# 统计学习方法》笔记

目录

[《统计学习方法》笔记 1](#_Toc515874205)

[**一、** **统计学习方法概论。** 1](#_Toc515874206)

[1.1. 统计学习三要素： 1](#_Toc515874207)

[1.2. 模型估计与模型选择 2](#_Toc515874208)

[1.3. 正则化和交叉验证。 2](#_Toc515874209)

[1.4. 泛化误差。 3](#_Toc515874210)

[1.5. 生成模型和判别模型。 3](#_Toc515874211)

[1.6. 分类问题。 3](#_Toc515874212)

[1.7. 标注问题。 4](#_Toc515874213)

[1.8. 回归问题。 4](#_Toc515874214)

[**二、** **感知机** 4](#_Toc515874215)

[2.1. 感知机模型。 4](#_Toc515874216)

[2.2. 感知机学习策略。 4](#_Toc515874217)

[2.3. 感知机学习算法。 5](#_Toc515874218)

[**三、** **K邻近算法。** 7](#_Toc515874219)

[**四、** **朴素贝叶斯法。** 7](#_Toc515874220)

[4.1. 朴素贝叶斯法的学习与分类。 7](#_Toc515874221)

[4.2. 朴素贝叶斯法的参数估计。 8](#_Toc515874222)

[**五、** **决策树。** 9](#_Toc515874223)

[5.1. 特征选择。 9](#_Toc515874224)

[5.2. 决策树的生成。 10](#_Toc515874225)

[5.3. 决策树的剪枝。 11](#_Toc515874226)

[5.4. CART算法。 12](#_Toc515874227)

[**六、** **逻辑斯蒂回归和最大熵模型。（LR回归）** 13](#_Toc515874228)

[6.1. 逻辑斯蒂回归模型。 13](#_Toc515874229)

[6.2. 最大熵模型。 14](#_Toc515874230)

[**七、** **支持向量机。** 14](#_Toc515874231)

[7.1. 线性可分SVM和硬间隔最大化。 14](#_Toc515874232)

[7.2. 线性支持向量机与软间隔最大化。 19](#_Toc515874233)

[7.3.非线性支持向量机和核函数。 19](#_Toc515874234)

[7.4.序列最小化算法（SMO） 21](#_Toc515874235)

[**八、** **提升方法（boosting）** 21](#_Toc515874236)

[8.1. 提升方法AdaBoost算法。 21](#_Toc515874237)

[8.2-8.3. AdaBoost算法的训练误差分析和算法解释。 22](#_Toc515874238)

[8.4. 提升树。 22](#_Toc515874239)

为了加深学习印象写这么一个笔记，基本是按照自己的思路来写，未见得有多系统，作为一个学习记录而已，本来是想要在简书上写的，后来发现编辑公式编程最头疼的问题。这份笔记可能有的地方写的比较系统和细致：一般是重点且自己没掌握的地方，有的地方可能写的比较粗略，可能是自己已经掌握的地方。

参考的书是李航的《统计学习方法》，这本书当时复习机器学习考试的时候草草看过一遍。

1. **统计学习方法概论。**

统计机器学习是现在热门的机器学习的基础，有以下几个特点：

1. 以计算机网络为平台，建立在计算机网络之上。
2. 以数据为研究对象，数据驱动。
3. 以方法为中心，构建模型进行预测及分析
4. 概率论，统计学，信息论，最优化方法及计算机科学等多领域的交叉学科。
   1. 统计学习三要素：

**方法=模型+策略+算法**

1. 模型。

模型就是所要学习的条件概率分布或者决策函数。

1. 策略

策略表示模型一般是按照什么样的准则学习或者是选择最优模型，一般使用的是两种策略。

损失函数和风险函数假设选择模型作为决策函数，对于给定的输入，可以由来得到输出，这个与真实的之间可能一致也可能不一致，这时候用一个损失函数（loss function）或者代价函数（cost function）来度量预测错误的程度，损失函数是和的实值函数，记作。常用的损失函数有。

* 0-1损失函数。



* 平方损失函数。



* 绝对损失函数。



* 对数损失函数。



损失函数越小，就说明模型越好，这是显而易见的。

由于输入输出（X.Y）是随机变量，遵循联合分布P(X,Y) ,所以损失函数的期望是：



但是实际上联合分布我们是不知道的，所以损失函数期望是没法去计算的，如果知道联合分布也就不用学习了。所以这样一来，期望风险最小学习模型要用到联合分布，而联合分布又是未知的，所以这样就变成一个病态问题了。

对于训练数据集：



我们把模型关于数据集的平均损失称为经验风险或者经验损失。



根据大数定律，当样本数量趋于无穷时，经验风险就趋于期望风险，所以一个很自然的想法就是用经验风险来估计期望风险，但是实际上数据样本可能常常不够用，所以单纯用经验风险来估计效果并不理想，所以要对其进行一些校正，这就关系到监督学习中常用的两个基本策略：经验风险最小化和结构风险最小化。

当数据量充足时，我们可以最小化经验风险来估计模型：



但是当样本比较少的时候，经验风险最小化的学习效果就未必小，这是因为会产生过拟合（overfitting）现象。

为了防止经验风险最小化带来的过拟合问题，通常会采用结构风险最小化（structural risk minimization）的策略，等价于正则化，给经验风险加上表示模型复杂度的正则化项或者是惩罚项。



所以，监督学习就转化为这样一个优化问题：



1. 算法

算法是指模型的具体计算方法，统计学习通常转化为一个最优化问题，这个最优化问题的解析解通常不存在，这就需要用数值计算的方法来求解，所以选择一个高效的算法就非常有用了。

* 1. 模型估计与模型选择

再说两个概念：训练误差与测试误差。

都比较好理解，训练误差就是模型关于训练数据集的平均损失，而测试误差就是模型关于测试集的平均损失，测试误差小的模型具有更好的预测能力，这种预测能力被称为泛化能力（generalization）。

过拟合和模型选择问题：

如果一味追求模型对于训练数据的预测能力，可能会导致模型的复杂度往往会比“真模型”的复杂度更高，这种现象称为过拟合，学习的模型包含的参数过多，可能导致对已知数据预测得很好，但是对未知数据预测的很差，模型选择旨在避免过拟合而提高模型的预测能力。

一个通俗的例子，比如我们现在有一个由二次函数产生的一个数据集，数据量非常有限，如果不限制模型的复杂度，可能为了降低预测误差，模型可能学习到3次或者4次以上的函数，这样虽然对现有的数据集有比较好的预测能力，但是对于未知的数据的预测结果可能就比较差。

* 1. 正则化和交叉验证。

模型选择的一个典型方法是正则化，正则化是结构风险最小化策略的实现。正则化一般式模型复杂度的单调函数，模型复杂度越高，正则化值就越大，比如，正则化可以是模型参数的范数（这很常用）。L1范数和L2范数都是经常采用的。

还有一个常用的方法是交叉验证：

如果样本量比较充足，可以随机的将数据分为训练集，验证集，和测试集。如果不是很充足的情况，一般只有训练集和测试集两部分。这种情况下可以采用交叉验证的方式来选择更好的模型，交叉验证的基本思想就是重复使用数据：把给定的数据进行切分，分成训练集和测试集，在此基础上进行反复训练，测试及模型选择。

* 简单交叉验证。

将数据随机分成两部分：训练集和测试集，然后在各种条件下（模型参数个数等）训练模型，最终选择测试误差最小的模型。KNN算法就常采用此种方法。

* S折交叉验证。

随机将已给的数据分成S个互不相交大小相同的子集，然后利用S-1个来进行训练，用余下的子集来测试，将这一过程在可能的S种选择中重复进行，然后选择S种测试误差最小的模型。

* 留一交叉验证。

S折交叉验证中当S=N时，就是只留下一个样本来做验证。往往在数据极度缺乏时才会采用这种方法。

* 1. 泛化误差。

泛化误差就是指的是学习到的模型对于未知数据的预测能力，一般采用的是测试误差对其进行评价，如果从理论上来说，泛化误差实际上就是学习到的模型的期望风险。



还有泛化误差上界的相关理论分析，不做重点关注。

* 1. 生成模型和判别模型。

监督学习的任务就是学习一个模型，利用这个模型，对于给定的输入预测响应的输出，监督学习方法又可以分为生成法和判别法。这种模型可以表示为决策函数：或者是一个条件概率分布。

生成方法由数据学习到联合概率分布P(X,Y),然后求出条件概率分布来作为预测模型：



这种模型被称作为是生成模型，因为其给出了由输入产生输出的生成关系，典型的有：朴素贝叶斯法和隐马尔科夫模型。

判别法是有数据直接学习到决策函数或者条件概率分布，而不关心其概率分布，典型的方法包括：最近邻，k近邻，感知机，决策树，逻辑斯蒂回归，最大熵模型，支持向量机，提升方法和条件随机场等。

生成方法特点：可以还原联合概率分布，收敛速度更快，当存在隐含变量时，仍然可以使用生成方法，判别方法就不行。（不是很理解）

判别方法：直接面对预测，往往准确率较高，可以对数据进行各种程度的抽象，定义特征并使用特征，因此可以简化学习问题。

下面再介绍监督学习中常见的几个问题。

* 1. 分类问题。

输出有限离散，输入可以离散也可以连续，这时候学习到的模型实际上就是一个分类问题。根据分类多少又可以分为二分类问题和多分类问题。（重点关注二分类：以关注的类为正类，其他类为负类）。

几个常见的概念：

TP—将正类预测为正类数。

FN—将正类预测为负类数。

FP—将负类预测为正类数。

TN—将负类预测为负类数。

精确率定义为：（分成正类中正确的百分比）。



召回率：（正类中被正确分类的百分比）。



还有一个F1值，是精确率和召回率的调和均值：



* 1. 标注问题。

标注问题也是一个监督学习问题，可以认为标注问题是分类问题的一个推广，标注问题又是更复杂的结构预测的简单形式，其输入是一个观测序列，输出是一个标记序列或者状态序列。可能标记状态是有限的，但是其组合而成的标记序列的个数是成指数增长的。常用的模型是马尔科夫模型和条件随机场。

* 1. 回归问题。

归回问题关注的是输入变量和输出变量之间的关系，特别是输入变量的值发生变化时，输出值是怎么变化的（输出的值可能在训练样本中就没有出现过）。回归问题的学习等价为函数拟合，最小二乘法就是一个典型的例子。

1. **感知机**

感知机是二分类的线性分类模型，输入为实例的特征向量，输出为实例的类别，取+1，-1二值。感知机对应于输入空间（特征空间）将两类分离的超平面，为此，导入基于误分类的损失函数，利用梯度下降法来对损失函数进行极小化，求得感知机模型。感知机学习算法简单易用，分为原始形式和对偶形式，1957年由Rosenblatt 提出，是神经网络和支持向量机的基础。

2.1. 感知机模型。

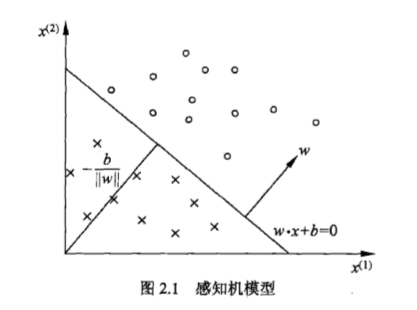
输入空间为，输出空间是Y={+1,-1}，输入是实例的特征向量，是输出的实例类别，输入空间到输出空间有如下函数：



其中：是权值向量，b称作偏置(bias)



感知机是一种线性分类模型，属于判别模型，感知机相当于一个超平面，可以把两类向量分开：

2.2. 感知机学习策略。

首先提出数据集可分的概念，对于一个数据集：



如果存在超平面：使得所有的正实例都有：，所有的负实例都有：，我们就说这个数据集是线性可分的，否则就是线性不可分的。

假设数据集是线性可分的，那么感知机的任务就是学习到这样一个超平面。如何找到这样一个超平面需要制定一个学习策略，即定义经验损失函数并将其最小化。

一个容易想到的就是错误分类的个数，但是这样的损失函数不是其参数w, b的可导函数，不易优化。另外一个可选择的是误分类点到超平面的总距离，这就是感知机所采用的，为此，首先写出输入任意一点到超平面的距离。



其次对于误分类的数据来说：是成立的，所以误分类点到超平面的距离则为：



这样的话，假设所有误分类点的集合是M，那么所有误分类点到超平面的总距离则是：



不考虑权重向量（一个正的比例因子），就得到了感知机学习的损失函数：



显然，这个损失函数是非负的，如果没有误分类点，那么这个损失函数值就是0，而且误分类点越少，误分类点离超平面就越近，损失函数值就越少，而且还是参数w，b的可导函数。

2.3. 感知机学习算法。

上节给出了感知机的学习策略，这节来看其学习算法。

感知机学习算法的原始形式：

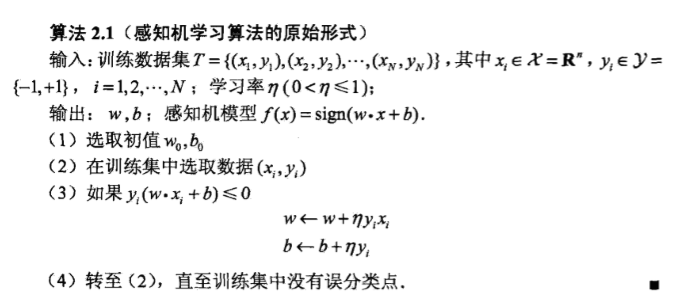
分别求其对于参数的梯度：



随机选择一个误分类点，对权值和偏置进行更新：



其中是学习步长，又称为学习率，我们是在梯度的负方向上进行学习，因为梯度表示的是增长最快的方向，这样的话通过迭代可以使得损失函数不断减小，直到为0。

所以我们可以得到这样的算法：

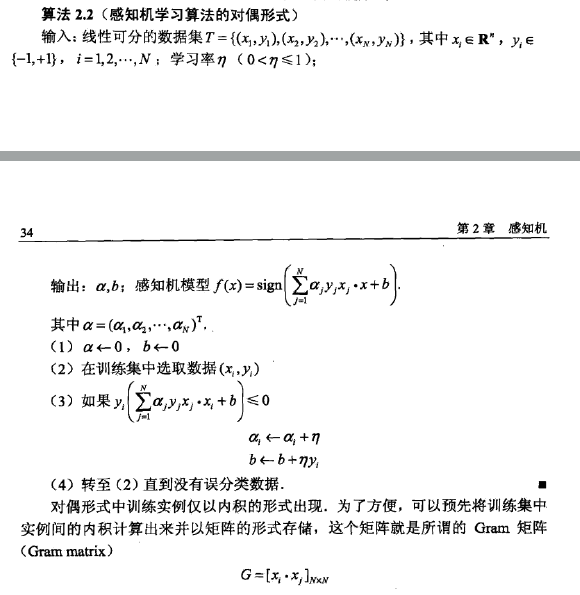
可以证明此算法是收敛的，参见原书P31.

下面看下感知机算法的对偶形式：

感知机的原始形式和对偶形式和支持向量机的原始形式和对偶形式是相对应的。

对偶形式的基本想法是，将表示为实例和标记的线性组合的形式，通过求解其系数从而求得，不失一般性。从算法2.1可以看出，是通过梯度负方向逐步来修改使其达到条件，学习过程中不难看出：逐步修改 ，设修改了n次，那么关于的增量就分别是和，其中，则最终学习到的可以分别被表示为：



这里，i=1,2,3…N, 当学习率时，就表示第i个实例点由于误分而进行更新的次数，实例点更新的次数越多，就意味着其距离超平面越近，也就越难正确分类，换句话说，这样的实例对学习结果的影响最大。

对偶形式的思路是最重要的，主要就是上面红字所述。然后响应修改了误判的判断条件，这样做的一个好处是训练实例仅以内积的形式出现，为了方便，可以在一开始就计算出表示内积的Gram矩阵，从而提高计算效率，原始形式中对于每一个实例都需要计算误判条件是否满足，这样实际上是包含了许多重复运算的。

从另一个角度来看，本来对于w的更新，每一次都要加上具体的数，而采用对偶形式之后，实际上只是记录了加多少次，然后再最后一次性加上去就好了。

可以看下例2.2的题目。

1. **K邻近算法。**

这个就不说了，已经很熟，距离评判的方式也有好多种，可以参考p39。K近邻算法是不存在训练过程的。

1. **朴素贝叶斯法。**

朴素贝叶斯法是基于贝叶斯定理和特征条件独立假设的分类方法。首先基于特征条件独立假设学习输入/输出的的联合概率分布，然后基于此模型，对于给定的输入x,利用贝叶斯定律求出后验概率最大的输出y，朴素贝叶斯法实现简单，学习和预测的效率都很高。

* 1. 朴素贝叶斯法的学习与分类。

基本方法：

设输入是n维向量的集合，输出空间的标记集合为，输入特征向量，输出标记,X是定义**X(这是n维向量的集合)**上的随机向量，Y是定义现在**Y**上的随机变量。P(X,Y)是X,Y的联合概率分布，训练数据是：



朴素贝叶斯法通过训练数据学习联合概率分布P(X,Y),具体地就是去学习以下先验概率分布以及条件概率分布，先验概率分布：



以及条件概率分布：



于是就可以学习到联合概率分布P(X,Y)。

条件概率分布:有指数量级的参数，其估计实际是不可行的。

朴素贝叶斯做了条件独立性的假设，这是一个较强的假设，朴素贝叶斯法也因此得名，这是一个比较强的假设，有了这条假设，概率分布就可以简化：



实际上是学到生成数据的机制，所以是生成模型，条件独立假设使得贝叶斯法变得简单，但是有时候会牺牲一定的准确率。

朴素贝叶斯分类时，对于给定的输入x来说，通过学习到的模型来计算其后验概率分布：,以后验概率最大的作为x的类输出，后验概率根据贝叶斯定理来计算：



由于条件独立性假设，则有：



这就是朴素贝叶斯法的基本公式，则其分类器可以表示为：



对于所有的来说，分母都是相同的，所以可以进一步把分母去掉，则变成：



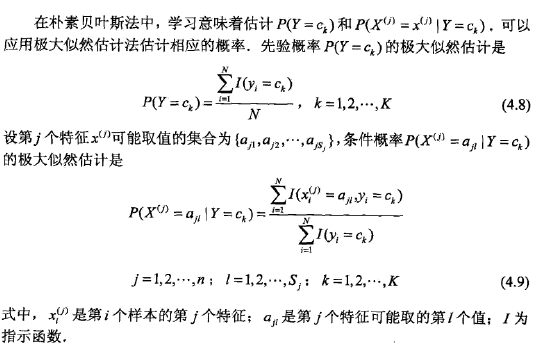
这就是朴素贝叶斯法所用的分类器。

后验概率最大化的意义：后验概率最大化，等价于期望风险最小化。证明详见P49，这里没有全看懂。

* 1. 朴素贝叶斯法的参数估计。

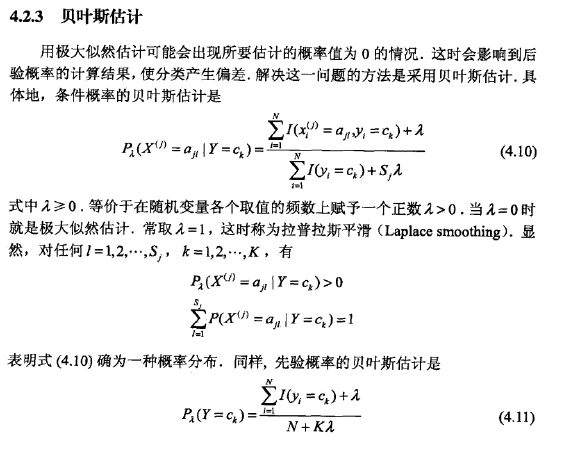
极大似然估计：

朴素贝叶斯法中，学习意味着可以用极大似然估计法来估计相应的概率。



这个我就直接贴图了，这实际上是一个用频率估计概率的方法。可以参见P50的例4.1。就是利用的这种方式。

但是用极大似然估计的时候，可能由于数据量缺少，出现概率值为0的情况，这时候就会影响到后验概率的计算结果，使得分类产生偏差，解决这一问题的一个方法就是采用贝叶斯估计，具体的就是在其后加上一个正数保证其不为0，这种方法称之为贝叶斯估计。



可以参见P52的例4.2。

1. **决策树。**

决策树简单的来讲就是一系列IF-THEN规则，本质上是从训练数据集中归纳出一组分类规则，与训练集不想矛盾的决策树可能有多个，也可能一个都没有，我们需要一个与训练集矛盾最小的决策树，同时具有较好的泛化能力。

5.1. 特征选择。

特征选择就是决定用哪个特征来划分特征空间，要说明这些问题，首先需要了解信息增益的概念。

为了便于说明，先给出熵和条件熵的概念。

熵是信息论和概率论中的概念，表示的随机变量不确定性的度量，设X是一个取优先值的离散随机变量，其概率分布为：



则随机变量X的熵定义为：



由于概率是0-1之间的，所以这个熵是个正数，而且不依赖于X的具体值，只与X的分布有关。随机变量的不确定性越大，其熵就越大。

条件熵：

设有随机变量，其联合概率分布为：



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X给定条件下Y的条件概率分布对X的数学期望。



其中，继续推导：



这说明，条件熵是条件概率分布在原分布下的期望（取了一个负对数）。

当熵和条件熵都是由数据统计得来的时候（特别是极大似然估计），分别称作经验熵和经验条件熵。

信息增益表示得知特性X的信息而使得类Y信息的不确定度的减少程度，通俗的来讲就是知道条件分布给熵带来的多少减少。

特征A对与训练数据集D的信息增益定义为：



一般的。熵H(Y)与条件熵H(Y|X)之差称为互信息，决策树学习中的信息上增益等价于训练数据集中类与特征的互信息。

决策树应用采用信息增益作为准则选取特征，给定训练数据集D和特征A，经验熵H(D)表示对数据集D进行分类的不确定性，而条件熵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，那么他们的差即信息增益表示了由于特征A而使得D的分类不确定性降低的程度，显然，D越依赖于A，则信息增益会越大，说明此特征具有更强的分类能力。

根据信息增益准则，对于训练集，计算每个特征的信息增益，并比较它们的大小，选择信息增益最大的特征。

关于信息增益的计算参见P62的例5.2。

以信息增益作为划分训练数据集的特征，会存在偏向于选取值较多的特征，使用信息增益比（相当于是一个归一化）可以对这个问题进行校正。

特征A对于训练数据集D的信息增益比定义为其信息增益与训练集D关于特征A的值得熵之比：



其中，参数意义看书吧。

5.2. 决策树的生成。

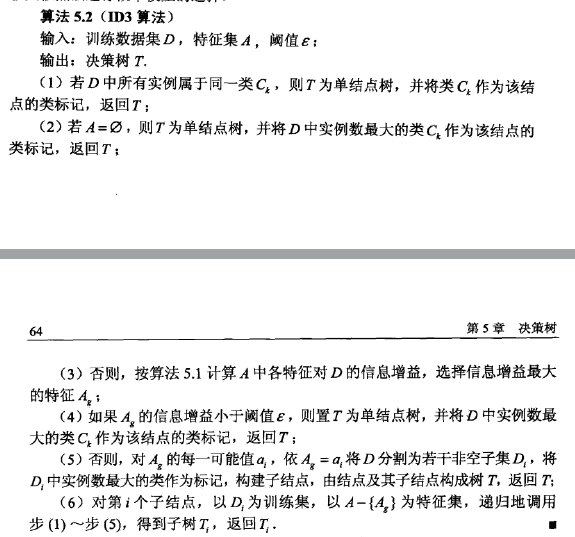
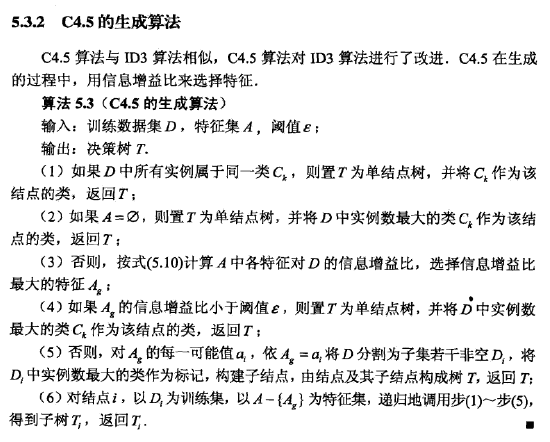
主要介绍两种算法，ID3和C4.5.

两种方法的主要思路是相同的，只是一个选择了信息增益作为准则，而另一个选择了信息增益比作为特征。算法我直接贴在下面。思路我试着用通俗的语言解释一下：

首先对于所有的特征来说，分别计算其对于分类的信息增益，选择增益最大的一种特征作为根节点，这样的分类可以根据特征把整体样本分枝，然后在各个分支中重复上述过程，知道满足停止条件。

停止的条件有以下几种：

* 当前节点全部属于同一类别，无需划分。
* 当前属性集为空，或者所在样本在特征上完全相同，无法划分。
* 当前节点包含的样本集为空，无法划分。

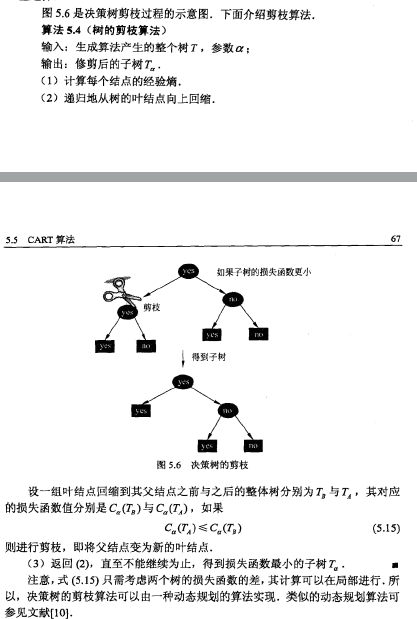
：

5.3. 决策树的剪枝。

决策树采用生成算法递归的产生决策树，这样产生的决策树往往对于熟练数据集比较准确，但是对于未知数据的预测却没那么准确，即产生了过拟合现象，过拟合产生的原因在于学习的时候过多追求训练数据的正确分类而构建出了过于复杂的决策树，解决这个问题的方法是考虑决策树的复杂度，对已生成的决策树进行简化处理。这个简化过程一般称之为剪枝。

这里介绍一种比较简单的剪枝方法。通过极小化决策树整体的损失函数或代价函数来实现。公式就不写了，P66页。

算法：



从叶节点开始剪枝，如果剪完后整体损失函数变小了，说明这个枝可以剪掉，则剪掉，否则这个枝不可剪掉。就这样一步一步来迭代，直到不能剪为止。

另外，剪枝还有一种思路：预留出一个训练集来进行剪枝处理。（这里参考西瓜书P81）

预留一部分来做训练集，依据在训练集上的预测精度来决定是否要开始剪枝。分别有两种处理方式：预剪枝和后剪枝，一种从上到下，一种从下往上，原理都很简单，参见西瓜书，我实在是懒得画图了。

5.4. CART算法。

分类与回归树（classification and regression tree）也是一种决策树，可用于分类和回归，同样由特征选择，树的生成和剪枝构成。

和以上两种算法不同的是其选择了一种叫做基尼系数的指标来划分属性。

假设有K个类，样本点属于第k类的概率是Pk，那么概率分布的基尼系数为：



这是多分类的，对于二分类来说，假设第1个类的概率是p，则概率分布的基尼系数为：



对于给定的集合D,其基尼系数为：



这里Ck是D中属于第k类的样本子集，K是类的个数。

如果样本被特征A分成两部分D1，D2，则在特征A的条件下，D的基尼系数定义为：



这个式子是最重要的，可以算出所有特征下的基尼系数，然后选择最小基尼系数的值来作为第一次分类。然后依次类推就可以进行生成决策树了，这样来看CART生成的决策树就一定是二叉树。

剪枝的算法可以参考上述的前剪枝和后剪枝的算法。

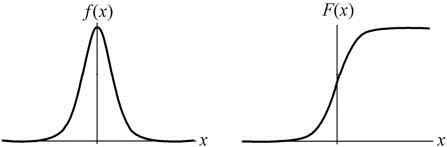
1. **逻辑斯蒂回归和最大熵模型。（LR回归）**

6.1. 逻辑斯蒂回归模型。

首先介绍下logistics分布：设X是随机变量，服从的分布函数和密度函数为：





可以画出他们的图像：

其中中心位置是，是一个形状参数，可以看出F(x)是关于(,0.5)中心对称的。整个曲线在中心附近增长比较快，而在两端增长比较慢，这是一个很好的归一化函数。

二项逻辑回归实际上是一种分类模型，来看最简单的二逻辑回归：我们用条件概率分布P(Y|X)表示，形式为参数化的logistics回归。



这里x是输入，Y是输出，对于给定的输入x，可以按照上式求得两个概率，logistic回归比较两个概率的大小，并将实例x分到概率大的那个类中。

有时候为了方便，将w进行扩充，最后一项为b，这样就算吧偏置一块计算进去了。即，这是logistic回归模型则会变为：



现考察logistic回归的特点，一个事件发生的几率（odds）指的是发生的概率与其不发生的概率的比值，如果其发生的概率是p，则该事件的几率是p/1—p，该事件的对数几率为：



对于logistic来说，由上式可以得到：



这就说明，在logistic回归模型之中，输出Y=1的对数几率是输入x的线性函数，或者说，输出Y=1的对数几率是由输入x线性表示的模型。

换言之，这实际上是一种归一化模型，的值域本身是实数空间，我们通过logistic转换之后转换到0-1之间，而且这个和实数空间是一个单调对应的映射，线性函数的值越大，这个logistic回归值就越接近与1，反之，线性函数的值越小则越接近于0。这样的模型就是logistic回归模型。

其模型参数估计可以采用极大似然函数法：

设则有：。

似然函数为： 

这个函数就是一个连乘，实际对于每一个i都只能取一项，根据的值来取。

套路是取对数似然函数：



对其取最大值，得到w的估计值。

这样的话，问题就变成了对对数似然函数的优化问题，常采用的回归学习通常采用的方法是梯度下降法即拟牛顿法。

这种策略可以推广到多项的logistic回归模型，用于多类分类。P79

6.2. 最大熵模型。

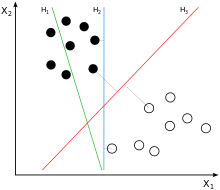
这一块全是公式，今晚实验室很乱，看不进去，先跳过吧，以后补。

1. **支持向量机。**

支持向量机是一种二分类模型，基本模型是定义在特征空间上的间隔最大的线性分类器，间隔最大使得它有别于感知机，支持向量机还包括核技巧，这使其成为实质上的非线性分类器。

7.1. 线性可分SVM和硬间隔最大化。

对于线性可分的的数据，在感知机那里我们就可以求解了，那个时候我们也指出了，这样的求解不是唯一的，因为可以使得样本集线性分开的超平面不唯一。但并非这些超平面都具有比较好的泛化能力。一个显而易见的可能是，如果这个超平面离两类都比较远的话，这个模型的泛化能力应该是说不错的。

就是出于这么一个思路，来引入支持向量机。

如图，对于一组训练数据：。我们可以用间隔最大化或等价的求解其响应的凸二次模型来学习到超平面：



决策函数为：



首先来介绍两个概念：函数间隔和几何间隔。

* 函数间隔。

对于给定的训练集和超平面来说，定义超平面关于样本点的函数间隔为：



这适中是一个正值，可以衡量样本点到超平面的距离。

我们定义超平面关于训练集的函数间隔为所有样本函数间隔的最小值：



但是选择超平面时，用函数间隔存在不合理的地方，因为我们只要成比例的改变w, b, 超平面关于数据集的函数间隔增加，然而超平面却没有发生变化。一个很自然的想法就是用几何间隔来代替函数间隔，几何间隔相当于用法向量对函数间隔做归一化。

* 几何间隔。

定义超平面关于样本点的几何间隔为：



定义超平面关于训练集的几何间隔为：所有样本几何间隔中的最小值：



所以几何间隔是函数间隔的规范化。

另外一个概念是间隔最大化，支持向量机的基本想法就是求解能够正确分类训练集的超平面并使得几何间隔最大化。几何间隔最大化的超平面时唯一的。这里的间隔又称作硬间隔。

最大间隔分离超平面：这其实是转化为一个最优化问题：



即最大化几何间隔，满足所有样本点的几何间隔都比这个几何间隔大。

我们把上述中的几何间隔换成函数间隔：



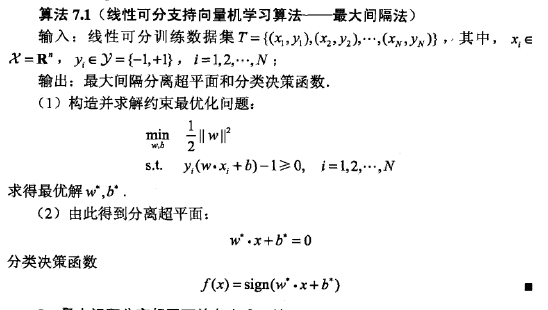
再化简：



实际上我们给优化目标乘以一个比例因子也是可以的，因为按照比例变换w，b对超平面并没有影响，所以我们可以令=1，然后把最大值问题改写成最小值问题，并且把一阶范数变成二阶范数。则有：

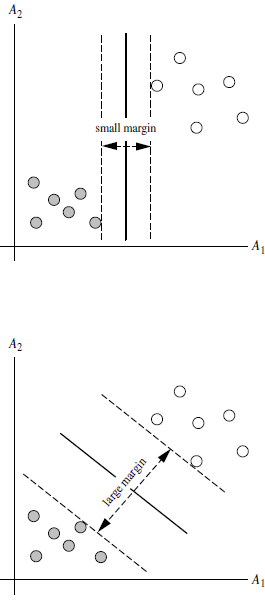


这是一个典型的凸二次规划问题。在最优化求解方法里面，这个是可解的，以此，我们提出了基于最大间隔法的支持向量机算法：



这样的算法是比较好理解的，在P100，书中给出了最大间隔分离超平面的存在性和唯一性证明。我就不看了。

还有两个概念要提：支持向量和间隔边界。

如左图所示，支持向量就是算法约束里使得等号成立的点，即满足：

的点，对于离分类面最近的正类点和负类点，分别满足：



这是两个超平面，这两个超平面之间的间隔称之为间隔，两个超平面称之为间隔边界。支持向量则是离分类面最近的实例。

在决定分类超平面时，只有支持向量起作用，其他点都不起作用，如果移动支持向量就会改变所求的解，但是移动其他实例点（间隔边界之外移动），则不会改变求解。支持向量的个数一般很少，所以支持向量机实际上是由很少的“重要的”训练样本确定。

上面算法所述把SVM转换为一个最优化问题，已经是可以求解了，而且是一个二次凸优化问题，用一般的二次优化的优化工具就可以求解了。但是我们可以由更高效的方法，这就是用其对偶形式。引入对偶形式一个是因为对偶形式往往更好求解，二是自然引入核函数，进而推广到非线性分类问题。

我们回顾前面得到的优化问题：



我们对每一个不等式约束引入拉格朗日乘子，并且定义拉格朗日函数：



其中称作拉格朗日乘子，通过拉格朗日乘子，我们把约束条件融入到目标函数里面，只用一个函数就可以表示我们的问题。

我们记：，容易验证，当某个约束调价不满足时：即，那么，这时我们只需要使趋于无穷就可以了。当所有条件都满足时，我们会得到（这个的得来主要是来控制，看求和号里面的那一部分，因为要满足的条件，所以对于小于零的那一部分是要取零的（这一部分不是支持向量）），这就是我们最初需要优化的值。所以在要求约束条件得到满足得条件下最小化，实际上是等价于直接最小化（这里当然也有约束条件：所有的）如果有约束条件没有得到满足，，自然不是我们要求的最小值。

所以现在我们的目标函数可以写作：



这里我们用来表示这个最优值，这个和我们最初的问题是等价的，不过我们现在把求最大和最小的位置交换一下，则会得到：



当然，交换以后的问题就不再等价于原问题，但是我们可以有：，这个从直观上也不难理解，从最小值里面取得最大值肯定是小于等于从最大值中取的最小值的，所以这个实际上是给出了的一个下界，在满足某些条件的情况下，这两者是相等的，这个时候我们可以用第二个问题来求解第一个问题。

这所谓的满足某些条件，就是首先要满足Slater条件，进而满足KKT条件。理由如下：

1. 在凸优化问题中，q∗ 和p∗ 相同的条件是Slater 条件，该条件保证鞍点Saddle Point 存在。

2. 至于KKT 条件，首先原问题的最优值可以通过求Lagrange 函数的鞍点（如果有的话）来得到，再

者，KKT 条件里面进一步引入了更强的前提，也就是在满足Slater 条件的同时（前面说了，Slater

条件保证鞍点存在），f(·) 和gi(·) 都是可微的，这样鞍点不仅存在，而且能通过对Lagrange 函数求

导得到，

3. 所以KKT 条件是一个点是最优解的条件，而不是p∗ = p∗ 的条件，当然这个KKT 条件对后边简化

对偶问题很关键。

我们接着往下算，对于上述的最小最大问题，我们先算：，我们队拉格朗日函数求偏导，并令偏导为0，则可以得到：



则可得到：



将这两个式子代入拉格朗日函数，可以得到：



即：



这个结果对求极大，则得到其对偶问题：



把极大转换为极小：



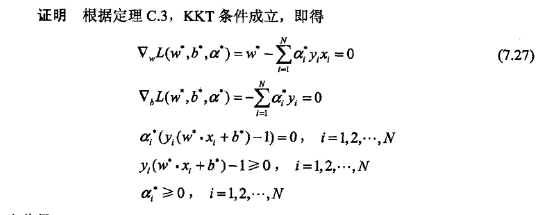
这就是我们得到的对偶问题：

这就是一个对偶优化问题了，假设我们求得了其解：，可由下式求得w和b。



关于KKT条件从拉格朗日乘子法一步一步得来的话可能容易很多，和高等数学当时学的那里比较像，具体的[这一篇博客](https://blog.csdn.net/the_lastest/article/details/78461566)我感觉讲的比较好，从等式约束一直讲到不等式约束，讲了对偶性和KKT条件的关系，即上面为什么\*p和\*q在KKT条件下的优化值是一样的。

KKT条件我截图放到下面了：



KKT条件主要有四部分构成：

* 拉格朗日函数关于各个变量的导数=0；（原始变量和引入的拉格朗日乘子）。
* 原始约束条件。
* 不等式约束乘以拉格朗日乘子=0；
* 拉格朗日乘子大于等于0.

7.2. 线性支持向量机与软间隔最大化。

这一节和上一节的推导基本是相同的，看书的P108—115。考虑到现实中的情况数据往往不是线性可分的，总有一些数据是离群的，不能满足函数间隔大于1 的条件，所以给他们机上一个松弛变量使得满足约束条件，同样在目标函数上，支付一个代价，还有一个惩罚参数，这样就可以进行继续推导了。

并且，在后面书中证明了线性支持向量机的原始优化问题（带软间隔）实际上等于合页损失下的最优化问题，在P114中给出了证明。

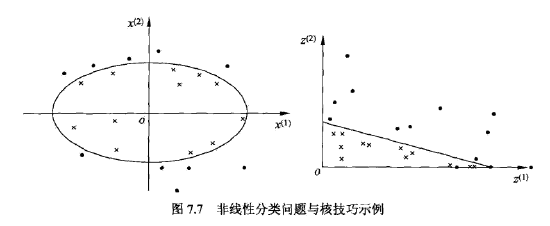
7.3.非线性支持向量机和核函数。

对于线性分类问题，线性支持向量机可以有效求解，但是对于非线性问题，这个时候线性SVM就无力了，这一节主要是通过核技巧来把非线性问题转换为线性问题，然后利用线性SVM求解，值得注意的是，核技巧不仅适用于SVM，也适用于其他统计学习问题。

通过一个简单的例子来说核技巧是个什么东西：

如上图所示，展示了一个可以被椭圆分开的一个数据集，我们如何通过变换使得这个数据集可以被直线分开呢（右边所示）。

对于这个问题，参考椭圆的约束方程，其实很简单，只需对原空间进行平方映射就可以了：

假设元空间是：,映射后的空间是：，做如下变换：



经过了非线性变换，原空间变换到了新的空间（虽然维数没有发生变化但是基底是变了的），元空间中的椭圆：



在新空间下变成了：



从线性不可分变成了线性可分，这个例子说明，可以通过对元空间进行非线性变换来使得线性不可分问题变成线性看可分问题。

变换函数我们称之为核函数：

设输入空间X，H是特征空间（希尔伯特空间），如果存在一个从X到H的映射：

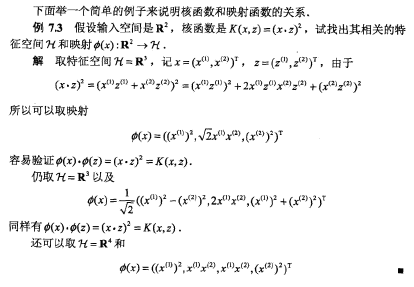


使得对于所有的，函数K(x,z)满足条件：



则称K(x,z)为核函数，点乘表示内积。

核技巧一般只定义核函数而不显示地定义映射函数，通常，直接计算核函数比较简单，而通过映射函数计算核函数则通常不容易，特征空间一般都是高维的或者无穷维的，而且对于给定的核函数，映射函数并不唯一，可以看下面的例子：



这个例子中，核函数没变，但是映射函数却不唯一，甚至映射的空间维数都不相同。

那么核技巧如何可以运用到SVM中呢？这就得从SVM的对偶问题说起，但是推导对偶问题的时候说了，算对偶优化问题主要是基于两点考虑，一是运算简单（SMO算法），二就是可以天然引入核函数。

我们来看对偶问题的优化函数：



优化目标函数中，或者说是决策超平面，都只涉及实例之间的内积，所以我们可以把其中实例的内积用核函数来代替，就等价于用映射函数将原输入空间变换到新的特征空间，在心的空间中用SVM来分类，学习到含有核技巧的SVM模型。一般往往依赖于领域知识直接选择核函数，核函数是否有效也只能通过实验来验证了。

不用构造映射函数怎么判断K(x,z)就是一个核函数呢？或者说，其满足怎样的条件才能称作核函数。结论是：正定核函数。证明有一大堆，感兴趣的话可以看P119。

常用的核函数：

* 多项式核函数。

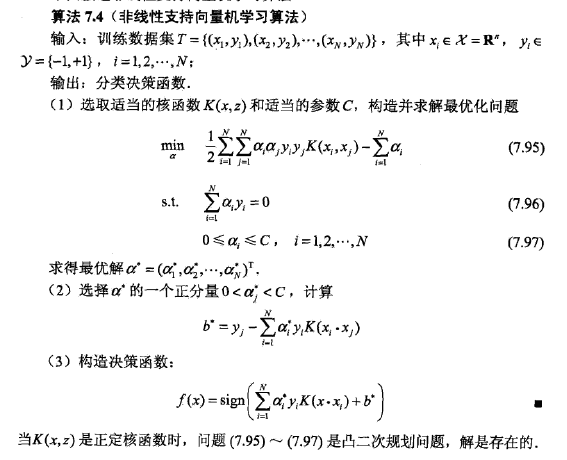


* 高斯核函数。



* 字符串核函数。

这个看书吧，看着挺复杂的，是定义在字符串集合上的，我的主要领域是图像，这部分就不看了。

综上，我们可以得到非线性支持向量机学习算法：

7.4.序列最小化算法（SMO）

上面提出的SVM是一个典型的凸二次规划问题，由一般的二次规划工具就可以求解了，但是在样本量非常大的情况下求解就会变得非常低效，以致无法使用，所以如何高效的实现SVM的学习变成一个非常重要的问题，人们提出了许多快速实现算法，其中最经典的就是SMO算法。

这个算法的证明书上有，暂时就当做一个坑吧，先放着，因为毕竟现在的计算包都很成熟，不会自己去编程实现SVM的，所以就暂且放下。证明过程参考P125。

1. **提升方法（boosting）**

提升方法是统计学习中常用的，在分类问题中，通过改变训练样本的权重，学习多个分类器，并将这些分类器进行组合，提高分类性能，这里面常用的就是AdaBoost算法。还有后续的提升树（Boosting Tree）。

8.1. 提升方法AdaBoost算法。

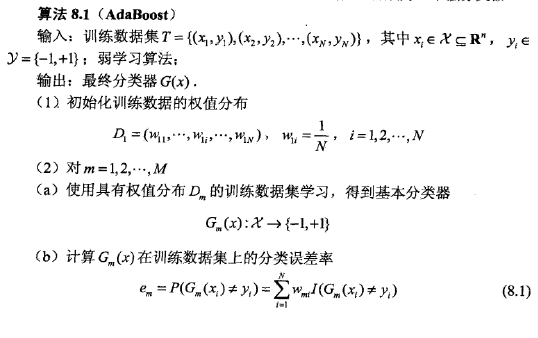
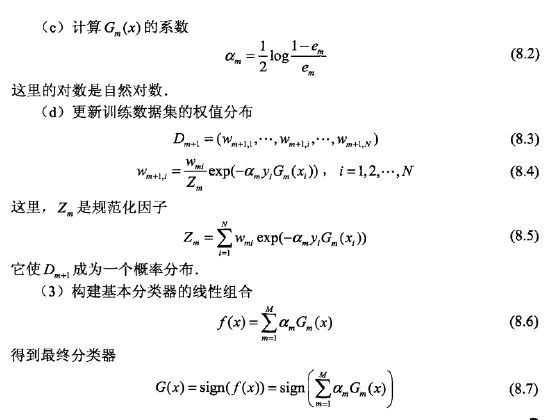
这个算法通俗的来讲就是通过一定的学习策略，把几个弱分类器按照一定的加权组合起来，得到一个强分类器，就是“三个臭皮匠赛过诸葛亮”的道理。

废话不说，直接上算法：

对于一个二分类的训练数据集：



其中y是标签属于{1，-1}。下面是这种算法的算法流程：



简单的来说，对于原始数据给定平均权重，然后在这个权重下，找到一个简单分类器，根据其误差率确定其系数（8.1式），然后更新原始数据的权值，更新的方向是增加错误分类的权重，使得下一个训练的分类器像这些数据倾斜。然后重复（2）中的步骤，值得算法收敛。

最终按照加权和的形式把这些弱分类器组合起来形成一个强分类器。

8.2-8.3. AdaBoost算法的训练误差分析和算法解释。

这部分没有很关注，都是一些理论上的东西。

8.4. 提升树。

提升树是以分类树或者回归树作为基本分类器的提升算法，被认为是统计学习中性能最好的方法之一。主要思路也是用不同的分类树或者回归树进行累加，得到性能更好的分类器或者回归树。

当用分类树时，这个和adaboost就没有太大区别了，这里主要讨论的是回归树。

提升树采用的前向分步算法，首先确定初始的树，第m步的模型是：



这个式子的意思就是模型是一步一步叠加起来的。其中是当前的模型，通过经验风险极小化来确定下一棵决策树的参数。



如果把输入空间化为J个互不相交的区域R1，R2，R3…RJ，并在每个区域确定输出的常量Cj，那么树可以表示为：



其中，参数{(R1，c1),( R2，c2),…(RJ,cJ)}表示树的区域划分和各区域的常数，J是回归树的复杂程度即叶节点的个数。

所以回归提升树使用下面的算法：



在前向算法的第m步，优化：

当采用平方误差函数时：



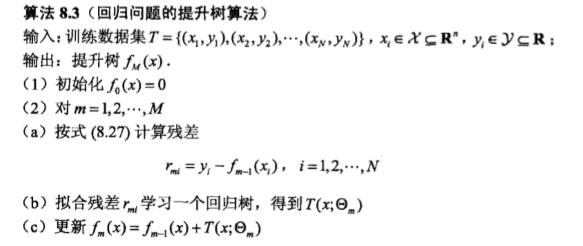
损失变为：

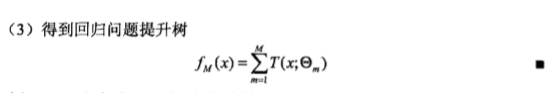


其中



这就是当前模型的拟合数据的残差，所以对于回归问题的提升树算法来说，在每一步只需要简单拟合当前模型的残差即可。





可以去看下P149的习题，说明了具体怎么去算残差，怎么进行树的融合。

另外，对于一般的损失函数而言，每一步优化可能不是那么简单，针对这一问题，提出了基于梯度提升的方法，这一步不是根据残差而是根据负梯度方向来更新模型。具体看P151。