-h_\theta\left(x^{(i)}\right)\right)\frac{\partial \theta^T x}{\partial \theta_j}$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)}\cdot\left(1-h_\theta\left(x^{(i)}\right)\right) - (1-y^{(i)})\cdot h_\theta\left(x^{(i)}\right)\right)\cdot x_j^{(i)}$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - y^{(i)}\cdot h_\theta\left(x^{(i)}\right) - h_\theta\left(x^{(i)}\right) + y^{(i)}\cdot h_\theta\left(x^{(i)}\right)\right)\cdot x_j^{(i)}$$

$$= -\frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - h_\theta\left(x^{(i)}\right)\right)\cdot x_j^{(i)}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)\cdot x_j^{(i)}$$

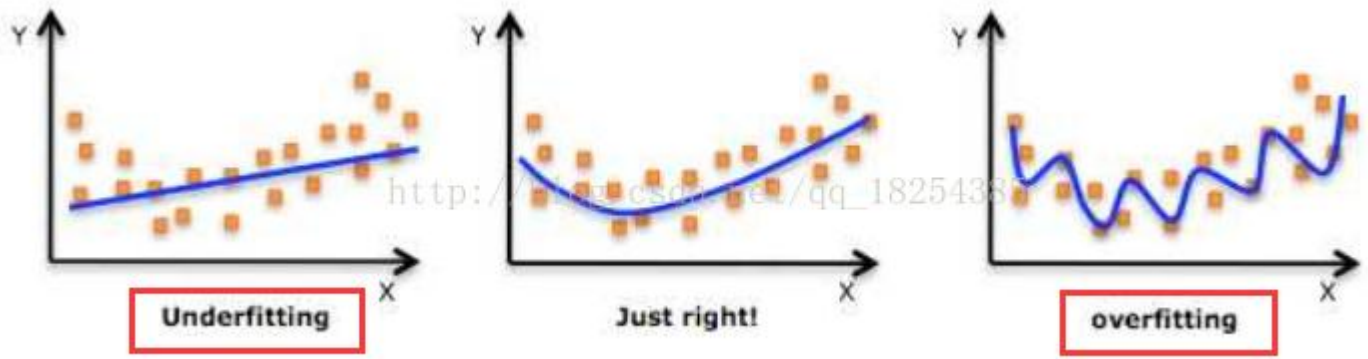
故参数更新公式为

$$\theta_j = \theta_j + \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - h_\theta(x^{(i)})\right)\cdot x_j^{(i)}$$

过拟合和欠拟合

过拟合（over-fitting）就是所建的机器学习模型在训练样本中表现得过于优秀，导致在验证数据集以及测试数据集中表现不佳。

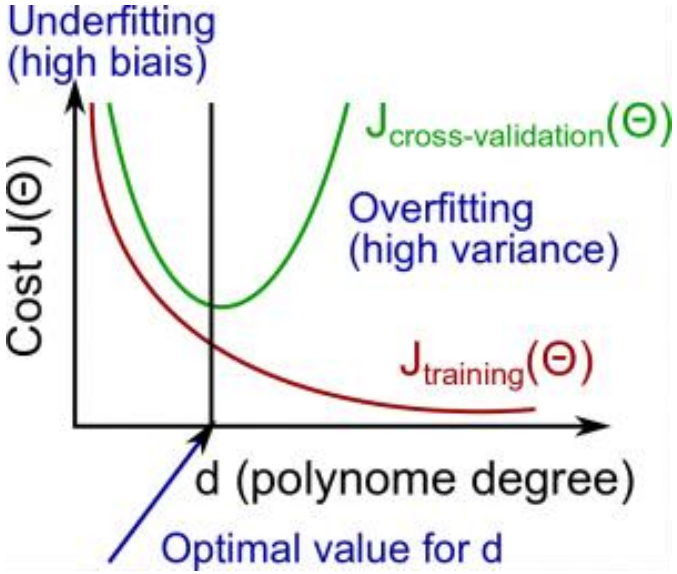
欠拟合就是模型没有很好地捕捉到数据特征，不能够很好地拟合数据。



偏差(bias)和方差(variance)

1、高偏差(high bias)对应着欠拟合，此时训练误差也较大，可以理解为对任何新数据（不论其是否属于训练集），都有着较大的测试误差，偏离真实预测较大。

2、高方差(high variance)对应着过拟合，此时训练误差很小，对于新数据来说，如果其属性与训练集类似，它的测试误差就会小些，如果属性与训练集不同，测试误差就会很大，因此有一个比较大的波动，因此说是高方差。



k折交叉验证

Regularization（正则化）

机器学习中经常可以看到损失函数后面会添加一个额外项，常用的额外项一般有两种，一般英文称作$L1-norm$和$L2-norm$，中文称作L1正则化和L2正则化，或者L1范数和L2范数。

L1正则化和L2正则化可以看做是损失函数的惩罚项。所谓『惩罚』是指对损失函数中的某些参数做一些限制。对于线性回归模型，使用L1正则化的模型建叫做Lasso回归，使用L2正则化的模型叫做Ridge回归（岭回归）。

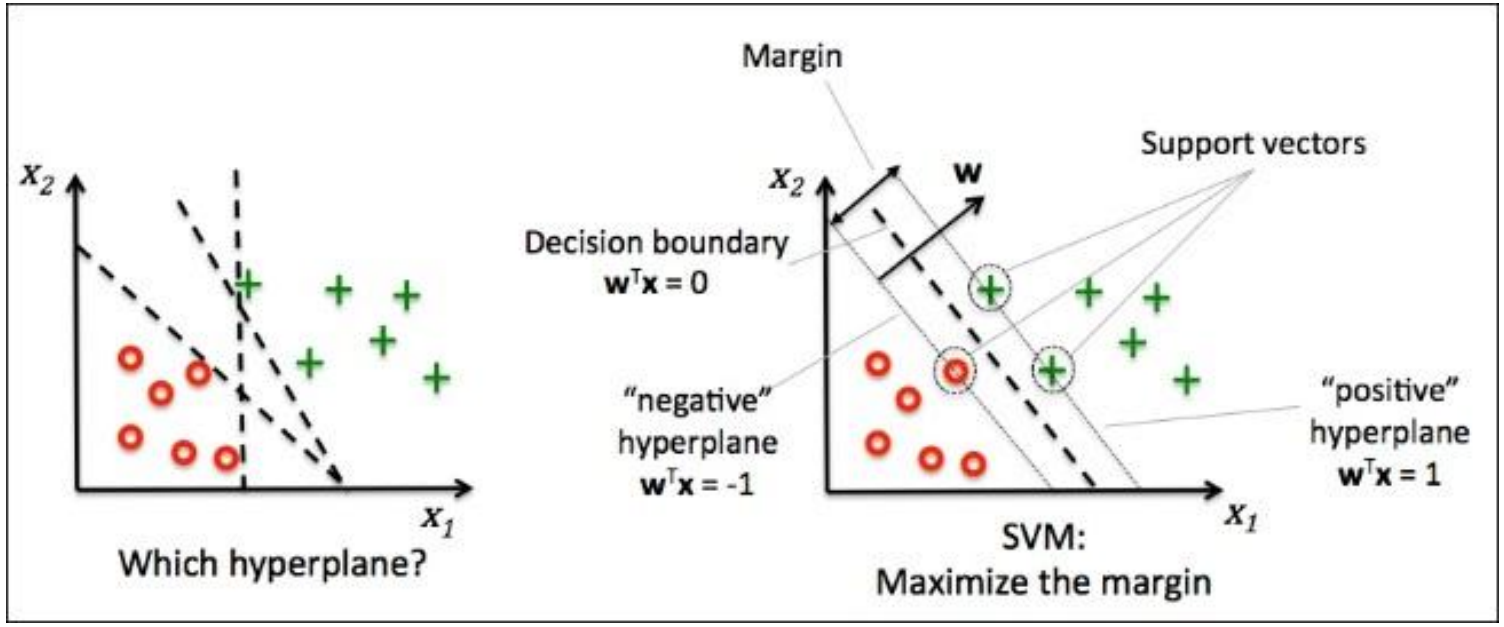
解决 overfitting最常用的解决方法叫做 L2 regulatization，在损失函数上加上下面的项

$$\frac{\lambda}{2}\|w\|^2 = \frac{\lambda}{2}\sum_{j=1}^{m} w_j^2$$

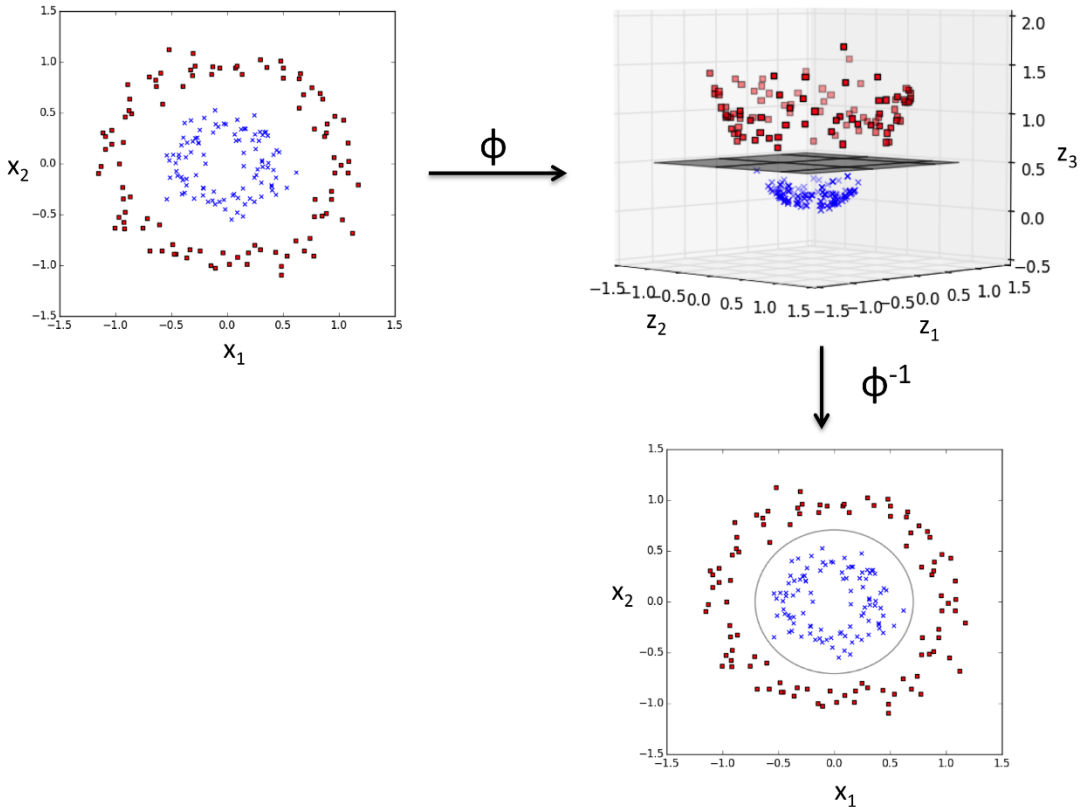
λ就是 regularization parameter, 可以用来控制拟合训练数据的好坏.

支持向量机

支持向量机，英文为Support Vector Machine，简称SVM。它是一种监督学习方法，广泛的应用于分类问题中。SVM的目标是寻找到一个超平面使样本分成两类，并且间隔最大。对于线性不可分的情况，通过使用核函数将低维输入空间线性不可分的样本转化到高维特征空间使其线性可分，从而可以在高维特征空间构造线性分离超平面。



对于非线性的情况，SVM 的处理方法是选择一个核函数 $\kappa(\cdot,\cdot)$ ，通过将数据映射到高维空间，来解决在原始空间中线性不可分的问题。支持向量机首先在低维空间中完成计算，然后通过核函数将输入空间映射到高维特征空间，最终在高维特征空间中构造出最优分离超平面，从而把平面上本身不好分的非线性数据分开。



K近邻算法

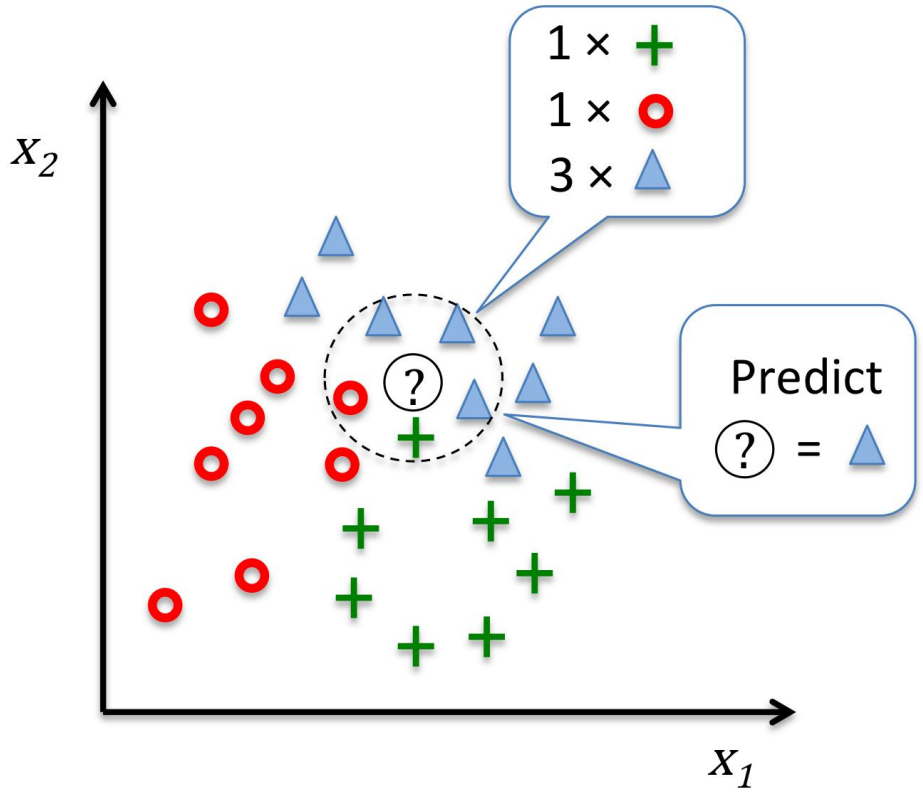
一种基于记忆的方法，不是直接学到一个模型，而是通过记住训练数据，对新的数据给出预测

1.选定k和一个距离度量

2.找到k个离待分类样本最近的邻居

3.通过投票，给出该样本的标签预测

这种方法好处在于新数据进来, 分类器可以马上学习并适应, 但是计算成本也是线性增长, 存储也是问题.



混淆矩阵（Confusion Matrix）

混淆矩阵是一个判断分类好坏程度的方法。

以下有几个相关概念：

TP(True Positive): 真实为0，预测也为0

FN(False Negative): 真实为0，预测为1

FP(False Positive): 真实为1，预测为0

TN(True Negative): 真实为1，预测也为1

二分类问题

Confusion Matrix		Predict	
		0	1
Real	0	a	b
	1	c	d

TP	FN
FP	TN

模型评价指标

分类模型总体判断的准确率

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

预测为0的准确率

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

预测为1的准确率

$$Negative\ Prediction = \frac{TN}{TN + FN}$$

真实为0的准确率

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

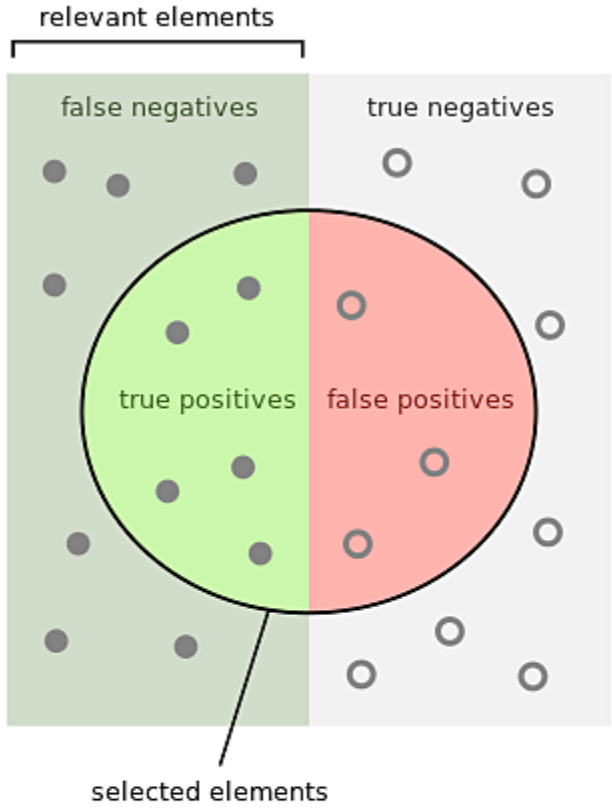
真实为1的准确率

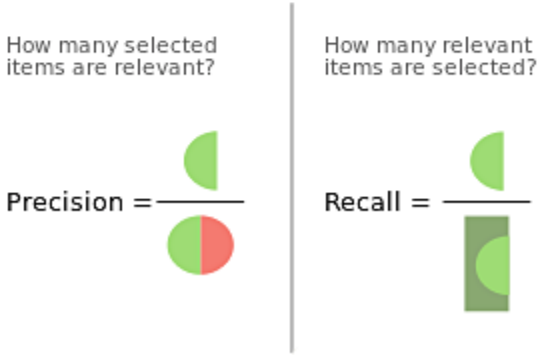
$$Specificity = \frac{TN}{TN + FP}$$

$F_1 - Score$

$$F_1\ Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

TP	FN
FP	TN





ROC曲线（receiver operating characteristic curve）

ROC曲线是另一个判断分类好坏程度的方法。

TP	FN
FP	TN

真正类率（True positive rate）：$TPR = \frac{TP}{TP+FN}$

分类器识别出的正例占所有正例的比例。

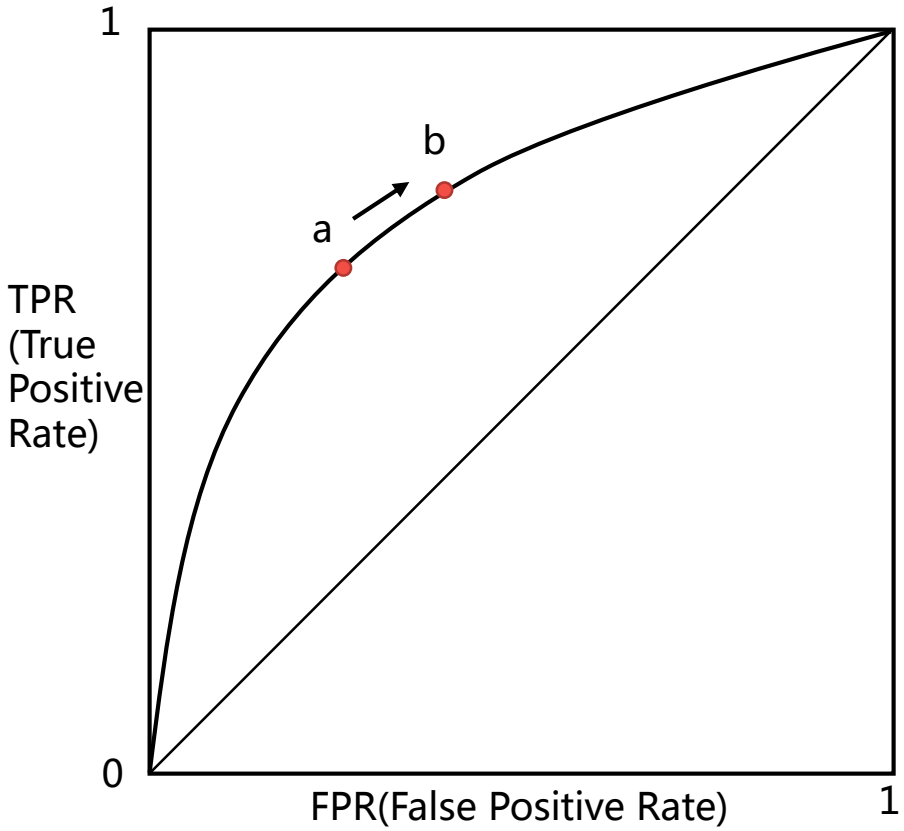
假正类率（False positive rate）：$FPR = \frac{FP}{FP+TN}$

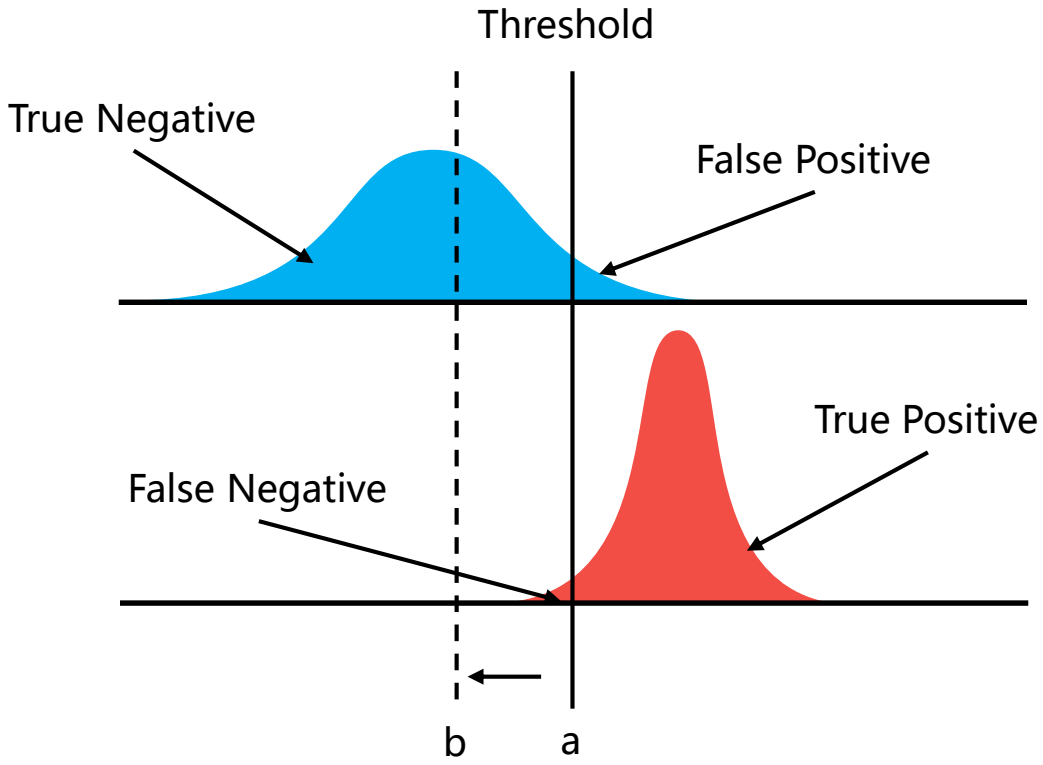
分类器错认为正类的负例占所有负例的比例。

ROC曲线：*FPR*为横坐标，*TPR*为纵坐标.

优点：当测试集中的正负样本的分布变换的时候，ROC曲线能够保持不变。

ROC 上的点如何移动





理想目标：TPR=1，FPR=0，即图中(0,1)点，故 ROC 曲线越靠近(0,1)点，越偏离 45 度对角线越好.

理解ROC曲线

$$FPR = \frac{FP}{FP + TN} \qquad TPR = \frac{TP}{TP + FN}$$

考虑ROC曲线图中的四个点和一条线:

•第一个点，(0,1)，即FPR=0, TPR=1，这意味着FN（false negative）=0，并且FP（false positive）=0。这是一个完美的分类器，它将所有的样本都正确分类。

•第二个点，(1,0)，即FPR=1，TPR=0，类似地分析可以发现这是一个最糟糕的分类器，因为它成功避开了所有的正确答案。

•第三个点，(0,0)，即FPR=TPR=0，即FP（false positive）=TP（true positive）=0，可以发现该分类器预测所有的样本都为负样本（negative）。

•第四个点（1,1），分类器实际上预测所有的样本都为正样本。

•考虑ROC曲线图中的虚线y=x上的点。这条对角线上的点表示的是一个采用随机猜测策略的分类器的结果，例如(0.5,0.5)，表示该分类器随机对于一半的样本猜测其为正样本，另外一半的样本为负样本。

因此：如果分类器效果很好, 那么图应该会在左上角. ROC曲线越接近左上角，该分类器的性能越好。

如何画ROC

Inst#	Class	Score	Inst#	Class	Score
1	**p**	.9	11	**p**	.4
2	**p**	.8	12	**n**	.39
3	**n**	.7	13	**p**	.38
4	**p**	.6	14	**n**	.37
5	**p**	.55	15	**n**	.36
6	**p**	.54	16	**n**	.35
7	**n**	.53	17	**p**	.34
8	**n**	.52	18	**n**	.33
9	**p**	.51	19	**p**	.30
10	**n**	.505	20	**n**	.1

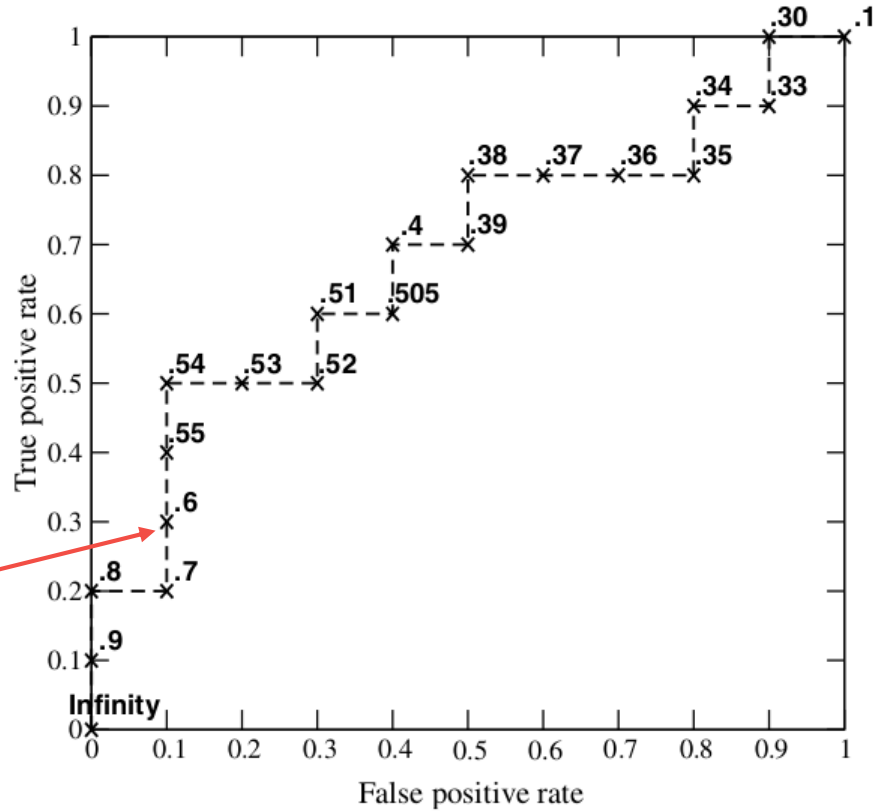
Class一栏表示每个测试样本真正的标签（p 表示正样本，n 表示负样本），Score表示每个测试样本属于正样本的概率。

从高到低，依次将Score值作为阈值 threshold，当测试样本属于正样本的概率大于或等于这个 threshold 时，被判断为正样本，否则判定为负样本。举例来说，对于图中的第 4 个样本，其Score值为 0.6，那么样本 1，2，3，4 都被认为是正样本，因为它们的Score值都大于等于 0.6，而其他样本则都认为是负样本。

$$TPR = \frac{TP}{TP + FN} = \frac{3}{10} = 0.3$$

$$FPR = \frac{FP}{FP + TN} = \frac{1}{10} = 0.1$$

所以threshold为0.6对应的点为(0.1,0.3).



每次选取一个不同的 threshold，我们就可以得到一组 FPR 和 TPR，即 ROC 曲线上的一点。

当 threshold 为 1 和 0 时，分别可以得到 ROC 曲线上的(0,0)和(1,1)两个点。将这些(FPR,TPR)对连接起来，就得到了 ROC 曲线。当 threshold 取值越多，ROC 曲线越平滑。

AUC

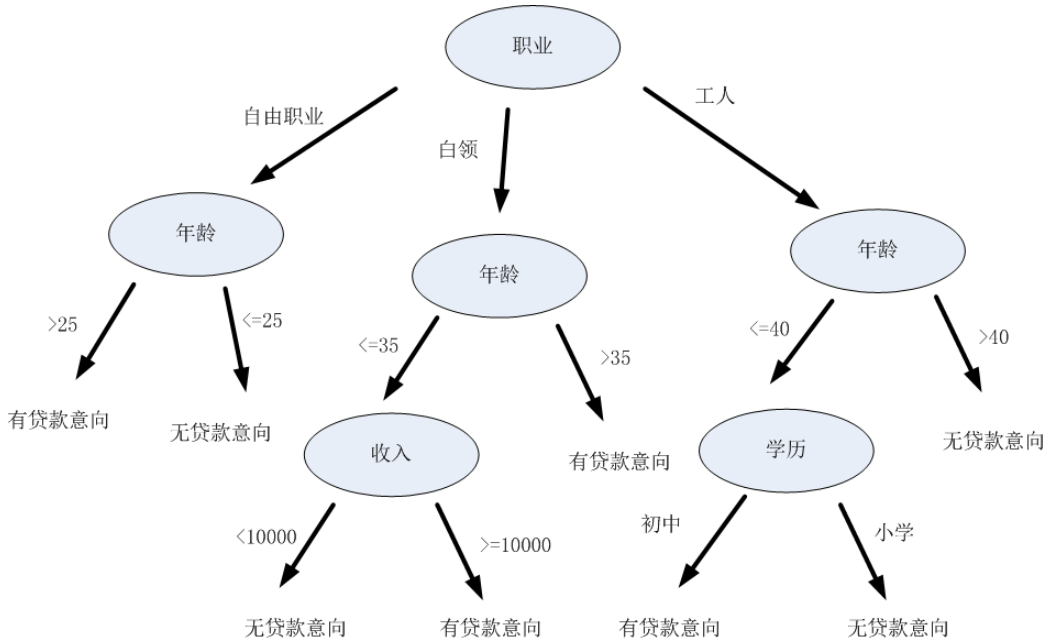
AUC (Area Under Curve) 被定义为 ROC 曲线下的面积，显然这个面积的数值不会大于 1。又由于 ROC 曲线一般都处于 $y=x$ 这条直线的上方，所以 AUC 的取值范围一般在 0.5 和 1 之间。使用 AUC 值作为评价标准是因为很多时候 ROC 曲线并不能清晰的说明哪个分类器的效果更好，而作为一个数值，对应 AUC 更大的分类器效果更好。

AUC 意味着什么

首先 AUC 值是一个概率值，当你随机挑选一个正样本以及一个负样本，当前的分类算法根据计算得到的 Score 值将这个正样本排在负样本前面的概率就是 AUC 值。当然，AUC 值越大，当前的分类算法越有可能将正样本排在负样本前面，即能够更好的分类。

决策树

决策树算法是一种监督学习方法，它基于特征属性进行分类。树中每个节点表示某个特征，而每个分叉路径则代表某个可能的属性值，而每个叶结点则对应从根节点到该叶节点所经历的路径所表示的对象的类别。其主要的优点是模型具有可读性，计算量小，分类速度快。决策树算法包括ID3，C4.5和CART。三个算法的主要区别是特征选择的准则不一样。ID3计算的是信息增益；C4.5比较的是信息增益比，而CART则依赖基尼指数。



决策树的构建

决策树的构建是数据逐步分裂的过程，构建的步骤如下:

步骤1：将所有的数据看成是一个节点，进入步骤2；

步骤2：从所有的数据特征中挑选一个数据特征对节点进行分割，进入步骤3；

步骤3：生成若干孩子节点，对每一个孩子节点进行判断，如果满足停止分裂的条件，进入步骤4；否则，进入步骤2；

步骤4：设置该节点是子节点，其输出的结果为该节点数量占比最大的类别。

从上述步骤可以看出，决策生成过程中有几个重要的问题：

（1）数据如何分割

（2）如何选择分裂的属性

（3）什么时候停止分裂

不同的纯度标准--分裂属性的选择

- 信息增益（ID3算法）
- 信息增益比（C4.5算法）
- 基尼指数（CART算法）

熵与基尼指数

设X是一个取有限个值的离散随机变量，其概率分布为：

$$P(X = x_i) = p_i$$

在信息论和概率统计中，熵(entropy)是表示随机变量不确定性的度量，定义为

$$entropy = -\sum(p_i log(p_i))$$

熵描述了数据的混乱程度，熵越大，混乱程度越高，也就是纯度越低；反之，熵越小，混乱程度越低，纯度越高。
基尼指数定义为

$$Gini(p) = \sum_{i=1}^{K} p_i(1 - p_i) = 1 - \sum_{i=1}^{K} p_i^2$$

基尼指数性质是：
1.类别个数越少，基尼系数越低，
2.类别个数相同时，类别集中度越高，基尼系数越低。
当类别越少，类别集中度越高的时候，基尼系数越低；当类别越多，类别集中度越低的时候，基尼系数越高。

条件熵与信息增益

条件熵

设有随机变量(X, Y)。条件熵$H(Y|X)$表示在已知随机变量X的条件下随机变量Y的不确定性。随机变量X给定的条件下随机变量Y的条件熵$H(Y|X)$定义为X给定条件下Y的条件概率分布的熵对X的数学期望

$$H(Y|X) = \sum_{i=1}^{n} p_i H(Y|X = x_i)$$

这里 $p_i = P(X = x_i), \quad i = 1, 2, \ldots, n.$

信息增益

信息增益表示得知特征X的信息而使得类的信息的不确定性减少程度。特征A对训练数据集D的信息增益$g(D, A)$,为集合D的熵$H(D)$与特征A给定条件下D的条件熵$H(D|A)$之差，即

$$g(D, A) = H(D) - H(D|A).$$

信息增益的算法

输入：训练数据集D，特征A

输出：特征A对训练数据集D的信息增益$g(D, A)$

(1):计算数据集D的熵$H(D)$

$$H(D) = -\sum_{k=1}^{K} \frac{|C_k|}{|D|} log_2 \frac{|C_k|}{|D|}$$

(2):计算特征A对训练数据集D的条件熵$H(D|A)$

$$H(D|A) = \sum_{i=1}^{n} \frac{|D_i|}{|D|} H(D_i) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} \sum_{k=1}^{K} \frac{|D_{ik}|}{|D_i|} log_2 \frac{|D_{ik}|}{|D_i|}.$$

(3):计算信息增益

$$g(D, A) = H(D) - H(D|A)$$

$|D|$为样本容量，设有K个类$Ck, k = 1, 2, \ldots, K$,$|C_k|$为属于类C_k的样本个数。设特征A有n个不同取值，根据特征A的取值将D划分为n个子集$D_1, D_2, \ldots, D_n$, D_{ik}为子集D_i中属于类C_k的集合。

例子

Day	Temperatrue	Outlook	Humidity	Windy	PlayGolf?
07-05	hot	rainy	high	false	no
07-06	hot	sunny	high	true	no
07-07	hot	overcast	high	false	yes
07-09	cool	rainy	normal	false	yes
07-10	cool	overcast	normal	true	yes
07-12	mild	sunny	high	false	no
07-14	cool	sunny	normal	false	yes
07-15	mild	rainy	normal	false	yes
07-20	mild	sunny	normal	true	yes
07-21	mild	overcast	high	true	yes
07-22	hot	overcast	normal	false	yes
07-23	mild	rainy	high	true	no
07-26	cool	sunny	normal	true	no
07-30	mild	rainy	high	false	yes

计算信息增益

计算整个数据集的熵

$$p_1 = Num(no)/(Num(no) + Num(yes))$$

$$p_2 = Num(yes)/(Num(no) + Num(yes))$$

$$H(D) = -\frac{5}{14}log\frac{5}{14} - \frac{9}{14}log\frac{9}{14} = 0.9403$$

假如把outlook作为分裂特征

$$E(Outlook = sunny) = -\frac{2}{5}log\frac{2}{5} - \frac{3}{5}log\frac{3}{5} = 0.971$$

$$E(Outlook = overcast) = -1log1 - 0log0 = 0$$

$$E(Outlook = rainy) = -\frac{3}{5}log\frac{3}{5} - \frac{2}{5}log\frac{2}{5} = 0.971$$

$$H(D|A) = \frac{5}{14} \cdot 0.971 + \frac{4}{14} \cdot 0 + \frac{5}{14} \cdot 0.971 = 0.693$$

$$g(D, Outlook) = 0.9403 - 0.693,$$

信息增益比

假如用Day来做为特征， H(D|Day)=0，那么g(D,Day)=0.9403，这特征可真是够“好”的！显然，每一天都可以将样本分开，这种样本分隔的结果就是形成了一棵叶子数量为14，深度只有两层的树。显然这种特征对于样本的分隔没有任何意义。解决办法自然就想到了如何能够对树分支过多的情况进行惩罚，这样就引入了信息增益比

特征A对训练数据集D的信息增益比$g_R(D, A)$定义为其信息增益$g(D, A)$与训练数据集D关于特征A的值的熵$H_A(D)$之比，即

$$g_R(D, A) = \frac{g(D, A)}{H_A(D)}$$

其中，$H_A(D) = -\sum_{i=1}^{n} \frac{|D_i|}{|D|} log_2 \frac{|D_i|}{|D|}$，$n$是特征$A$取值的个数。

停止分裂的条件

（1）最小节点数

当节点的数据量小于一个指定的数量时，不继续分裂。两个原因：一是数据量较少时，再做分裂容易强化噪声数据的作用；二是降低树生长的复杂性。提前结束分裂一定程度上有利于降低过拟合的影响。

（2）熵或者基尼值小于阈值。

由上述可知，熵和基尼值的大小表示数据的复杂程度，当熵或者基尼值过小时，表示数据的纯度比较大，如果熵或者基尼值小于一定程度数，节点停止分裂。

（3）决策树的深度达到指定的条件

节点的深度可以理解为节点与决策树跟节点的距离，如根节点的子节点的深度为1，因为这些节点与跟节点的距离为1，子节点的深度要比父节点的深度大1。决策树的深度是所有叶子节点的最大深度，当深度到达指定的上限大小时，停止分裂。

（4）所有特征已经使用完毕，不能继续进行分裂。

被动式停止分裂的条件，当已经没有可分的属性时，直接将当前节点设置为叶子节点。

集成学习--随机森林

随机森林就是通过集成学习的思想将多棵树集成的一种算法，它的基本单元是决策树，而它的本质属于机器学习的一大分支——集成学习（Ensemble Learning）方法。

随机森林生成过程：

1.随机且有放回地从训练集中抽取N个训练样本（这种采样方式称为自助采样法（bootstrap sample）方法）

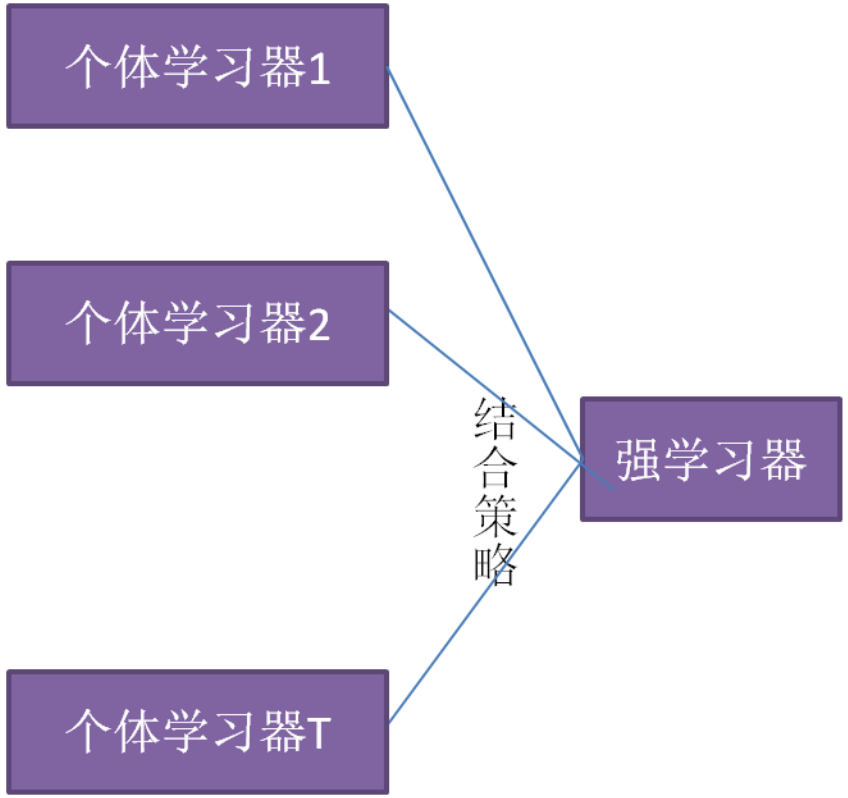
2.如果每个样本的特征维度为M，指定一个常数m<<M，随机地从M个特征中选取m个特征子集，每次树进行分裂时，从这m个特征中选择最优的

3.重复步骤1和步骤2，k次

4.最后的预测是每棵树的判断做投票

集成学习

对于训练集数据，我们通过训练若干个个体学习器，通过一定的结合策略，就可以最终形成一个强学习器，以达到博采众长的目的。



集成学习分类

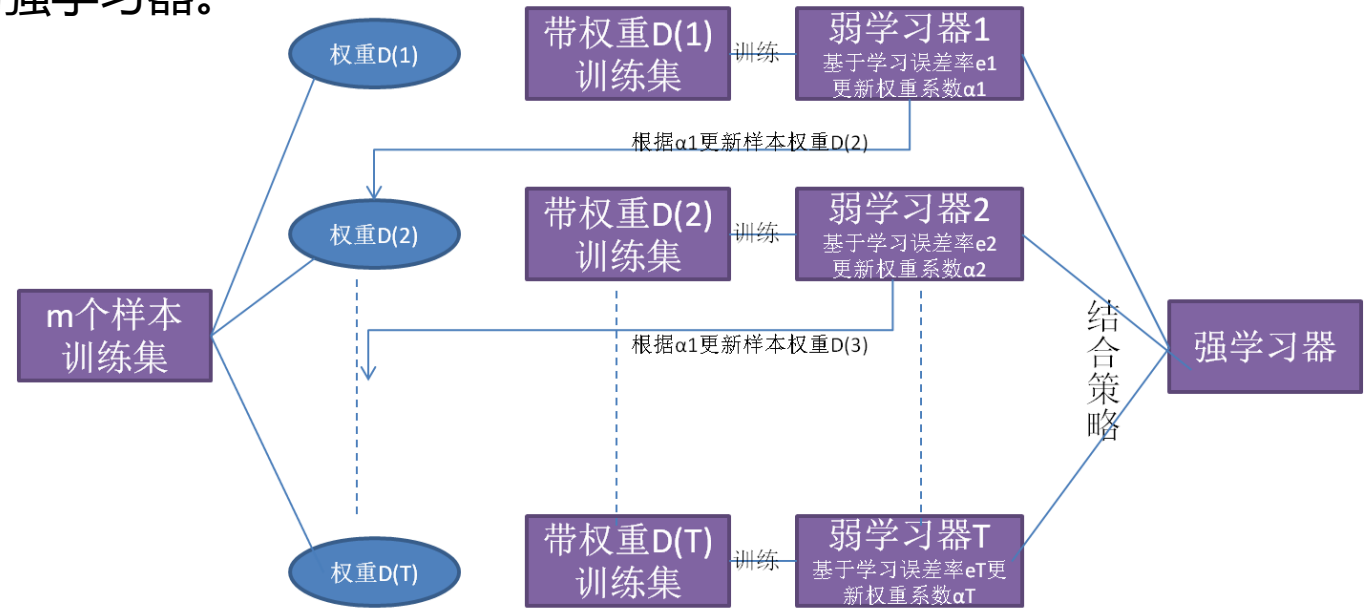
集成学习有两个主要的问题需要解决，第一是如何得到若干个个体学习器，第二是如何选择一种结合策略，将这些个体学习器集合成一个强学习器。

根据个体学习器是否存在依赖关系分为两类：

- 个体学习器之间存在强依赖关系，一系列个体学习器基本都需要串行生成，代表算法是boosting系列算法，
- 个体学习器之间不存在强依赖关系，一系列个体学习器可以并行生成，代表算法是bagging和随机森林系列算法。

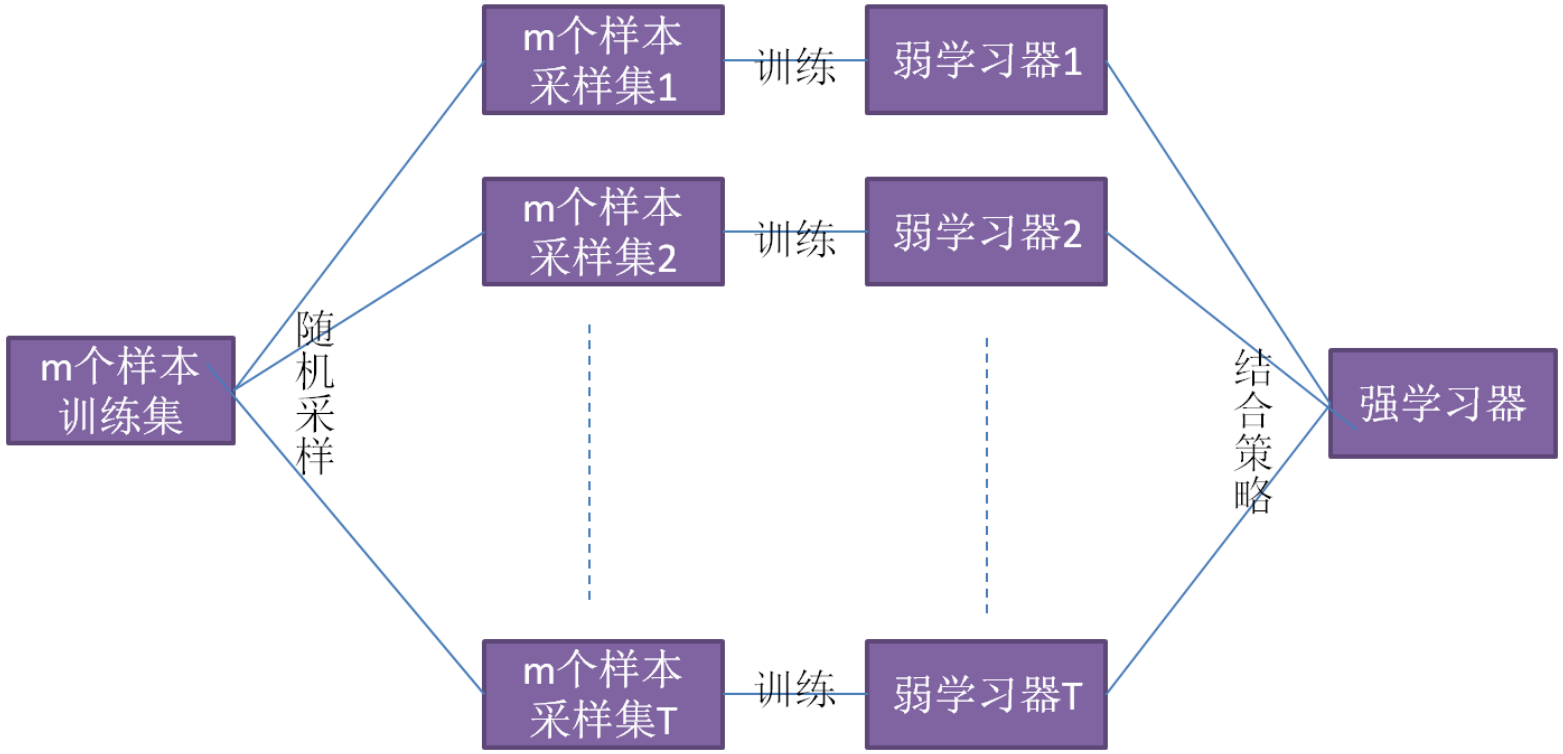
集成学习之boosting

Boosting算法的工作机制是首先从训练集用初始权重训练出一个弱学习器1，根据弱学习的学习误差率表现来更新训练样本的权重，使得之前弱学习器1学习误差率高的训练样本点的权重变高，使得这些误差率高的点在后面的弱学习器2中得到更多的重视。然后基于调整权重后的训练集来训练弱学习器2.，如此重复进行，直到弱学习器数达到事先指定的数目T，最终将这T个弱学习器通过集合策略进行整合，得到最终的强学习器。



集成学习之bagging

bagging的个体弱学习器的训练集是通过随机采样得到的。通过T次的随机采样，我们就可以得到T个采样集，对于这T个采样集，我们可以分别独立的训练出T个弱学习器，再对这T个弱学习器通过集合策略来得到最终的强学习器。



集成学习之结合策略

投票分类

多个弱学习器的对样本x的预测结果中，数量最多的类别c_i为最终的分类类别。如果不止一个类别获得最高票，则随机选择一个做最终类别。

若每个分类器对应不同的权重w_j则每个弱学习器的分类票数要乘以一个权重，最终将各个类别的加权票数求和，最大的值对应的类别为最终类别。

平均法

对于数值类的回归预测问题，通常使用的结合策略是平均法，也就是说，对于若干个弱学习器的输出进行平均得到最终的预测输出。

学习法

学习法的代表方法是stacking，当使用stacking的结合策略时， 我们不是对弱学习器的结果做简单的逻辑处理，而是再加上一层学习器，也就是说，我们将训练集弱学习器的学习结果作为输入，将训练集的输出作为输出，重新训练一个学习器来得到最终结果。

回归树

回归树是可以用于回归的决策树模型，一个回归树对应着输入空间（即特征空间）的一个划分以及在划分单元上的输出值.与分类树不同的是，回归树对输入空间的划分采用一种启发式的方法，会遍历所有输入变量，找到最优的切分变量j和最优的切分点s，即选择第j个特征x_j和它的取值s将输入空间划分为两部分，然后重复这个操作。

而如何找到最优的j和s是通过比较不同的划分的误差来得到的。一个输入空间的划分的误差是用真实值和划分区域的预测值的平方和来衡量的，即

$$\sum_{x_i \in R_m} (y_i - f(x_i))^2$$

其中，$f(x_i)$是每个划分单元的预测值，这个预测值是该单元内每个样本点的值的均值，即

$$f(x_i) = c_m = ave(y_i | x_i \in R_m)$$

其中，$R_1(j,s) = \{x | x^j \leq s\}$ 和$R_2(j,s) = \{x | x^j > s\}$是被划分后的两个区域。

例子

为了便于理解，下面举一个简单实例。训练数据见下表，目标是得到一棵最小二乘回归树。

x	1	2	3	4	5	6	7	8	9	10
y	5.56	5.7	5.91	6.4	6.8	7.05	8.9	8.7	9	9.05

选择最优切分变量j与最优切分点s

在本数据集中，只有一个变量，因此最优切分变量自然是x。

接下来我们考虑9个切分点[1.5,2.5,3.5,4.5,5.5,6.5,7.5,8.5,9.5].

例如，取$s = 1.5$。此时 $R_1 = 1, R_2 = 2, 3, 4, 5, 6, 7, 8, 9, 10$，这两个区域的输出值分别为：
$c_1 = 5.56, c2 = 1/9(5.7 + 5.91 + 6.4 + 6.8 + 7.05 + 8.9 + 8.7 + 9 + 9.05) = 7.50$。得到下表：

s	1.5	2.5	3.5	4.5	5.5	6.5	7.5	8.5	9.5
c1	5.56	5.63	5.72	5.89	6.07	6.24	6.62	6.88	7.11
c2	7.5	7.73	7.99	8.25	8.54	8.91	8.92	9.03	9.05

把c_1, c_2的值代入误差计算公式，如：$m(1.5) = 0 + 15.72 = 15.72$。同理，可获得下表：

s	1.5	2.5	3.5	4.5	5.5	6.5	7.5	8.5	9.5
m(s)	15.72	12.07	8.36	5.78	3.91	1.93	8.01	11.73	15.74

显然取 $s = 6.5$时，$m(s)$最小。因此，第一个划分变量$j = x, s = 6.5$.

对两个子区域继续调用上述步骤，假设在生成3个区域之后停止划分，那么最终生成的回归树形式如下：

$$T = \begin{cases} 5.72 & x \leq 3.5 \\ 6.75 & 3.5 \leqslant x \leq 6.5 \\ 8.91 & x > 6.5 \end{cases}$$