摘要：R分类判别函数

```
> myfit = classifier(y~x, data=data.train)
> predict(myfit, newdata=data.test)

# 这里classifier是任何分类器，比如
  两类(线性)logistic回归：glm(y~x, family=binomial),
  多类(线性)logisitc回归：multinom()
  非线性logistic回归/单层神经网络：nnet()
# 深度学习R包：keras、tensorflow、deepnet
```

1. 判别分类问题概述

分类 classification 或判别 discriminate 问题以自变量 (variable) 或特征 (feature) $\mathbf{x} \in R^p$ 预测其类别标签 $y \in \{1,...,K\}$。假设训练数据为

$$(\mathbf{x}_i, y_i), i = 1, 2, ..., n$$

其中 $\mathbf{x}_i, y_i$ 分别是第 i 个训练样本的特征和类别。对于两类问题 ($K = 2$), 对于任一特征 $\mathbf{x}$ （其对应的类别标号为 $y = 1$ 或 2）, 我们希望得到判别函数 $b(\mathbf{x})$（也成为分类器，classifier）

$$\text{当 } b(\mathbf{x}) > C \quad \text{预测/判定} y = 1.$$

仅考虑两类判别。我们先应用贝叶斯方法推导出一般的 logistic 回归分类判别模型。假设随机向量 $\mathbf{x}$, 其对应的类别标签 $y = 0, 1$, 假设

$$\mathbf{x}|y = k \sim f_k(\mathbf{x}),\ k = 0, 1$$

记总体概率 $p = P(y = 1)$, 记 $b(\mathbf{x}) = \log(f_1(\mathbf{x})/f_0(\mathbf{x})), a = \log(p/(1-p))$, 则由贝叶斯公式得到如下（一般形式的）logistic 回归判别/分类模型或贝叶斯判别：

$$P(y = 1|\mathbf{x}) = \frac{pf_1(\mathbf{x})}{pf_1(\mathbf{x}) + (1-p)f_0(\mathbf{x})} = \frac{\exp(a + b(\mathbf{x}))}{1 + \exp(a + b(\mathbf{x})}, \tag{1}$$

当 $P(y = 1|\mathbf{x})$ 较大（比如大于 0.5）或等价地，当 $b(\mathbf{x})$ 较大时，我们预测 y 等于 1.

类似地，对于多类问题 (各类概率分布分别是 $f_1, ..., f_K$)

$$P(y = k) = \frac{\exp(a_k + b_k(\mathbf{x}))}{1 + \exp(a_k + b_k(\mathbf{x})}, b_k(\mathbf{x}) = \log\{f_k(\mathbf{x})/f_1(\mathbf{x})\}$$

其中 $a_1 = 0, b_k(\mathbf{x}) = 0$，这称为多项 logistic 回归模型 (multinomial logistic). 对于给定的 feature $\mathbf{x}$, 判别准则如下

$$\text{若 } k_0 = argmax_{k=1,...,K} P(y = k|\mathbf{x}), \text{ 则预测 } y = k。$$

下面以二类问题为例，介绍线性 logistic 判别、非线性（分段线性）logistic 判别（神经网络）。

2. 线性 logistic 回归：线性判别

若模型 (1) 中的 $b(\mathbf{x})$ 是线性的：$b(\mathbf{x}) = \mathbf{b}^\top \mathbf{x}$，即是通常的 (线性) logistic 回归模型:

$$P(y=1|\mathbf{x}) = \frac{\exp(a+\mathbf{b}^\top \mathbf{x})}{1+\exp(a+\mathbf{b}^\top \mathbf{x})}$$

当 $P(y=1|\mathbf{x}) > C$ 即 $\mathbf{b}^\top \mathbf{x} > C'$ 时，我们预测 $y=1$。所以基于该模型的判别是线性判别。容易验证当 f_1, f_0 是方差相同的多元正态分布时，$b(\mathbf{x}) = \log(f_1(\mathbf{x})/f_0(\mathbf{x}))$ 确实关于 $\mathbf{x}$ 线性，而正态假设下我们有 Fisher 线性判别方法，此时 logistic 回归和 Fisher 判别方法接近但不完全相同，这是因为 logistic 回归的计算不依赖于正态假设而 Fisher 判别依赖于正态假设。

另外，如果 $b(\mathbf{x})$ 是二次函数，则称为是二次判别。线性 logistic 回归的 R 函数为glm, Fisher 线性判别和 Fisher 二次判别的 R 函数为 lda, qda (in MASS)。更多细节参见课件第 21 讲。

```
#线性logistic 回归 （指数上关于x线性）
myfit=glm(Country~Marine+Freshwater,data=training.data,
         family=binomial)
predict(myfit,newdata=testing.data)
#Fisher线性判别（正态假设）
library(MASS)
myclassifer = lda(class~variables,data=...)
#二次判别（正态假设）
myclassifer = qda(class~variables,data=...)
#预测新数据newdata的类别标签
predict(classifer,newdata)
```

例 **1.** (阿拉斯加和加拿大三文鱼的判别分类，课本 Example 11.7) 三文鱼出生于淡水河流中，出生 1-2 年后会游到海里生活，若干年再次回到淡水区域。为了判定一条三文鱼是来自美国阿拉斯加还是加拿大，研究人员收集了 50 条阿拉斯加、50 条加拿大三文鱼，测量了其在淡水中第一年的生长速度（Freshwater）和海水中第一年的生长速度（Marine）。生长速度以年轮直径代表。基于这两种鱼的两种生长速度，Fisher 判别法可以计算出线性判别准则函数，即上图的直线。对任意一条不知道其来与地区的三文鱼，我们可以利用该准则预测它是阿拉斯加的还是加拿大的。

```
salmon.train = read.table("http://staff.ustc.edu.cn/~ynyang/vector/data/salmon-train.xls",head=T)
salmon.test= read.table("http://staff.ustc.edu.cn/~ynyang/vector/data/salmon-test.xls",head=T)
plot(salmon.train[,2:3], col=salmon.train[,"Country"] ) # alaska: black, canada: red

#logistic 回归
logistic.fit = glm(Country-1~Freshwater + Marine, data=salmon.train, family=binomial)
y.fit=predict(logistic.fit,  newdata=salmon.test[,-1],type="response")
country.pred.log = round(y.fit)

table(true.label=salmon.test[,1],country.pred.log)

country.pred.log
true.label  0  1
1 27  1
2  3 19
# 错误率 (1+3)/50=0.08
```

```
#LDA（Fisher 线性判别，正态方差相同的情形）
library(MASS)
salmon.lda = lda(Country~Freshwater+Marine,prior=c(0.5,0.5), data=salmon.train)
country.predicted=predict(salmon.lda,newdata=salmon.test[,-1])$class    # 预测/判别
table(true.label=salmon.test[,1], predicted=country.predicted ) # 与真实的类别进行比较：

predicted
true.label  1  2
1 25  3
2  2 20

# 错误率 (2+3)/50=0.1

#qda: （Fisher 二次判别，正态方差不同的情形）
salmon.qda = qda(Country~Freshwater+Marine,prior=c(0.5,0.5), data=salmon.train)
country.predicted =predict(salmon.qda,newdata=salmon.test[,-1])$ class  # 预测
table(true.label=salmon.test[,1], predicted=country.predicted ) # 与真实的类别进行比较：

predicted
true.label  1  2
1 27  1
2  3 19
```

3. 多项 logistic 回归和单层神经网络

R 包 nnet 提供了多项 logistic 回归（响应为多个类别的因子变量），其中 multinom 与线性 logistic 回归类似（指数上线性）。

```
library(nnet)
myfit = multinom(y~x,data=train.data) # 线性（指数上线性） #y: 多类因子
myfit = nnet(y~x,size=, data=train.data) # 非线性 logistic 回归（指数上非线性），
  单层神经网络，size: 神经元个数
predict(myfit,newdata=test.data)
```

若模型 (1) 中 $b(\mathbf{x})$ 是分段线性（复合的仿射变换 +ReLU 激活）或其他非线性函数（复合的仿射变换 + 其它非线性激活），即是神经网络模型。R 包 nnet 中的函数 nnet() 是单层神经网络。R 环境下的深度学习程序包：

keras、tensorflow、deepnet

感兴趣者可以下载学习使用。

尽管基于深度神经网络的深度学习分类方法近些年已经占据机器学习的主导地位，但其它传统的分类方法仍有可借鉴之处。下页简介经典的支持向量机和回归分类树。

附录 1：支持向量机 (SVM: support vector machine)

SVM 模型试图将空间中的不同类的点尽可能地用比较宽的间隔 (margin，下图方框) 分开：间隔边缘的点称为支持向量（support vector），支持向量是最容易判错的数据点，换言之，分类器（即分类判别准则）主要由支持向量决定，其它（各类内部的）向量对分类器影响不大。间隔的宽度即两类支持向量之间的距离，SVM 最大化间隔的宽度，即希望最难分类的点之间的距离尽量大。

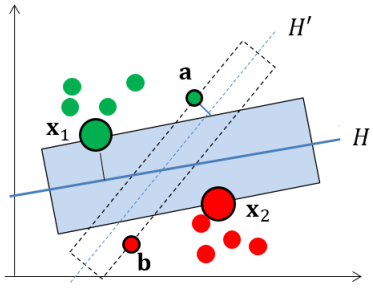


Figure 1: H' 的支持向量 $\mathbf{a},\mathbf{b}$ 比 H 的支持向量更难区分.

以线性可分的两类问题为例，假设 H 是任一线性分类器（线性判别函数）可完全分离两类数据点（上图），两类距离 H 最近（垂直距离）的点分别是 $\mathbf{x}_1,\mathbf{x}_2$, 我们可调整 H 的位置使得 $d(\mathbf{x}_1,H)=d(\mathbf{x}_2,H)$. 两类间隔 (margin) 宽度为

$$m(H)=2d(\mathbf{x}_1,H)$$

(如果 H 不平分间隔，则可定义间隔宽度 $m(H)=d(\mathbf{x}_1,H)+d(\mathbf{x}_2,H)$)

上图中 H' 是另外一个分类器，对应的支持向量为 $\mathbf{a},\mathbf{b}$, 间隔宽度 $m(H')=2d(\mathbf{a},H')<m(H)=2d(\mathbf{x}_1,H)$, 使用分类器 H' 比使用 H 更容易出现误判（$\mathbf{a},\mathbf{b}$ 更容易判错），因此 H 是更不容易出错的分类方法。一般地，SVM 分类器定义为最大间隔的分类器：

$$H_{svm}=\max_H m(H)=\max_H \min_{\mathbf{x}} d(\mathbf{x},H).$$

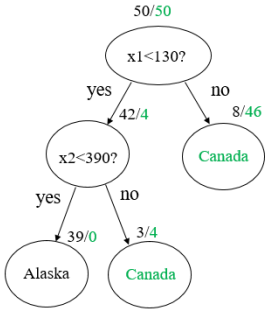
对于不可分的情形，讨论类似。SVM 使用核函数，可用于非线性分类。R 包 e1071 中的函数 svm(), 或 R 包 kernlab 中的 ksvm()，分别执行普通的（线性）svm 分类和带有核技巧的（非线性）svm 分类。

```
library(e1071) #svm() : svm
library(kernlab) #ksvm(): kernel svm


mysvm=svm(factor(Country) ~ Marine +Freshwater, data=salmon.train)
country.predicted.svm =predict(mysvm, newdata=salmon.test[,-1])
table(true.label=salmon.test[,1], predicted=country.predicted.svm )

myksvm=ksvm(factor(Country) ~ Marine +Freshwater, data=salmon.train)
country.predicted.ksvm =predict(myksvm, newdata=salmon.test[,-1])
table(true.label=salmon.test[,1], predicted=country.predicted.ksvm )
```

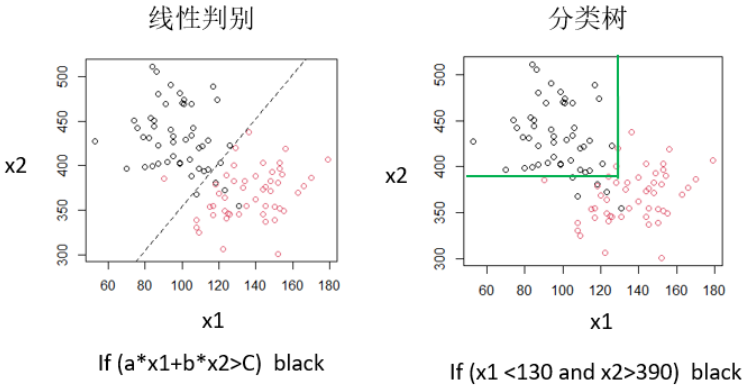
附录 2：决策树/随机森林

决策树或回归决策树是一种将特征分量递归划分的分类或回归方法, 其分类判别方法由一系列逻辑判断构成，呈树状结构，比如

```
x1=Freshwater; x2=Marine ;

若x1<130，且
        若x2<390，  则判为Canada；
        若x2>=390，  则判为Alaska；
若x1>130，则判为Alaska；
```



上述决策树在 (x1,x2) 平面上的划分（分类）如下：



决策树方法简洁和直观，而且具有可解释性：递归划分特征 $\mathbf{x}=(x_1,...,x_p)^\top$ 的分量 $x_i > C$ 或 $x_i \le C$，通过划分分量（而不是分量的线性组合或非线性函数）进行分类显然具有较好的可解释性，因此之故，回归决策树 (Classification and Regression Trees, CART) 在商业领域应用广泛。随机森林 (random forest) 是通过反复抽样生成多个树（称为森林）分类方法，是 CART 的加强版本。

```
library(rpart)
mytree =rpart(Country~Freshwater+Marine, data=salmon.train,method="class" )
country.predicted.tree=predict(mytree,newdata=salmon.test)

# 随机森林:
install.packages("randomForest")
library(randomForest)
myfit=randomForest(Country~Marine+Freshwater,data=salmon.train, ntree=1000)
#1000 棵树
predict(myfit, newdata=salmon.test)
```