机器学习模型介绍

讲师：武晟然

主要内容

➢ 监督学习

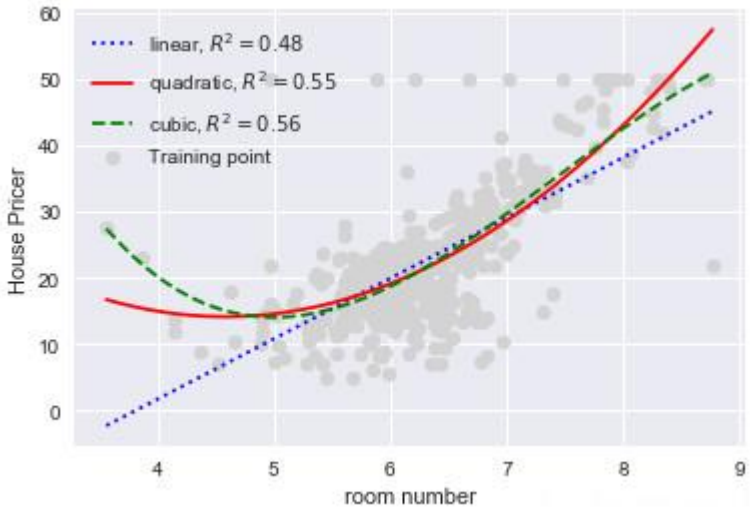
- 回归模型
  - 线性回归
- 分类模型
  - k近邻（kNN）
  - 决策树
  - 逻辑斯谛回归

➢ 无监督学习

- 聚类
  - k均值（k-means）
- 降维

监督学习 —— 回归模型

- 线性回归模型
  - 一元线性回归
  - 多元线性回归
- 非线性回归模型
- 最小二乘法



线性回归模型

- 线性回归（linear regression）是一种线性模型，它假设输入变量 x 和单个输出变量 y 之间存在线性关系
- 具体来说，利用线性回归模型，可以从一组输入变量 x 的线性组合中，计算输出变量 y

$$y=ax+b$$

$$f(\boldsymbol{x}) = w_1 x_1 + w_2 x_2 + \ldots + w_d x_d + b$$

线性方程求解

假设我们有一个如下的二元一次方程：

$$y = ax + b$$

我们已知两组数据：x = 1 时，y = 3，即（1, 3）

x = 2 时，y = 5，即（2, 5）

将数据输入方程中，可得：

$$a + b = 3$$

$$2a + b = 5$$

解得：a = 2，b = 1

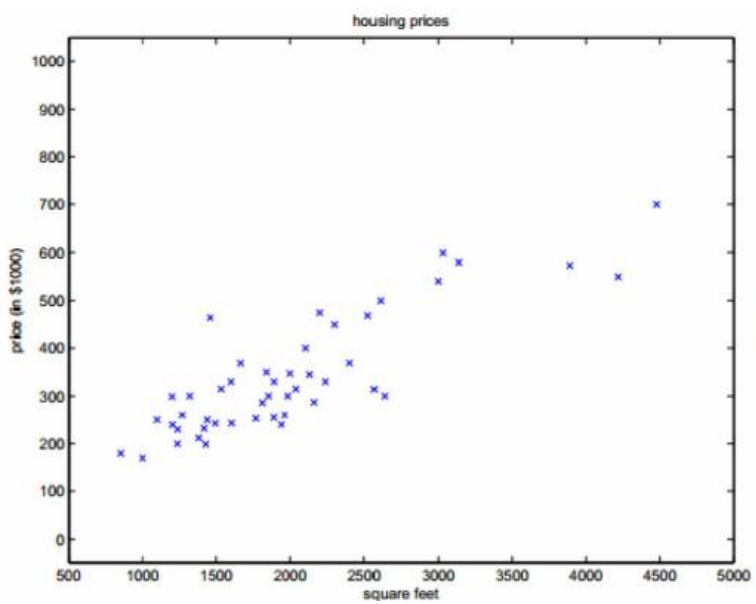
即方程为：2x + 1 = y

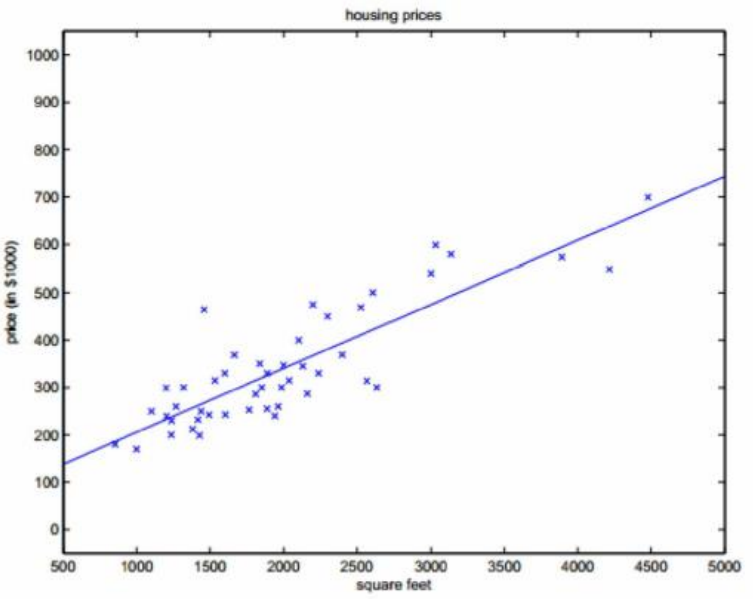
当我们有任意一个 x 时，输入方程，就可以得到对应的 y

例如 x = 5时，y = 11。

线性回归模型

$$y=ax+b$$





线性回归模型

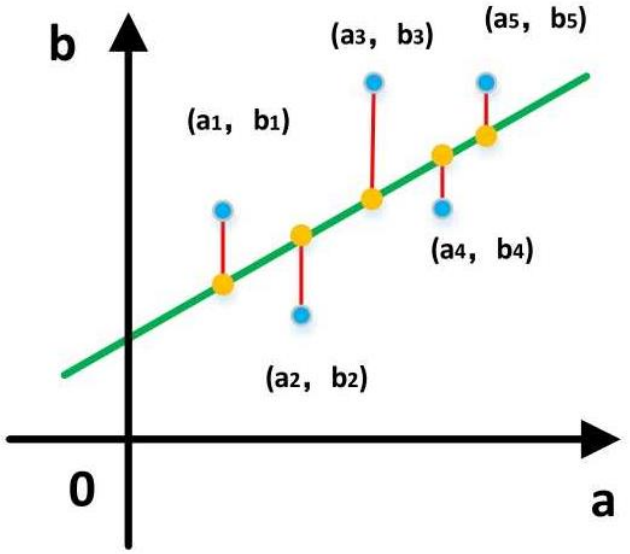
- 给定有d个属性（特征）描述的示例 x =（x1; x2; …; xd），其中xi是x在第i个属性（特征）上的取值，线性模型（linear model）试图学得一个通过属性（特征）的线性组合来进行预测的函数，即：

$$f(\boldsymbol{x}) = w_1 x_1 + w_2 x_2 + \ldots + w_d x_d + b$$

- 一般用向量形式写成：　　$f(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x} + b$
  - 其中　　$\boldsymbol{w} = (w_1; w_2; \ldots; w_d)$
- 假设特征和结果都满足线性，即不大于一次方。
- w和b学得之后，模型就得以确定。
- 许多功能更为强大的非线性模型可在线性模型的基础上通过引入层级结构或高维映射而得。

最小二乘法

- 基于均方误差最小化来进行模型求解的方法称为“最小二乘法”（least square method）
- 它的主要思想就是选择未知参数，使得理论值与观测值之差的平方和达到最小。



最小二乘法

- 我们假设输入属性（特征）的数目只有一个：

$$f(x_i) = wx_i + b, \text{ 使得 } f(x_i) \simeq y_i$$

- 在线性回归中，最小二乘法就是试图找到一条直线，使所有样本到直线上的欧式距离之和最小。

$$\begin{aligned}(w^*, b^*) &= \arg\min_{(w,b)} \sum_{i=1}^{m} (f(x_i) - y_i)^2 \\ &= \arg\min_{(w,b)} \sum_{i=1}^{m} (y_i - wx_i - b)^2\end{aligned}$$

求解线性回归

- 求解w和b，使得 $E_{(w,b)} = \sum_{i=1}^{m}(y_i - wx_i - b)^2$ 最小化的过程，称为线性回归模型的“最小二乘参数估计”
- 将 $E_{(w,b)}$ 分别对w和b求导，可以得到

$$\frac{\partial E_{(w,b)}}{\partial w} = 2\left(w\sum_{i=1}^{m}x_i^2 - \sum_{i=1}^{m}(y_i - b)\,x_i\right)$$

$$\frac{\partial E_{(w,b)}}{\partial b} = 2\left(mb - \sum_{i=1}^{m}(y_i - wx_i)\right)$$

- 令偏导数都为0，可以得到

$$w = \frac{\sum_{i=1}^{m} y_i(x_i - \bar{x})}{\sum_{i=1}^{m} x_i^2 - \frac{1}{m}\left(\sum_{i=1}^{m} x_i\right)^2} \qquad b = \frac{1}{m}\sum_{i=1}^{m}(y_i - wx_i)$$

—— 其中

$$\bar{x} = \frac{1}{m}\sum_{i=1}^{m} x_i$$

多元线性回归

- 如果有两个或两个以上的自变量，这样的线性回归分析就称为多元线性回归
- 实际问题中，一个现象往往是受多个因素影响的，所以多元线性回归比一元线性回归的实际应用更广

$$f_{好瓜}(\boldsymbol{x}) = 0.2 \cdot x_{色泽} + 0.5 \cdot x_{根蒂} + 0.3 \cdot x_{敲声} + 1$$

$$f(\boldsymbol{x}) = w_1 x_1 + w_2 x_2 + \ldots + w_d x_d + b$$

$$h_\theta(x) = X^T\theta = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

梯度下降法求解线性回归

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 \quad h_\theta(x) = X^T\theta = \sum_{k=0}^{n}\theta_k x_k$$

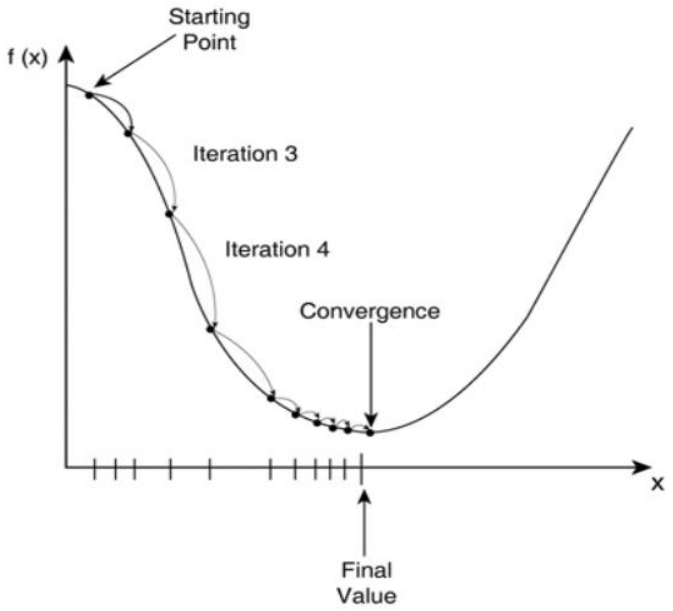
$$\theta := \theta - \alpha \cdot \frac{\partial J(\theta)}{\partial \theta}$$

$$\frac{\partial}{\partial \theta_j}J(\theta) = \frac{\partial}{\partial \theta_j}\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

$$= 2 \cdot \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) \cdot \frac{\partial}{\partial \theta_j}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)$$

$$= \frac{2}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) \cdot \frac{\partial}{\partial \theta_j}\left(\sum_{k=0}^{n}\theta_k x_k^{(i)} - y^{(i)}\right)$$

$$= \frac{2}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) \cdot x_j^{(i)}$$



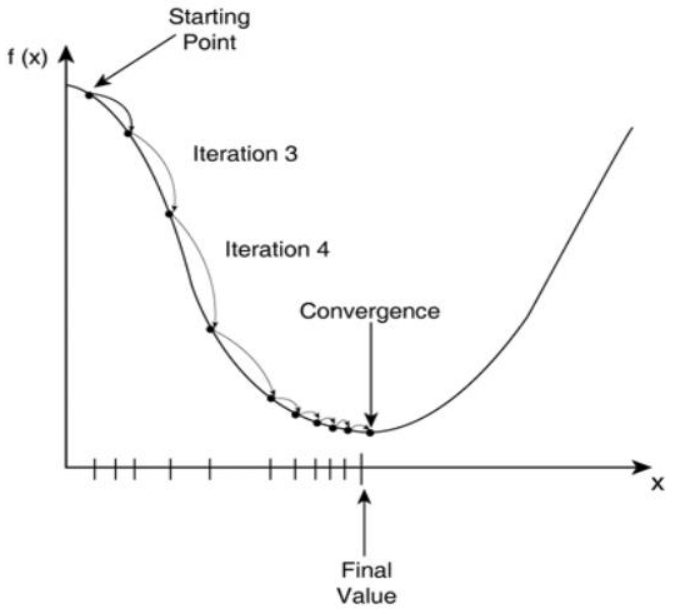
梯度下降法求解线性回归

- 退化到一元线性回归，就有

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

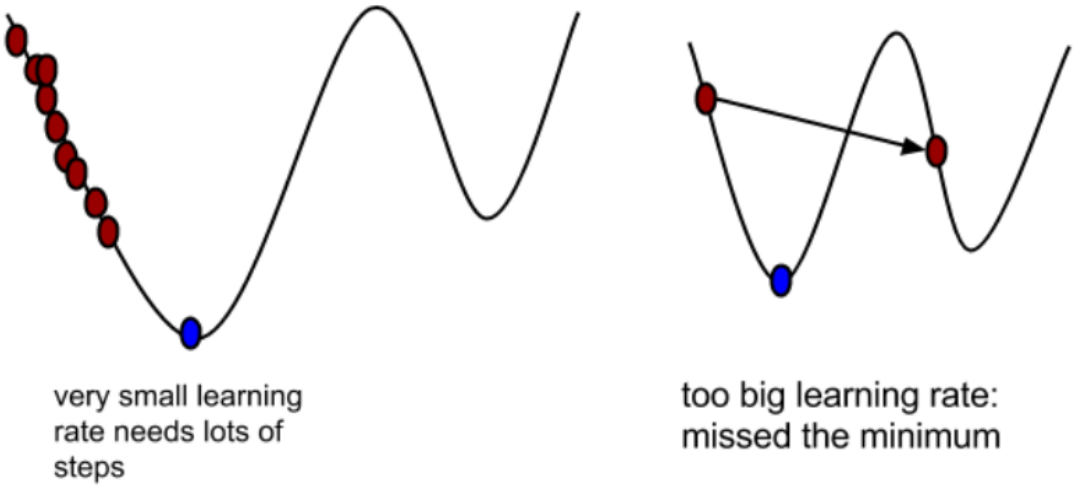
$$\frac{\partial}{\partial \theta_1} J(\theta) = \frac{2}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) \cdot x^{(i)}$$

$$\frac{\partial}{\partial \theta_0} J(\theta) = \frac{2}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right)$$



梯度下降法求解线性回归

- α 在梯度下降算法中被称作为**学习率**或者**步长**
- 这意味着我们可以通过 α 来控制每一步走的距离，以保证不要走太快，错过了最低点；同时也要保证收敛速度不要太慢
- 所以 α 的选择在梯度下降法中往往是很重要的，不能太大也不能太小

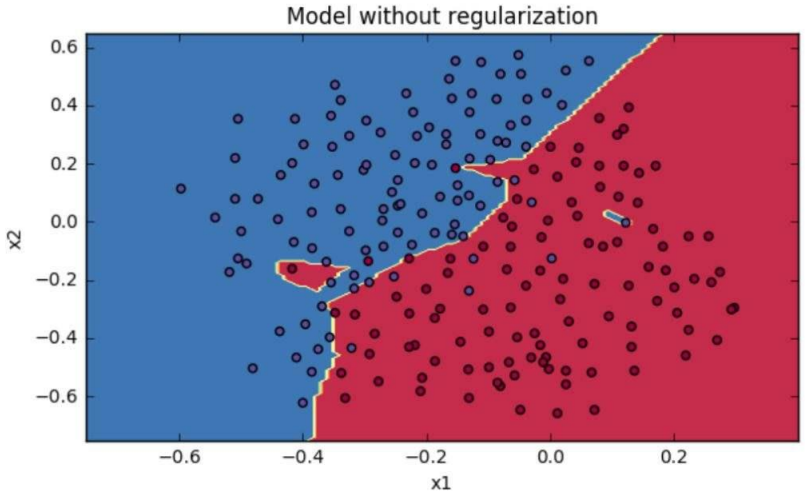


梯度下降法和最小二乘法

- 相同点
  - 本质和目标相同：两种方法都是经典的学习算法，在给定已知数据的前提下利用求导算出一个模型（函数），使得损失函数最小，然后对给定的新数据进行估算预测
- 不同点
  - 损失函数：梯度下降可以选取其它损失函数，而最小二乘一定是平方损失函数
  - 实现方法：最小二乘法是直接求导找出全局最小；而梯度下降是一种迭代法
  - 效果：最小二乘找到的一定是全局最小，但计算繁琐，且复杂情况下未必有解；梯度下降迭代计算简单，但找到的一般是局部最小，只有在目标函数是凸函数时才是全局最小；到最小点附近时收敛速度会变慢，且对初始点的选择极为敏感

分类模型

- K近邻
- 逻辑斯谛回归
- 决策树

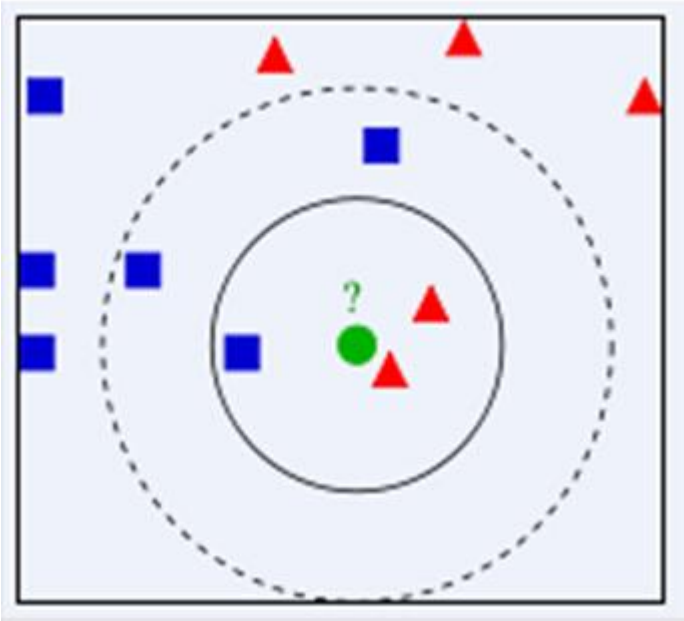


K近邻（KNN）

- 最简单最初级的分类器，就是将全部的训练数据所对应的类别都记录下来，当测试对象的属性和某个训练对象的属性完全匹配时，便可以对其进行分类
- K近邻（k-nearest neighbour, KNN）是一种基本分类方法，通过测量不同特征值之间的距离进行分类。它的思路是：如果一个样本在特征空间中的k个最相似（即特征空间中最邻近）的样本中的大多数属于某一个类别，则该样本也属于这个类别，其中K通常是不大于20的整数
- KNN算法中，所选择的邻居都是已经正确分类的对象

KNN示例

- 绿色圆要被决定赋予哪个类，是红色三角形还是蓝色四方形?
- 如果K=3，由于红色三角形所占比例为2/3，绿色圆将被赋予红色三角形那个类，如果K=5，由于蓝色四方形比例为3/5，因此绿色圆被赋予蓝色四方形类



- *KNN算法的结果很大程度取决于K的选择*

KNN距离计算

- KNN中，通过计算对象间距离来作为各个对象之间的非相似性指标，避免了对象之间的匹配问题，在这里距离一般使用欧氏距离或曼哈顿距离：

欧式距离： $d(x,y)=\sqrt{\sum_{k=1}^{n}(x_k-y_k)^2}$

曼哈顿距离： $d(x,y)=\sqrt{\sum_{k=1}^{n}|x_k-y_k|}$

KNN算法

- 在训练集中数据和标签已知的情况下，输入测试数据，将测试数据的特征与训练集中对应的特征进行相互比较，找到训练集中与之最为相似的前K个数据，则该测试数据对应的类别就是K个数据中出现次数最多的那个分类，其算法的描述为：

  a) 计算测试数据与各个训练数据之间的距离；

  b) 按照距离的递增关系进行排序；

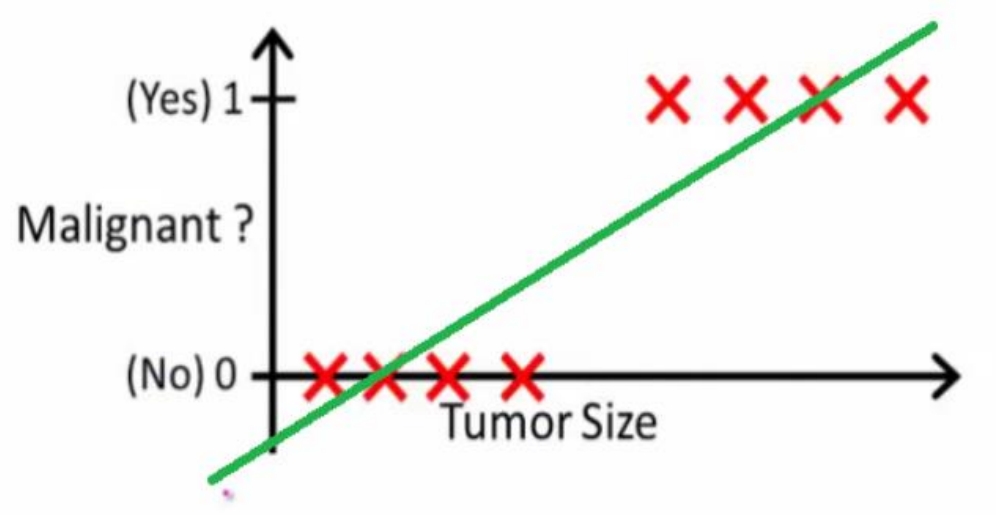
  c) 选取距离最小的K个点；

  d) 确定前K个点所在类别的出现频率；

  e) 返回前K个点中出现频率最高的类别作为测试数据的预测分类。

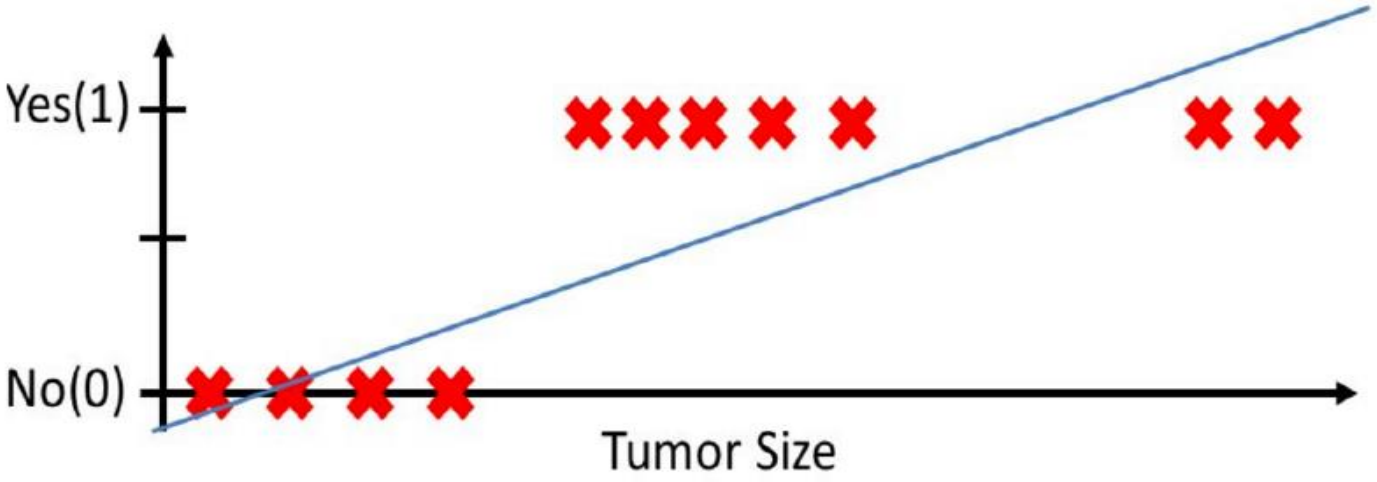
逻辑斯蒂回归

- 线性回归的问题 —— 怎样判断肿瘤是否恶性？

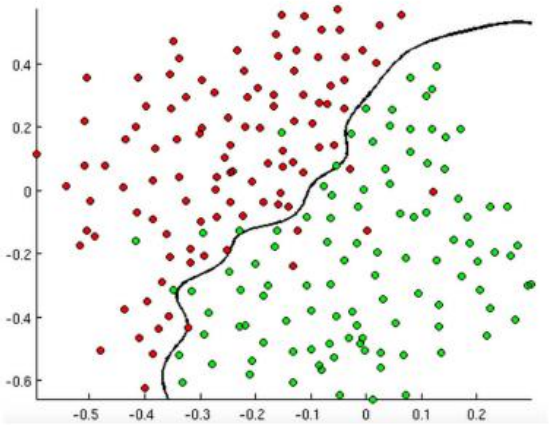


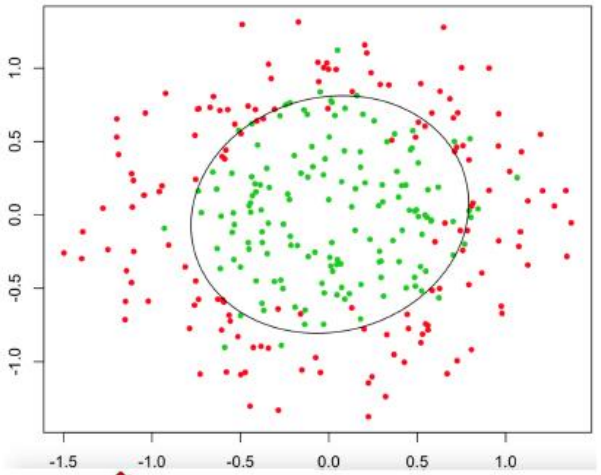
逻辑斯蒂回归

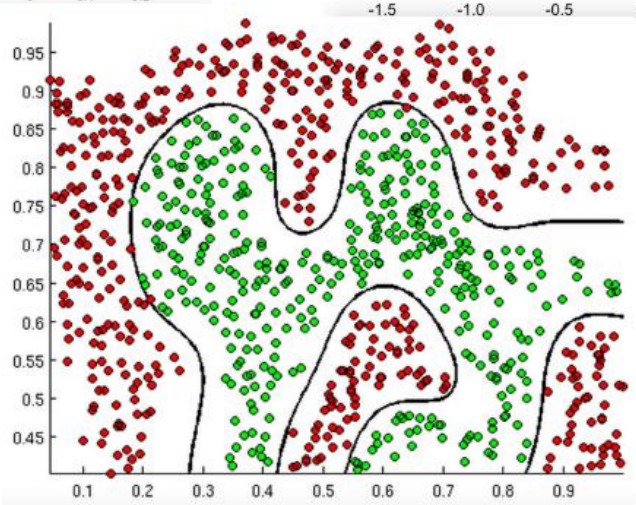
- 线性回归健壮性不够，一旦有噪声，立刻“投降”



逻辑斯蒂回归 —— 分类问题

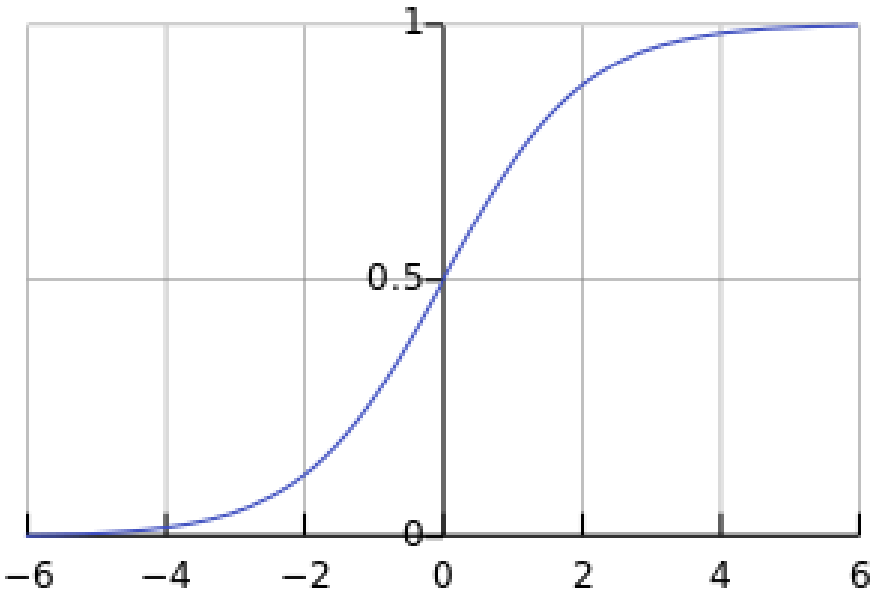


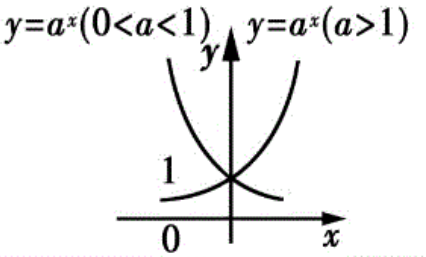




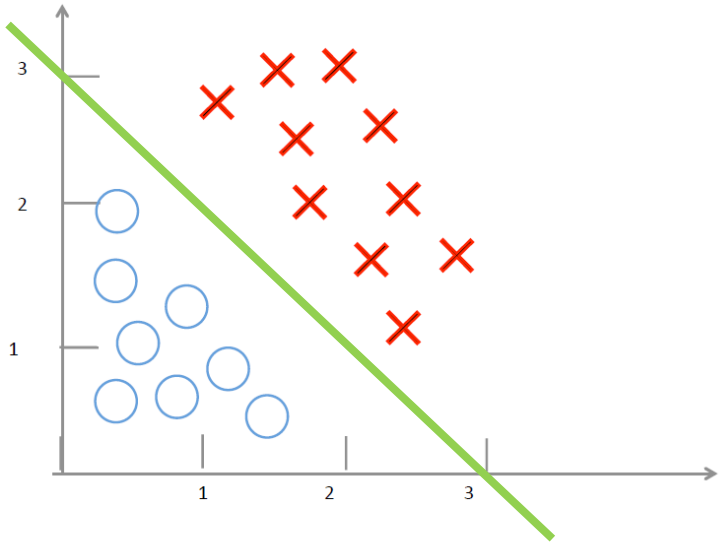
Sigmoid函数（压缩函数）

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





Sigmoid函数（压缩函数）



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

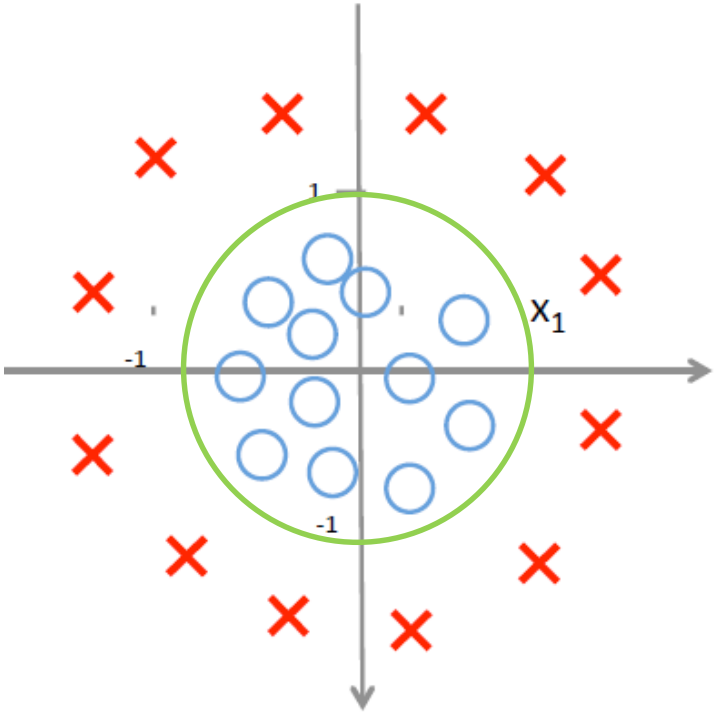
$$g(z) = \frac{1}{1+e^{-z}}$$

取 $\theta_0 = -3$，$\theta_1 = 1$，$\theta_2 = 1$：

得到分类函数：$-3 + x_1 + x_2 = 0$

- 我们将线性回归拟合出来的值用压缩函数进行压缩，压缩完成后用 0.5 做一个概率的判定边界，就能把样本分成两类，即正样本和负样本

Sigmoid函数（压缩函数）



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

取 $\theta_0 = -1$， $\theta_1 = 0$， $\theta_2 = 0$， $\theta_3 = 1$，$\theta_4 = 1$：

得到分类曲线： $x_1^2 + x_2^2 - 1 = 0$

Sigmoid函数（压缩函数）

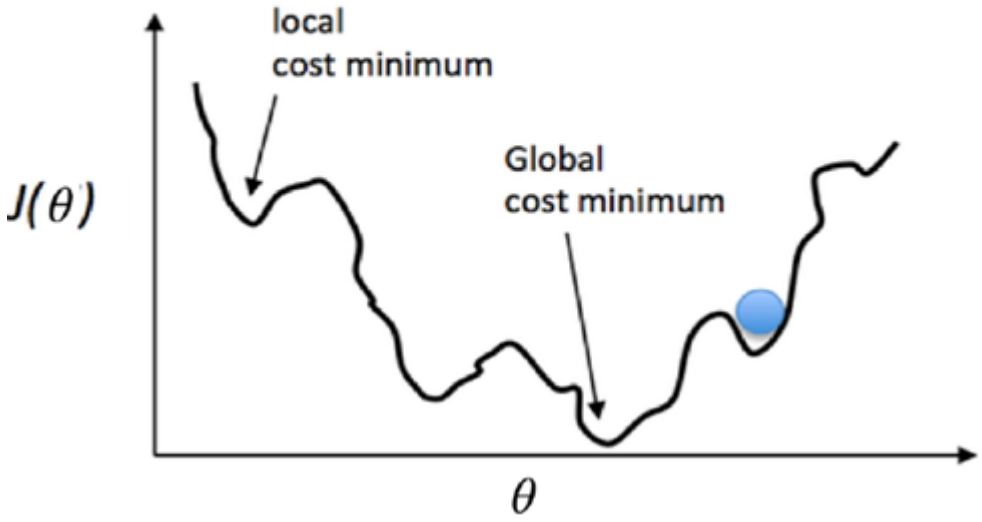
$$g(z)=\frac{1}{1+e^{-z}}$$

- sigmoid函数中，e^{-z} 中 z 的正负决定了 g(z) 的值最后是大于 0.5 还是小于 0.5；即 z 大于 0 时，g(z) 大于 0.5，z 小于 0 时，g(z)小于 0.5
- 当 z 对应的表达式为分类边界时，恰好有分类边界两侧对应 z 正负不同，也就使得分类边界两边分别对应 g(z)>0.5 和 g(z)<0.5，因此根据 g(z) 与 0.5 的大小关系，就可以实现分类

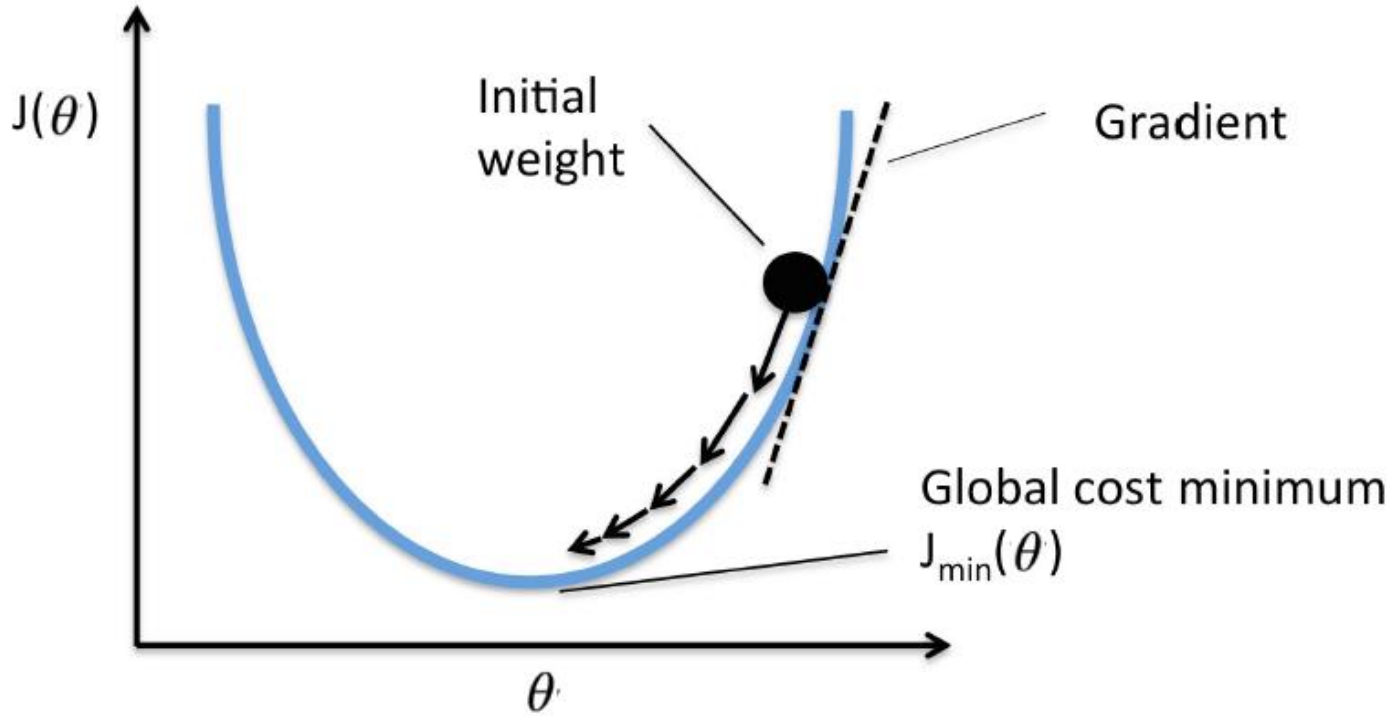
逻辑斯谛回归损失函数

- 平方损失函数的问题

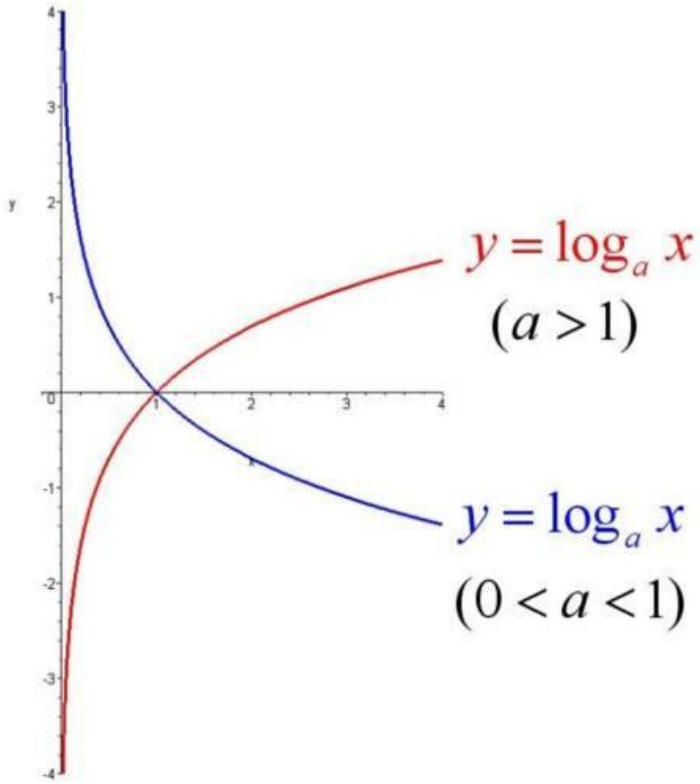
$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$



损失函数

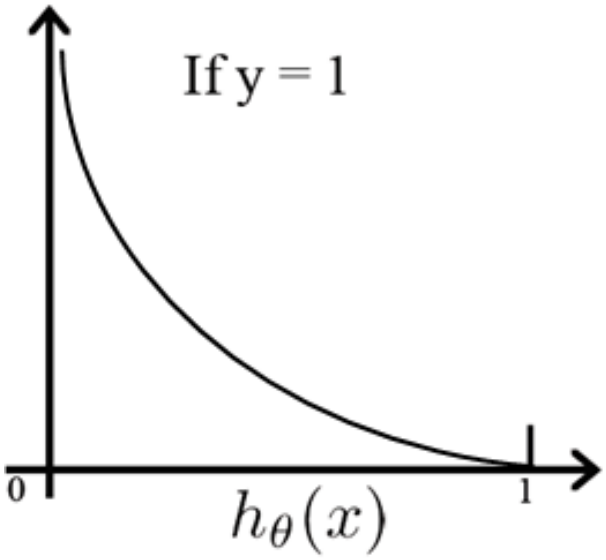


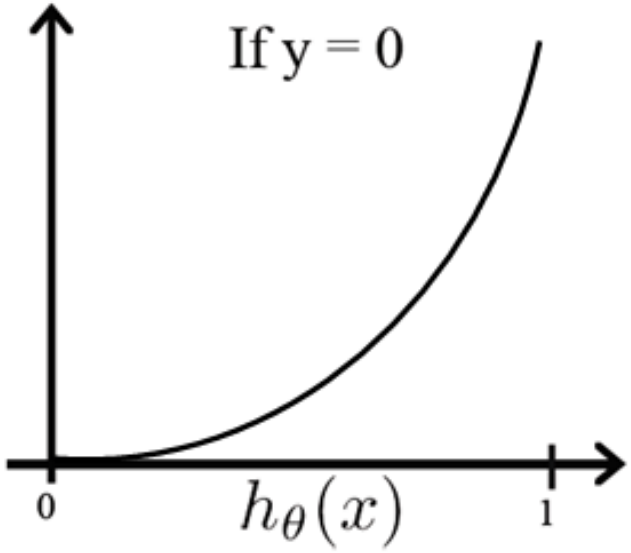
损失函数



损失函数

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$





损失函数

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

$$-(y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)})))$$

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

损失函数

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

为了防止过拟合，加入正则化项

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log h_\theta(x^{(i)}) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

这样，我们获得了一个**凸函数**。

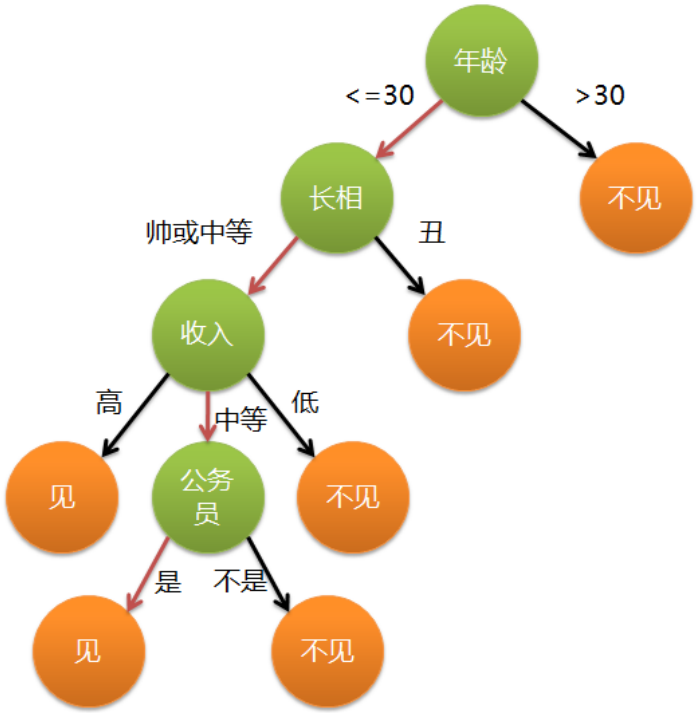
梯度下降法求解

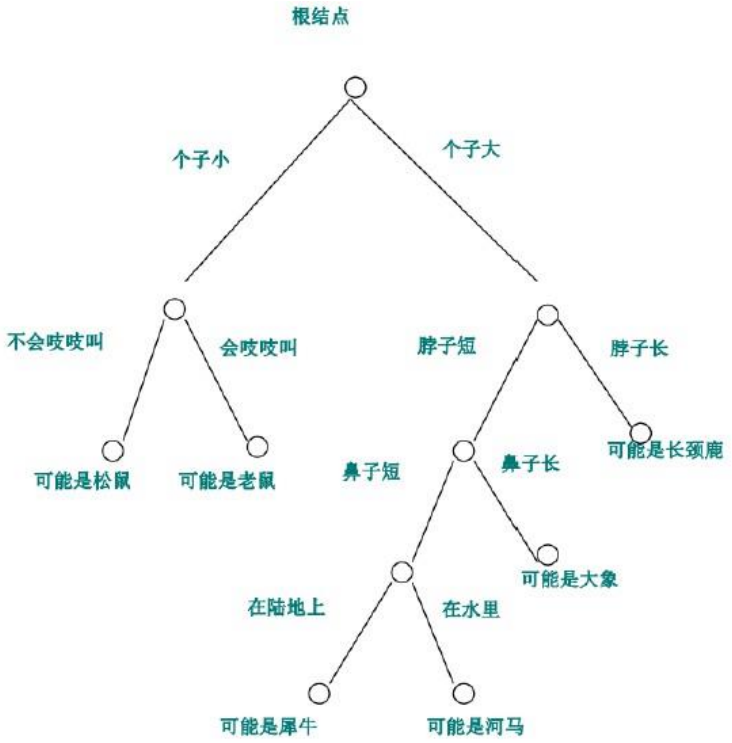
$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log h_\theta(x^{(i)}) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

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

决策树

- 决策树是一种简单高效并且具有强解释性的模型，广泛应用于数据分析领域。其本质是一颗自上而下的由多个判断节点组成的树



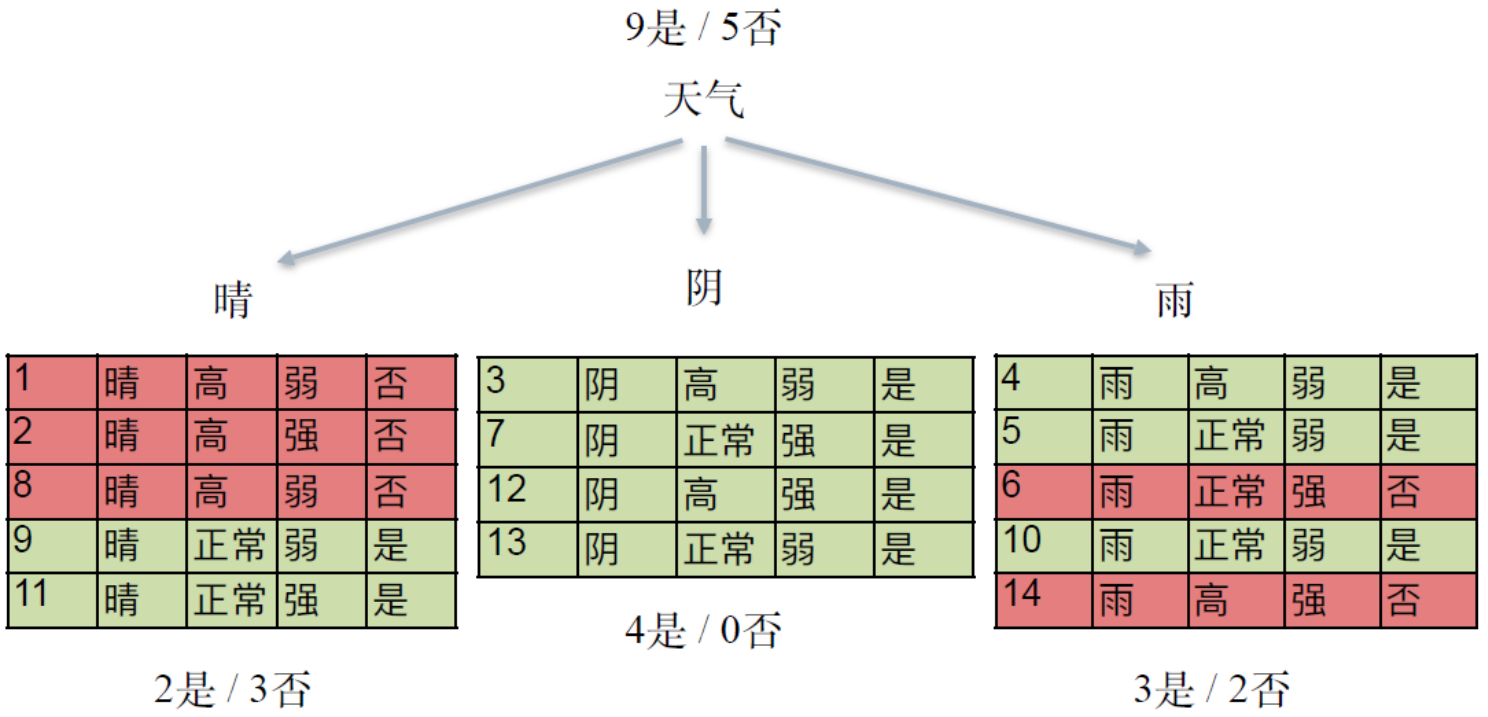


决策树示例

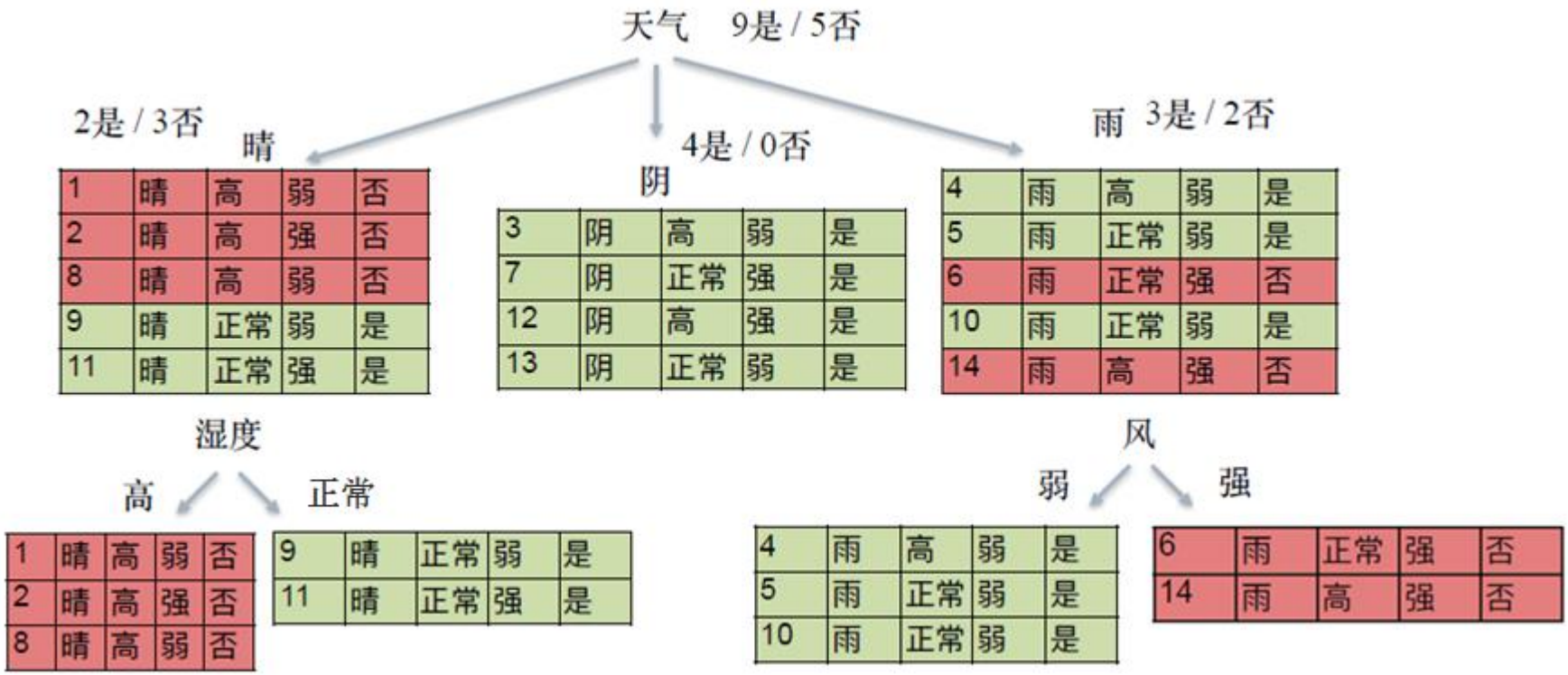
- 预测小明今天是否会出门打球

日期	天气	湿度	风级	打球
1	晴	高	弱	否
2	晴	高	强	否
3	阴	高	弱	是
4	雨	高	弱	是
5	雨	正常	弱	是
6	雨	正常	强	否
7	阴	正常	强	是
8	晴	高	弱	否
9	晴	正常	弱	是
10	雨	正常	弱	是
11	晴	正常	强	是
12	阴	高	强	是
13	阴	正常	弱	是
14	雨	高	强	否

决策树示例



决策树示例



1	晴	高	弱	否
2	晴	高	强	否
8	晴	高	弱	否
9	晴	正常	弱	是
11	晴	正常	强	是

3	阴	高	弱	是
7	阴	正常	强	是
12	阴	高	强	是
13	阴	正常	弱	是

4	雨	高	弱	是
5	雨	正常	弱	是
6	雨	正常	强	否
10	雨	正常	弱	是
14	雨	高	强	否

1	晴	高	弱	否
2	晴	高	强	否
8	晴	高	弱	否

9	晴	正常	弱	是
11	晴	正常	强	是

4	雨	高	弱	是
5	雨	正常	弱	是
10	雨	正常	弱	是

6	雨	正常	强	否
14	雨	高	强	否

决策树与 if-then 规则

- 决策树可以看作一个 if-then 规则的集合
- 由决策树的根节点到叶节点的每一条路径，构建一条规则：路径上内部节点的特征对应着规则的条件（condition），叶节点对应规则的结论
- 决策树的 if-then 规则集合有一个重要性质：互斥并且完备。这就是说，每个实例都被一条规则（一条路径）所覆盖，并且只被这一条规则覆盖

if (condition)
then
else
then

决策树中的 Condition 是什么？

Condition 的确定过程就是**特征选择**的过程

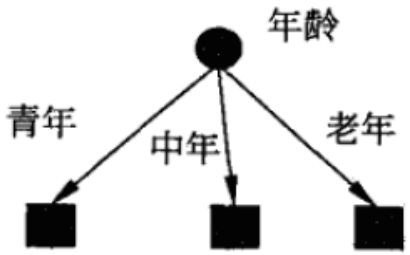
决策树的目标

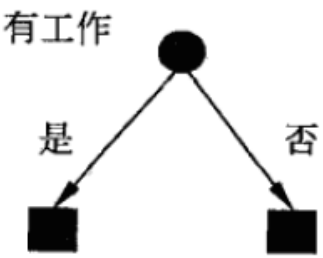
- 决策树学习的本质，是从训练数据集中归纳出一组 if-then 分类规则
- 与训练集不相矛盾的决策树，可能有很多个，也可能一个也没有；所以我们需要选择一个与训练数据集矛盾较小的决策树
- 另一角度，我们可以把决策树看成一个条件概率模型，我们的目标是将实例分配到条件概率更大的那一类中去
- 从所有可能的情况中选择最优决策树，是一个**NP完全问题**，所以我们通常采用启发式算法求解决策树，得到一个**次最优解**
- 采用的算法通常是递归地进行以下过程：**选择最优特征**，并根据该特征对训练数据进行分割，使得各个子数据集都有一个最好的分类

特征选择

- 特征选择就是决定用哪个特征来划分特征空间

ID	年龄	有工作	有自己的房子	信贷情况	类别
1	青年	否	否	一般	否
2	青年	否	否	好	否
3	青年	是	否	好	是
4	青年	是	是	一般	是
5	青年	否	否	一般	否
6	中年	否	否	一般	否
7	中年	否	否	好	否
8	中年	是	是	好	是
9	中年	否	是	非常好	是
10	中年	否	是	非常好	是
11	老年	否	是	非常好	是
12	老年	否	是	好	是
13	老年	是	否	好	是
14	老年	是	否	非常好	是
15	老年	否	否	一般	否





随机变量

- 随机变量（random variable）的本质是一个函数，是从样本空间的子集到实数的映射，将**事件**转换成一个**数值**

实验结果	随机变量
HH	2
HT	1
TH	1
TT	0

- 根据样本空间中的元素不同(即不同的实验结果)，随机变量的值也将随机产生。可以说，随机变量是“数值化”的实验结果
- 在现实生活中，实验结果是描述性的词汇，比如 “硬币的正面”、“反面”。在数学家眼里，这些文字化的叙述太过繁琐，所以拿数字来代表它们

熵

- 熵（entropy）用来衡量随机变量的**不确定性**
- 变量的不确定性越大，熵也就越大

设X是一个取有限个值的离散随机变量，其概率分布为：

$$P(X=x_i)=p_i, \quad i=1,2,\cdots,n$$

则随机变量X的熵定义为：

$$H(X)=-\sum_{i=1}^{n} p_i \log p_i$$

通常，上式中的对数以2为底或者以e为底（自然对数），这时熵的单位分别称为比特（bit）或纳特（nat）。

熵

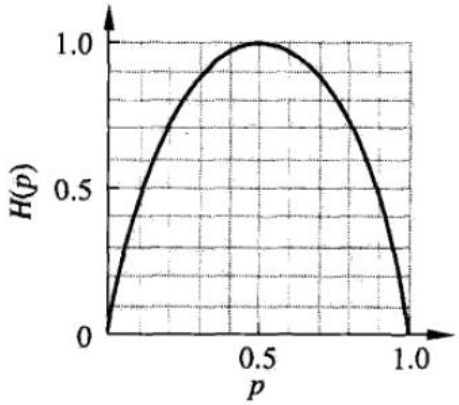
当随机变量只取两个值，例如 1,0 时，则 X 的分布为：

$$P(X=1)=p\,,\quad P(X=0)=1-p\,,\quad 0\leqslant p\leqslant 1$$

熵为：

$$H(p)=-p\log_2 p-(1-p)\log_2(1-p)$$

这时，熵H(p)随概率p变化的曲线如下图所示（单位为比特）：

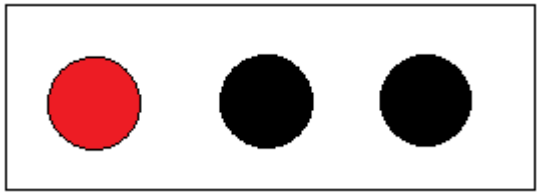


熵的示例

- 给三个球分类

  → 显然一眼就可以看出把红球独自一组，黑球一组；

  → 那么从熵的观点来看，是什么情况呢？



初始状态的熵：

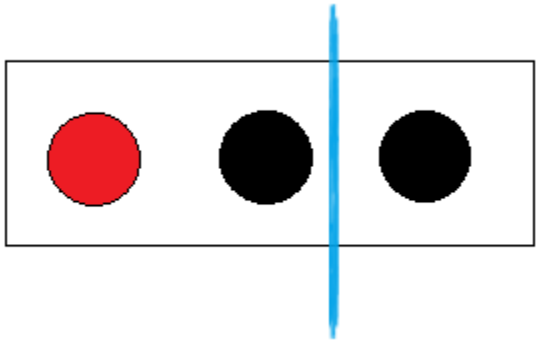
E(三个球) = - 1/3 * log(1/3) - 2/3 * log(2/3) = 0.918

熵的示例

- 第一种分类方法是一个红球、一个黑球一组，另一个黑球自己一组。

  → 在红黑一组中有红球和黑球, 红黑球各自出现的概率是 1/2.

  → 在另一组 100% 出现黑球, 红球的概率是 0



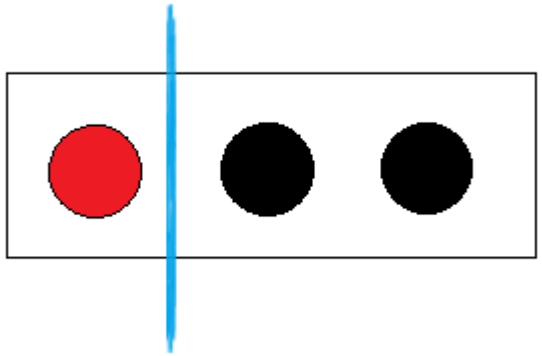
E(红黑|黑) = E(红黑) + E(黑) = - 1/2 * log(1/2) - 1/2 * log (1/2) - 1 * log(1) = 1

可以看到，分类之后熵反而增大了

熵的示例

- 第二种分法就是红球自己一组，剩下两个黑球一组

  → 在红球组中出现黑球的概率是0, 在黑球组中出现红球的概率是0, 这样的分类已经“纯”了，也就是分类后子集中的随机变量已经变成确定性的了



$$E(红|黑黑) = E(红) + E(黑黑) = -1 * \log(1) - 1 * \log(1) = 0$$

决策树的目标

- 我们使用决策树模型的最终目的是利用决策树模型进行分类预测，预测我们给出的一组数据最终属于哪一种类别，这是一个由不确定到确定的过程
- 最终理想的分类是，每一组数据，都能确定性地按照决策树分支找到对应的类别
- 所以我们就选择使数据信息熵下降最快的特征作为分类节点，使得决策树尽快地趋于确定

条件熵（conditional entropy）

- 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：

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

其中，$p_i = P(X = x_i)$

- 熵 H(D) 表示对数据集 D 进行分类的不确定性。
- 条件熵 H(D|A) 指在给定特征 A 的条件下数据集分类的不确定性
- 当熵和条件熵的概率由数据估计得到时，所对应的熵与条件熵分别称为经验熵（empirical entropy）和经验条件熵（empirical conditional entropy）

信息增益

- 特征A对训练数据集D的信息增益 g(D, A)，定义为集合D的经验熵H(D)与特征A给定条件下D的条件熵 H(D|A)之差，即

$$g(D, A) = H(D) - H(D \mid A)$$

- 决策树学习应用**信息增益准则**选择特征
- 经验熵H(D)表示对数据集D进行分类的不确定性。而经验条件熵H(D|A)表示在特征A给定的条件下对数据集D进行分类的不确定性。那么它们的差，即信息增益，就表示由于特征A而使得对数据集D的分类的不确定性减少的程度
- 对于数据集D而言，信息增益依赖于特征，不同的特征往往具有不同的信息增益
- 信息增益大的特征具有更强的分类能力

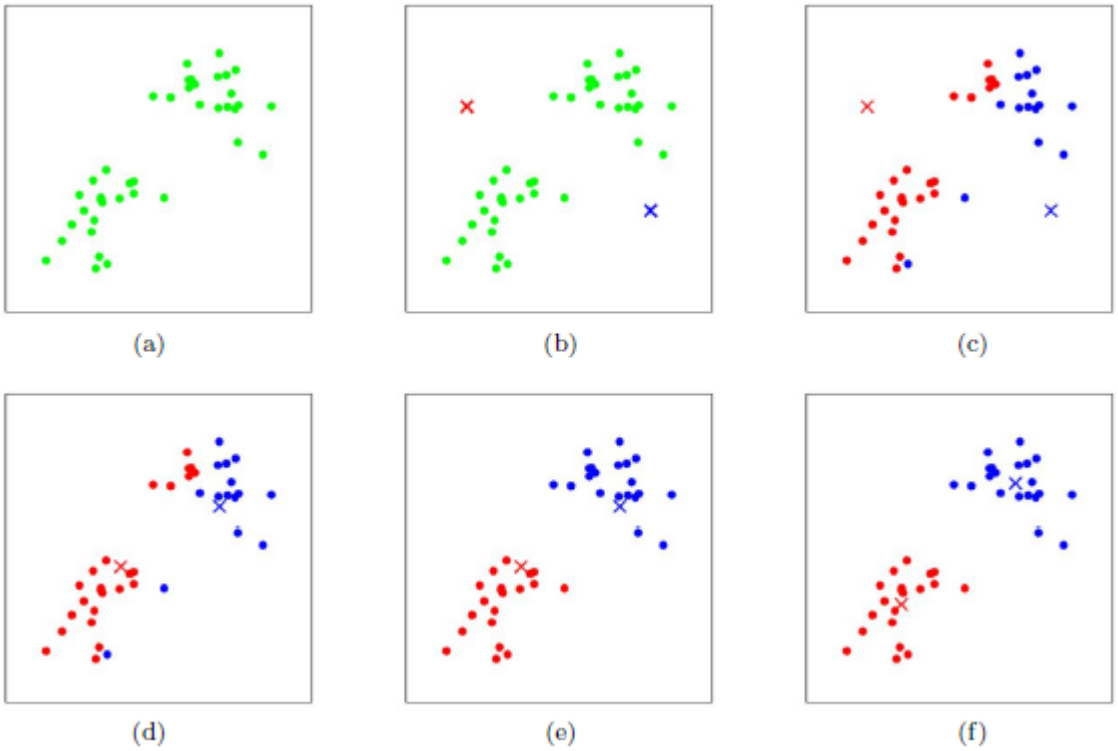
决策树的生成算法

- ID3
  - 决策树（ID3）的训练过程就是找到信息增益最大的特征，然后按照此特征进行分类，然后再找到各类型子集中信息增益最大的特征，然后按照此特征进行分类，最终得到符合要求的模型。
- C4.5
  - C4.5算法在ID3基础上做了改进，用信息增益比来选择特征
- 分类与回归树（CART）
  - 由特征选择、树的生成和剪枝三部分组成，既可以用于分类也可以用于回归

无监督学习

- 聚类
  - k均值
  - 基于密度的聚类
  - 最大期望聚类
- 降维
  - 潜语义分析（LSA）
  - 主成分分析（PCA）
  - 奇异值分解（SVD）

聚类 —— k均值



聚类 —— k均值

- k 均值（k-means）是聚类算法中最为简单、高效的，属于无监督学习算法
- 核心思想：由用户指定k个初始质心（initial centroids)，以作为聚类的类别（cluster），重复迭代直至算法收敛
- 基本算法流程：
  - 选取k个初始质心（作为初始cluster）；
  - repeat:
    - 对每个样本点，计算得到距其最近的质心，将其类别标为该质心所对应的cluster;
    - 重新计算k个cluser对应的质心;
  - until 质心不再发生变化或迭代达到上限

Q & A