Machine Learning

于海悦

2017 年 5 月 8 日

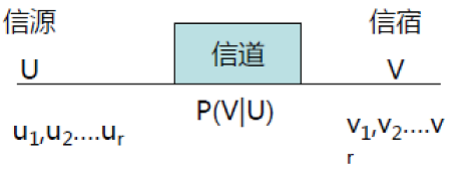
1 决策树

1.1 理论

1.1.1 信息论原理

- 信息论一般用于分类问题，从大量数据中获取知识。例如在各个实例的类别数据中，找出确定类别的关键条件属性，如在决策树中的 ID3 算法和 C4.5 算法，把信息量最大的属性作为树或者子树的根节点，属性的取值作为分支。
- 信道模型

  一个传递信息的系统是由发送端（信源）和接收端（信宿）以及连接两者的通道（信道）三者组成。



- 先验不确定性

在进行实际的通信之前，收信者（信宿）不可能确切了解信源究竟会发出什么样的具体信息，不可能判断信源会处于什么样的状态。这种情形就称为信宿对于信源状态具有不确定性。而且这种不确定性是存在于通信之前的。因而又叫做先验不确定性，表示成：信息熵 H(U)

- 后验不确定性

在进行了通信之后，信宿收到了信源发来的信息，这种先验不确定性才会被消除或者被减少。通信结束之后，信源仍然具有一定程度的不确定性。这就是后验不确定性，用条件熵表示 H(U/V)。

- 信息量

  后验不确定性总要小于先验不确定性: H（U/V）< H（U）

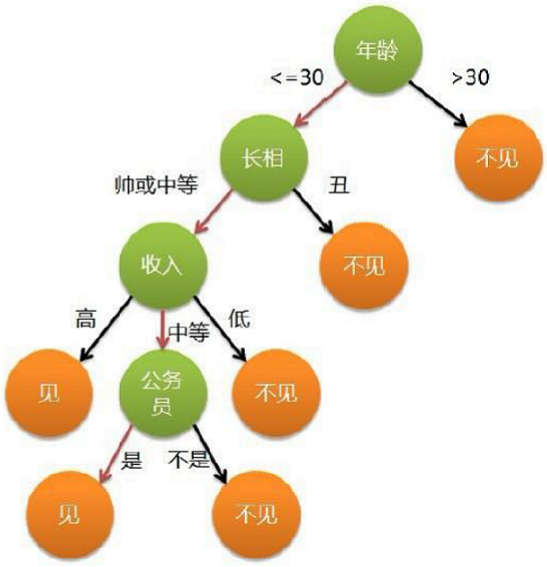
  信息量用互信息来表示，即：I（U，V）= H（U）− H（U/V）

1.1.2. 决策树概念

* 举一个简单的例子：

女孩的母亲要给这个女孩介绍男朋友

女儿：多大年纪了？
母亲：26。
女儿：长的帅不帅？
母亲：挺帅的。
女儿：收入高不？
母亲：不算很高，中等情况。
女儿：是公务员不？
母亲：是，在税务局上班呢。
女儿：那好，我去见见。



决策树用样本的属性作为结点，用属性的取值作为分支的树结构。其中，根结点是所有样本中信息量最大的属性。叶结点是样本的类别值。

1.1.3.ID3 算法

1. 算法的基本思想

首先找出最有判别力的特征，把数据分成多个子集，每个子集又选择最有判别力的特征进行划分，一直进行到所有子集仅包含同一类型的数据为止。最后得到一棵决策树。

引进信息论中的互信息 (信息增益, information gain)，作为特征判别能力的度量；将建树的方法嵌在一个迭代的外壳之中。

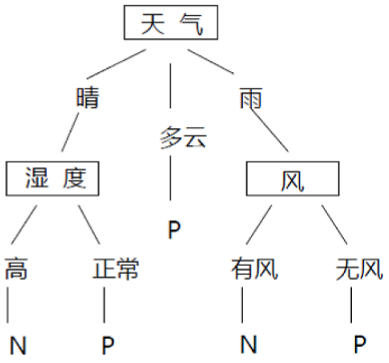
ID3 算法能得出结点最少的决策树。

- 举例：气候数据，特征分别为:
  - 天气取值为：晴，多云，雨
  - 气温取值为：冷，适中，热
  - 湿度取值为：高，正常
  - 风取值为：风，无风

为简单起见，这里我们假定样本仅有两个类别，分别为 P(正例)，N(反例)

NO.	属性				类别
	天气	气温	湿度	风	
1	晴	热	高	无风	N
2	晴	热	高	有风	N
3	多云	热	高	无风	P
4	雨	适中	高	无风	P
5	雨	冷	正常	无风	P
6	雨	冷	正常	有风	N
7	多云	冷	正常	有风	P
8	晴	适中	高	无风	N
9	晴	冷	正常	无风	P
10	雨	适中	正常	无风	P
11	晴	适中	正常	有风	P
12	多云	适中	高	有风	P
13	多云	热	正常	无风	P
14	雨	适中	高	有风	N

训练集



ID3决策树

2. 算法实现

主算法

1. 从训练集中随机选择一个既含正例又含反例的子集（称为“窗口”）；
2. 用“建树算法”对当前窗口形成一棵决策树；
3. 对训练集（窗口除外）中例子用所得决策树进行类别判定，找出错判的例子；
4. 若存在错判的例子，把它们插入窗口，转 2，否则结束。

建树算法

1. 对当前例子集合，计算各特征的互信息；
2. 选择互信息最大的特征A_k；
3. 把在A_k 处取值相同的例子归于同一子集，A_k 取几个值就得几个子集；
4. 对既含正例又含反例的子集，递归调用建树算法；
5. 若子集仅含正例或反例，对应分枝标上 P 或 N，返回调用处。

3. 举例

NO.	属性				类别
	天气	气温	湿度	风	
1	晴	热	高	无风	N
2	晴	热	高	有风	N
3	多云	热	高	无风	P
4	雨	适中	高	无风	P
5	雨	冷	正常	无风	P
6	雨	冷	正常	有风	N
7	多云	冷	正常	有风	P
8	晴	适中	高	无风	N
9	晴	冷	正常	无风	P
10	雨	适中	正常	无风	P
11	晴	适中	正常	有风	P
12	多云	适中	高	有风	P
13	多云	热	正常	无风	P
14	雨	适中	高	有风	N

训练集

(1) 信息熵的计算

信息熵:$H(U) = -\sum_i P(U_i) log P(U_i)$

类别出现概率：$P(U_i) = \frac{|U_i|}{|S|}$

|S| 表示例子集 S 的总数；

$|U_i|$ 表示类别U_i 的例子数。

对 9 个正例和 5 个反例有：

$P(U_1) = 9/14$, $P(U_1) = 5/14$

H(U)=(9/14)log(14/9)+(5/14)log(14/5)=0.94bit

(2) 条件熵计算

条件熵：$H(U/V) = -\sum_j P(v_j) \sum_i P(u_i/v_i) log P(u_i/v_i)$

属性 A1 取值v_j 时，类别u_i 的条件概率：$P(u_i/v_i) = \frac{|u_i|}{|v_j|}$

A_1= 天气取值 v_1= 晴，v_2= 多云，v_3= 雨

在A_1 处取值晴的例子 5 个, 取值多云的例子 4 个, 取值雨的例子 5 个，故：

P（v_1）=5/14 P（v_2）=4/14 P（v_3）=5/14

取值为晴的 5 个例子中有 2 个正例、3 个反例，故：

P（u_1/v_1）=2/5，P（u_2/v_1）=3/5

同理：P（u_1/v_2）=4/4，P（u_2/v_2）=0 ；P（u_1/v_3）=2/5，P（u_2/v_3）=3/5

H(U/V)=(5/14)((2/5)log(5/2)+(3/5)log(5/3))+(4/14)((4/4)log(4/4)+0) +(5/14)((2/5)log(5/2)+(3/5)log(5/3)) = 0.694bit

(3) 互信息计算

对 “A1= 天气” 有：

I（天气）=H（U）- H（U|V）= 0.94 - 0.694 = 0.246 bit

类似可得：

I（气温）=0.029 bit

I（湿度）=0.151 bit

I（风）=0.048 bit

(4) 建决策树的树根和分枝

选择互信息最大的特征天气作为树根，在 14 个例子中对天气的 3 个取值进行分枝，3 个分枝对应 3 个子集，分别是:

F1={1，2，8，9，11}

F2={3，7，12，13}

F3={4，5，6，10，14}

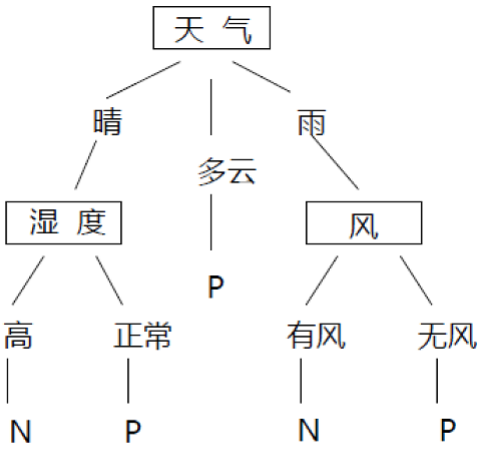
其中 F2 中的例子全属于 P 类，因此对应分枝标记为 P，其余两个子集既含有正例又含有反例，将递归调用建树算法。

(5) 递归建树

分别对 F1 和 F3 子集利用 ID3 算法，在每个子集中对各特征（仍为四个特征）求互信息。

– F1 中的天气全取晴值，则 H（U）=H（U|V），有 I（U|V）=0，在余下三个特征中求出湿度互信息最大，以它为该分枝的根结点，再向下分枝。湿度取高的例子全为 N 类，该分枝标记 N。取值正常的例子全为 P 类，该分枝标记 P。

– 在 F3 中，对四个特征求互信息，得到风特征互信息最大，则以它为该分枝根结点。再向下分枝，风取有风时全为 N 类，该分枝标记 N。取无风时全为 P 类，该分枝标记 P。



ID3决策树

4. 优缺点

优点：ID3 在选择重要特征时利用了互信息的概念，算法的基础理论清晰，使得算法较简单。

缺点：

(1) 互信息的计算依赖于特征取值的数目较多的特征。一种简单的办法是对特征进行分解，化为二值特征。如天气取值晴，多云，雨，分解为三个特征；天气 —晴，天气 —多云，天气 —雨。取值都为 “是” 或 “否”。

(2) 用互信息作为特征选择量存在一个假设，即训练例子集中的正，反例的比例应与实际问题领域里正、反例比例相同。一般情况不能保证相同，这样计算训练集的互信息就有偏差。

(3) ID3 对噪声较为敏感。噪声包含两方面，一是特征值取错，二是类别给错。

(4) 当训练集增加时，ID3 的决策树会随之变化。当训练例子不断增加时会出现一些麻烦。

1.1.4.C4.5 算法

C4.5 是在 ID3 基础上发展起来的决策树生成算法，由 J.R.Quinlan 在 1993 年提出。

- 算法优点

C4.5 克服了 ID3 在应用中存在的不足，主要体现在以下几个方面：

- 用信息增益率来选择属性，它克服了用信息增益选择属性时偏向选择取值多的属性的不足；

在 ID3 中使用信息论中的信息增益（gain）来选择属性，而 C4.5 采用属性的信息增益率（gain ratio）来选择属性。

理论和实验表明，采用 “信息增益率”（C4.5 方法）比采用 “信息增益更好，主要是克服了 ID3 方法选择偏向取值多的属性。

- 决策树构造过程

(1) 类别的信息熵 $H(C) = -\sum_j P(C_j)log(P(C_j))$

(2) 类别条件熵按照属性 V 把集合 T 分割，分割后的类别条件熵为：$H(C|V) = -\sum_j p(v_j)\sum_i p(C_j|v_i)logp(C_j|v_i)$

(3) 信息增益（gain），即互信息 $I(C,V) = H(C) - H(C|V)$

(4) 属性 V 的信息熵 $H(V) = -\sum_i p(v_i)log(p(v_i))$

(5) 信息增益率 gain_ratio=I(C,V)/H(V)

- 在树构造过程中进行剪枝；由于噪声和随机因素的影响，决策树一般会很复杂。因此需要进行剪枝操作。

C4.5 可以对生成好的树剪去某些结点和分枝。剪枝之后的决策树的叶结点不再只包含一类实例。通常是用叶结点替代一个或者多个子树，然后选择出现概率最高的类作为该结点的类别。

- 能够完成对连续属性的离散化处理；采用的方法是二分法

1.1.4.CART 算法

Classification And Regression Tree，即分类回归树算法，简称 CART 算法，它是决策树的另一种实现。既可以是分类树，也可以是回归树。

1. 算法原理

CART 算法是一种二分递归分割技术，把当前样本划分为两个子样本，使得生成的每个非叶子结点都有两个分支，因此 CART 算法生成的决策树是结构简洁的二叉树。由于 CART 算法构成的是一个二叉树，它在每一步的决策时只能是 “是” 或者 “否”，即使一个 feature 有多个取值，也是把数据分为两部分。

那么如何构建这样一颗二叉树呢？如果是分类树，CART 采用 GINI 值衡量节点纯度；如果是回归树，采用样本方差衡量节点纯度。节点越不纯，节点分类的效果就越差。

GINI 值的计算公式为：$GINI = 1 - \sum_{i \in I} p_i^2$

节点越不纯，GINI 值越大。以二分类为例，如果节点的所有数据只有一个类别，则 GINI=0，如果两类数量相同，则 GINI=0.5。

回归方差计算公式：$\sigma = \sqrt{\sum_{i \in I}(x_i - \mu)^2} = \sqrt{\sum_{i \in I} x_i^2 - n\mu^2}$

方差越大，表示该节点的数据越分散，预测的效果就越差。如果一个节点的所有数据都相同，那么方差就为 0，此时可以很肯定得认为该节点的输出值；如果节点的数据相差很大，那么输出的值有很大的可能与实际值相差较大。

因此，无论是分类树还是回归树，CART 都要选择使子节点的 GINI 值或者回归方差最小的属性作为分裂的方案。即最小化（分类树）：$Gain = \sum_{i \in I} p_i * Gini_i$ 或者（回归树）：$Gain = \sum_{i \in I} \sigma_i$

2. 举例：

对于离散型属性，理论上有多少个离散值就应该分裂成多少个节点。但 CART 是一棵二叉树，每一次分裂只会产生两个节点，怎么办呢？很简单，只要将其中一个离散值独立作为一个节点，其他的离散值生成另外一个节点即可。这种分裂方案有多少个离散值就有多少种划分的方法，例如：某离散属性一个有三个离散值 X，Y，Z，则该属性的分裂方法有 {X}、{Y，Z}，{Y}、{X，Z}，{Z}、{X，Y}，分别计算每种划分方法的基尼值或者样本方差确定最优的方法。

以属性 “职业” 为例，一共有三个离散值，“学生”、“老师”、“上班族”。该属性有三种划分的方案，分别为 {“学生”}、{“老师”、“上班族”}，{“老师”}、{“学生”、“上班族”}，{“上班族”}、{“学生”、“老师”}，分别计算三种划分方案的子节点 GINI 值或者样本方差，选择最优的划分方法：

- 第一种划分方法：{“学生”}、{“老师”、“上班族”}

看电视时间	是否已婚	职业	年龄
3	否	学生	12
4	否	学生	18
2	是	老师	26
5	是	上班族	47
2.5	是	上班族	36
3.5	否	老师	29
4	是	学生	21

看电视时间	是否已婚	职业	年龄
2	是	老师	26
3.5	否	老师	29

看电视时间	是否已婚	职业	年龄
3	否	学生	12
4	否	学生	18
5	是	上班族	47
2.5	是	上班族	36
4	是	学生	21

预测是否已婚（分类）：

$$Gain = \sum_{i \in I} p_i \cdot Gini_i = \frac{3}{7} \cdot (1 - ((\frac{2}{3})^2 + (\frac{1}{3})^2)) + \frac{4}{7} \cdot (1 - ((\frac{3}{4})^2 + (\frac{1}{4})^2)) = 0.4$$

预测年龄（回归）：

$$Gain = \sum_{i \in I} \sigma_i = \sqrt{12^2 + 18^2 + 21^2 - 3 \cdot 17^2} + \sqrt{26^2 + 47^2 + 36^2 + 29^2 - 4 \cdot 32.5^2} = 34.71$$

- 第二种划分方法：{“老师”}、{“学生”、“上班族”}

看电视时间	是否已婚	职业	年龄
3	否	学生	12
4	否	学生	18
2	是	老师	26
5	是	上班族	47
2.5	是	上班族	36
3.5	否	老师	29
4	是	学生	21

看电视时间	是否已婚	职业	年龄
2	是	老师	26
3.5	否	老师	29

看电视时间	是否已婚	职业	年龄
3	否	学生	12
4	否	学生	18
5	是	上班族	47
2.5	是	上班族	36
4	是	学生	21

预测是否已婚（分类）：

$$Gain = \sum_{i \in I} p_i \cdot Gini_i = \frac{2}{7} \cdot (1 - ((\frac{1}{2})^2 + (\frac{1}{2})^2)) + \frac{5}{7} \cdot (1 - ((\frac{3}{5})^2 + (\frac{2}{5})^2)) = 0.49$$

预测年龄（回归）：

$$Gain = \sum_{i \in I} \sigma_i = \sqrt{26^2 + 29^2 - 2 \cdot 27.5^2} + \sqrt{12^2 + 18^2 + 47^2 + 36^2 + 21^2 - 5 \cdot 26.8^2} = 16.36$$

- 第三种划分方法：{“上班族”}、{“学生”、“老师”}

看电视时间	是否已婚	职业	年龄
3	否	学生	12
4	否	学生	18
2	是	老师	26
5	是	上班族	47
2.5	是	上班族	36
3.5	否	老师	29
4	是	学生	21

看电视时间	是否已婚	职业	年龄
5	是	上班族	47
2.5	是	上班族	36

看电视时间	是否已婚	职业	年龄
3	否	学生	12
4	否	学生	18
2	是	老师	26
3.5	否	老师	29
4	是	学生	21

预测是否已婚（分类）：

$$Gain = \sum_{i \in I} p_i \cdot Gini_i = \frac{2}{7} \cdot (1-1) + \frac{5}{7} \cdot (1 - ((\frac{3}{5})^2 + (\frac{2}{5})^2)) = 0.34$$

预测年龄（回归）：

$$Gain = \sum_{i \in I} \sigma_i = \sqrt{47^2 + 36^2 - 2 \cdot 41.5^2} + \sqrt{12^2 + 18^2 + 26^2 + 29^2 + 21^2 - 5 \cdot 21.2^2} = 21.14$$

1.2 应用

1.2.1 rpart、party 包：

基于 CART 算法的实现：

1. 用 party 包做分类树

数据采用的是 R 中自带的鸢尾花数据集（iris），属性包括 Sepal.Length（萼片长度）、Sepal.Width（萼片宽度）、Petal.Length（花瓣长度）以及 Petal.Width（花瓣宽度），我们用这四个属性来预测鸢尾花的 Species（种类）。

在建立模型之前，iris 数据集被分为两个子集：训练集（70%）和测试集（30%）。用随机种子设置固定的随机数，使其结果可以再现。在这个包中，我们用 ctree() 函数建立一个决策树，predict() 函数来预测测试集。

使用 ctree 函数的基本语法为：ctree(formula, data)，其中，formula 是一个公式描述的预测和响应变量。data 是所使用的数据集的名称。

```
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
```

```
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
set.seed(1234)
# 使用 sample 函数抽取样本，将数据集中观测值分为两个子集
ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.7, 0.3))
# 选取一部分样本作为训练集
trainData <- iris[ind==1,]
# 另外一部分作为测试集
testData <- iris[ind==2,]
library(party)
```

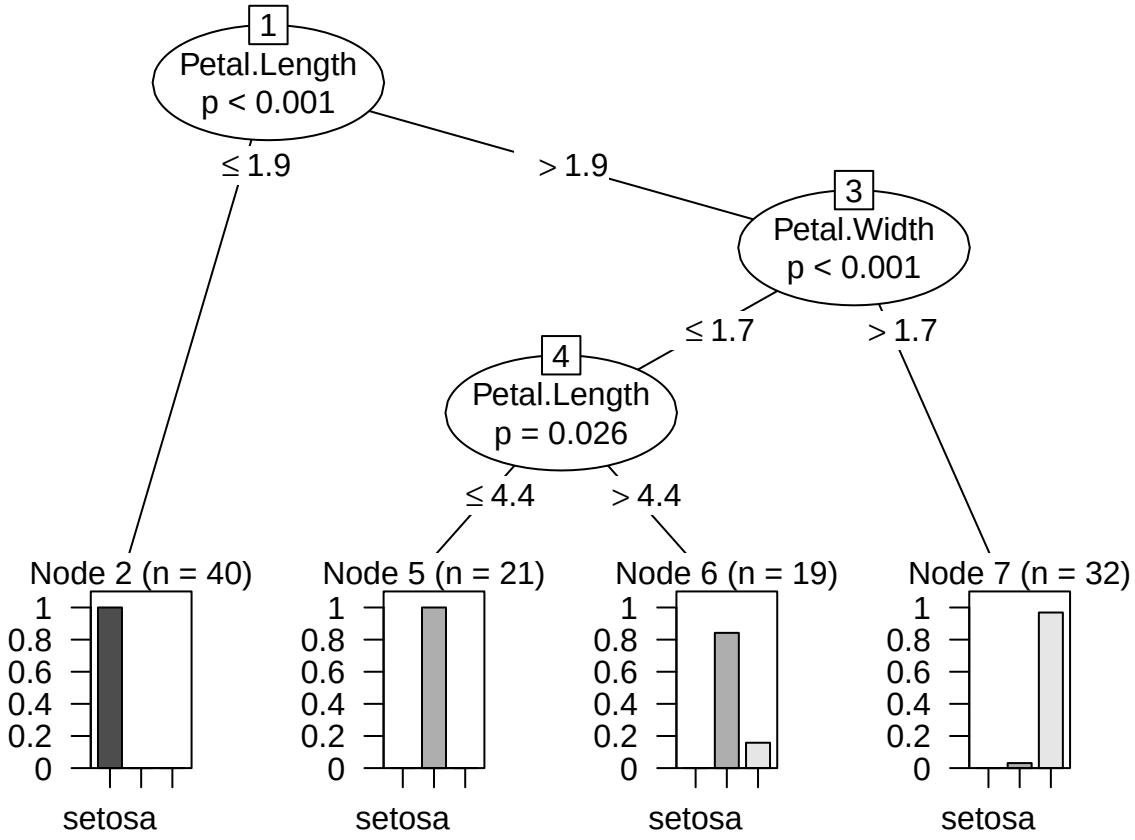
```
## Loading required package: grid
## Loading required package: mvtnorm
## Loading required package: modeltools
## Loading required package: stats4
## Loading required package: strucchange
## Loading required package: zoo
##
## Attaching package: 'zoo'
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
## Loading required package: sandwich
```

```
# 找到自变量和因变量
myFormula <- Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width
# 建立决策树
iris_ctree <- ctree(myFormula, data=trainData)
# 检测预测值
table(predict(iris_ctree), trainData$Species)
```

```
##
##              setosa versicolor virginica
##   setosa         40          0         0
##   versicolor      0         37         3
##   virginica       0          1        31
```

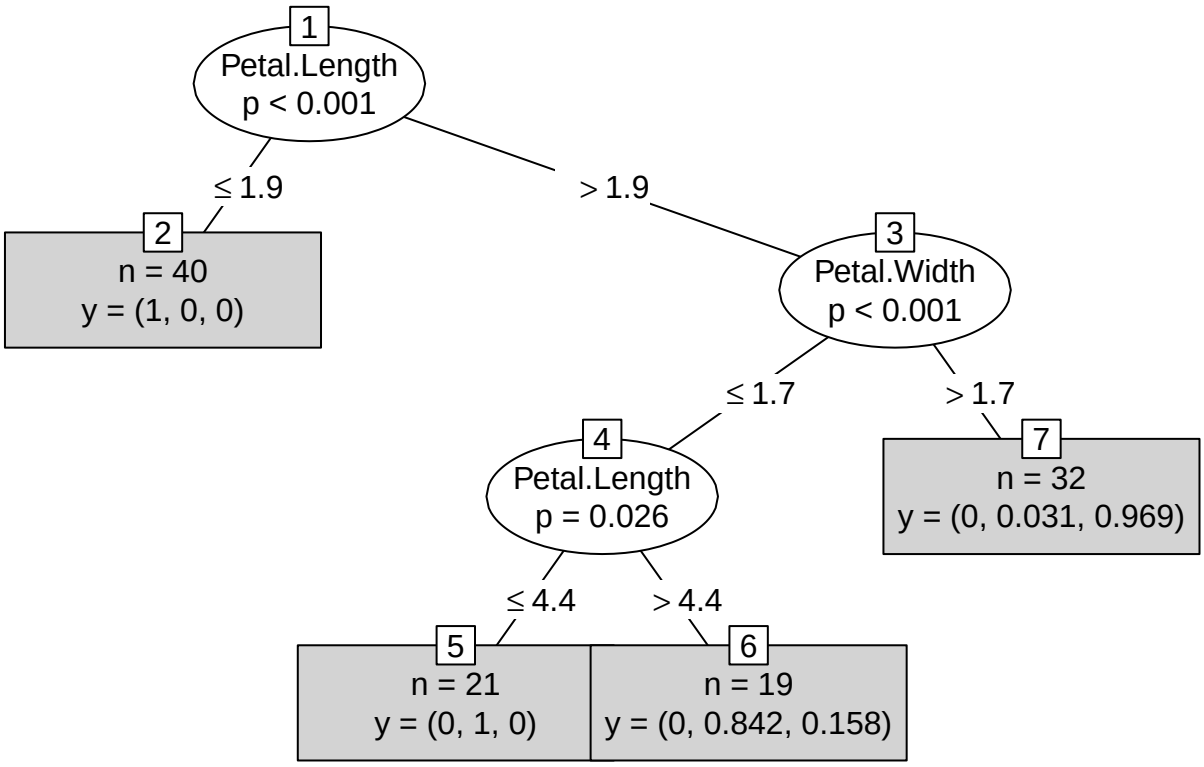
由上表可知，setosa（山鸢尾）40 观测值全部正确预测，而 versicolor（变色鸢尾）有一个观测值被误判为 virginica（维吉尼亚鸢尾），virginica（维吉尼亚鸢尾）有 3 个观测值被误判为 versicolor（变色鸢尾）。

```
# 输出决策树图，其中每一个叶子的节点的条形图都显示了观测值落入三个品种的概率。
plot(iris_ctree)
```




```
# 简化输出，y 值表示每个叶子结点中的不同种类的出现概率。例如：结点 2 里面的标签是“n=40 y=(1,0,0)”，指的是这

plot(iris_ctree, type="simple")
```




```
# 接下来，使用测试集测试决策树的预测效果。
testPred <- predict(iris_ctree, newdata = testData)
table(testPred, testData$Species)
```

```
##
## testPred     setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0         12         2
##   virginica       0          0        14
```

2. 用 rpart 包做回归树

通常默认 anova 用来作回归树，以汽车消费数据 car90 为例, 该数据包括 34 个变量 110 条观察值。

```
library(rpart)
library(maptree)
```

```
## Loading required package: cluster
```

```
data(car90)
# 剔除轮胎尺寸、型号等 3 个因素型变量 (factor variable): "Rim", "Tires", "Model2"
cars = car90[, -match(c("Rim", "Tires", "Model2"), names(car90))]
# 建立回归树模型
```

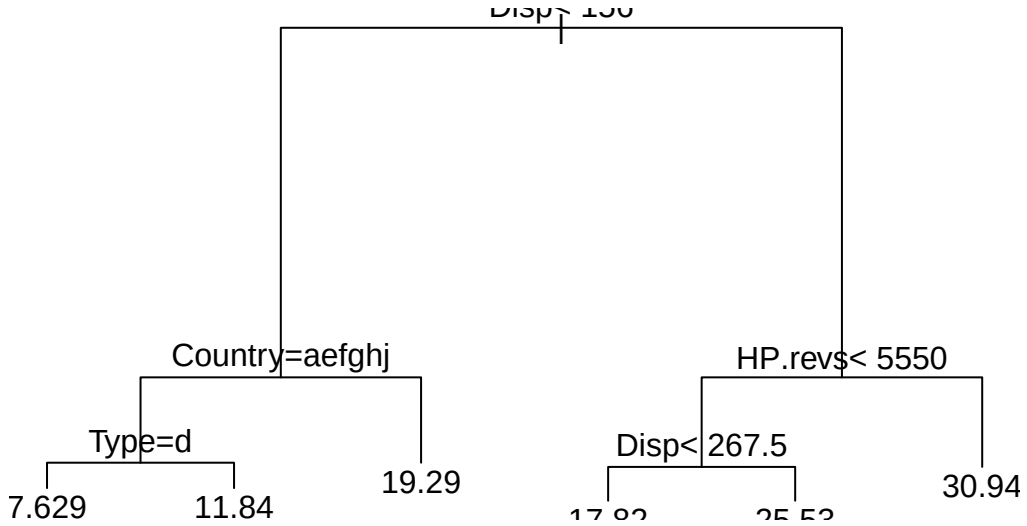
```
carfit = rpart(Price/1000 ~ ., data=cars)
carfit
```

```
## n=105 (6 observations deleted due to missingness)
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 105 7118.26700 15.805220
##    2) Disp< 156 70 1491.62800 11.855870
##      4) Country=Brazil,Japan,Japan/USA,Korea,Mexico,USA 58  421.21470 10.318470
##        8) Type=Small 21   50.30983  7.629048 *
##        9) Type=Compact,Medium,Sporty,Van 37  132.80330 11.844890 *
##      5) Country=France,Germany,Sweden 12  270.72330 19.286670 *
##    3) Disp>=156 35 2351.19600 23.703910
##      6) HP.revs< 5550 24  980.27790 20.388710
##       12) Disp< 267.5 16  395.99670 17.820060 *
##       13) Disp>=267.5 8  267.58000 25.526000 *
##      7) HP.revs>=5550 11  531.63680 30.937090 *
```

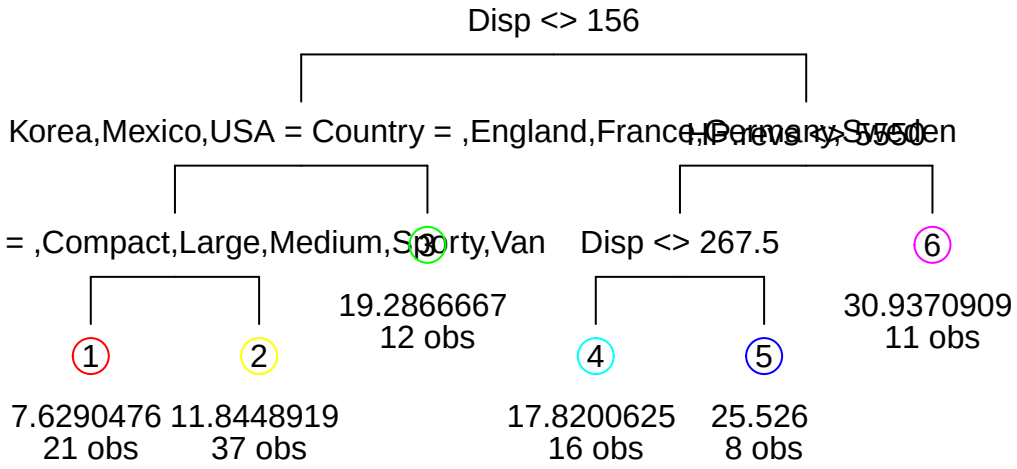
```
printcp(carfit)
```

```
##
## Regression tree:
## rpart(formula = Price/1000 ~ ., data = cars)
##
## Variables actually used in tree construction:
## [1] Country Disp    HP.revs Type
##
## Root node error: 7118.3/105 = 67.793
##
## n=105 (6 observations deleted due to missingness)
##
##         CP nsplit rel error  xerror     xstd
## 1 0.460146      0   1.00000 1.03166 0.166225
## 2 0.117905      1   0.53985 0.71005 0.108712
## 3 0.112343      2   0.42195 0.65722 0.101208
## 4 0.044491      3   0.30961 0.48894 0.090147
## 5 0.033449      4   0.26511 0.47565 0.087640
## 6 0.010000      5   0.23166 0.46342 0.105544
```

```
plot(carfit)
text(carfit)
```




```
# 让图片好看一点
draw.tree(carfit)
```



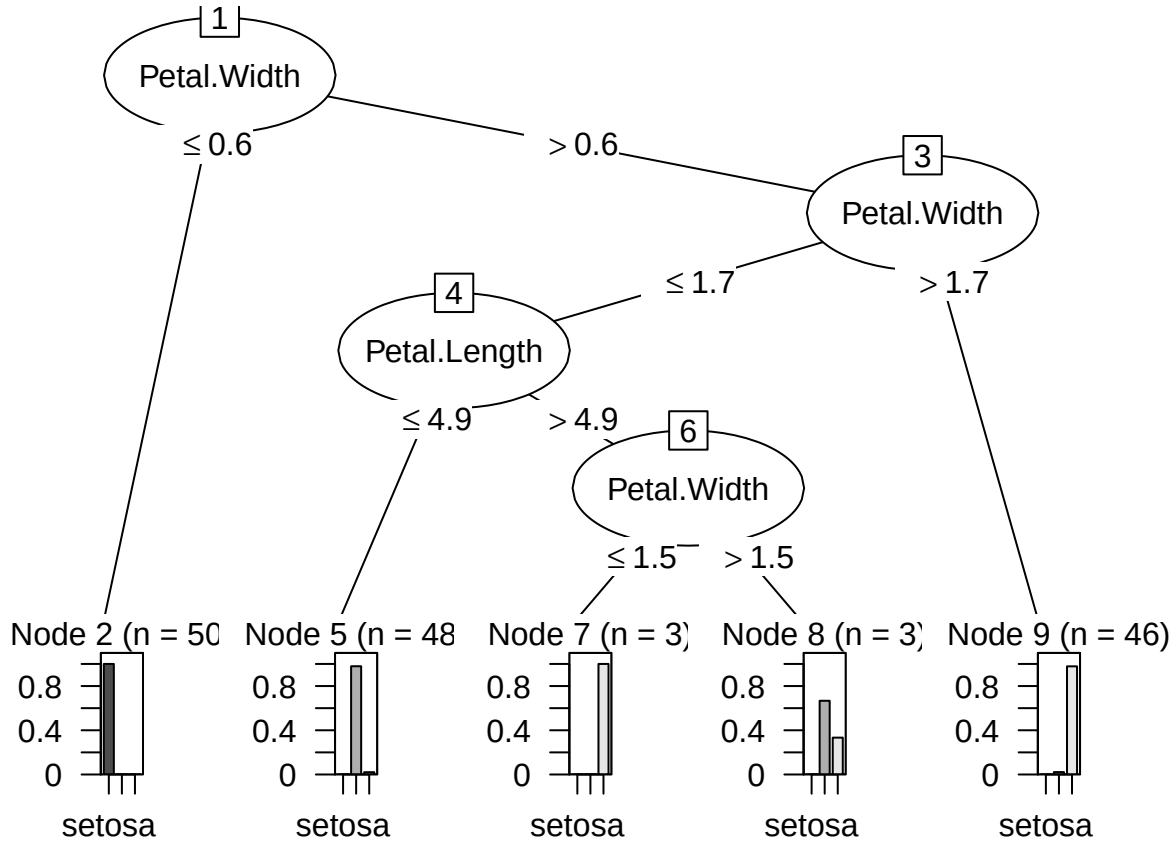
1.2.2 rweka 包

J48() 是基于 C4.5 算法的实现：

Weka 里有非常全面的机器学习算法，包括数据预处理、分类、回归、聚类、关联规则等。

```
library(RWeka)

data(iris)
m1<-J48(Species~.,data=iris)
plot(m1)
```




```
#m1
```

1.3 练习

用 solder 数据做 Possion 回归树：

```
head(solder)
str(solder)
# 建立 poisson 回归树
sfit = rpart(skips ~ Opening + Solder + Mask + PadType + Panel,data = solder, method = 'poisson',contro
sfit
printcp(sfit)
summary(sfit,cp=.1)
plot(sfit)
text(sfit)
draw.tree(sfit)
table(iris$Species,predict(m1))
```

1.4 延伸阅读

决策树是一个弱分类器，集成学习的方法可以将弱学习器组合在一起，根据投票法得到强学习器。感兴趣的童鞋可以进一步学习 AdaBoost，Bagging, 以及 random forest。具体建议参考书籍:《机器学习》（周志华）。

2. 线形模型

2.1 理论

2.1.1 基本形式

例如用 d 个属性描述示例$x = (x_1, x_2, ..., x_d)$ 其中，x_i 是 x 在第 i 个属性上的取值。

线性模型（linear model) 就是试图用一个线性组合来描述：

$f(x) = w_1x_1 + w_2x_2 + ... + w_dx_d + b$

我们在其他很多的课程中也接触过用线性模型去近似非线性模型（nonlinear model）。因为线性模型的较好的可解释性（comprehensibility）。

下面介绍几种经典的线性模型 ——回归任务，二分类以及多分类。

2.1.2 线形回归

“线性回归”（linear regression）试图学得一个线性模型以尽可能准确地预测实值输出标记线性回归试图学习得到

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

通常，我们用均方误差来衡量 f(x) 和 y 之间的差别。均方误差是回归中最常用的应用了性能指标，均方误差的几何意义就是欧氏距离。（最小二乘法的几何意义就是试图找到一条直线，让所有样本到直线的距离之和最小）。

线形模型虽然简单，但是却有丰富的变化，例如，对于输出标记是在指数尺度上变化时，可以用输出标记的对数作为线性模型逼近的目标，即：$ln(y) = w^Tx + b$ 这就是对数线性回归，但实质上是在求取输入空间到输出空间的非线性映射。

更一般地，考虑单调可微函数 g()，令$y = g^{-1}(w^tx + b)$，这个模型就是 GLM（广义线性）模型，其中$g(\cdot)$ 称为联系函数（link function）。显然对数线性模型就是广义线性模型在$g(\cdot) = ln(\cdot)$ 时的特例。

2.1.3 对数几率回归（Logistic Regression）

广义线形模型在二分类问题中的应用。

很多时候我们需要作的是把一堆东西进行二分类，而非进行回归，而回归产生的函数是$z = w^Tx + b$，所以我们希望将 z 转化为 0/1 值，最为理想的就是单位阶跃函数。

$$y=\begin{cases}0, & z<0;\\ 0.5, & z=0;\\ 1, & z>0,\end{cases}$$

然而单位阶跃函数并不是连续的，故而我们需要构造一个替代函数，常用的是$y=\frac{1}{1+e^{-z}}$，其中$z=w^Tx+b$。

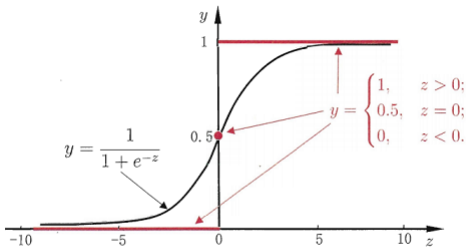


图 3.2 单位阶跃函数与对数几率函数

可以推出$ln\frac{y}{1-y}=w^Tx+b$，y 可以看作正例的可能性，1-y 看作反例的可能性。两者的比值$\frac{y}{1-y}$ 成为几率（odds），反映了 x 作为正例的相对可能性，对几率取对数，得到对数几率（logit）$ln\frac{y}{1-y}$

对应模型就为 “对数几率回归”，注意，虽然它的名字是回归，但却是一种分类学习的方法。这种方法有很多优点，它不仅可预测出类别，也可以得到近似概率预测。这对于很多需要概率辅助决策的任务很有用。通常用极大似然估计的方法结合梯度下降法和牛顿法可以找出参数的最优解。

2.1.4 线形判别分析（LDA）

LDA 是一种经典分类方法，也叫做 Fisher 判别分析，与 PCA 思想类似，但目的不同：PCA 是找一个低维的子空间，样本投影在这个空间基本不丢失信息。而 fisher 是寻找这样的一个空间，样本投影在这个空间上，类内距离最小，类间距离最大。

LDA 思想：把样例投影到一条直线，使得同样例尽可能靠近，异类尽可能远离。二维情况如图所示：

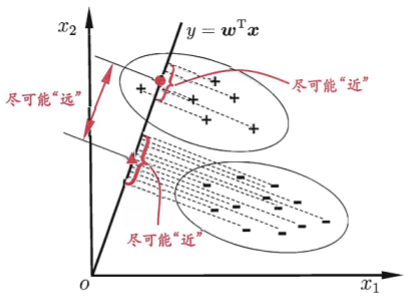


图 3.3 LDA 的二维示意图. “+”、“–” 分别代表正例和反例, 椭圆表示数据簇的外轮廓, 虚线表示投影, 红色实心圆和实心三角形分别表示两类样本投影后的中心点.

这里就会遇到一个问题 ——怎么来衡量这些投影点的 “靠近” 与 “远离” 程度呢？

要使得同类投影点的协方差尽可能小, 即$w^T\Sigma_0w+w^T\Sigma_1w$

要使得异类投影点距离尽可能大，即$||w^T\mu_0-w^T\mu_1||_2^2$，其中$X_i,\mu_i,\Sigma_i$ 分别表示 i 类示例的集合，均值向量和协方差矩阵。因此，只要使得

$J = \frac{||w^T\mu_0 - w^T\mu_1||_2^2}{w^T\Sigma_0 w + w^T\Sigma_1 w}$

尽可能大就好，求解出对应的 w 就好。

LDA 推广到多类的方法：若要分为 m 类，则投影到 m-1 维的空间，利用构造的 “散度” 作为评判准则再求解 w。

值得一提的是，LDA 可从贝叶斯决策的角度来解释，可以证明，当两类数据同先验，满足高斯分布并且协方差相等时，LDA 可以达到最优分类。

2.2 应用

2.2.1 glm() 函数

logistic Regression

经典的 LR 模型只能做二分类，但有一些从 LR 中衍生出来的方法也可以解决多分类问题，具体如下：

1. 普通二分类 logistic 回归用系统的 glm

2. 因变量多分类 logistic 回归

a. 有序分类因变量：用 MASS 包里的 polr b. 无序分类因变量：用 nnet 包里的 multinom

3. 条件 logistic 回归用 survival 包里的 clogit

```
# 这里举一个基础的例子
# 由于 LR 是一种经典的二分类算法，先将三种类别的数据取出两类
data<-iris[51:150,]
names(data)<-c('sl','sw','pl','pw','species')
logit.fit<-glm(data$species~sw+pw,binomial(link='logit'),data=data)
logit.pred <- ifelse(predict(logit.fit) > 0,"virginica","versicolor")
table(data$species,logit.pred)
```

```
##             logit.pred
##              versicolor virginica
##   setosa              0         0
##   versicolor         46         4
##   virginica           3        47
```

2.3 延伸阅读

多分类 Logistic 回归,Lasso Logistic 回归 QDA

2.2.2 MASS 包中的 lda() 函数

```r
library(MASS)
data(iris)
iris.lda=lda(Species~.,data=iris)
iris.lda
```

```
## Call:
## lda(Species ~ ., data = iris)
##
## Prior probabilities of groups:
##     setosa versicolor  virginica
##  0.3333333  0.3333333  0.3333333
##
## Group means:
##            Sepal.Length Sepal.Width Petal.Length Petal.Width
## setosa            5.006       3.428        1.462       0.246
## versicolor        5.936       2.770        4.260       1.326
## virginica         6.588       2.974        5.552       2.026
##
## Coefficients of linear discriminants:
##                     LD1         LD2
## Sepal.Length  0.8293776  0.02410215
## Sepal.Width   1.5344731  2.16452123
## Petal.Length -2.2012117 -0.93192121
## Petal.Width  -2.8104603  2.83918785
##
## Proportion of trace:
##    LD1    LD2
## 0.9912 0.0088
```

```r
table<-table(iris$Species,predict(iris.lda,iris)$class)
table
```

```
##
##              setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         48         2
##   virginica       0          1        49
```

```r
### 判对率
sum(diag(prop.table(table)))
```

```
## [1] 0.98
```

3. 支持向量机 (SVM)

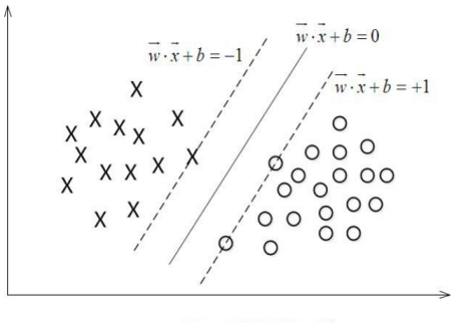
SVM 与 LR 回归的联系？

3.1 理论

3.1.1 基本原理

举个简单的例子。现在有一个二维平面，平面上有两种不同的数据，分别用圈和叉表示。由于这些数据是线性可分的，所以可以用一条直线将这两类数据分开，这条直线就相当于一个超平面，超平面一边的数据点所对应的 y 全是 -1，另一边所对应的 y 全是 1。

这个超平面可以用分类函数$f(x) = w^T x + b$ 表示，当 f(x) 等于 0 时，x 是位于超平面上的点，而 f(x) 大于 0 的点对应 y=1 的数据点，f(x) 小于 0 的点对应 y=-1 的点，如下图所示：



换言之，在进行分类的时候，遇到一个新的数据点 x，将 x 代入 f(x) 中，如果 f(x) 小于 0 则将 x 的类别赋为 -1，如果 f(x) 大于 0 则将 x 的类别赋为 1。

接下来的问题是，如何确定这个超平面呢？从直观上而言，这个超平面应该是最适合分开两类数据的直线。而判定“最适合”的标准就是这条直线离直线两边的数据的间隔最大。所以，得寻找有着最大间隔的超平面。

3.1.2 函数间隔和几何间隔

在超平面$w^T x+b=0$ 确定的情况下，$|w^T x+b|$ 能够表示点 x 到距离超平面的远近，而通过观察$w^T x+b$ 的符号与类标记 y 的符号是否一致可判断分类是否正确，所以，可以用$(y*(w^T x+b))$ 的正负性来判定或表示分类的正确性。于此，我们便引出了函数间隔（functional margin）的概念。

定义函数间隔（用$\hat{\gamma}$ 表示）为：

$$\hat{\gamma} = y(w^T x + b) = yf(x)$$

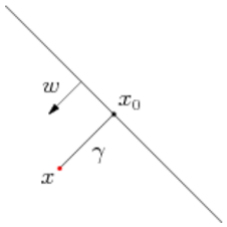
而超平面(w, b) 关于 T 中所有样本点(x_i, y_i) 的函数间隔最小值（其中，x 是特征，y 是结果标签，i 表示第 i 个样本），便为超平面(w, b) 关于训练数据集 T 的函数间隔：

$$\hat{\gamma} = min\hat{\gamma}_i i = 1, ...n$$

但这样定义的函数间隔有问题，即如果成比例的改变 w 和 b（如将它们改成 2w 和 2b），则函数间隔的值 f(x) 却变成了原来的 2 倍（虽然此时超平面没有改变），所以只有函数间隔还远远不够。

事实上，我们可以对法向量 w 加些约束条件，从而引出真正定义点到超平面的距离 –几何间隔（geometrical margin）的概念。

假定对于一个点 x，令其垂直投影到超平面上的对应点为x_0，w 是垂直于超平面的一个向量，γ 为样本 x 到超平面的距离，如下图所示：



根据平面几何知识，有:

$$x = x_0 + \gamma \frac{w}{||w||}$$

其中$||w||$ 为w 的二阶范数（范数是一个类似于模的表示长度的概念），$\frac{w}{||w||}$ 是单位向量（一个向量除以它的模称之为单位向量）。

又由于x_0 是超平面上的点，满足$f(x_0) = 0$, 代入超平面的方程$w^T x + b = 0$，可得$w^T x_0 + b = 0$，即$w^T x_0 = -b$。

随即令$x = x_0 + \gamma \frac{w}{||w||}$ 的两边同时乘以w^T，得出：

$$\gamma = \frac{w^T x + b}{||w||} = \frac{f(x)}{||w||}$$

为了得到γ 的绝对值，令γ 乘上对应的类别 y，即可得出几何间隔（用$\tilde{\gamma}$ 表示）的定义：

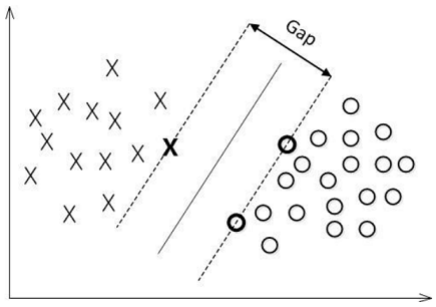
$$\tilde{\gamma} = y\gamma = \frac{\hat{\gamma}}{||w||}$$

从上述函数间隔和几何间隔的定义可以看出：几何间隔就是函数间隔除以$||w||$，而且函数间隔实际上就是$|f(x)|$，只是人为定义的一个间隔度量，而几何间隔才是直观上的点到超平面的距离。

3.1.3 最大间隔分类器

对一个数据点进行分类，当超平面离数据点的 “间隔” 越大，分类的确信度（confidence）也越大。所以，为了使得分类的确信度尽量高，需要让所选择的超平面能够最大化这个 “间隔” 值。这个间隔就是下图中的 Gap 的一半。

这里要找的最大间隔分类超平面中的 “间隔” 指的是几何间隔。



于是最大间隔分类器（maximum margin classifier）的目标函数可以定义为：

$$max\tilde{\gamma}$$

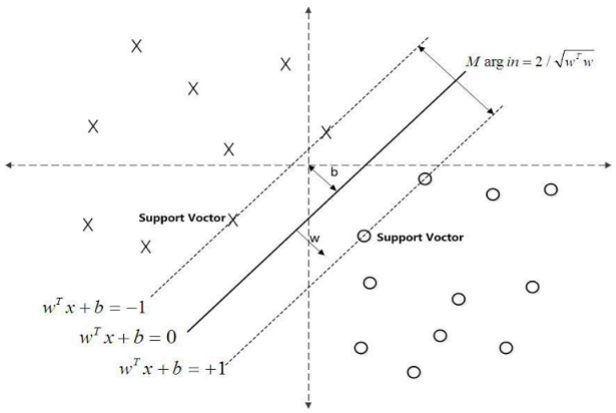
约束条件为：

$$y_i(w^T x_i + b) = \hat{\gamma}_i \geq \hat{\gamma} i = 1, ..., n$$

令函数间隔$\tilde{\gamma} = 1$，对优化结果没有影响，则优化问题转换为：

$$max \frac{1}{||w||}, s.t., y_i(w^T x_i + b) \geq 1, i = 1, ..., n$$

相当于在上述约束条件下最大化$\frac{1}{||w||}$，而$\frac{1}{||w||}$ 便是几何间隔$\tilde{\gamma}$。

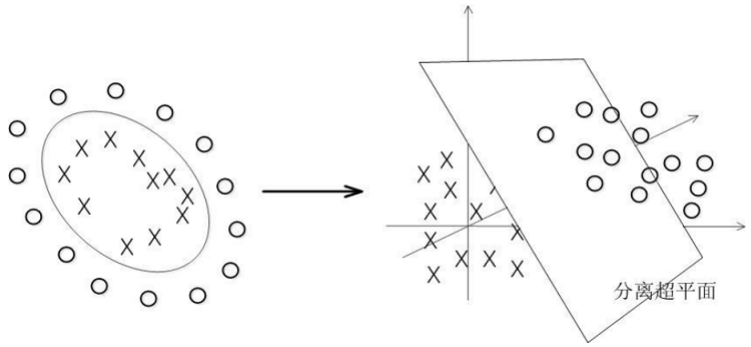


3.1.4 线形不可分 (Kernel 核函数)

到目前为止，我们的 SVM 还比较弱，只能处理线性的情况，下面我们将引入核函数，进而推广到非线性分类问题。

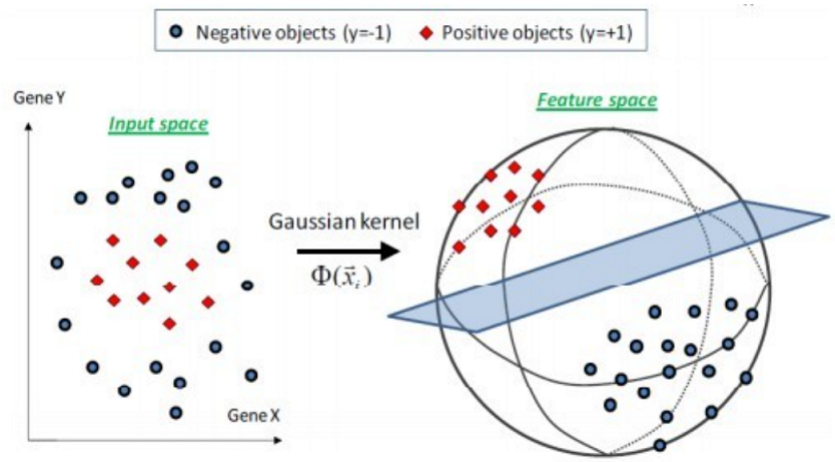
大部分时候数据并不是线性可分的，这个时候满足这样条件的超平面就根本不存在。在上文中，我们已经了解到了 SVM 处理线性可分的情况，那对于非线性的数据 SVM 咋处理呢？对于非线性的情况，SVM 的处理方法是选择一个核函数 (,)，通过将数据映射到高维空间，来解决在原始空间中线性不可分的问题。

具体来说，在线性不可分的情况下，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。如所示，一堆数据在二维空间无法划分，从而映射到三维空间里划分：



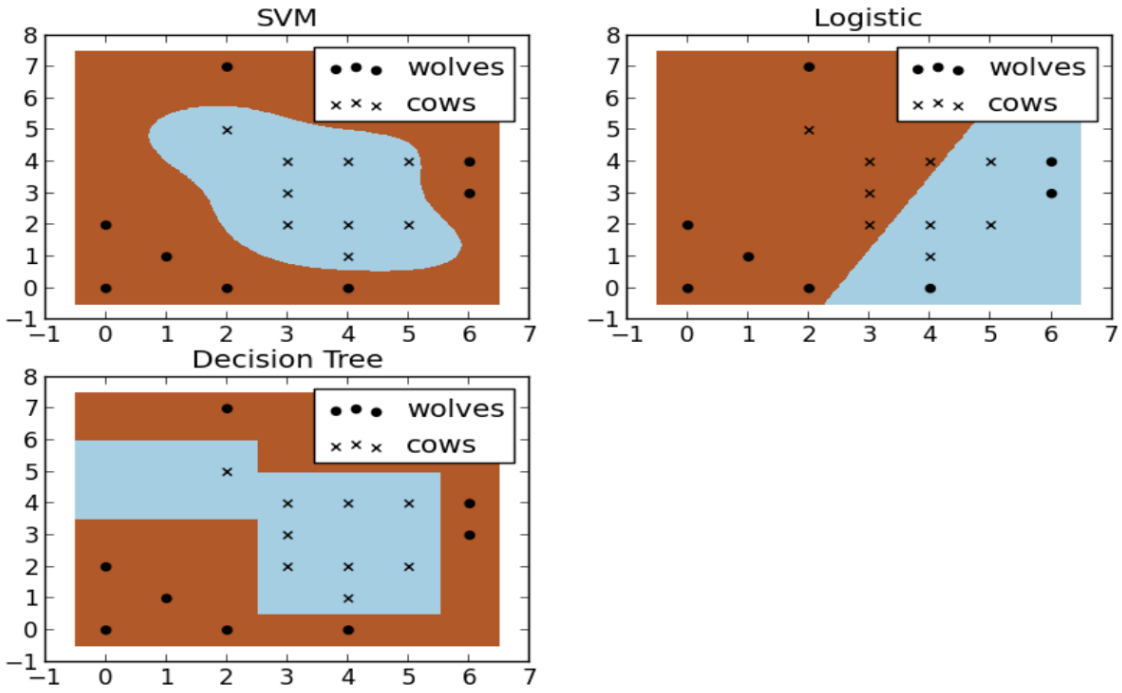
几个核函数:

（1）多项式核函数（2）高斯核函数（3）线形核函数



引用一个例子举例说明下核函数解决非线性问题的直观效果。

假设现在你是一个农场主，养了一批羊群，但为预防狼群袭击羊群，如何搭建一个篱笆来把羊群和狼分开呢？

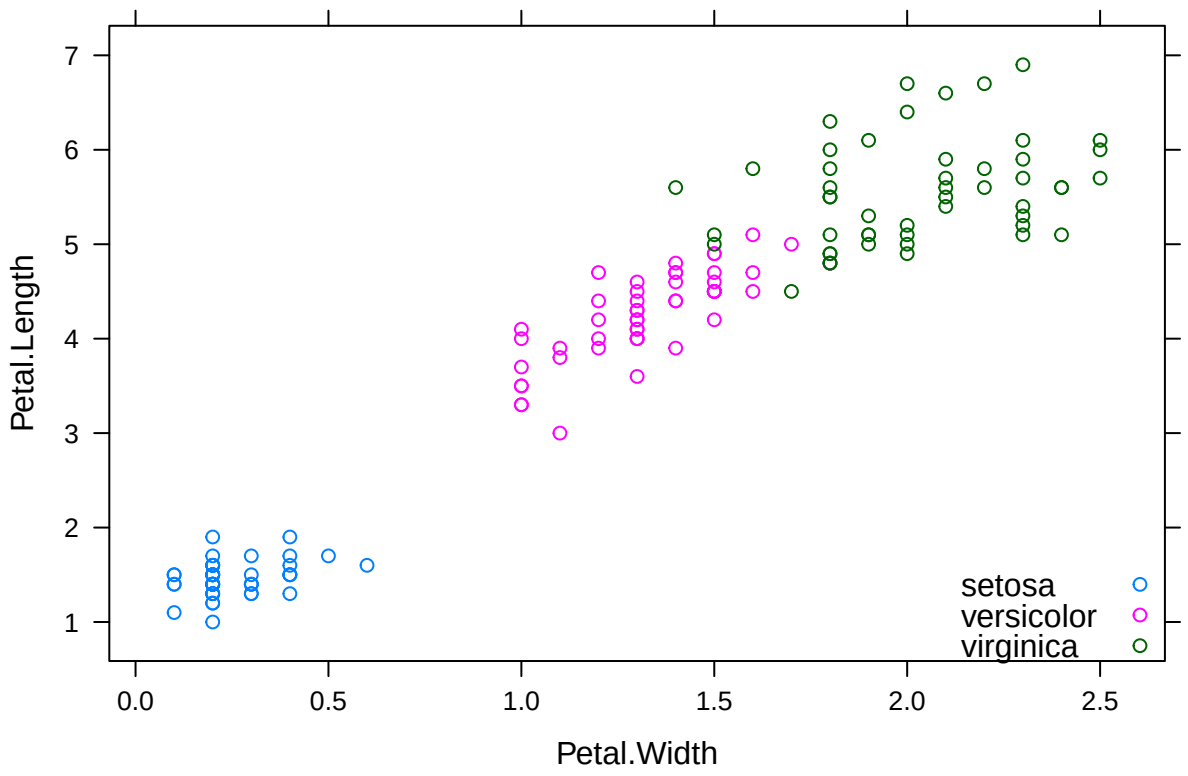


这个例子从侧面简单说明了 SVM 使用非线性分类器的优势，而逻辑模式以及决策树模式都是使用了直线方法。

3.2 应用

用 e1071 程序包所提供 svm() 函数来实现：

```
library(lattice)
library(e1071)
data(iris)
xyplot(Petal.Length~Petal.Width,data=iris,groups=Species,auto.key=list(corner=c(1,0)))
```




```
# 标记为 setosa 的鸢尾花可以很容易地被划分出来。但仅使用 Petal.Length 和 Petal.Width 这两个特征时，versico
subdata<-iris[iris$Species!='virginica',]
subdata$Species<-factor(subdata$Species)
iris.svm<-svm(Species~Petal.Length+Petal.Width,data=subdata)
summary(iris.svm)
```

```
##
## Call:
## svm(formula = Species ~ Petal.Length + Petal.Width, data = subdata)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.5
##
## Number of Support Vectors:  7
##
##  ( 3 4 )
```

```
##
##
## Number of Classes:  2
##
## Levels:
##  setosa versicolor
```

```
plot(iris.svm,subdata,Petal.Length~Petal.Width)
```


```
pred <- predict(iris.svm,subdata[,1:4])
table(pred,subdata$Species)
```

```
##
## pred         setosa versicolor
##   setosa         50          0
##   versicolor      0         50
```

参考文献：http://blog.csdn.net/v_july_v/article/details/7624837