Mestrado em Inteligência Artificial e Sistemas Inteligentes

Base de Dados e Programação

"Classificador de Mahalanobis"

28 de Janeiro de 2005

António Jesus Monteiro de Castro

Aluno 040594004

Faculdade de Engenharia da Universidade do Porto

ÍNDICE

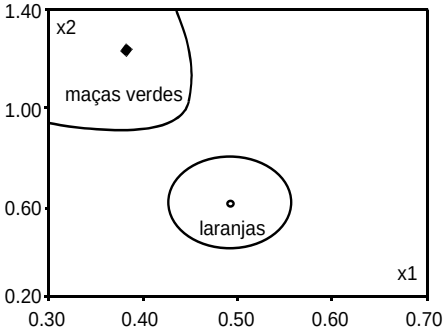
Introdução Teórica

O objectivo deste trabalho é implementarmos em R o Classificador de Mahalanobis. Nesta introdução teórica ao tema vamos começar por dar um exemplo do que é um classificador (em termos gerais) e, depois, um pouco do que é um Classificador de Mahalanobis.

O que é um classificador?

Usando um exemplo que encontramos em [1]:

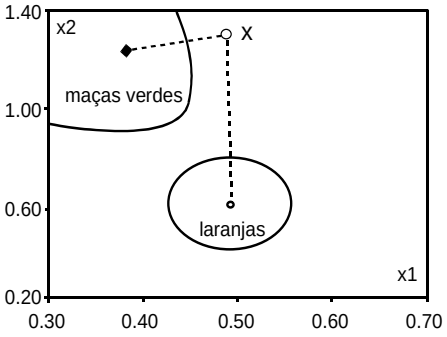
"(...) Para avaliar as semelhanças entre objectos recorre-se a características ou atributos de natureza diferente dos objectos.
Considere-se o exemplo onde se pretende distinguir maças verdes de laranjas. Pode considerar-se as características cor e forma. Para obter uma representação numérica da característica cor pode começar-se por separar a imagem do objecto nas componentes vermelho-verde-azul. Escolhe-se uma região central de interesse no objecto e calcula-se, para essa região, atendendo ao histograma de intensidade de luz, o valor para as componentes vermelho e verde nos respectivos valores possíveis ([0, 255]: 0= sem cor; 255= cor total). Considere-se que os valores obtidos são 186 para a cor verde e 150 para a cor vermelha para um determinado protótipo. Para obter uma representação numérica da característica forma pode-se, por exemplo, medir a distância que difere a largura máxima do objecto com a altura do objecto e normalizar com a altura do objecto (x/h, com x = h –l, h = altura e l = largura máxima). Considere-se que para o mesmo exemplar se tem: x/h= 0.37. Efectuando uma escolha adequada de exemplares espera-se que os exemplos representativos de maças verdes e laranjas definam concentrações de pontos dos referidos exemplares, no espaço característico bidimensional. Na figura estão representadas essas concentrações.



Ao fazer uma boa escolha das características, espera-se que as concentrações estejam razoavelmente separadas, permitindo distinguir as suas classes de frutos.
A tarefa de classificação consiste em relacionar um objecto (exemplo) a uma determinada classe.
Matematicamente, nas tarefas de classificação é conveniente representar um exemplo por um vector de características. Neste caso teríamos:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} cor \\ forma \end{bmatrix}$$

Considere-se que se tem uma máquina para separar maças verdes das laranjas usando as características descritas.
O procedimento consiste em dada uma peça de fruta calcular os valores das características. Seja **x** o valor obtido como se mostra na figura:

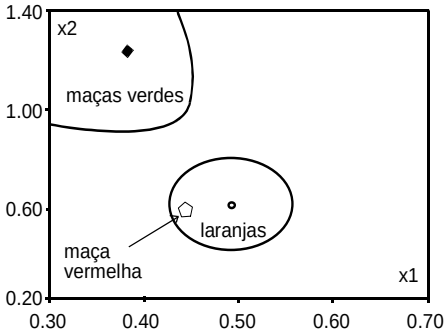


A máquina tem que decidir, tendo como *inputs* os valores calculados das características (**x**), se a peça de fruta é uma maça verde ou uma laranja. Uma decisão razoável baseia-se na distância Euclidiana do ponto **x** aos exemplares, ou seja, para a máquina a semelhança é a distância e neste caso decide que a peça de fruta é uma maça verde. O *output* da máquina para este caso de duas variáveis pode ser, por exemplo:

0 se é maça verde
1 se é laranja

Assim a máquina é chamada de **classificador**.

Observe-se que, por exemplo, se considerarmos uma maça vermelha, pode obter-se os valores como se mostra na figura:



A maça vermelha é erradamente classificada como laranja pois está mais próxima dos exemplos de laranja.
Geralmente os sistemas de classificação não são infalíveis e podem esperar-se erros que podem dever-se a várias causas:

- características inadequadas e insuficientes;
- o conjunto de exemplos usados para a determinação do classificador podem não ser suficientemente significativos;
- o classificador pode não ser suficientemente eficiente para separar as classes;
- pode existir uma sobreposição intrínseca das classes.

Assim é importante a selecção de características adequadas e a determinação de classificadores eficientes. (...) "

O que é o Classificador de Mahalanobis?

Basicamente o Classificador Mahalanobis permite classificar um novo exemplo (uma observação, instância, um objecto) por semelhança com exemplos que já existam, usando a distância de Mahalanobis como métrica para encontrar o exemplo mais semelhante. O exemplo mais semelhante é aquele cuja distância de Mahalanobis até ao novo exemplo é mais curta e, assim, passa a ser classificado com a classe corresponde a esse exemplo.

O que distingue este classificador de outros é que este toma em consideração erros associados com as medidas de previsão, tal como o ruído, usando a Matriz co-variância dos atributos exemplo para escalar os atributos de acordo com as suas variâncias. Para melhor compreendermos o processo, tomemos o seguinte exemplo [4]:

Sendo X o conjunto dos exemplos observados que são mais semelhantes ao vector exemplo f a ser classificado, o classificador de Mahalanobis usa a distância de Mahalanobis [DM] para calcular a distância entre o vector f e o conjunto X. A fórmula para a DM é

$$DM_i(f) = \frac{1}{n}(f - m_i)' C_i^{-1} (f - m_i)$$

Em que f é o vector a classificar, m é o vector com a média dos exemplos e C é a matriz co-variância calculada a partir dos n exemplos guardados como linhas nos vectores de dados de exemplo X_i. O índice i=1,2,...k indica as classes do exemplo.

Para classificar o f calcula-se o valor da distância DM para cada classe usando a equação indicada acima. Como resultado obtemos um número de k diferentes DM para o vector exemplo f. A classe a atribuir é aquela cuja DM for mais pequena.

Como já referimos a distância de Mahalanobis considera, não só a média como também a variância e a covariância das variáveis de medida. Isto permite compensar as interacções entre variáveis.
Pode-se concluir que a distância de Mahalanobis é muito útil para determinar a "semelhança" entre um exemplo "desconhecido" e um conjunto de valores determinados por uma colecção de exemplos "conhecidos". É superior à distância Euclidiana pois tem em consideração a distribuição dos pontos (correlação).
A distância de Mahalanobis deve ser usada para classificar observações em diferentes grupos.

Para além de termos apresentado a formula matemática da Distância de Mahalanobis, gostaríamos, também, de apresentar algumas definições não matemáticas que encontramos durante a nossa pesquisa. Assim e segundo [2]:

"As distancias de Mahalanobis são uma generalização das distancias Euclidianas clássicas que permitem que mudanças em determinada direcção tenham mais peso, ou mais custo, que para outra determinada direcção. O respectivo custo das diferenças é resumido numa matriz de peso W e a distância é calculada como raiz

quadrada de (x-y)'W(xy), onde x e y são os vectores característica dos respectivos objectos que estão a ser comparados."

[Taguchi & Jugulum, 2002] dão-nos a seguinte definição:

"Medida de distância: baseada em correlações entre as variáveis, onde diferentes padrões podem ser identificados e analisados tendo como referencia um determinado ponto."

No anexo A – Classificador de Mahalanobis apresentamos uma explicação teórica mais detalhada deste assunto.

Aplicações Práticas

As distâncias de *Mahalanobis* aparecem relacionadas com métodos de *Instance Based Learning* (Aprendizagem Baseada em Instâncias) e, também, de *Pattern Recognition* (Reconhecimento de Padrões). Compreende-se esta relação uma vez que, quer num caso quer noutro, procura-se semelhanças.

Na Aprendizagem Baseada em Instâncias procura-se classificar novas instâncias através da semelhança com instâncias já existentes. Ao fazer essa comparação usa-se uma métrica de distância para encontrar as instâncias que estão mais perto. Um dos métodos que usa uma métrica de distância é o *Nearest – Neighbor Classification*.

As aplicações das técnicas e sistemas de reconhecimento de padrões são numerosas e cobrem um basto número de actividades. Por exemplo:

- Agricultura
  - Analises de produções
  - Avaliações de solos
- Astronomia
  - Analises de imagens telescópicas
- Biologia
  - Propriedades dos cromossomas
  - Estudos genéticos
- Geologia
  - Classificação das rochas
  - Analises sísmicas
  - Estimação dos recursos da exploração mineira
- Medicina
  - Analise de electrocardiogramas
  - Analises de imagens medicinais
- Segurança
  - Identificação de imagens digitais
  - Vigilância e sistemas de alarme

Descrição da Implementação

Foram criadas duas funções:

(1) calcula.distancia.mahalanobis
(2) classificador.mahalanobis

A primeira tem como objectivo calcular a distância de Mahalanobis entre a observação a classificar e cada um dos elementos do conjunto de treino. Para invocar esta função são necessários os seguintes parâmetros (pela ordem indicada):

- Observação a classificar sem as variáveis nominais.
- Conjunto de treino também sem as variáveis nominais.
- A matriz de variâncias-covariâncias do conjunto de treino.

A função vai garantir que todos os dados são numéricos e, depois, calcula a distância de acordo com a formula apresentada na introdução deste trabalho e, também, de acordo com o que está no Anexo A.

O código é o seguinte:

```
#
# função para calcular a distância de Mahalanobis
#
calcula.distancia.mahalanobis <- function(obs.classificar.corrigidas, novo.conj.treino,
covariancia.novo.conj.treino)
{
        novo.conj.treino <- as.matrix(novo.conj.treino)
        obs.classificar.corrigidas <-
        matrix(data=as.numeric(obs.classificar.corrigidas),
                                nrow=nrow(novo.conj.treino),
                                ncol=ncol(novo.conj.treino),
                                byrow = TRUE,
                                dimnames = NULL)
        distancia.calculada <- (novo.conj.treino-
        obs.classificar.corrigidas)%*%(solve(covariancia.novo.conj.treino))%*%(t(no
        vo.conj.treino-obs.classificar.corrigidas))
        return(diag(distancia.calculada))
}
```

A segunda formula contém o classificador propriamente dito, recebendo os vários valores, preparando-os e mostrando o resultado final. Para invocar esta função são necessários os seguintes parâmetros (pela ordem indicada):

- Observação a classificar de acordo com a estrutura dos dados de treino.
- Conjunto de treino.
- Número da coluna que contém o atributo com a classificação.
- Números das colunas dos atributos a serem usados na classificação.

O código comentado desta função é o seguinte:

```
classificador.mahalanobis <- function(obs.classificar, conj.treino, atr.classificar,
atr.usar)
{
#
# novo conjunto de treino somente com os atributos indicados
#
num.colunas <- ncol(conj.treino)
num.linhas <- nrow(conj.treino)
novo.conj.treino <- conj.treino[,atr.usar]
vector1 <- c(1:num.colunas)
vector1 <- vector1[-atr.classificar]
#
# retira os atributos nominais
#
vector2 <- c()
if (is.null(ncol(novo.conj.treino))==TRUE) {
        vector2 <- is.numeric(novo.conj.treino)
        novo.conj.treino <-novo.conj.treino[vector2]
        }
else {
                for (i in 1:ncol(novo.conj.treino)) {
                        vector2 <- c(vector2,is.numeric(novo.conj.treino[,i]))
                        }
                novo.conj.treino <- novo.conj.treino[,vector2]
                }
#
# actualiza o vector com a observação a classificar de acordo com os atributos
# indicados
#
vector1 <- obs.classificar[match(atr.usar,vector1)]
obs.classificar.corrigidas <- vector1[vector2]
#
# Usa a co-variância ou a variância do atributo para calcular a
# distância de Mahalanobis
#
if (is.null(nrow(novo.conj.treino))==TRUE) {
        covariancia.novo.conj.treino <- var(novo.conj.treino)
        }
else {
                covariancia.novo.conj.treino <- cov(novo.conj.treino)
                }
#
# Calcula a distância de Mahalanobis entre a observação a classificar e
# cada um dos elementos do conjunto de treino
#
distancias <- calcula.distancia.mahalanobis(obs.classificar.corrigidas,
novo.conj.treino, covariancia.novo.conj.treino)
#
# Classifica em função da distancia mais próxima
# que, se tudo correr bem é a que estará em primeiro lugar após ser ordenada
#
classes.possiveis.atribuir <- conj.treino[, atr.classificar]
```

```
distancias.ordenadas <- order(c(distancias))
valores <- classes.possiveis.atribuir[distancias.ordenadas[1]]
factores <- levels(valores)
classificacao.obtida <- factores[which.max(summary(valores))]
cat("A classificação obtida foi: ",classificacao.obtida,"\n")
}
```

Experiências Efectuadas

Fiz experiências usando o data frame IRIS que já vem com o R. Aliás, estas experiências foram fundamentais porque só assim é que foi possível corrigir alguns erros que no início existiam no algoritmo.
Segue-se os resultados dessas experiências.

Observações Utilizadas do Data Frame Iris

```
> iris[c(10),]
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
10          4.9         3.1          1.5         0.1  setosa

> iris[c(100),]
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
100          5.7         2.8          4.1         1.3 versicolor

> iris[c(150),]
    Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
150          5.9           3          5.1         1.8 virginica
```

Resultados obtidos com o Classificador

Para a observação 10:

```
> classificador.mahalanobis(iris[c(10),-5],iris[c(10),],5,c(1:4))
A classificação obtida foi:  setosa
```

Para a observação 100:

```
> classificador.mahalanobis(iris[c(100),-5],iris[c(100),],5,c(1:4))
A classificação obtida foi:  versicolor
```

Para a observação 150:

```
> classificador.mahalanobis(iris[c(150),-5],iris[c(150),],5,c(1:4))
A classificação obtida foi:  virginica
```

Conclusões

Limitações

A implementação que fiz tem, sobretudo, duas limitações:

(1) Só trata uma observação de cada vez. Será mais lógico que possa receber um data frame com várias observações para classificar.
(2) Não verifica os parâmetros de entrada, ou seja, espera que o utilizador forneça os parâmetros da forma indicada e correctos. Se alguma coisa não estiver bem, "estoura".

Possíveis Desenvolvimentos

Existem três possíveis desenvolvimentos que eu considero importantes:

(1) Possibilidade de passar um data.frame com várias observações a classificar, removendo desta forma a limitação (1) indicada anteriormente.
(2) Validar os parâmetros de entrada para que se prever os erros antecipadamente.
(3) Possibilidade de ler que o conjunto de treino quer as observações a classificar a partir de um ficheiro de texto.

Package

Infelizmente e devido a um problema para o qual não encontrei solução, não consegui fazer o build do package, apesar de o ter todo preparado. Em anexo envio o directório do package devidamente configurado. O erro que obtive penso que está ligado a um problema de instalação em que deverá faltar um qualquer software adicional. O Erro obtido é:

```
C:\Program Files\R\rw2000pat>bin\r CMD build ClassMahalanobis
* checking for file 'ClassMahalanobis/DESCRIPTION' ... OK
* preparing 'ClassMahalanobis':
* cleaning src
* removing junk files
* building 'ClassMahalanobis_1.0.tar.gz'
'sh' is not recognized as an internal or external command,
operable program or batch file.
'sh' is not recognized as an internal or external command,
operable program or batch file.
Error: cannot open file 'ClassMahalanobis/DESCRIPTION' for reading
```

Bibliografia

[1] Sá, J. P. Marques de: *Pattern Recognition, Concepts, Methods and Applications.*

[2] William J. Raynor, Jr.: *The International Dictionary of Artificial Intelligence*.

[3] Witten, Ian H e Eibe, Frank: *Data Mining Pratical Machine Learning*.

[4] Hacker, Engelhardt, Frey: *Robust Manufacturing Inspection and Classification with Machine Vision*.

[5] http://www.cs.princeton.edu/courses/archive/fall04/cos436/Duda/PR_Mahal/PR_Mahal.htm:
Mahalanobis Classifiers.

Nota Final

Gostaria de salientar que a primeira parte deste trabalho foi efectuado em conjunto com o Mário Wilson que, entretanto, optou por desistir do mestrado.

Anexo A – Classificador de Mahalanobis

(resumo baseado na referência [5] da bibliografia)

Vamos supor que temos um atributo x com n exemplos que pertencem à mesma classe. Os diferentes valores de x são x(1), x(2), ..., x(n).

Média

A **média** será $m = [x(1) + x(2) + ... + x(n)] / n$

Nota:
se os dados "caírem" num *cluster* a média ficará mais ou menos no meio do *cluster*.

Variância

A **variância** será $v = [(x(1) - m)^2 + (x(2) - m)^2 + ... + (x(n) - m)^2] / (n - 1)$

Nota:
A variância é a medida do tamanho do *cluster*.

Desvio Padrão

$s = \sqrt{v}$

Escala (Scale)

O valor numérico de x depende das unidades usadas, ie, da escala. Se x for multiplicado por um factor escala a, então

$m' = ma \; ; \; s' = sa \; ; \; v' = va^2$

- Algumas vezes é desejável escalar os dados de forma a que o s resultante seja a unidade. Para isso basta fazer $\frac{x}{s}$
- Ao medir a distância de x a m, muitas vezes faz sentido medi-la relativamente ao s. A isto chama-se a **distância padrão** e é dada por
  $r = \left| \frac{x - m}{s} \right|$

  Nota: r é invariável com a *translation* e a *scale*.

Se x(i) for o valor do atributo i,
Se m(i,j) for a média do atributo i para a classe j
Se s(i,j) for o desvio padrão do atributo i para a classe j
Ao medir a distância entre o vector de atributo x e o vector da média m_j para a classe j e utilizando a **distância padrão**, fica

$$r(x, m_j)^2 = \left[\frac{x(1) - m(1, j)}{s(1, j)} \right]^2 + \left[\frac{x(2) - m(2, j)}{s(2, j)} \right]^2 + ... + \left[\frac{x(d) - m(d, j)}{s(d, j)} \right]^2$$

Esta distância tem uma propriedade importante que é ser invariável com a escala, ou seja, se medirmos a distância desta forma, as unidades que usarmos para os vários atributos não terão efeito nas distâncias resultantes e, assim, sem efeito na classificação final.

A **Distância de Mahalanobis** é uma generalização da **distância padrão**.

Co-variância

É a média dos produtos do desvio dos valores dos atributos para as suas médias, ou seja, considerando o atributo i com os seguintes n exemplos:

$$\{x(1,i), x(2,i), ..., x(n,i)\}$$

e o atributo j com os seguintes n exemplos:

$\{x(1,j), x(2,j), ..., x(n,j)\}$ em que x(k,i) e x(k,j) são atributos do mesmo padrão k.

e sendo m(i) média do atributo i e m(j) média do atributo j
a co-variância do atributo i e do atributo j é

$$c(i,j) = \frac{\{[x(1,i) - m(i)][x(i,j) - m(j)] + ... + [x(n,i) - m(i)][x(n,j) - m(j)]\}}{(n-1)}$$

Propriedades importantes da co-variância:

- Se o atributo i e o atributo j tendem a aumentar juntos, então c(i,j)>0
- Se o atributo i tende a diminuir e o atributo j a aumentar, então, c(i,j)<0
- Se são independentes então c(i,j)=0
- $c(i,j) = s(i)^2 = v(i)$

Então, a co-variância c(i,j) é um número entre –s(i) s(j) e +s(i)s(j) que mede a dependência entre o atributo i e o atributo j, com c(i,j)=0 se não há dependências.

A Matriz Co-variância

Todas as co-variâncias c(i,j) podem ser colocadas numa matriz

$$C = \begin{pmatrix} c(1,1) & c(1,2)\cdots & c(1,d) \\ c(2,1) & c(2,2)\cdots & c(2,d) \\ \cdots & \cdots & \cdots \\ c(d,1) & c(d,2)\cdots & c(d,d) \end{pmatrix}$$

Esta matriz providencia-nos uma maneira de medir a distância que é invariante com a transformação linear dos dados.
Se começarmos com um atributo vector x de dimensão d que tem um vector média m_x e uma matriz de co-variância C_x. Se usarmos a matriz A para transformar x em y através de $y = Ax$, então o vector média para y é dado por $m_y = Am_x$ e a matrix co-variância para y é dada por $C_y = AC_xA'$

Se quisermos medir a distância de x a m_x ou de y a m_y o que faremos é normalizar a distância, de uma forma semelhante ao que fizemos quando definimos a **distância padrão** para um único atributo. Assim, o que queremos é saber qual a matriz generalização da expressão escalar

$$r^2 = \left[\frac{x-m}{s}\right]^2 = (x-m)\frac{1}{s^2}(x-m)$$

A resposta é

$$r^2 = (x-m_x)'C_x^{-1}(x-m_x)$$

Esta expressão é invariante para qualquer transformação linear não singular, ou seja, se se substituir y=Ax e usar as formulas acima para m_y e C_y obteremos o mesmo valor numérico para r, independentemente do que a matriz A seja.

Agora suponhamos que existe um espaço de atributos em que os *clusters* são esféricos e a métrica Euclidiana providencia a maneira certa para medir a distância de y a m_y. Neste espaço, a matriz co-variância é a matriz identidade e r é exactamente a **distância Euclidiana** de y a m_y. Mas, uma vez que podemos chegar a esse espaço a partir do espaço x através duma transformação linear, e uma vez que r é invariante com as transformações lineares, podemos computar r a partir da expressão

$$r^2 = (x-m_x)'C_x^{-1}(x-m_x)$$

A Métrica de Mahalanobis e o Classificador de Mahalanobis

O valor e em $r^2 = (x-m_x)'C_x^{-1}(x-m_x)$ é a **Distância de Mahalanobis** do vector atributo x ao vector média m_x, onde C_x é a matriz co-variância para x.

Pode-se provar que as superfícies onde r é constante são elipsoidais centradas à volta da média m_x. No caso especial em que os atributos não são correlacionados e as variâncias entre todas as direcções são as mesmas, estas superfícies são esféricas e a **Distância de Mahalanobis** torna-se equivalente à **Distância Euclidiana**.

Pode-se usar a **Distância de Mahalanobis** num **Classificador de Distâncias Mínimas** da seguinte forma:

Sejam $m_1, m_2, ..., m_c$ as médias (modelos) para as classes c e $C_1, C_2, ..., C_c$ as correspondentes matrizes de co-variância.
Classificamos um vector atributo x medindo a distância de Mahalanobis de cada x para cada uma das médias e atribuímos a x a classe para a qual a distância de Mahalanobis é mínima.

Esta métrica remove várias limitações da métrica Euclidiana:

- Tem em conta automaticamente a escala dos eixos das coordenadas.
- Corrige para a correlação entre atributos diferentes.
- Pode providenciar fronteiras de decisão curvas e lineares.

O preço a pagar:

- As matrizes de co-variância podem ser difíceis de determinar com exactidão.
- A memória e o tempo de computação crescem quadraticamente em vez de linearmente com o número de atributos.

Este problemas podem ser insignificantes quando são precisos poucos atributos mas podem tornar-se mais sérios quando há mais atributos.

Discriminantes Lineares Invariáveis

Existe um caso especial importante em que a métrica de Mahalanobis resulta novamente em funções discriminantes lineares.
Ocorre quando os *clusters* em todas as classes c têm a mesma matriz de co-variância C.
Geometricamente esta situação acontece quando os *clusters* para todas as categorias têm a mesma forma elipsoidal.
Fisicamente esta situação acontece quando o problema é para detectar um entre c sinais diferentes na presença de ruído adicional, onde o ruído não depende em função de quais dos sinais está presente.
Neste caso podemos expandir o quadrado da distância de Mahalanobis do vector atributo x ao vector média m_k da seguinte forma:

$$r^2 = (x - m_k)' C^{-1} (x - m_k) \Leftrightarrow$$
$$r^2 = x' C^{-1} x - m_k' C^{-1} x - x' C^{-1} m_k + m_k' C^{-1} m_k \Leftrightarrow$$
$$r^2 = -2\left[(C^{-1} m_k)' x - 0.5 m_k' C^{-1} m_k \right] + x' C^{-1} x$$

Esta expressão faz lembrar uma expressão semelhante obtida para o **Classificador de Distância Euclidiana Mínima**.

Podemos obter funções discriminantes lineares maximizando a expressão entre parêntesis rectos. Desta vez definimos a função discriminante linear como

$g_k(x) = w_k' x + w_{k0}$ em que $w_k = C^{-1} m_k$ e $w_{k0} = -0.5 m_k' C^{-1} m_k$

Este resultado é mais útil apesar de desistir das vantagens de ter fronteiras de decisão curvas. Mantém a vantagem de ser invariante a transformações lineares. Adicionalmente reduz os requisitos de memória e aumenta a velocidade de computação.
Finalmente, quando a matriz co-variação é a mesma para todas as c classes, podemos obter dados de todas as classes e obter resultados muito melhores a partir dum volume limitado de dados.

Anexo B - Manual do Utilizador

Juntamente com este documento foram fornecidos dois ficheiros, um com o código correspondente e com o nome ClassificadorMahalanobis.R e um outro em forma de Package.

Para se poder utilizar o código enviado basta proceder da seguinte maneira:

1) Arrancar com o gui do R.
2) Escolher FILE/Source R Code.
3) Escolher o ficheiro com o nome ClassificadorMahalanobis.R

Para experimentar com os dados indicados neste documento, proceda da seguinte maneira:

Carregue o data set IRIS
> data(íris)

Agora escreva o seguinte para cada um dos exemplos:

```
>íris[c(10),];classificador.mahalanobis(iris[c(10),-5],iris[c(10),],5,c(1:4))
>       Sepal.Length  Sepal.Width   Petal.Length   Petal.Width    Species
10              4.9           3.1            1.5            0.1     setosa
A classificação obtida foi:  setosa

> iris[c(100),];classificador.mahalanobis(iris[c(100),-5],iris[c(100),],5,c(1:4))
>       Sepal.Length  Sepal.Width   Petal.Length   Petal.Width   Species
100             5.7           2.8            4.1            1.3     versicolor
A classificação obtida foi:  versicolor

> iris[c(150),];classificador.mahalanobis(iris[c(150),-5],iris[c(150),],5,c(1:4))
>       Sepal.Length  Sepal.Width   Petal.Length   Petal.Width    Species
150             5.9           3              5.1            1.8     virginica
A classificação obtida foi:  virginica
```