# 一般线性模型

## 线性模型概述

首先，给定一个示例，它具有d个属性，我们将它表示为，其中表示在第i个属性上的取值。线性模型试图学得一个与属性有关的线性组合并据此来进行预测的函数，即

，（1.1）

一般用向量形式表示，即

，（1.2）

其中。和b的值确定之后，模型就能得到确定了。

给定数据集，其中。线性回归即假设特征满足线性关系，根据给定的训练集D训练一个线性模型，并用此模型进行预测。

我们先考虑一种最简单的情形：输入属性的数目只有一个。那么预测函数为。接下来就是和b的确定。显然，关键在于使与实际值的差别尽可能的小。因为均方误差是回归任务中最常用的性能度量，所以我们将均方误差作为衡量差别的工具，目的是让均方误差最小化，即

，（1.3）

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”。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氏距离之和最小。均方误差的几何意义对应的就是欧几里得距离，简称欧式距离。

求解w和b使最小化的过程，称为线性回归模型的最小二乘“参数估计”。我们可将分别对w和b求导，得到

，（1.4）

，（1.5）

然后令偏导数为零，即可得到w和b最优解的闭式解

，（1.6）

，（1.7）

其中为的均值。

讨论完只有一个属性的简单形式，我们来更一般的情形，即样本具有d个属性，此时我们试图学得的函数为。这称为“多元线性回归”。

类似的，可利用最小二乘法来对和b进行估计。为了便于讨论，我们把和b用向量形式来表示，把数据集D表示为一个大小的矩阵，其中每行对应一个示例，该行前个元素对应于示例的d个属性值，最后一个元素都为1，即

，

再把标记也写成向量形式，则类似于式（1.3），有

.（1.8）

令，对求导得到

. （1.9）

同样的，令上式为零就可以得到最优解的闭式解。但是由于涉及矩阵的逆运算，我们需要做一个简单的讨论。

当为满秩矩阵或者正定矩阵时，令式（1.9）为零可得

.（1.10）

令，则最终学得的多元线性回归模型为

.（1.11）

然而，现实任务中往往不是满秩矩阵。例如在有的任务中，我们会遇到示例的属性个数超过样例数，也就是的列数要多于行数，显然不满秩。这时候解出的不唯一，它们都能够使均方误差最小化。这时候决定选择哪一个解作为输出，就要由学习算法的归纳偏好决定了。常见的做法就是引入正则化项。

参考资料：《机器学习》周志华

# k-邻近算法

## k-邻近算法概述

k-近邻（k-Nearest Neighbor，简称kNN）是一种常用的监督学习方法。它具有精度高、对异常值不敏感、无数据输入假定的特点。不足之处在于计算复杂度高、空间复杂度高。适用于数值型和标称型数据。

### 基本思想

它的工作原理是：给定一个训练样本集，基于某种距离度量找出训练集中与新数据最靠近的k个训练样本，然后将根据这k个“邻居”的特征来预测新数据的分类标签。它并没有显示的训练过程，而仅仅在训练阶段将样本保存起来，待收到测试样本后再进行处理，它的训练时间开销为零。

### 算法概述

近邻算法是基于实例学习的分类算法中比较常用的一种方法。令，其中每一个样本所属的类别均已知。对于测试样本点，在集合中距离它最近的点记为，那么，最近邻规则的分类方法就是把点分为所属的类别。最近邻点的标记是一个随机变量。的概率为后验概率。

当样本个数非常大的时候，有理由认为距离足够近，使得。

因为这恰好是状态位于的概率，因此最近邻规则自然是真实概率的一个有效的近似。

K近邻规则是最近邻规则的一个推广。从这个规则名词可知，该规则将是一

个测试数据点分类为与它最接近的K个近邻中出现最多的那个类别。K近邻算

法从测试样本点开始生长，不断的扩大区域，直到包含进K 个训练样本点为止，并且把测试样本点归为这最近的K个训练样本点中出现频率最大的类别。

如果K值固定，并且允许训练样本个数趋向于无穷大，那么，所有的这K 个

近邻都将收敛于x 。如同最近邻规则一样，K个近邻的标记都是随机变量，概率

都是相互独立的。而最近邻规则则以概率选取类别。而根据K近邻规则，只有当K个最近邻中的大多数的标记记为,才判定为类别。做出这样判定的概率为



通常K值越大，选择类别的概率也越大。

K近邻规则可以被看作是另一种从样本中估计后验概率的方法。为了得到可靠的估计必须使得 K值越大越好。另一方面，又希望x的K个近邻距离x越近越好，因为这样才能保证尽可能逼近。在选取K值的时候，就不得不做出某种折衷。只有当n趋近于无穷大时，才能保证K近邻规则几乎是最优的分类规则。

算法步骤：

（1）构建训练样本集合X。

（2）设定K的初值。K值的确定没有一个统一的方法（根据具体问题选取的K值可能有较大的区别）。一般方法是先确定一个初始值，然后根据实验结果不断调试，最终达到最优。

（3）在训练样本集中选出与待测样本最近的K个样本。假定样本点x属于n维空间，样本之间的“近邻”一般由欧式距离来度量。设第i个样本，其中表示第i个样本第个特征属性值。那么两个样本和之间的欧氏距离定义为：



1. 给定一个待分类的样本，表示与距离最近的K个样本，设离散的目标函数（分类问题）为，表示第i个类别的标签，标签集合定义为，

，

表示对的估计，当时，；否则，。

1. 就是待测样本的类别。

参考资料：基于K近邻的分类算法研究\_桑应宾.caj

《机器学习》周志华

《机器学习实战》

# 支持向量机

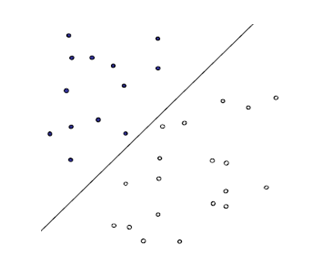
## 概述

支持向量机(Support Vector Machine，简称SVM)是一种高效的的监督学习算法。它的主要思想可以概括为2点：（1）针对线性可分情况进行分析。（2）对于线性不可分的情况，通过使用核函数，将低维线性不可分空间转化为高维线性可分的情况，然后再进行分析。

## 支持向量机算法

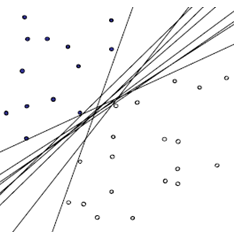
### 线性分类器

首先给出一个非常非常简单的分类问题（线性可分），我们要用一条直线，将下图中黑色的点和白色的点分开，很显然，图上的这条直线就是我们要求的直线之一（可以有无数条这样的直线）。



假如说，我们将黑色的点用-1表示， 白色的点用+1表示，直线，这儿的、是向量，其实写成这种形式也是等价的，当向量的维度=2的时候，表示二维空间中的一条直线，当的维度=3的时候，表示3维空间中的一个平面，当x的维度=n > 3的时候，表示n维空间中的n-1维超平面。

但是，我们怎样才能取得一个最优的划分直线呢？下图的直线表示几条可能的。



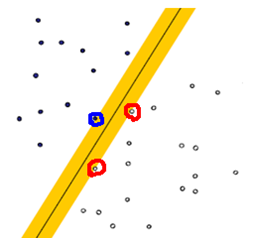
从直观上来说，就是分割的间隙越大越好，把两个类别的点分得越开越好。在SVM中，称为最大间隔，是SVM的一个理论基础之一。选择使得间隙最大的函数作为分割平面是由很多道理的，比如说从概率的角度上来说，就是使得置信度最小的点置信度最大，从实践的角度来说，这样的效果非常好，等等。

假设超平面能将训练样本正确分类，即对于，则有

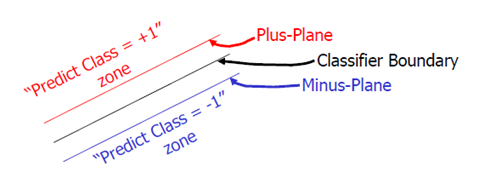
 （3.1）

如下图，被红色和蓝色的线圈出来的点，即距离超平面最近的几个训练样本点就是所谓的支持向量(support vector)。两个异类支持向量到超平面的距离之和被称为“间隔”，

， （3.2）



下图就是一个对之前说的类别中的间隙的一个描述。表示的是超平面，红色和蓝色的线就是支持向量所在的面，红色、蓝色线之间的间隙就是我们要最大化的分类间的间隙。



想要找到具有最大间隔的划分超平面，也就是要找到能满足式（3.1）中约束的参数和b，使得最大，即

（3.3）

显然，为了最大化间隔，仅需最大化，这等价于最小化。于是，式（3.3）可重写为

（3.4）

这就是支持向量机（SVM）的基本型。

### 对偶问题

式（3.4）优化问题可以用拉格朗日乘子法得到其“对偶问题”并求解，使用了KKT条件的理论。具体地说就是对（3.4）的每条约束添加拉格朗日乘子，则该问题的拉格朗日函数可写为

 （3.5）

其中。令式（3.5）对和b的偏导为零可得

，（3.6）

。（3.7）

将两式带回（3.5）得到对偶问题的表达式

（3.8）



解出后，求出和b即可得到模型

 （3.9）

（3.8）就是我们需要最终优化的式子。至此，得到了线性可分问题的优化式子。求解这个式子，有很多的方法，比如SMO等等。

### 软间隔

接下来谈谈线性不可分的情况，因为线性可分这种假设实在是太有局限性了。线性不可分的分类图就是我们没有办法用一条直线去将其分成两个区域，每个区域只包含一种颜色的点。

要想在这种情况下的分类器，有两种方式，一种是用曲线去将其完全分开，曲线就是一种非线性的情况，跟之后将谈到的核函数有一定的关系；另外一种还是用直线，不过不用去保证可分性，就是包容那些分错的情况，不过我们得加入惩罚函数，使得点分错的情况越合理越好。其实在很多时候，不是在训练的时候分类函数越完美越好，因为训练函数中有些数据本来就是噪声，可能就是在人工加上分类标签的时候加错了，如果我们在训练（学习）的时候把这些错误的点学习到了，那么模型在下次碰到这些错误情况的时候就难免出错了。这种学习的时候学到了“噪声”的过程就是一个过拟合（over-fitting），这在机器学习中是一个大忌，我们宁愿少学一些内容，也坚决杜绝多学一些错误的知识。还是回到主题，用直线怎么去分割线性不可分的点：

我们可以为分错的点加上一点惩罚，对一个分错的点的惩罚函数就是这个点到其正确位置的距离。我们在原函数上面加上一个惩罚函数，并且带上其限制条件为：

，（3.10）



这就是常用的“软间隔支持向量机”。C是一个由用户去指定的系数，表示对分错的点加入多少的惩罚，当C很大的时候，分错的点就会更少，但是过拟合的情况可能会比较严重，当C很小的时候，分错的点可能会很多，不过可能由此得到的模型也会不太正确，所以如何选择C是有很多学问的，不过在大部分情况下就是通过经验尝试得到的。

接下来就是同样的，求解一个拉格朗日对偶问题，得到一个原问题的对偶问题的表达式：

 （3.12）



是与线性可分的对偶问题表达式的不同之处。在线性不可分情况下得到的对偶问题，不同的地方就是的范围从[0, +∞)，变为了[0, C]，增加的惩罚没有为对偶问题增加什么复杂度。

### 核函数

线性不可分的情况下，如果使用某些非线性的方法，可以得到将两个分类完美划分的曲线，比如接下来将要说的核函数。一个低维的样本集映射到高维则可以变成线性可分，那样才能使用SVM工作。

设映射函数为，则映射后的空间分类函数变成



但是，如果拿到低维数据直接映射到高维的话，维度的数目会呈现爆炸性增长。所以这里需要引入核函数（kernal function）。核函数的思想是寻找一个函数，这个函数使得在低维空间中进行计算的结果和映射到高维空间中计算内积的结果相同。

这样就避开直接在高维空间中进行计算，而最后的结果却是等价的。现在，分类函数就变成了这样：



其中就是核函数。由于对任意数据集找到它合适的映射是困难的且没有必要，所以通常会从常用核函数中选择。

参考资料：1、机器学习中的算法(2)-支持向量机(SVM)基础 - LeftNotEasy - 博客园 <https://www.cnblogs.com/LeftNotEasy/archive/2011/05/02/basic-of-svm.html>

1. SVM支持向量机算法 - CSDN博客 <http://blog.csdn.net/androidlushangderen/article/details/42780439>
2. 《机器学习》周志华

# AdaBoost算法

## 基本思想

AdaBoost，是英文"Adaptive Boosting"（自适应增强）的缩写，由Yoav Freund和Robert Schapire在1995年提出。它的自适应在于：前一个基本分类器分错的样本会得到加强，加权后的全体样本再次被用来训练下一个基本分类器。同时，在每一轮中加入一个新的弱分类器，直到达到某个预定的足够小的错误率或达到预先指定的最大迭代次数。

具体说来，整个AdaBoost 迭代算法就3步：

（1）初始化训练数据的权值分布。如果有N个样本，则每一个训练样本最开始时都被赋予相同的权重：1/N。

（2）训练弱分类器。具体训练过程中，如果某个样本点已经被准确地分类，那么在构造下一个训练集中，它的权重就被降低；相反，如果某个样本点没有被准确地分类，那么它的权重就得到提高。然后，权重更新过的样本集被用于训练下一个分类器，整个训练过程如此迭代地进行下去。

（3）将各个训练得到的弱分类器组合成强分类器。各个弱分类器的训练过程结束后，加大分类误差率小的弱分类器的权重，使其在最终的分类函数中起着较大的决定作用，而降低分类误差率大的弱分类器的权重，使其在最终的分类函数中起着较小的决定作用。换言之，误差率低的弱分类器在最终分类器中占的权重较大，否则较小。

## 算法流程

给定一个训练数据集，其中实例，而实例空间，属于标记集合{-1,+1}，AdaBoost的目的就是从训练数据中学习一系列弱分类器或基本分类器，然后将这些弱分类器组合成一个强分类器。

其算法流程如下：

步骤1。首先，初始化训练数据的权值分布。每一个训练样本最开始时都被赋予相同的权重：1/N。



步骤2。进行多轮迭代，用表示迭代的第多少轮。

a.使用具有权值分布的训练数据集学习，得到基本分类器：



b.计算在训练数据集上的分类误差率：



由上述式子可知，在训练数据集上的误差率就是被误分类样本的权值之和。

c. 计算的系数，表示在最终分类器中的重要程度，目的是为了得到基本分类器在最终分类器中所占的权重：



由上述式子可知，时，，且随着的减小而增大，意味着分类误差率越小的基本分类器在最终分类器中的作用越大。

d.更新训练数据集的权值分布（目的：得到样本的新的权值分布），用于下一轮迭代:





使得被基本分类器误分类样本的权值增大，而被正确分类样本的权值减小。就这样，通过这样的方式，AdaBoost方法能“聚焦于”那些较难分的样本上。其中，是规范化因子，使得成为一个概率分布：



步骤3。组合各个弱分类器：



从而得到最终分类器，如下：



参考资料：Adaboost 算法的原理与推导（读书笔记） <http://www.360doc.com/content/18/0112/12/51961713_721308110.shtml>