《机器学习》课程——

机器学习常见分类算法

主讲人： 徐义田教授

学　校：中国农业大学

目 录

- 朴素贝叶斯
- 决策树
- K近邻 (KNN)
- 支持向量机 (SVM)
- 逻辑回归 (LR)

目前还没有一种被共同接受的机器学习理论框架，但大致可分为：

- 经典的参数估计方法，如传统贝叶斯方法。基于传统统计学，参数相关分布（形式）已知，只需用训练样本来估计参数值；但该方法必须已知样本分布形式，而且其理论往往基于样本数目无穷大。

- 基于经验风险的非线性方法，如人工神经网络。该方法利用已知样本建立非线性模型，克服了参数估计方法的困难。但是该方法更像一个“黑盒子”，缺乏统一的数学理论。

- 统计学习理论，如支持向量机。它是专门用于研究小样本情况下机器学习规律的理论。有一套完整的理论体系，可以解决许多原来难以解决的问题，比如局部极小点问题等。

➢朴素贝叶斯

一、基本概念

1. 先验概率

从以往的数据中直接简单分析得到的概率；

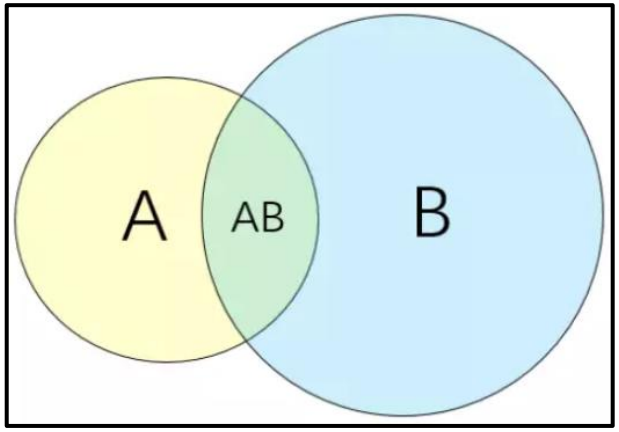
2. 后验概率

利用得到的概率信息，重新加以修正加工后的概率；

3. 朴素贝叶斯算法

Bayes算法可简单概括为：数据+先验概率=后验概率。它属于生成性方法，不需要构建模型或者映射，仅进行简单计算即可进行分类，在垃圾邮件分类和文本分类等众多领域有广泛应用。

二、相关公式



1. 条件独立公式 $P(X,Y)=P(X)P(Y)$

2. 条件概率公式 $P(Y|X)=\frac{P(X,Y)}{P(X)}$，从而有$P(Y|X)=\frac{P(X|Y)P(Y)}{P(X)}$

3. 贝叶斯公式

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

所属类别　　测试数据

贝叶斯算法是用训练数据中的信息，为测试样本提供类别标签的方法。

三、朴素贝叶斯算法

1. 模型

假设样本集合为：

$$(x_1^{(1)}, x_2^{(1)},\ldots, x_n^{(1)},y_1), (x_1^{(2)}, x_2^{(2)},\ldots, x_n^{(2)},y_2),\ldots, (x_1^{(m)}, x_2^{(m)},\ldots, x_n^{(m)},y_1)$$

共m个样本，每个样本n维，标签有K个类别：$C_1, C_2, \cdots, C_K$，目标是找出最大的条件概率对应的类别：

$$C_{label} = \arg\max_{C_k} P(Y = C_K | X = X^{test})$$

利用条件概率转化为先验分布可得：

相同

$$C_{label} = \arg\max_{C_k} P(X = X^{test} | Y = C_k) P(Y = C_k) / P(X = X^{test})$$

简化为

$$C_{label} = \arg\max_{C_k} P(X = X^{test} | Y = C_k) P(Y = C_k)$$

因此，只需考虑如何从数据中获得分布$P(X = X^{test} | Y = C_k)$以及$P(Y = C_k)$.

2. 参数估计

① $P(Y=C_k)$的获得利用频率来近似，即类别C_k在训练集里面出现的频数，即样本类别 C_k 出现的次数 m_k 除以样本总数 m；

②$P(X=X^{test}|Y=C_k)$的获得则需要假设每个样本各特征相互独立。即：

$$P(X=X^{test}|Y=C_k)=P(X_1=X_1^{test},X_2=X_2^{test},\cdots,X_n=X_n^{test}|Y=C_k)$$

假设特征相互独立

$$P(X=X^{test}|Y=C_k)=P(X_1=X_1^{test}|Y=C_k)\times\cdots\times P(X_n=X_n^{test}|Y=C_k)$$

如果特征之间非常不独立，可能导致预测不准确。但是一般情况下，样本的特征之间独立这个条件的确是弱成立的，尤其是数据量非常大的时候。牺牲准确性来简化计算就是贝叶斯模型所做的必要取舍。

2. 参数估计

- 只需考虑$P(X_j = X_j^{test} | Y = C_k)$如何计算。

① 如果X_j是离散值，则假设X_j符合多项式分布，则$P(X_j = X_j^{test} | Y = C_k)$就是在样本类别$C_k$中，特征$X_j^{test}$出现的频率，即

$$P(X_j = X_j^{test} | Y = C_k) = \frac{m_{kj}^{test}}{m_k}$$

第 j 维特征在类别为C_k的样本中出现的次数

类别为C_k的样本出现的所有不同特征个数

注：测试样本的某个特征在样本中没出现过，则会导致$P(X_j = X_j^{test} | Y = C_k)$为0，因此需引入拉普拉斯平滑，即：

$$P(X_j = X_j^{test} | Y = C_k) = \frac{m_{kj}^{test} + \lambda}{m_k + O_j\lambda}$$

其中 λ 为一个大于0的常数，一般取1； O_j为第 j 个特征的取值个数。

2. 参数估计

- 只需考虑 $P(X_j = X_j^{test} | Y = C_k)$如何计算。

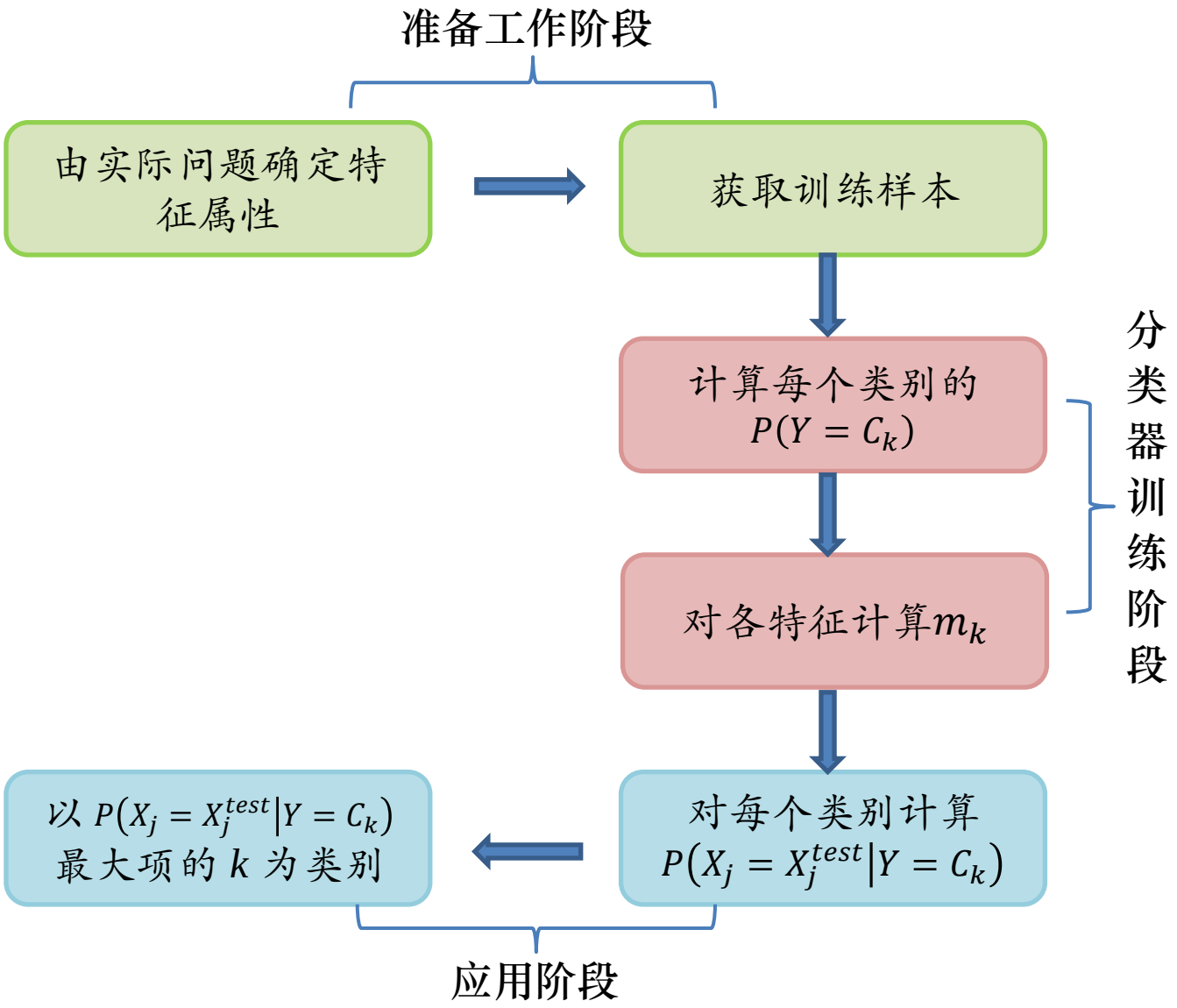
② 如果X_j是连续值，则假设X_j符合正态分布，则

$$P(X_j = X_j^{test} | Y = C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{(X_j^{test} - \mu_k)^2}{2\sigma_k^2}\right)$$

其中μ_k 和σ_k^2是正态分布的期望和方差。其中μ_k为在样本类别C_k中，所有 X_j 的平均值； σ_k^2为在样本类别C_k中，所有X_j的方差，对于一个连续的样本值，带入上述正态分布公式，即可得概率分布。

三、朴素贝叶斯算法

3. 流程



4. 小结

- 优点：

① 理论上，朴素贝叶斯模型具有最小的误差率，有稳定的分类效率，因此常被用做测试分类精度的基准算法。

② 对小规模的数据表现很好，能处理多分类任务，当数据量超出内存时，可以一批批进行增量训练。

- 缺点：

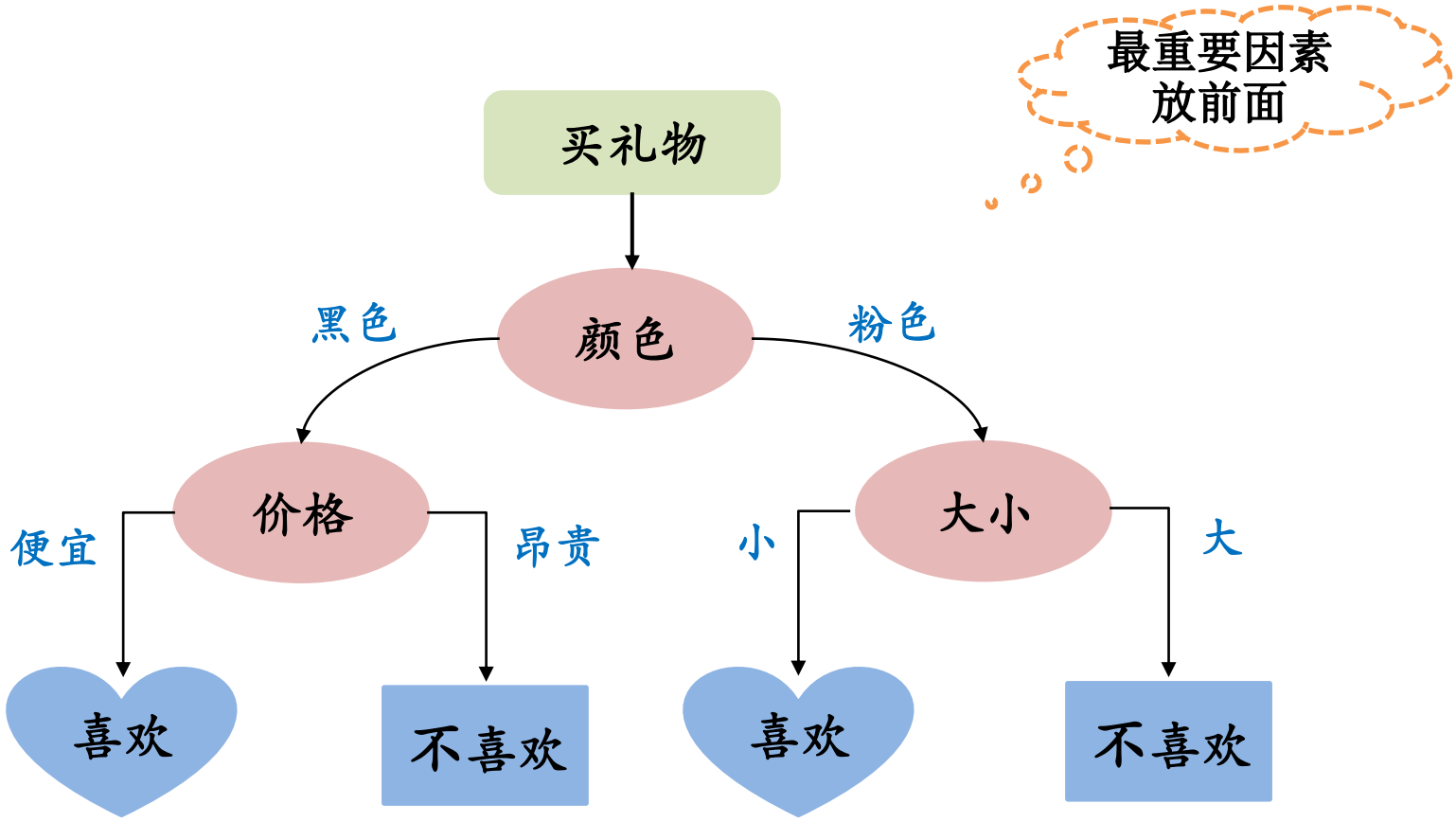
① 属性之间相关性较大时，分类效果不好。

② 先验概率很多时候取决于假设，假设的模型可以有很多种，因此在某些时候会由于假设的先验模型的原因导致预测效果不佳。

决策树

一、决策树基本概念

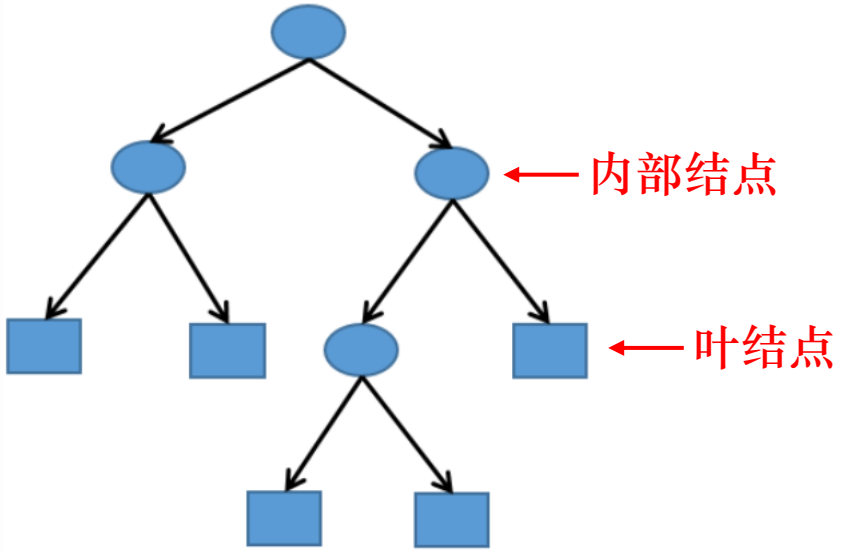
决策树的建立过程与人脑针对某事件做决策时的思维类似，比如你想要给朋友买一件生日礼物，往往需要经历以下过程：



一、决策树基本概念

决策树是一种基于“树”结构进行决策的经典机器学习方法。其基本思想是以信息熵为度量构造一颗熵值下降最快的树，到叶子节点处，熵值为0。其构成如下：

- 结点
  - 内部结点：表示一个特征或属性
  - 叶结点：预测结果
- 有向边



二、决策树常见算法

- 第一个决策树： CLS（Concept Learning System）

  [E. B. Hunt, J. Marin, and P. T. Stone's book "Experiments in Induction" published by Academic Press in 1966]

- 使决策树受到关注，成为机器学习主流技术的算法：ID3

  [J. R. Quinlan's paper in a book "Expert Systems in the Micro Electronic Age" edited by D. Michie, published by Edinburgh University Press in 1979]

- 最常用的决策树算法：C4.5

  [J. R. Quinlan's book "C4.5: Programs for Machine Learning" , published by Morgan Kaufmann in 1993]

- 可以用于回归任务的决策树算法：CART

  [L. Breiman, J. H. Fridman, R. A. Olshen, and C. J.Stone's book "Classification and Regression Trees" published by Wadsworth in1984]

- 基于决策树的强大算法：RF（Random Forest）

  [L. Breiman's MLJ'01 paper "Random Forest"]

三、决策树构造过程

1. 特征选择

从众多特征中选择一个作为当前节点分裂的标准，不同的选择标准导致了不同的决策树算法，如ID3(以信息增益为标准)、CART(以Gini指数为标准)。

2. 决策树生成

根据每个节点选出的特征，从上至下递归的生成子节点，直到数据集不可再分。每一步划分都使得分类的不确定性降低。

3. 决策树裁剪

为解决过拟合问题，利用剪枝来缩小结构规模。

四、决策树经典算法 ID3

1. 信息增益

- 熵

在信息论中，熵是随机变量不确定性的度量，熵越大，随机变量的不确定性越大。设X是取有限个值的离散随机变量，其概率分布为：

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

则随机变量X的熵定义为：

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

熵的取值不会无限大，被控制在如下范围内：

$$0 \le H(p) \le \log n$$

四、决策树经典算法 ID3

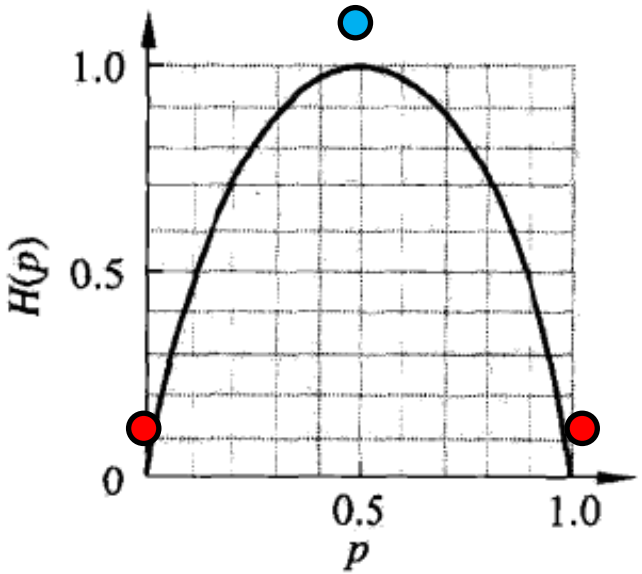
1. 信息增益

- 熵

当随机变量X服从伯努利分布，即：

$$P(X=1)=p, \qquad P(X=0)=1-p, \qquad 0 \le p \le 1$$

则熵为：$H(p)=-plog_2p-(1-p)log_2(1-p)$



分布为伯努利分布时，熵与概率的关系图

1. 信息增益

- 条件熵

设随机变量(X,Y)的联合概率分布为：$P(X=x_i,Y=y_i)=p_{ij}$

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，定义为：

$$H(Y|X)=\sum_{i=1}^{n}p_iH(Y|X=x_i)，\text{其中 } p_i=P(X=x_i)$$

注：当熵和条件熵中的概率分布是通过数据集，利用参数估计方法得到时，被称为经验熵和经验条件熵，该形式更为常用。

四、决策树经典算法 ID3

1. 信息增益

- 信息增益定义

表示由于得知特征A的信息后，数据集D的分类不确定性减少的程度，定义为：Gain(D,A)=H(D)-H(D|A),即集合D的经验熵H(D)与特征A给定下的经验条件熵H(D|A)之差。

- 信息增益理解

① 选择信息增益最大的特征作为划分标准，说明利用该特征划分会减小分类不确定性，因此往往取得更好的分类精度。

② 然而，信息增益偏向取值较多的特征，因为取值多的特征更容易得到纯度更高的子集，使得不确定性更低。C4.5算法利用信息增益率克服了这一缺点。

四、决策树经典算法 ID3

2. 具体算法

输入：数据集 D，特征集 A，阈值 ε；

输出：决策树 T.

Step 1：若D中所有实例属于同一类C_k，则T为单结点树，将C_k作为该结点的类标记，返回T；

Step 2：若A=∅，则T为单结点树，并将D中实例数最大的类C_k作为该节点的类标记，返回T；

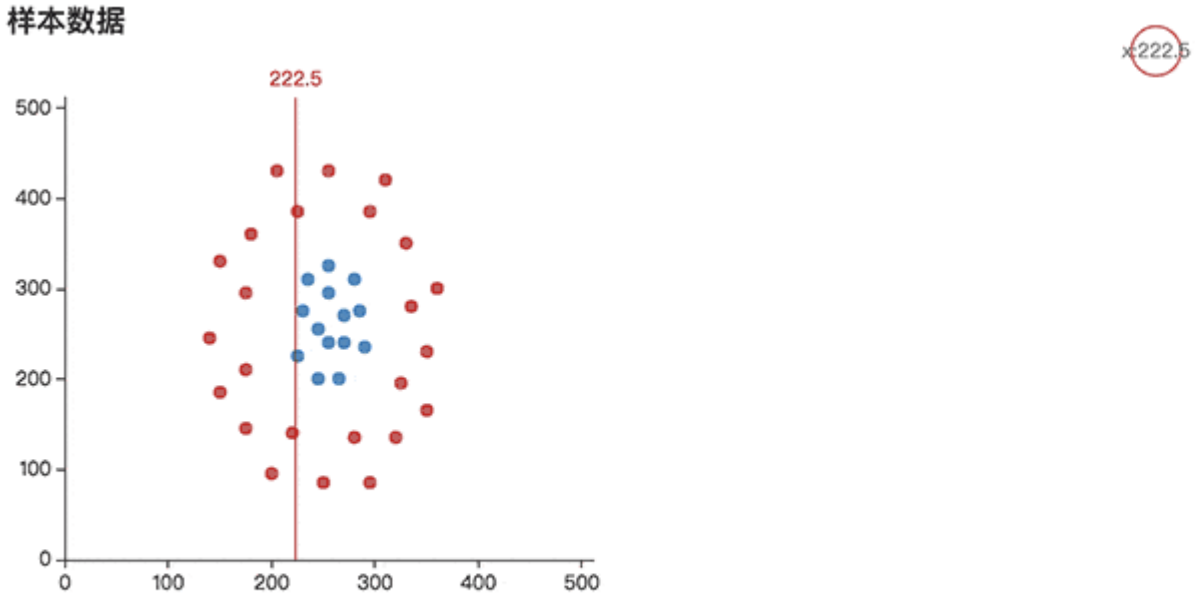
Step 3：计算A中各特征对D的信息增益，选择最大的特征A_k；

Step 4：若所有的信息增益小于阈值ε，则T为单结点树，并将D中实例数最大的类C_k作为该节点的类标记，返回T；否则按照最大的特征将数据集D分割为若干非空子集D_i，并将D_i中实例数最大的类作为标记，返回T；

Step 5：对第i个结点，以D_i为训练集，以剩余特征为特征集合，递归调用Step1~Step4，得到子树T_i，返回T_i.

四、决策树经典算法 ID3

四、决策树经典算法 ID3



以上图像来自SIGAI的云端实验室

3. 剪枝

为了尽可能正确分类训练样本，有可能造成分支过多，可通过主动去掉一些分支来降低过拟合的风险。基本策略如下：

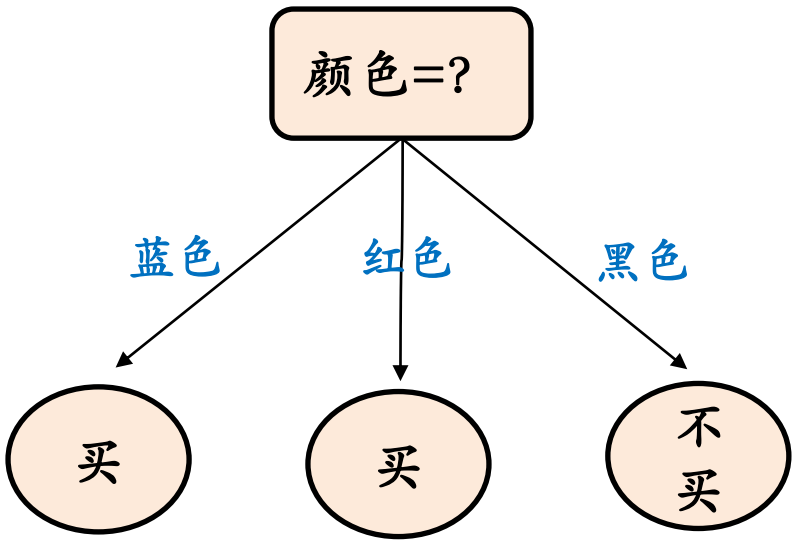
- 预剪枝：在决策树生成的过程中，提前终止某些分支的生长
- 后剪枝：生成一颗完整的树之后，再“回头”剪枝（自下往上）

剪枝过程中需评估剪枝前后决策树的优劣，常用方法有：

- 加入正则化项：叶节点的个数|T|代表模型的复杂程度，因此常会在树的分类精确率与树的叶节点个数之间做一个平衡；
- 预留法：预留一部分数据进行剪枝前后的精度测试，若剪枝使得精度变高，则剪枝，否则不剪。

3.1 预剪枝

基本思想：按照某一节点划分前和划分后，对于测试数据集的分类准确率是提高还是降低，若提高则划分，否则该节点为叶子结点，不再向下划分。



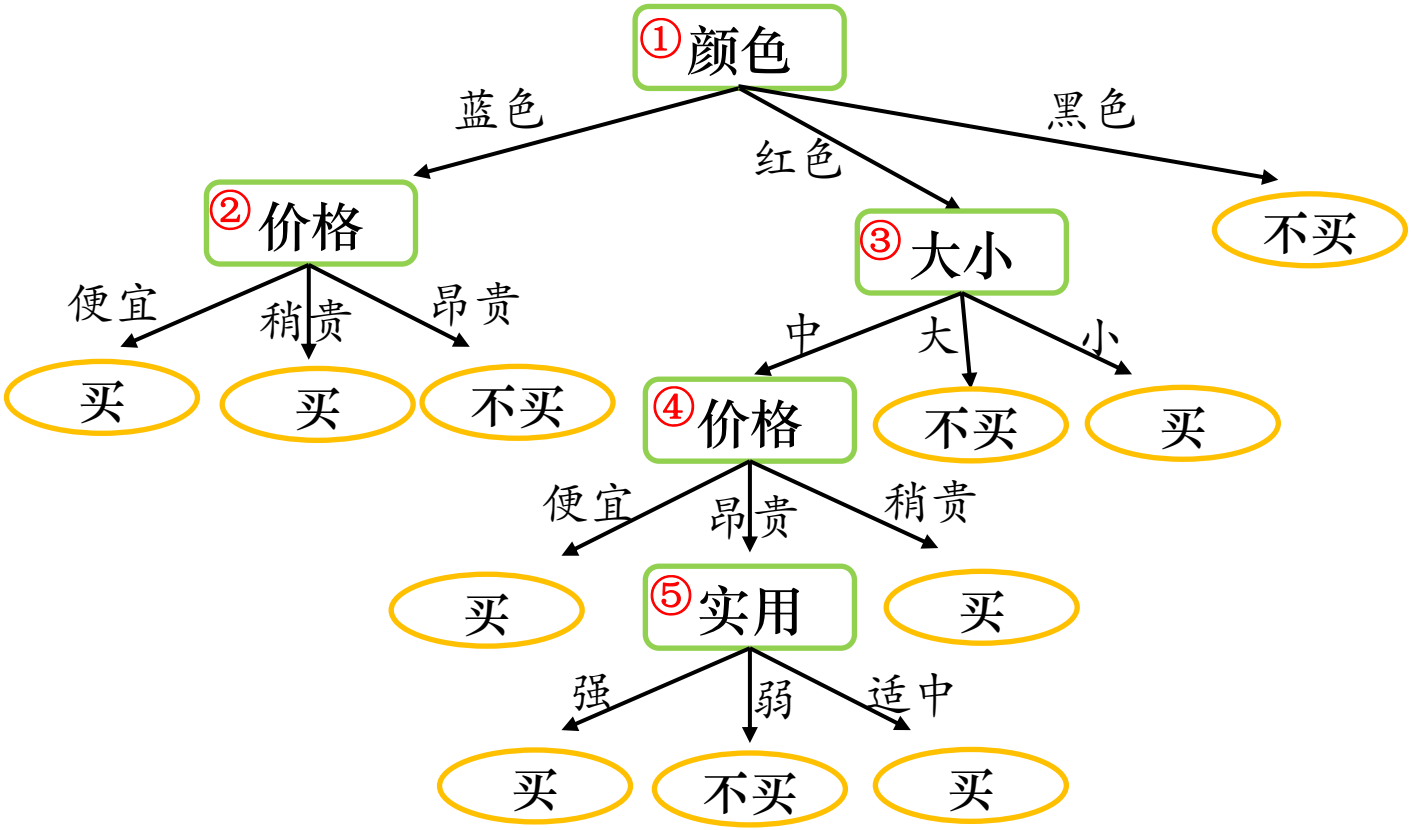
“天气=？”这一节点
划分前，测试集精度：76%
划分后，测试机精度：82%
预剪枝决策：划分

“天气=？”这一节点
划分前，测试集精度：76%
划分后，测试机精度：60%
预剪枝决策：不划分

四、决策树经典算法 ID3

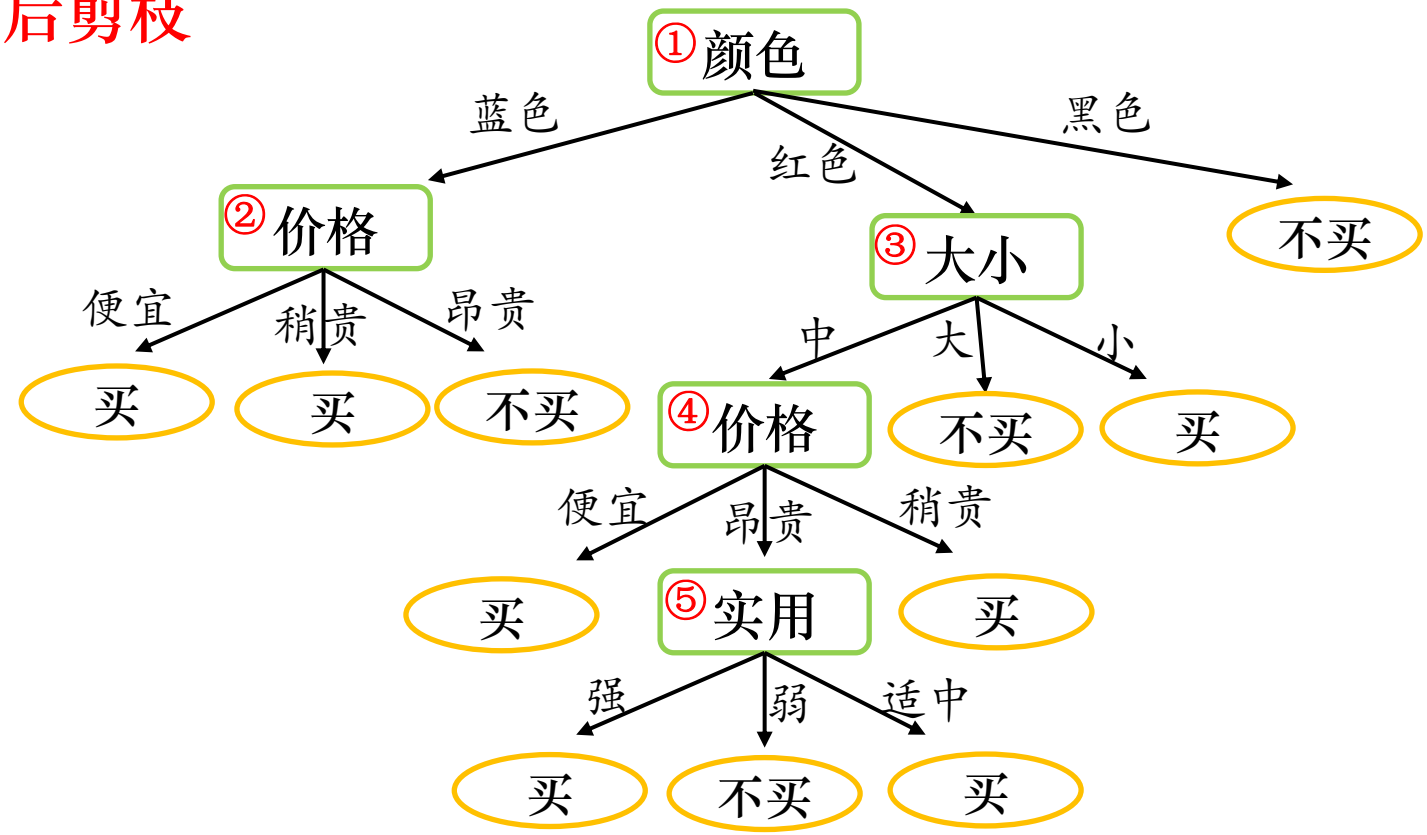
3.2 后剪枝

基本思想：构建一棵树，然后自下向上(由⑤到①)考虑每一个内部节点，判断该节点划分前后精度的变化，若不划分会使得测试精度变高，则剪枝；



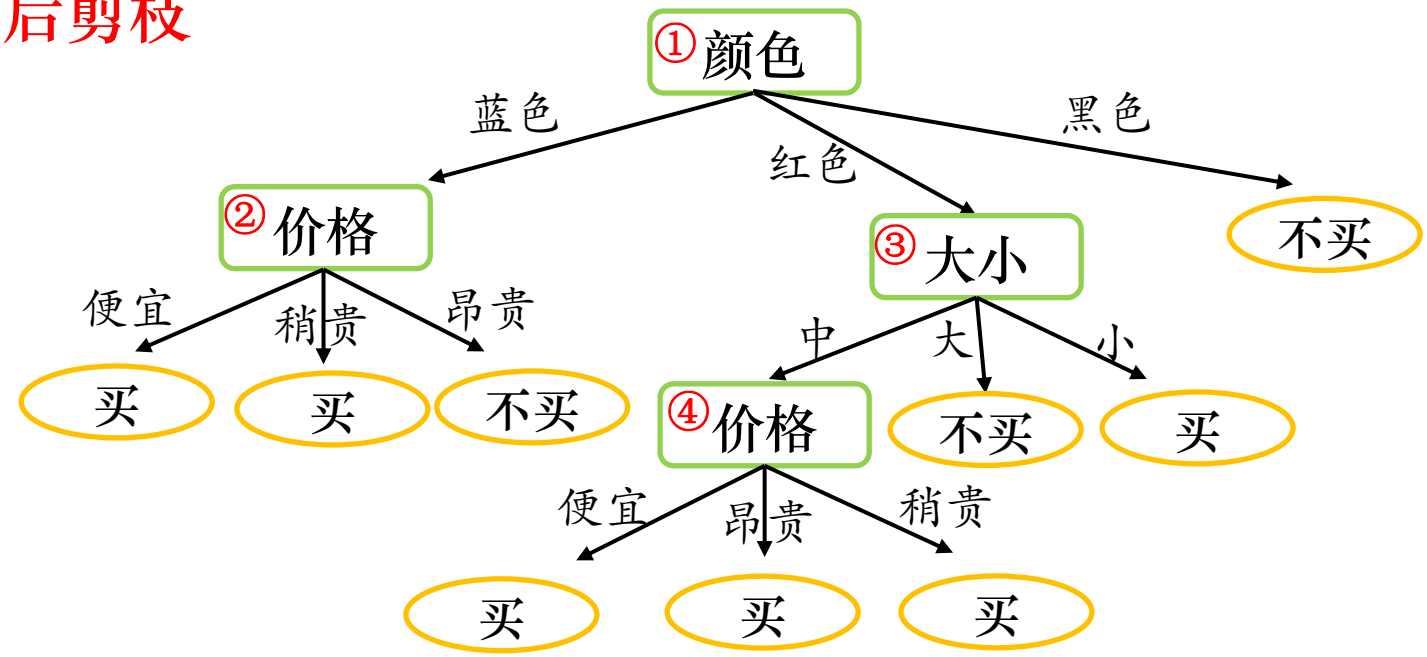
四、决策树经典算法 ID3

3.2 后剪枝



对于⑤而言，假设剪枝前精度为56%，若剪枝，该节点对应两个买，一个不买，因为替换为叶节点“买”，此时若精度高于56%，则确定剪枝；若低于56%则不剪枝。然后依次考虑④、③、②即可。

3.2 后剪枝



精度上升，确定剪枝；
精度下降，不剪枝。

对于⑤而言，假设剪枝前精度为56%，若剪枝，该节点对应两个买，一个不买，因为替换为叶节点“买”，此时若精度高于56%，则确定剪枝；若低于56%则不剪枝。然后依次考虑④、③、②即可。

五、决策树算法小结

前面只介绍了ID3，但常用的还有C4.5，CART以及由此衍生出的随机森林算法、集成决策树等，他们针对不同的情况对ID3进行了改进。

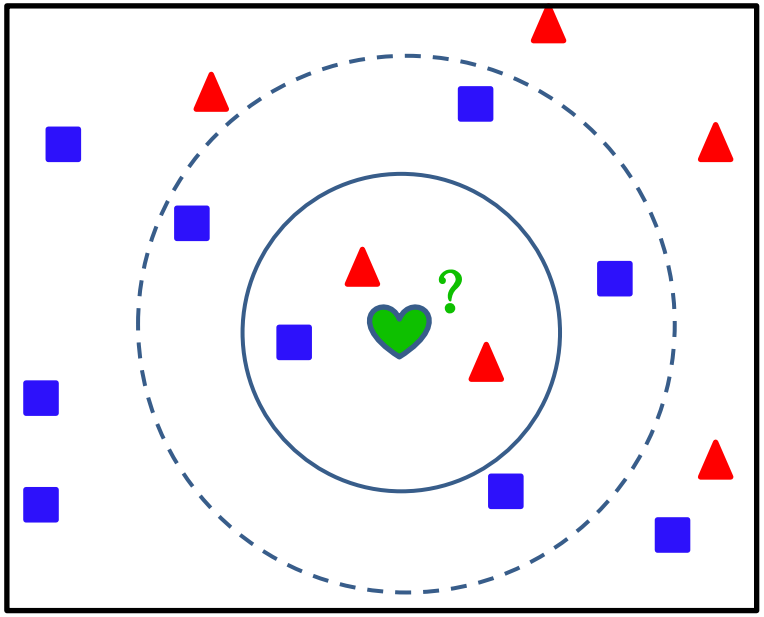
- 优点
  - 理解和解释起来简单，且决策树模型可以想象；
  - 需要准备的数据量不大且分类速度很快；
  - 能够处理多输出的问题。

- 缺点
  - 决策树算法学习者可以创建复杂的树，但容易过拟合；
  - 决策树的结果可能是不稳定的，因为在数据中一个很小的变化可能导致生成一个完全不同的树。

➢K近邻(KNN)

一、K近邻算法概念

K 最近邻 (k-Nearest Neighbor，KNN)，是一种常用的有监督分类算法。其基本思路是：如果一个待分类样本在特征空间中的 k 个最相似(即特征空间中K近邻)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，即近朱者赤，近墨者黑。



K值选取很重要

图中k=3时，将绿色待分类点预测为红色三角▲；k=7时，预测为蓝色方块■。

二、KNN算法四要素

1. 数据集

必须给定一个带标签的数据集，计算待分类样本与所有数据点的距离，继而得出分类结果。并且由于计算距离的需要，每个样本必须以向量的形式存在，每一个维度必须是量化可比较的；

2. 样本间距离计算

多数情况下欧氏距离即可，但也可采用其他距离：

- 欧式距离： $d_{xy} = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$
- 曼哈顿距离： $d_{xy} = \sum_{i=1}^{n}|x_i - y_i|$
- 闵可夫斯基距离： $d_{xy} = \sqrt[p]{\sum_{i=1}^{n}(x_i - y_i)^p}$

3. K值的选取

K值的选取无固定经验，一般通过交叉验证等来选择合适的值。不同K值会有不同分类效果，从而影响算法的偏差与方差。

二、KNN算法四要素

3. K值的选取

- 偏差

模型输出值与真实值之间的差异。偏差越高，则数据越容易欠拟合(Underfitting)，未能充分利用数据中的有效信息。K值很大时偏差大。

- 方差

对数据微小改变的敏感程度。假如有一组同一类的样本，它们的特征之间只有微小差异，我们的模型能很好处理这些微小的变化则方差应该接近0；但现实中存在很多噪声(即存在不同类别的样本，其特征向量差异很小)，方差越高，越容易过拟合(Overfiiting)，对噪声越敏感。K值很小时方差大。

三、KNN算法实现方式

- 蛮力实现

要找到k个最近的邻居来做预测，那么只需要计算预测样本和所有训练集中的样本的距离，然后计算出最小的k个距离即可，接着多数表决，很容易做出预测。

该方法简单直接，在样本量及特征少的时候有效。但是当样本数和特征数成千上万时，算法时间效率很低。因此，这个方法我们一般称之为蛮力实现。

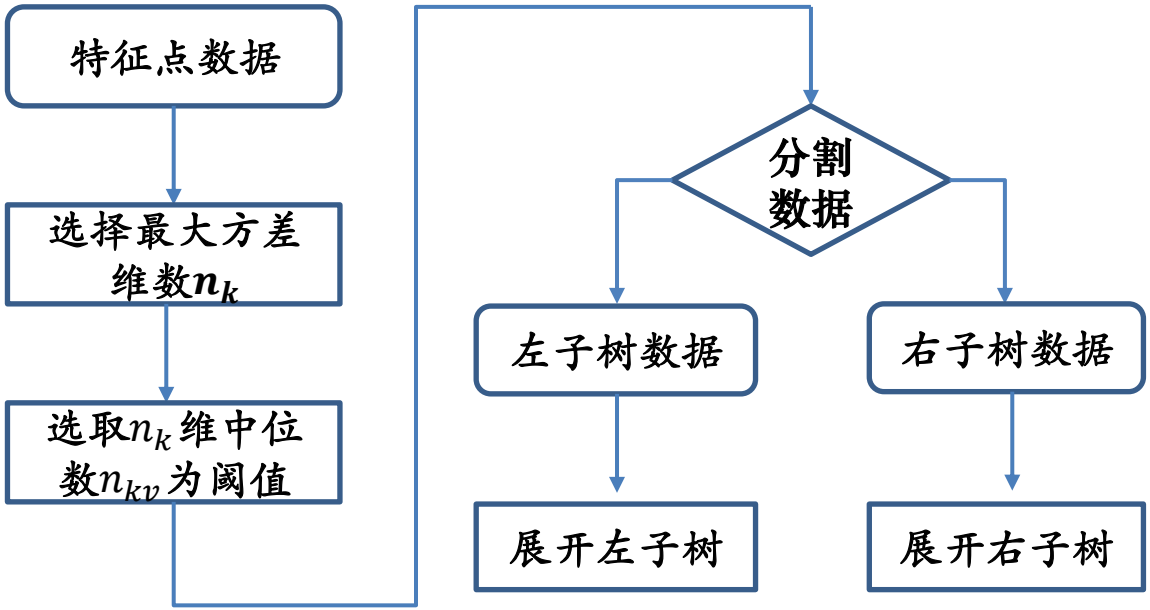
- 更高效的KNN算法之KD树

当处理大规模问题时，通常会采用KD树(K-Dimension tree)结构来提高KNN性能，KD树先对搜索空间进行层次划分，继而进行预测。KD中的K指的不再是最近的样本个数，而是特征维数。

四、KD树

1. KD树的建立

KD树对m个样本的n维特征，分别计算n个特征取值的方差，用方差最大的第k维特征n_k作为根节点。选取该特征的中位数n_{kv}对应的样本作为划分点，将所有第k维特征的取值小于n_{kv}的样本划入左侧区域(左子树)，其他划入右子树；继而针对左右两侧，继续分别寻找剩余特征里面方差的最大的作为划分点，递归的生成KD树。



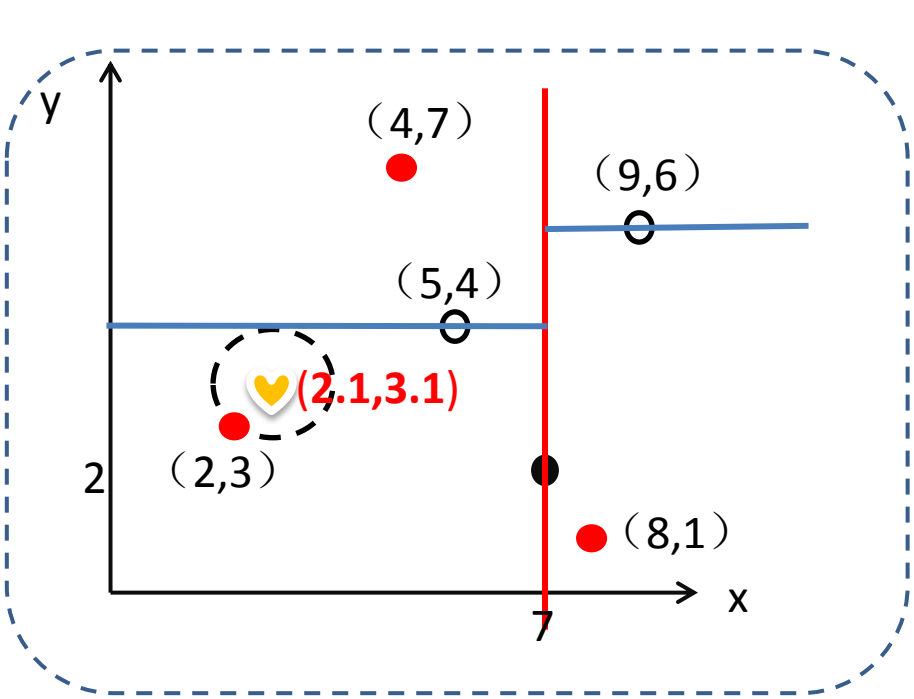
2. KD树搜索最近邻

对于一个目标点，我们首先在KD树里面找到与目标点在同一区域的叶子节点。以目标点为圆心，以目标点到叶子节点样本实例的距离为半径，得到一个球，该区域内最近邻的点一定在这个超球体内部。

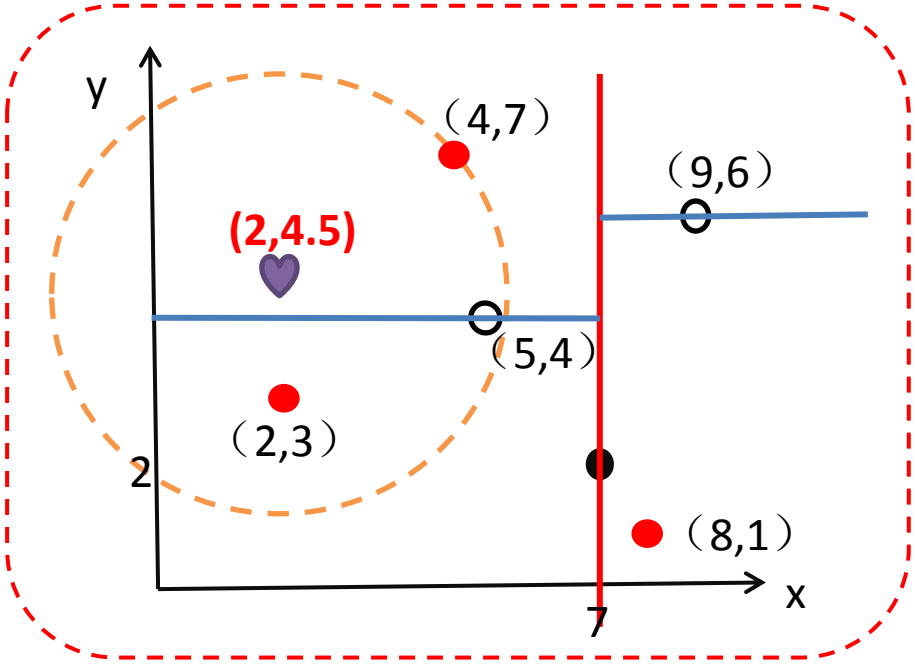
然后检查该球体与搜索区域的边界是否相交，若不相交，则该区域内的最近邻点即为最近邻。若相交，则需在另一侧区域内继续寻找最近邻点，保留更近的最近邻点。若每一步进行下去都会相交，则至多回溯到根节点时，算法结束，此时保存的最近邻节点就是最终的最近邻。

四、KD树

2. KD树搜索最近邻(例)



不相交



相交

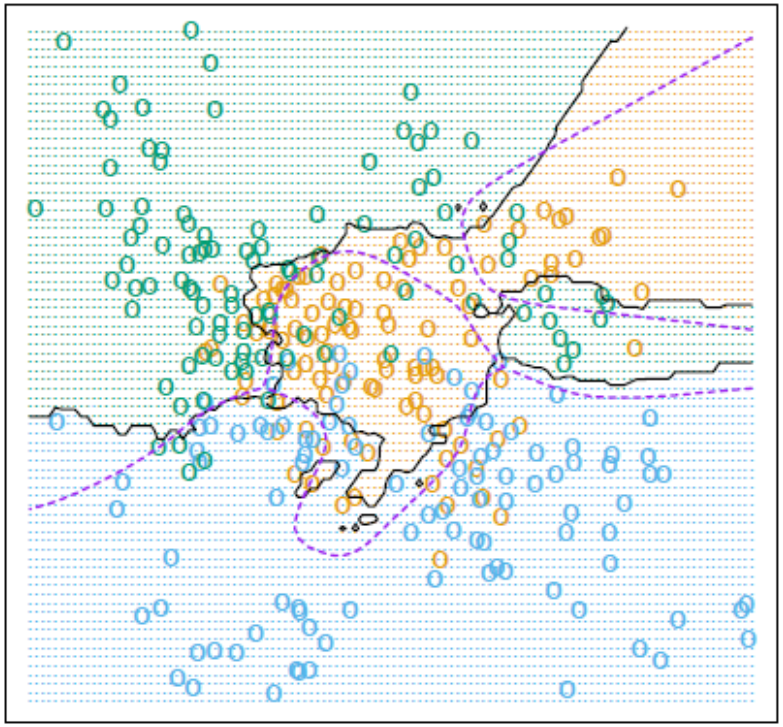
3. KD预测

在KD树搜索最近邻的基础上，我们选择到了第一个最近邻样本，就把它置为已选。在第二轮中，我们忽略置为已选的样本，重新选择最近邻，这样跑K次，就得到了目标的K个最近邻，然后根据多数表决法，预测为K个最近邻里面有最多类别数的类别。

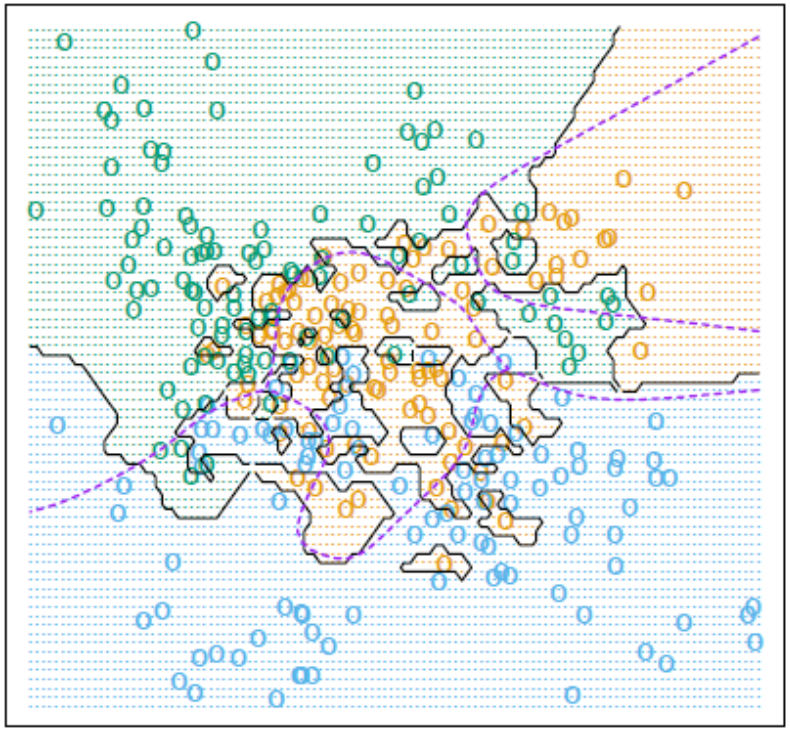
KD树将最近邻搜索的时间复杂度从暴力搜索的O(n)降低至O(log(n))!

五、KNN算法

KNN算法尽管简单，但却取得了不错的效果：



15-近邻

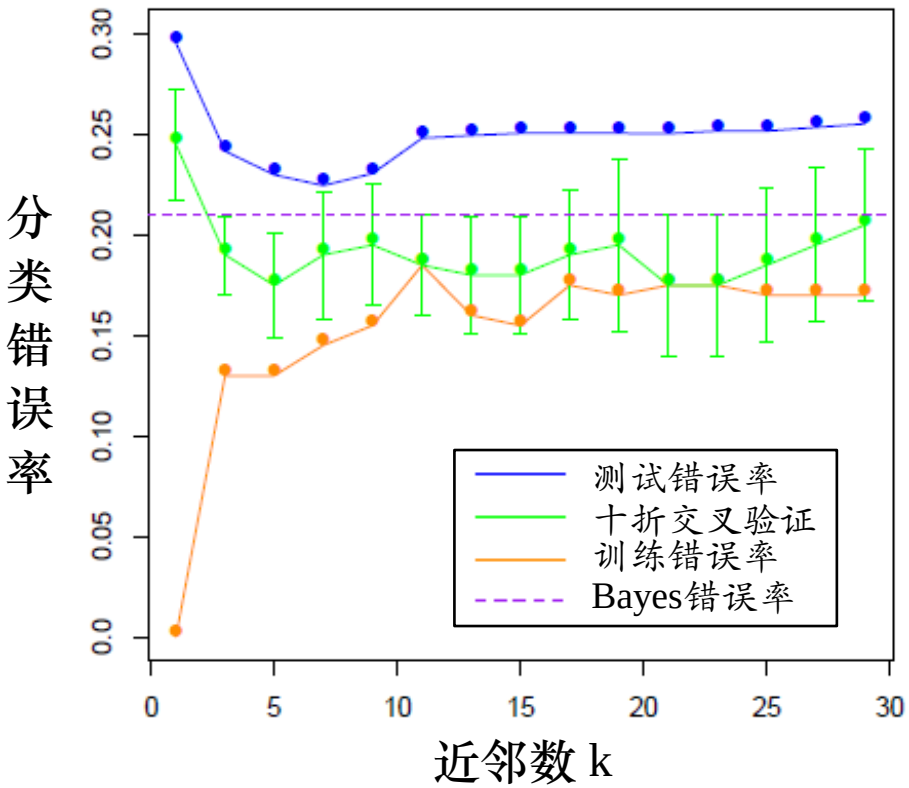


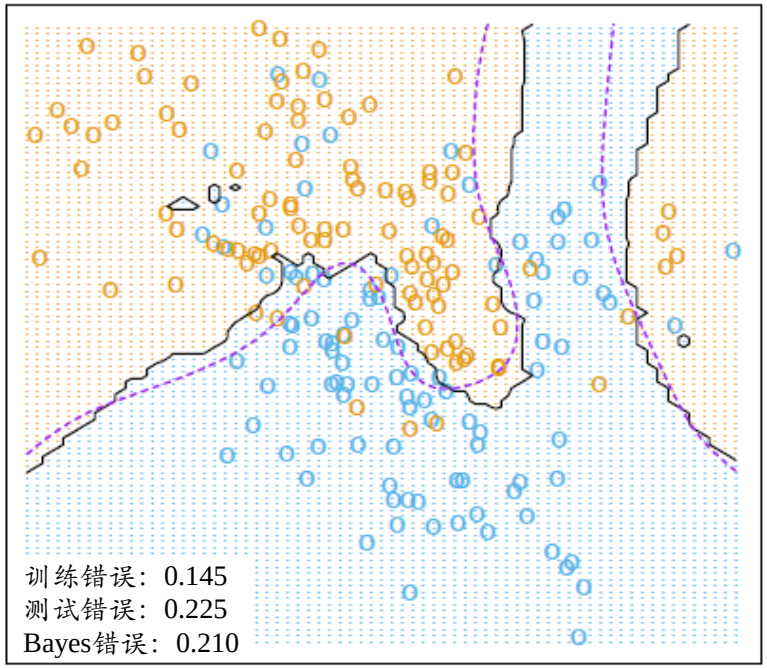
1-近邻

图中黑色实线为KNN方法的决策边界；紫色虚线为Bayes方法的决策边界

分类结果与k取值有关！

五、KNN算法





7-近邻(最优)

五、KNN算法

- 优点：

① 理论成熟，思想简单，既可以用来做分类也可以用来做回归；

② 训练时间复杂度比支持向量机之类的算法低，最差仅为O(n)；

③ 和朴素贝叶斯比，对数据没有假设，准确度高；

- 缺点：

① 计算量大，尤其是特征数非常多的时候；

② 样本不平衡的时候，对稀有类别的预测准确率低；

③ 使用懒散学习方法，基本上不学习，导致预测时速度比起逻辑回归之类的算法慢；而利用KD树加速时可能会导致内存溢出；

➢支持向量机(SVM)

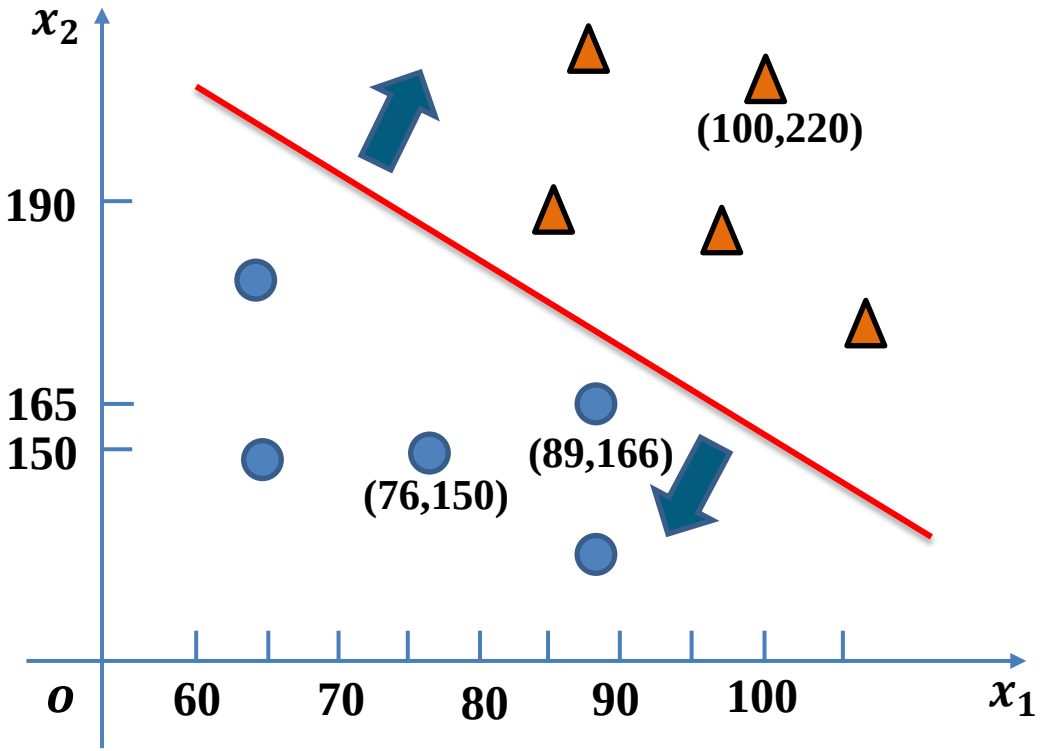
一、支持向量机背景(引例)

进行心脏病检测时，有些检查十分昂贵或具有创伤性，从而希望利用一些有关的容易获得的临床指标进行辅助性推断。假定是否患有心脏病与病人的血压和胆固醇水平密切相关，下表列出了10个病人的临床数据。表中y表示病人所属类别的标号：y=1表示病人有心脏病；y=-1表示病人无心脏病。在这里第一位病人的数据是2维向量：$x_1 = ([x_1]_1, [x_1]_2)^T = (73,150)^T$和标号$y_1 = -1$；……；第十位病人的数据是2维向量 $x_{10} = ([x_{10}]_1, [x_{10}]_2)^T = (110,190)^T$和标号$y_{10} = 1$。

病人编号	血压$[x]_1$	胆固醇水平$[x]_2$	是否心脏病y
1	$[x]_1 = 73$	$[x]_2 = 150$	$y = -1$
2	$[x]_1 = 85$	$[x]_2 = 165$	$y = -1$
⋮	⋮	⋮	⋮
10	$[x]_1 = 110$	$[x]_2 = 190$	$y = 1$

一、支持向量机背景(引例)

现在的问题是，对新来的一位病人，已测得他的血压和胆固醇水平，即已得知他对应的2维向量$x = ([x]_1, [x]_2)^T$，推断他是否有心脏病，即求对应的y为 1还是 -1.



一、支持向量机背景(概述)

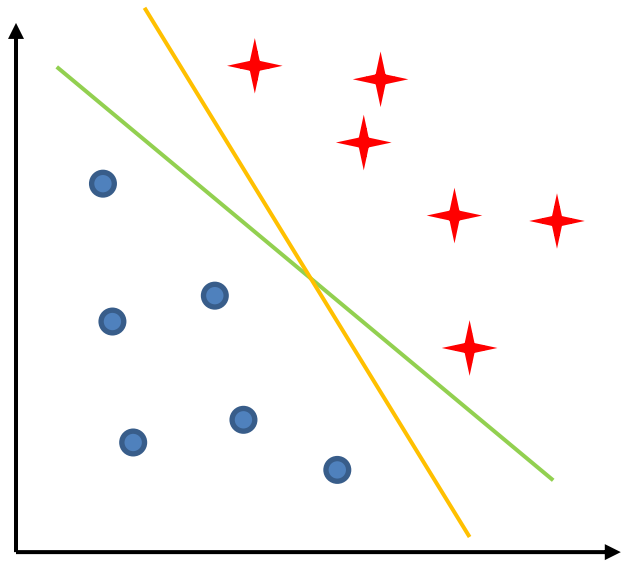
1995年，Vapnik等人基于有限样本的统计学习理论提出了SVM。SVM是一种经典的二分类模型，它的目的是寻找一个分类超平面，使得两类样本能够很好地分开。

SVM的提出有力的反驳了“复杂的理论是没有用的”错误观点；同时，通过结构风险最小化和间隔最大化原则，它克服了传统机器学习方法的局限性。在解决小样本、非线性以及高维模式识别问题中表现出了许多特有的优势。SVM可分为三类：

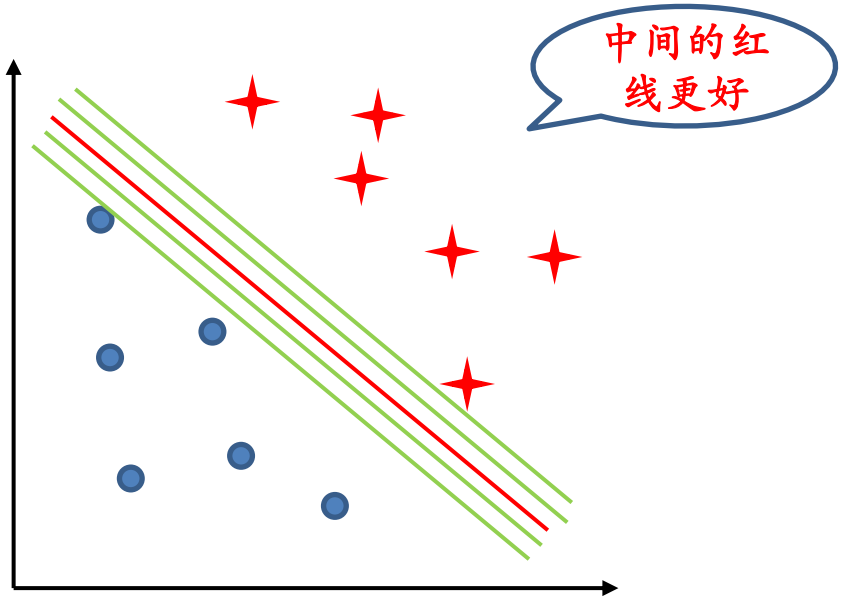
- 当样本线性可分时，通过硬间隔最大化，学习一个线性可分SVM；
- 当样本近似线性可分时，通过软间隔最大化，学习一个线性SVM；
- 当样本线性不可分时，通过核技巧和软间隔最大化，学习一个非线性SVM；

二、线性可分 SVM

给定训练集$D=(x_1,y_1),(x_2,y_2),\cdots,(x_m,y_m)$,其中$y_i\in\{-1,+1\}$，而训练集线性可分，就是指可以找到一个超平面$f(x)=w^Tx+b$，使得两类点分别严格的在该超平面两侧:



w不确定，有无数条



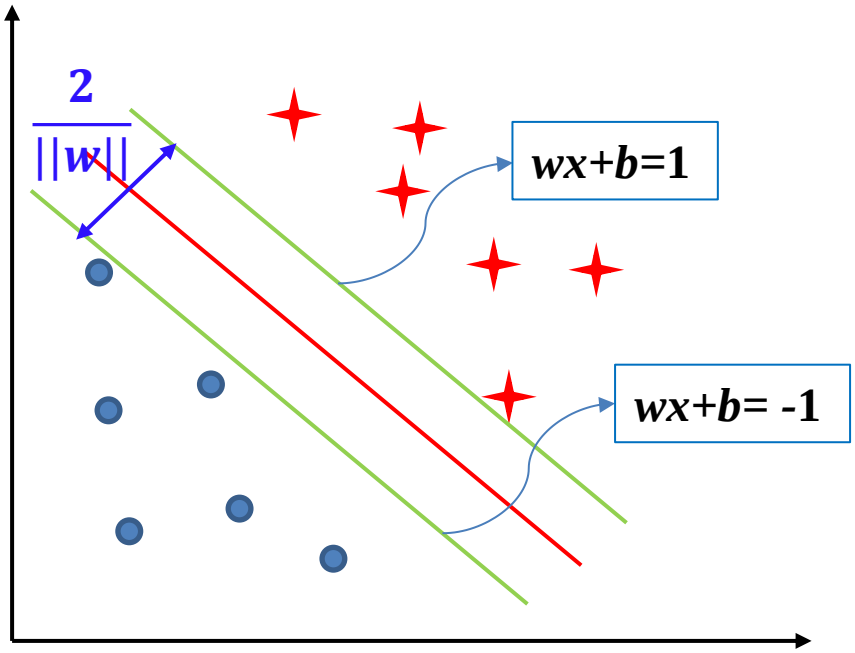
w确定，上下平移有无数条

分类直线不唯一，根据间隔最大化可提升鲁棒性!

二、线性可分 SVM

1. 如何实现间隔最大化?

设法向量为w，则上下平移可产生无数条平行直线，直到碰到某个训练点则停止平移，此时我们可得到在上方和下方碰到的直线l_1和l_2，这两条极端直线为支持直线，其正中间的$w^T x+b=0$ 即为分类超平面。w的获取只需使得两个支持直线之间的“间隔”$\frac{2}{||w||}$尽可能大:



2. 约束条件

约束条件应该使得两类点尽可能正确的分开，也就是对于任意一个训练点，下列不等式满足：

$$\begin{cases} w^T x_i + b \geq +1, & if\ y_i = +1 \\ w^T x_i + b \leq -1, & if\ y_i = -1 \end{cases}$$

这里，$y_i = +1$表示正类样本，$y_i = -1$表示负类样本。约束条件中≥+1和≤-1只是为了计算方便，取任意常数皆可。整合上述两个不等式，可得到下列等价约束：

$$y_i(w^T x_i + b) \geq 1,$$

3. 模型

二次规划

$$\max \quad \frac{2}{||w||} \qquad s.t. \quad y_i\left(w^T x^i + b\right) \geq 1$$

$$\Rightarrow$$

$$\min \quad \frac{1}{2}||w||^2 \qquad s.t. \quad y_i(w^T x_i + b) \geq 1$$

4. 模型求解

为了简化模型求解过程，通过拉格朗日乘子法转化为对偶问题：

$$L(w,b,\alpha)=\frac{1}{2}||w||^2+\sum_{i=1}^{m}\alpha_i(1-y_i(w^Tx_i+b))$$

对w和b求导，并令其等于0可得：

$$\begin{cases}\dfrac{\partial L}{\partial w}=w-\sum_{i=1}^{m}\alpha_i y_i x_i=0\\ \dfrac{\partial L}{\partial b}=\sum_{i=1}^{m}\alpha_i y_i=0\end{cases}$$

代入到拉格朗日函数中可得对偶问题：

$$\max_{\alpha}\ \sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i x_j$$

$$s.t.\ \sum_{i=1}^{m}\alpha_i y_i=0$$

$$\alpha_i\geq 0,\qquad i=1,\dots,m$$

4. 模型求解

过程中的KKT条件是“凸问题的解是唯一最优解”的保证：

KKT条件：

$$w - \sum_{i=1}^{m} \alpha_i y_i x_i = 0$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0$$

→ 1. 求导所得

$$\alpha_i (1 - y_i (w^T x_i + b)) = 0$$

→ 2. 互补松弛条件

$$y_i (w^T x_i + b) \geq 1$$

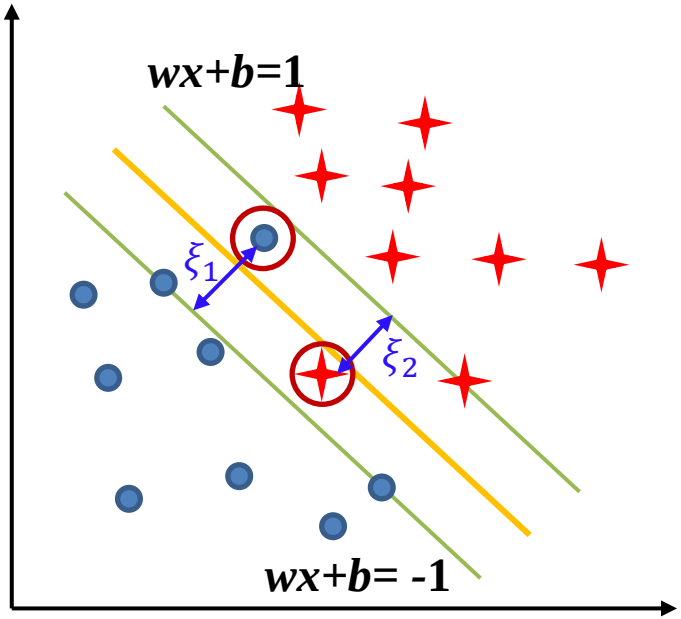
$$\alpha_i \geq 0$$

→ 3. 约束条件

三、近似线性可分 SVM

1. 原问题

近似线性可分意味着，在忽略少数点的前提下，仍然可找到直线来划分两类点，无需选用曲线来增加模型复杂度，如下图所示，只需对两个不满足 $y_i(w^T x_i + b) \geq 1$的点引入松弛变量(代价) $\xi_i \geq 0$，使得间隔加上松弛变量大于等于1。并且所有点的代价和应该尽可能小：



惩罚参数，权衡　　松弛变量

$$\min \quad \frac{1}{2}||w||^2 + C\sum_i \xi_i$$

$$s.t. \quad y_i(w^T x_i + b) \geq 1 - \xi_i,$$

$$\xi_i \geq 0, \ i = 1,2,\dots,m$$

2. 对偶问题

引入拉格朗日乘子α_i , μ_i ，构建拉格朗日函数：

$$L(w,b,\alpha,\mu)=\frac{1}{2}||w||^2+C\sum_i\xi_i+\sum_{i=1}^{m}\alpha_i\left(1-\xi_i-y_i(w^Tx_i+b)\right)-\sum_{i=1}^{m}\mu_i\xi_i$$

对w和b求导，并令其等于0可得：

$$\begin{cases}\dfrac{\partial L}{\partial w}=w-\sum_{i=1}^{m}\alpha_i y_i x_i=0\\ \dfrac{\partial L}{\partial b}=\sum_{i=1}^{m}\alpha_i y_i=0\\ C=\alpha_i+\mu_i\end{cases}$$

代入到拉格朗日函数中,并进行简化，可得对偶问题：

$$\max_{\alpha}\ \sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j x_i x_j$$

$$s.t.\ \sum_{i=1}^{m}\alpha_i y_i=0$$

$$0\le\alpha_i\le C,\qquad i=1,\dots,m$$

$\alpha_i=0$ 为非支持向量

$\alpha_i>0$ 为支持向量

3. KKT条件与样本点分布

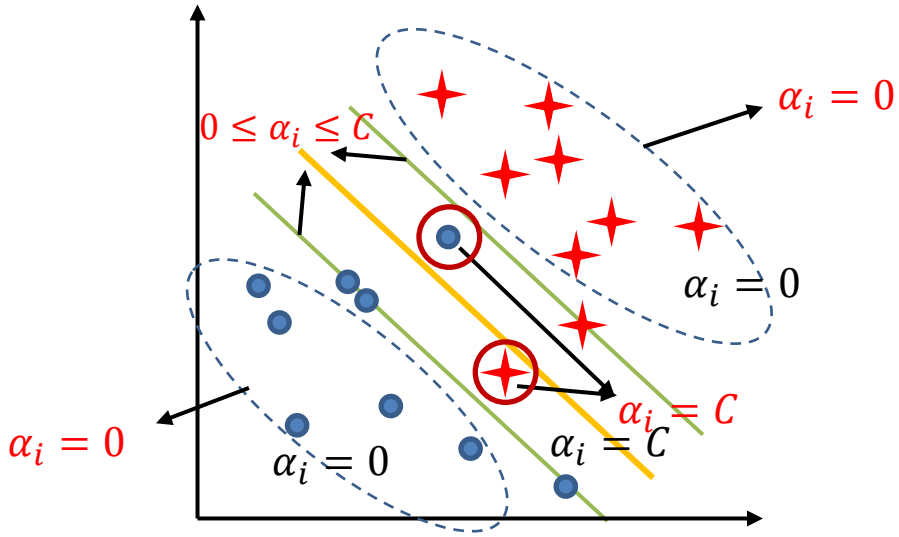
通过KKT条件可将所有样本点划分为三个区域：

①KKT：

$$\begin{cases} w-\sum_{i=1}^{m}\alpha_i y_i x_i=0\,;\ \sum_{i=1}^{m}\alpha_i y_i=0 \\ \alpha_i\left(1-\xi_i-y_i(w^T x_i+b)\right);\quad \mu_i\xi_i=0 \\ y_i(w^T x_i+b)\geq 1-\xi_i\,;\ \alpha_i+\mu_i=C;\ \ \alpha_i,\mu_i,\xi_i\ \geq\ 0 \end{cases}$$

②区域：

$$\begin{cases} R\rightarrow\{i: y_i(w^T x_i+b)>1\} \\ E\rightarrow\{i: y_i(w^T x_i+b)=1\} \\ L\rightarrow\{i: y_i(w^T x_i+b)<1\} \end{cases}$$

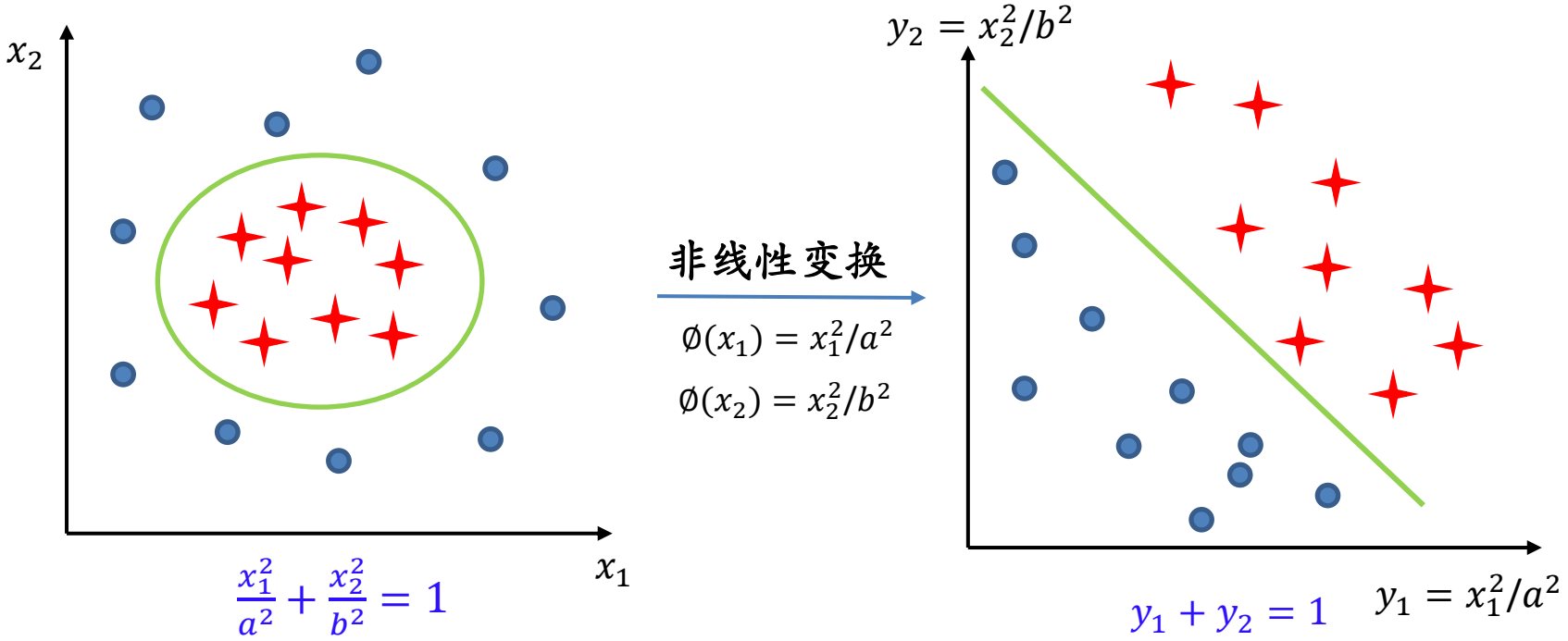


SVM 有非常强的样本稀疏性，可在求解前删除无用的非支持向量！

四、非线性 SVM

1. 原问题

对于非线性可分问题，用直线并不能将两类点有效分开(如图)，而椭圆可很好的将其分开。但是非线性问题(求椭圆)较为困难，因此希望引入非线性变换，将非线性曲线转换为高维空间中的线性问题，从而容易求解。



1. 原问题

也就是说，可将训练样本从原始空间映射到另外一个空间(往往维数会升高)，使得样本在这个新空间中线性可分，令$\emptyset(x)$表示将x映射后的特征向量，于是在新的空间中分类超平面所对应的形式为：

$$f(x) = w^T \emptyset(x) + b$$

非线性SVM的原问题对应模型为：

$$\begin{aligned} \min \quad & \frac{1}{2}||w||^2 + C\sum_i \xi_i \\ s.t. \quad & y_i(w^T\emptyset(x_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \ i = 1,2,\dots,m \end{aligned}$$

2. 对偶问题及核函数

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \emptyset(x_i)^T \emptyset(x_j)$$

$$s.t. \sum_{i=1}^{m} \alpha_i y_i = 0$$

$$0 \le \alpha_i \le C, \qquad i = 1, \dots, m$$

此时，一方面不容易确定$\emptyset(\cdot)$的形式，另一方面$\emptyset(\cdot)$的维数往往比较高甚至是无穷维，计算$\emptyset(x_i)^T \emptyset(x_j)$较为困难，因此需引入核函数：

$$k(x_i, x_j) =< \emptyset(x_i), \emptyset(x_j) = \emptyset(x_i)^T \emptyset(x_j)$$

常用的核函数有：

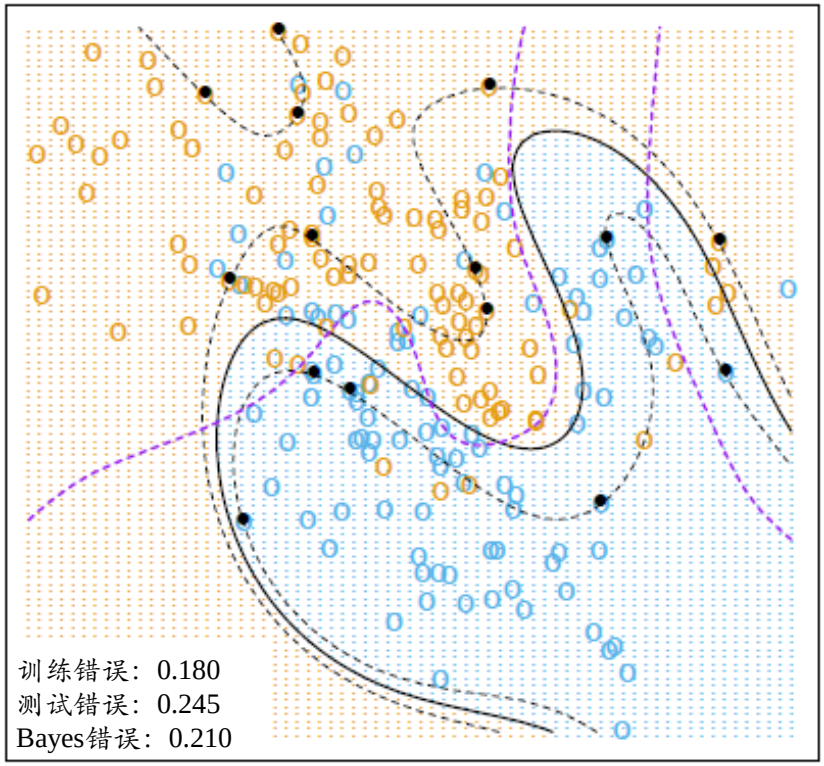
线性核：$k(x_i, x_j) = x_i^T x_j$

多项式核：$k(x_i, x_j) = (x_i^T x_j)^d$

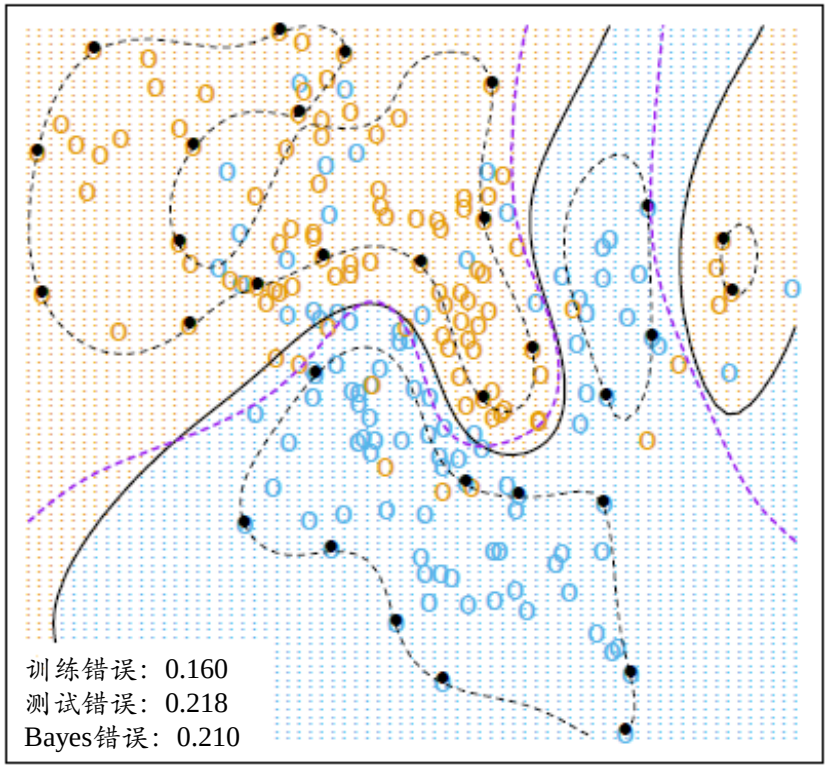
高斯核：$k(x_i, x_j) = \exp(-\frac{||x_i - x_j||^2}{2\sigma^2})$

样本成对出现时使用核函数

五、实验结果



4次多项式核函数



径向基核函数

六、小结

1.优点

- 对于线性不可分的情况可以通过核函数，映射到高维特征空间实现线性可分；
- SVM学习问题可以表示为凸优化问题，因此可以利用已知的有效算法发现目标函数的全局最小值。而其他分类方法 (如基于规则的分类器和人工神经网络)一般只能获得局部最优解；
- 对小样本分类效果很好。

2.缺点

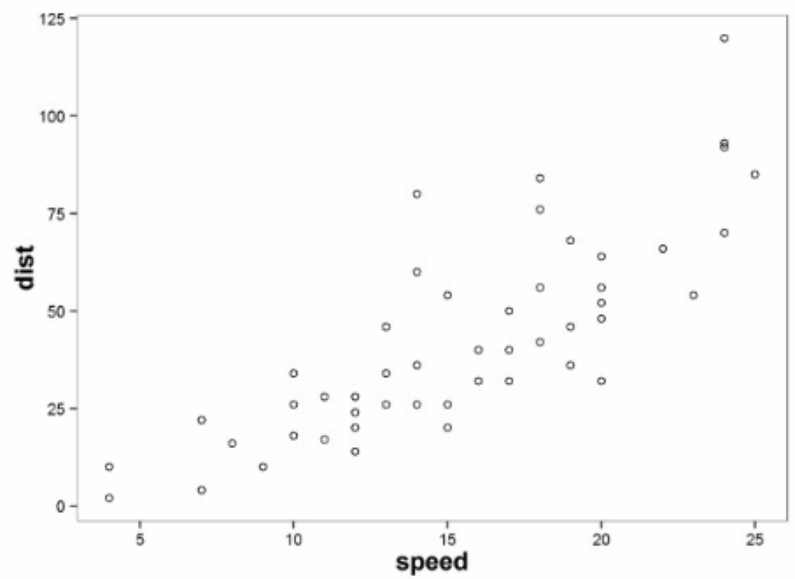
- SVM仅仅只限于一个二类分类问题，将其直接应用于多分类问题效果并不好；
- 仅局限于小集群样本，对于观测样本太多时，效率较低；
- 寻求合适的核函数相对困难。

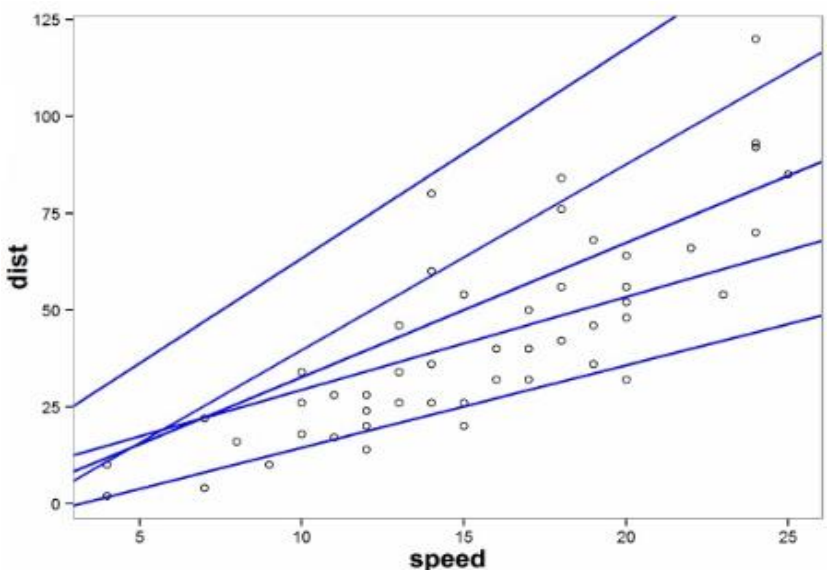
➢逻辑回归(LR)

一、逻辑回归研究背景

- 实际工作中我们经常需要预测某些事物是否发生。
  ① 通过财务信息预测公司是否破产；
  ② 通过驾驶记录预测驾驶员是否会出事故；
  ③ 通过购物和还款记录预测信用卡持卡人是否诚信；
- 线性回归分析预测的被解释变量Y是连续的数值型，即：

$$y = \theta^T x = \theta_{0+}\theta_1 x_1 + \cdots + \theta_n x_n$$



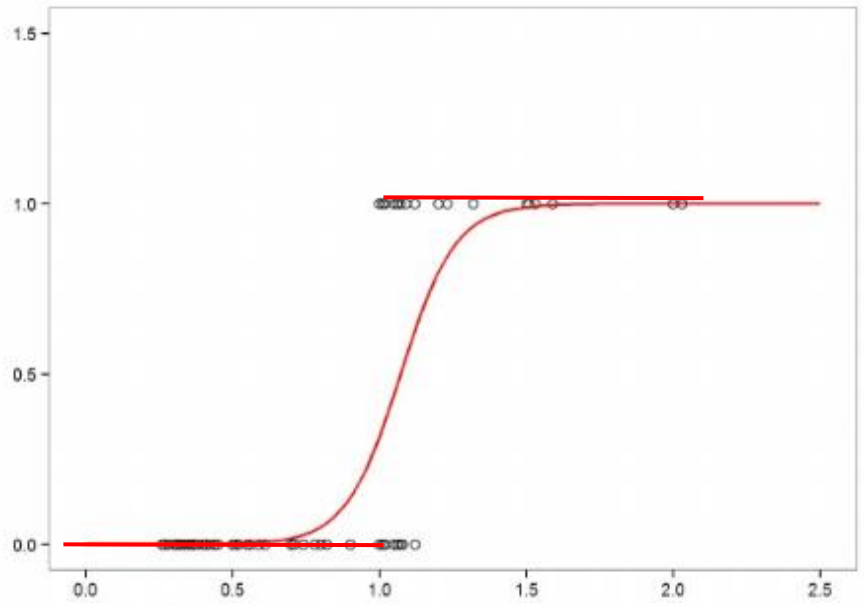


如何将回归分析用于处理二分类问题？

一、逻辑回归研究背景

$y \in \{0,1\}$

$y \sim X$?



$\theta^T x$ → 只包含0,1的离散值（跃阶函数） → [0,1] 区间内概率值

- 逻辑回归(Logistic Regression, LR)模型，主要是基于线性回归，对受多因素影响的事件进行概率预测，根据预测结果进行**分类**。

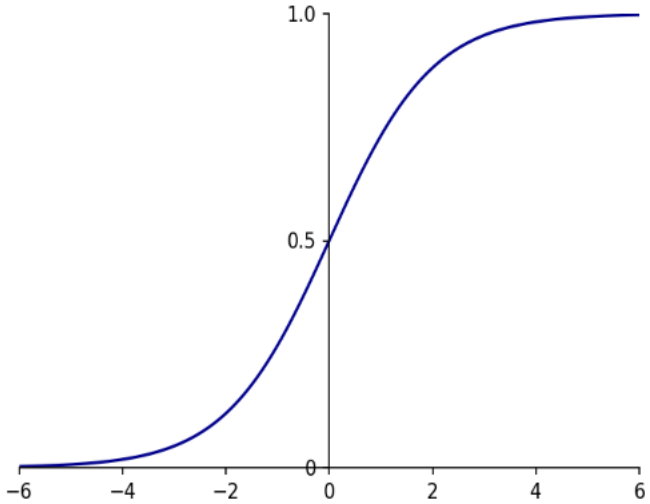
二、逻辑回归模型

因此，逻辑回归是将线性回归模型$\theta^T x$转化成一个取值在(0,1)之间的概率函数的过程，这个转化通过Sigmoid函数来完成：

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

该函数有非常好的性质，当z趋于无穷时，g(z)趋于1；当z趋于负无穷时，g(z)趋于0，这非常适合于分类模型，且它具有很好的导数性质：

$$g'(z) = g(z)(1 - g(z))$$



二、逻辑回归模型

当令g(z)中的z等于$\theta^T x$时，就得到了逻辑回归模型的一般形式：

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

其中x为样本，$h_\theta(x)$为输出，可理解为被分为某一类的概率大小，θ为模型参数。可以发现，若$h_\theta(x)$>0.5，即$\theta^T x$>0，则y=1。如果$h_\theta(x)$<0.5，即$\theta^T x$<0，则y=0。当 $h_\theta(x)$>0.5无法确定分类。即：

$$P(y = 1|x, \theta) = h_\theta(x)$$

$$P(y = 0|x, \theta) = 1 - h_\theta(x)$$

合并两式可得：

$$P(y|x, \theta) = h_\theta(x)^y (1 - h_\theta(x))^{1-y}$$

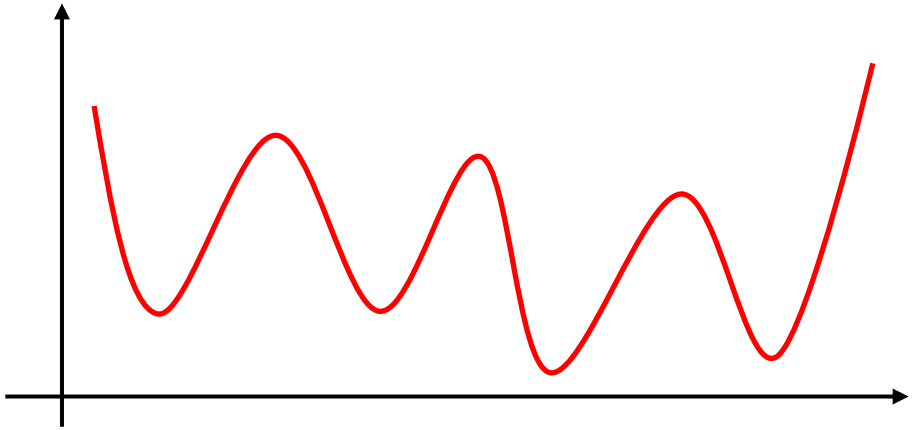
$h_\theta(x)$值越小，分类为0的概率越高；反之，值越大分类为1的概率越高。若靠近临界点，则分类准确率会下降。因而接下来我们考虑LR的损失函数是什么。

三、逻辑回归损失函数

由于LR可视为广义线性模型，而线性模型最常用的损失函数为误差平方和函数，因此我们首先考虑能够对LR应用误差平方和函数：

$$J(\theta)=\frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}(h_\theta(x_i)-y_i)^2$$

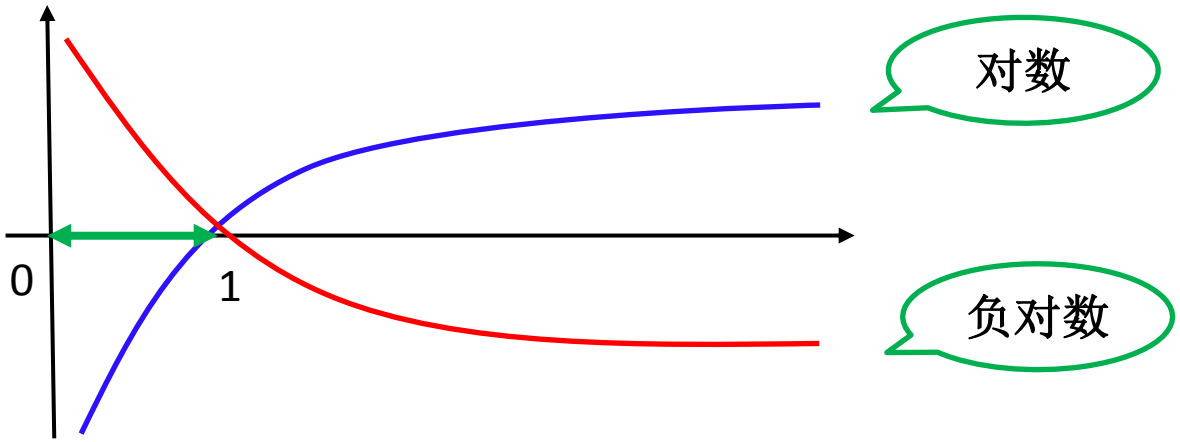
将$h_\theta(x_i)=\frac{1}{1+e^{-\theta^T x_i}}$代入上式时，由于Sigmoid函数是一个复杂的非线性函数，这就导致我们最后得到的$J(\theta)$是非凸函数：



该函数有多个局部极小值，从而使得梯度下降法等求解算法时，得到的结果并非全局最小，因此不能直接采用误差平方和函数。

三、逻辑回归损失函数

接下来我们要为LR找到一个凸的损失函数，常用的为对数损失函数：



该图像在[0,1]区间有很好的性质，如红色部分，当z=1时，函数值为0；当z=0时，函数值为无穷大。这就可以和损失函数联系起来：

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

当实际标签和预测结果相同时，损失为0；当预测相反时，损失为无穷大。为了简化计算，将损失函数整合为下式：

$$Cost(h_\theta(x), y) = -y\log(h_\theta(x)) - (1 - y)\log\left(1 - h_\theta(x)\right)$$

四、逻辑回归求解

LR最终的模型即为下列无约束优化问题：

$$J(\theta)=\frac{1}{m}\sum_{i=1}^{m}Cost(h_\theta(x_i),y_i)=-\frac{1}{m}\sum_{i=1}^{m}y_i\log(h_\theta(x_i))+(1-y_i)\log\big(1-h_\theta(x_i)\big)$$

利用Sigmoid函数的导数性质$g'(z)=g(z)(1-g(z))$，采用梯度下降法：

$$\begin{aligned}\frac{\partial J(\theta)}{\partial\theta_j}&=-\frac{1}{m}\sum_{i=1}^{m}(y_i\cdot\frac{1}{h_\theta(x_i)}-(1-y_i)\cdot\frac{1}{1-h_\theta(x_i)})\cdot\frac{\partial h_\theta(x_i)}{\partial\theta_j}\\&=-\frac{1}{m}\sum_{i=1}^{m}(y_i\cdot\frac{1}{h_\theta(x_i)}-(1-y_i)\cdot\frac{1}{1-h_\theta(x_i)})\cdot\frac{\partial\theta^Tx}{\partial\theta_j}\\&=-\frac{1}{m}\sum_{i=1}^{m}(y_i\cdot(1-h_\theta(x_i))-(1-y_i)\cdot h_\theta(x_i))\cdot x_j\\&=\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x_i)-y_i)\cdot x_j\end{aligned}$$

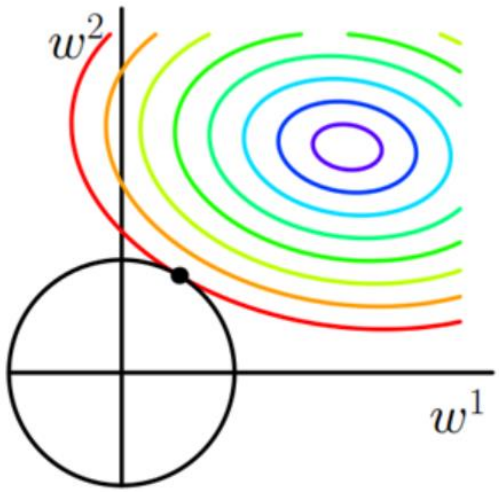
所以参数更新公式为：$\theta_j=\theta_j-\alpha\sum_{i=1}^{m}(h_\theta(x_i)-y_i)\cdot x_j$

五、逻辑回归正则化项

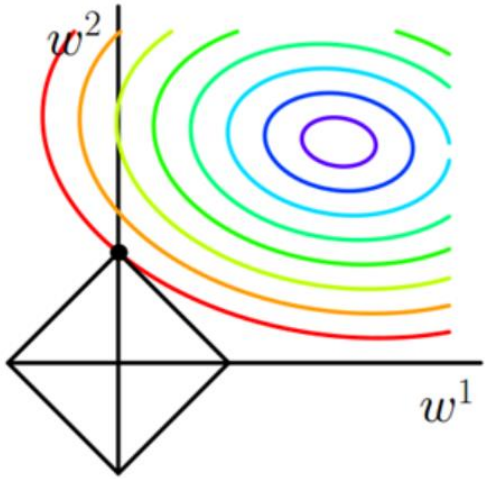
目标函数如下：

$$J_1(\theta)=-\frac{1}{m}\sum_{i=1}^{m} y_i \log(h_\theta(x_i)) + (1-y_i)\log(1-h_\theta(x_i))+\frac{\lambda}{2}\|\theta\|_2^2$$

$$J_2(\theta)=-\frac{1}{m}\sum_{i=1}^{m} y_i \log(h_\theta(x_i)) + (1-y_i)\log(1-h_\theta(x_i))+\frac{\lambda}{2}\|\theta\|_1$$



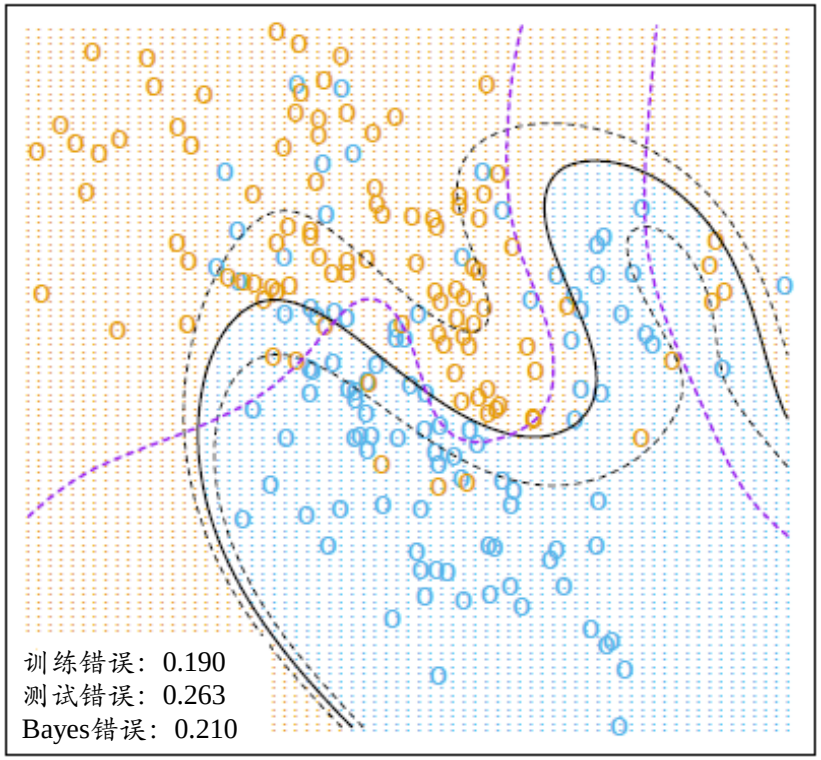
L_2正则化项



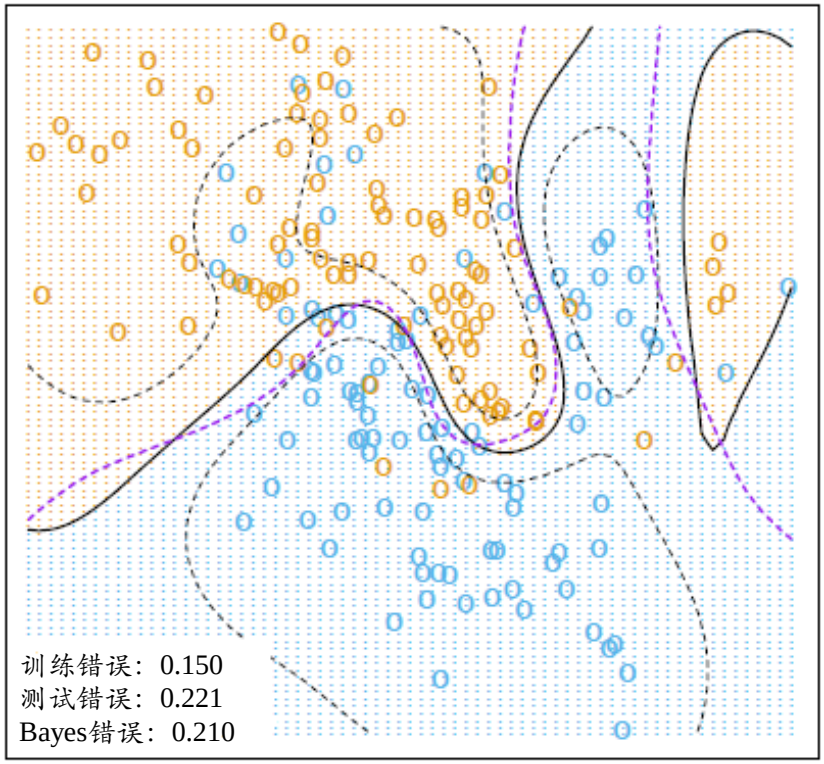
L_1正则化项

LR可通过与SVM相似的方式加核函数提升性能

六、逻辑回归实验结果



4次多项式核函数



径向基核函数

七、逻辑回归优缺点比较

优点：

- 可直接对分类的可能性进行建模，无需事先假设数据分布；
- 不仅预测出类别，还可得到近似概率预测；
- 对数函数是任意阶可导的凸函数，有很好的数学性质，现有的许多数值优化方法都可直接用于求取最优解；计算代价不高，容易理解实现。

缺点：

- 容易欠拟合，分类精度不高；
- 数据特征有缺失或者特征空间很大时表现效果并不好；
- 很难处理数据不平衡的问题。举例：如果正负样本比 10000:1.我们把所有样本都预测为正也能使损失函数的值比较小。但这样显然不合理。

中国农业大学
China Agricultural University

谢　谢！