Traballo Fin de Máster

Clasificación supervisada en datos multivariantes, de alta dimensión e funcionais

Xián López Álvarez

Máster en Técnicas Estatísticas

Curso 2014-2015

Proposta de Traballo Fin de Máster

Título en galego: Clasificación supervisada en datos multivariantes, de alta dimensión e funcionais

Título en español: Clasificación supervisada en datos multivariantes, de alta dimensión y funcionales

English title: Supervised classification with multivariate, high dimensional and functional data

Modalidade: A

Autor: Xián López Álvarez, Universidade de Santiago de Compostela

Breve resumo do traballo:

Este traballo revisa varias técnicas da clasificación estatística supervisada, detallando a teoría sobre a que se sustentan. Tamén se explica a representación de datos funcionais en bases de B-splines e compoñentes principais, e como os coeficientes nestas poden ser empregadas para a clasificación. Escóllense unha serie de bases de datos, obtidas no repositorio da UCI e en certas librerías de R, e aplícanse os modelos seguindo un determinado procedemento de sintonización de parámetros e estimación da precisión. Finalmente amósanse os resultados e compáranse os diferentes clasificadores, analizando cuestións como o tipo de variables ou o tamaño dos grupos. Tamén se estuda a dependencia, no caso dos datos funcionais, fronte ó número de elementos das bases.

Recomendacións:

Outras observacións:

Agradecementos

Debo comezar os agradecementos citando a Manuel Fernández Delgado, que aínda que non figura oficialmente como co-director deste traballo, actuou como tal, e de feito a temática aquí tratada xurdiu, en parte, por iniciativa súa.

Tamén quero agradecer a Manuel Febrero Bande a dirección e correccións e suxerencias aportadas a este traballo.

Finalmente, e como xa empeza a ser costume, debo agradecer á miña nai a revisión lingüística.

Índice xeral

Resumo

Resumo en galego

Este traballo revisa varias técnicas da clasificación estatística supervisada, detallando a teoría sobre a que se sustentan. Tamén se explica a representación de datos funcionais en bases de B-splines e compoñentes principais, e como os coeficientes nestas poden ser empregadas para a clasificación. Escóllense unha serie de bases de datos, obtidas no repositorio da UCI e en certas librerías de R, e aplícanse os modelos seguindo un determinado procedemento de sintonización de parámetros e estimación da precisión. Finalmente amósanse os resultados e compáranse os diferentes clasificadores, analizando cuestións como o tipo de variables ou o tamaño dos grupos. Tamén se estuda a dependencia, no caso dos datos funcionais, fronte ó número de elementos das bases.

English abstract

This document examines several techniques belonging to the field of supervised statistical classification, detailing the theory on which they are based. Besides, basis representation of funcional data is explained, including B-splines and principal components bases, and how their coefficients can be used for classification. A group of datasets is chosen, obtained from the UCI repository and some R packages, and the classification models are applied following a specific procedure for parameter tunning and accuracy estimation. Finally, the results are shown and the different classifiers compared, analyzing things such as the variable types or the group size. In the case of functional data, the dependence on the number of elements of the bases is also studied.

Introdución

Este traballo é a tarefa final do Máster en Técnicas Estatísticas da Universidade de Santiago de Compostela, e pretende axuntar todos os coñecementos nel adquiridos, e aplicalos a un campo concreto da estatística. En particular, céntrase na clasificación supervisada, tanto con datos multivariantes como con funcionais.

Os principais obxectivos son revisar, empregar e comparar diferentes modelos de clasificación estatística supervisada. Co fin de abranguer un espectro amplo, escolléronse tanto métodos clásicos como modernos. Entre os primeiros citamos a análise discriminante de Fisher (tanto linear como cadrática), a regresión loxística e os K veciños máis próximos. Do outro lado eliximos técnicas de amplo uso hoxe en día, como o Perceptrón Multicapa, a Máquina de Vectores de Soporte e o Random Forest.

No Capítulo 1, que pode ser considerado a parte máis importante do traballo, detállanse todos os modelos de clasificación anteriormente citados, desenvolvendo toda a teoría sobre a que se sustentan. No seguinte capítulo faise unha introdución teórica do campo dos datos funcionais, e explícase o procedemento escollido para levar a cabo a clasificación: a representación en bases (de B-splines e de compoñentes principais). Deste xeito, ós datos funcionais aplícanselles os mesmos clasificadores que ós multivariantes, considerando directamente as súas compoñentes sobre as devanditas bases.

As bases de datos escollidas, a metodoloxía empregada para a súa análise, e o código utilizado son descritos no Capítulo 3. Preséntanse as bases de datos multivariantes e funcionais escollidas. Explícase que parámetros axustables ten cada modelo, así como a maneira de levar a cabo a súa sintonización, e como se estima a precisión mediante validación cruzada.

O capítulo 4 adícase á presentación dos resultados experimentais obtidos, así como á súa interpretación. Razóase por que nuns casos uns clasificadores resultan mellor que outros, por que fallan noutras situacións, estúdase a súa dependencia fronte ó ruído, e compáranse os tempos de execución.

Aclaracións sobre a notación

O marco sobre o que se traballa no ámbito da clasificación estatística supervisada é o dunha poboación, da cal podemos observar unha serie de variables numéricas ou categóricas, na que os seus individuos se reparten en varias clases disxuntas.

En todo este documento, empregaremos a letra p para referirnos ó número de variables observables, e K para a cantidade de clases. Tamén empregaremos indistintamente a palabra "grupos" para referirnos a estas últimas.

Dependendo do libro que se consulte, é posible atopar nomes moi diferentes para ditas variables: entradas ou *inputs*, variables explicativas... Igualmente, as observacións da poboación tamén reciben cantidade de nomes, como individuos, patróns...

En canto á notación puramente estatística, X será a variable aleatoria p-dimensional coas variables de entrada, e representaremos por x unha realización particular dela. Polo tanto, x_i será a observación i-ésima, con $i = 1, ..., n$ (n é o número de observacións). Por outra banda, G será a variable aleatoria que identifica o grupo ó que pertencen as observacións.

Capítulo 1

Clasificadores Multivariantes

Este primeiro capítulo está adicado ós seis modelos de clasificación considerados neste traballo. Xa que todo o resto da obra se basea no que aquí se expón, decidiuse converter esta parte nunha das principais, adicándolle un gran esforzo para conseguir desenvolver, de maneira clara e precisa, toda a teoría dos distintos métodos.

As tres primeiras seccións teñen conta dos modelos máis clásicos: a análise discriminante de Fisher, a regresión loxística e os K veciños máis próximos. A continuación, outras tres seccións detallan os métodos máis modernos: o Random Forest, a Máquina de Soporte Vectorial e o Perceptrón Multicapa.

Co obxectivo de obter fórmulas o máis claras e compactas posibles, nalgúns casos cometeremos un pequeno abuso de notación, poñendo $\mathbb{P}(x \,|\, i)$ en lugar de $\mathbb{P}(X = x \,|\, G = i)$, e $\mathbb{P}(i \,|\, x)$ en lugar de $\mathbb{P}(G = i \,|\, X = x)$. A omisión das variables aleatorias X e G non debería inducir a ningún erro, pois nunca cabe dúbida de a cal pode estar referida unha observación x, ou un enteiro i ou j.

1.1. Análise discriminante

Dicir análise discriminante é tanto como dicir clasificación estatística, mais neste traballo referímonos concretamente á análise discriminante en poboacións normais, que inclúe o discriminante linear de Fisher (LDA, *Linear Discriminant Analysis*), e a súa xeneralización, a análise cadrática discriminante (QDA *Quadratic Discriminant Analysis*). Comezaremos coa segunda, por ser máis xeral, xa que logo a primeira dedúcese facendo unhas sinxelas simplificacións.

A hipótese fundamental sobre a que se sustenta a análise discriminante é a de que os grupos seguen unha distribución normal, é dicir, a probabilidade de que unha observación x pertenza ó grupo i vén dada por

$$\mathbb{P}(x \,|\, i) = \frac{1}{\sqrt{(2\pi)^D |\Sigma_i|}} \exp\left\{ -\frac{1}{2}(x - \mu_i)^t \Sigma_i^{-1} (x - \mu_i) \right\}.$$

No caso da análise discriminante linear, asumimos tamén que $\Sigma_i = \Sigma_j \ \forall i \neq j$, é dicir, que todos os grupos teñen a mesma matriz de varianzas-covarianzas. Se temos que as probabilidades *a priori* de cada grupo son $\pi_1, \ldots, \pi_K$, a probabilidade *a posteriori* de pertencer ó grupo i virá dada por

$$\mathbb{P}(i \,|\, x) = \frac{\pi_i \, \mathbb{P}(x \,|\, i)}{\mathbb{P}(x)}$$

Cómpre mencionar que, na práctica, os resultados que se obteñen con estas técnicas de análise discriminante son realmente bos, a pesar de que a hipótese de normalidade raramente se cumpre. Ademais, sitúanse entre os procedementos que menor esforzo computacional requiren, e non precisan do axuste de ningún parámetro.

Outro aspecto a salientar é que, aínda que o QDA trata un problema máis xeral, pode non ser sempre máis axeitado que o LDA. En principio, se se pode asegurar que se cumpre a hipótese da

igualdade de matrices de varianzas-covarianzas, é aconsellable empregar o LDA. Ademais, o QDA require da estimación dunha cantidade maior de parámetros (unha matriz de varianzas-covarianzas por cada clase), reducindo polo tanto o número de graos de liberdade. Isto pode redundar en serios problemas no caso de ter poucas observacións nalgún grupo, ou moitas variables (matrices moi grandes).

1.1.1. QDA

Para calquera dos dous grupos i e j tense, baixo a hipótese de normalidade, que o logaritmo do cociente das probabilidades *a posteriori*, dada unha observación x, é

$$\begin{aligned}\log\left(\frac{\mathbb{P}(i\,|\,x)}{\mathbb{P}(j\,|\,x)}\right) = \frac{1}{2}x^t(\Sigma_j^{-1}-\Sigma_i^{-1})x \; + x^t(\Sigma_i^{-1}\mu_i-\Sigma_j^{-1}\mu_j) \; + \\ \log\left(\frac{\pi_i}{\pi_j}\right)+\frac{1}{2}\log\left(\frac{|\Sigma_j|}{|\Sigma_i|}\right)+\frac{1}{2}\left(\mu_j^t\Sigma_j^{-1}\mu_j-\mu_i^t\Sigma_i^{-1}\mu_i\right)\end{aligned} \tag{1.1}$$

(véxase o Apéndice A.1).

Esta expresión pon de manifesto o tipo de sumandos que definen as separacións entre as clases: apréciase un termo cadrático, un linear, e finalmente varios constantes (con respecto a x, todos). Este é o motivo polo que se fala de análise **cadrática** discriminante.

Habendo K clases resultan, en principio, $\binom{K}{2}$ posibles comparacións entre grupos, mais na práctica só é preciso calcular $K-1$, pois pódese facer o seguinte:

$$\log\left(\frac{\mathbb{P}(i\,|\,x)}{\mathbb{P}(j\,|\,x)}\right) = \log\left(\frac{\mathbb{P}(i\,|\,x)}{\mathbb{P}(K\,|\,x)}\frac{\mathbb{P}(K\,|\,x)}{\mathbb{P}(j\,|\,x)}\right) =$$

$$\log\left(\frac{\mathbb{P}(i\,|\,x)}{\mathbb{P}(K\,|\,x)}\right)-\log\left(\frac{\mathbb{P}(j\,|\,x)}{\mathbb{P}(K\,|\,x)}\right).$$

É dicir, calculamos os cocientes dos $K-1$ primeiros grupos, por exemplo, respecto do último, e as comparacións cruzadas obtéñense logo restando estas cantidades.

O feito de coñecer os logaritmos dos cocientes entre as probabilidades *a posteriori* permite determinar, de xeito analítico, as superficies (hiperplanos no caso do LDA) de separación entre os grupos. Sen embargo, para clasificar unha nova observación, só é preciso calcular a probabilidade *a posteriori* de cada grupo (correspondente a dita observación) e asignarlle aquel para o cal sexa maior.

1.1.2. LDA

Se consideramos o caso máis restritivo sobre o que se aplica o LDA, é dicir, que as matrices de varianzas-covarianzas son iguais para todos os grupos, na expresión (1.1) desaparecen o termo cadrático e un dos constantes, resultando

$$\log\left(\frac{\mathbb{P}(i\,|\,x)}{\mathbb{P}(j\,|\,x)}\right) = \log\left(\frac{\pi_i}{\pi_j}\right)+\frac{1}{2}\left(\mu_j^t\Sigma^{-1}\mu_j-\mu_i^t\Sigma^{-1}\mu_i\right)+x^t\Sigma^{-1}(\mu_i-\mu_j),$$

sendo $\Sigma=\Sigma_1,\ldots,\Sigma_K$. É por iso que neste caso falamos de análise **linear** discriminante. Como comentabamos anteriormente, dada a linealidade da función de clasificación resultante, as fronteiras de separación entre os distintos grupos pasan a ser hiperplanos.

Ademais, a diferenza doutros métodos de clasificación que se estudarán máis adiante, na análise discriminante (tanto linear como cadrática) todas as observacións contribúen á determinación de ditas fronteiras (teñen unha influencia proporcinal á súa distancia á media). En teoría, se non se cumpren as hipóteses do modelo, isto pode supoñer un axuste incorrecto. Por outra banda, este feito tamén implica unha gran dependencia fronte a datos atípicos.

1.2. Regresión Loxística

Nesta sección abordamos o problema da clasificación dende a óptica da regresión. Para isto, consideramos o modelo de regresión loxística, que permite aplicar a metodoloxía da regresión para modelar as probabilidades a posteriori de pertenza a cada grupo (usaremos RL como abreviación de Regresión Loxística). Veremos como, a pesar das funcións non lineais que involucra, a regra de clasificación que resulta é linear. Comezamos estudando o caso binario, e desenvolvemos o algoritmo para o seu axuste por máxima verosimilitude, coñecido como *Iteratively Reweighted Least Squares* (IRLS). De seguido, abordamos o caso de ter varios grupos distintos.

A aplicación do modelo de regresión loxística para un determinado problema de clasificación está supeditada, evidentemente, a que se cumpran as hipóteses sobre as que este se constrúe. Como veremos, estas son que as *log-odds*, é dicir, os logaritmos dos cocientes das probabilidades de pertenza a cada grupo, sexan funcións lineais. No caso particular de clasificación binaria, isto implica que a probabilidade de pertenza a un dos grupos teña a forma descrita na Figura 1.1 (o eixo horizontal sería unha certa combinación linear das variables explicativas, ou a propia variable se só hai unha).

1.2.1. Regresión loxística binaria

Consideremos que temos dous grupos, que codificaremos como $G = 1$ e $G = 0$. Estamos interesados en modelar as probabilidades a posteriori, dada unha observación x:

$$\mathbb{P}\big(G = 1 \,|\, X = x\big), \qquad \mathbb{P}\big(G = 0 \,|\, X = x\big).$$

A idea máis sinxela que se nos pode ocorrer é modelar estas probabilidades como funcións lineares de x:

$$\mathbb{P}\big(1 \,|\, x\big) = \beta_{10} + \beta_1^t x,$$

$$\mathbb{P}\big(0 \,|\, x\big) = \beta_{00} + \beta_0^t x.$$

Sen embargo, este enfoque presenta algúns problemas. Primeiro, deste xeito as probabilidades non estarían acoutadas entre 0 e 1. Ademais, empiricamente obsérvase que as probabilidades non seguen un patrón linear: un aumento nunha certa cantidade dunha variable explicativa non produce o mesmo efecto se a probabilidade xa é moi alta, que se está próxima a 1/2, por exemplo.

O seguinte paso sería considerar linear o logaritmo da probabilidade. Deste xeito, nunca habería probabilidades negativas. Nembargantes, seguiríamos a ter probabilidades por encima de 1, o cal non pode acontecer. Polo tanto, este enfoque segue sen resultar axeitado.

A alternativa que solventa todos os problemas anteriormente citados é a función loxística (ou *logit*). Esta vén dada por

$$g(q) = \log \frac{q}{1-q},$$

e o modelo ten entón a seguinte forma:

$$\log \frac{\mathbb{P}\big(1 \,|\, x\big)}{\mathbb{P}\big(0 \,|\, x\big)} = \beta_0 + \beta^t x,$$

tendo en conta que $\mathbb{P}\big(0 \,|\, x\big) = 1 - \mathbb{P}\big(1 \,|\, x\big)$. O cociente entre as probabilidades de pertencer a cada grupo coñécese como *odds*, e o seu logaritmo como *log-odds*. Polo tanto, o modelo loxístico consiste en considerar as *log-odds* como funcións lineais de x.

Se aplicamos a inversa da función *logit*,

$$g^{-1}(y) = \frac{1}{1 + \mathrm{e}^{-y}},$$

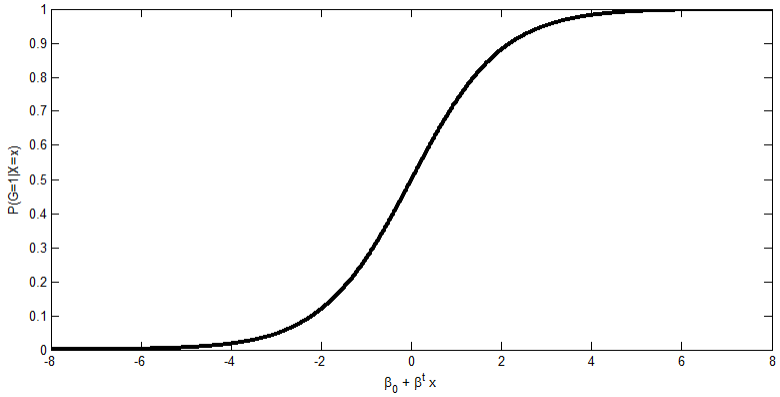

Figura 1.1: Modelo loxístico para a probabilidade a posteriori.

podemos despexar a probabilidade de pertencer ó grupo 1:

$$\mathbb{P}\big(1 \,|\, x\big) = \frac{1}{1+\exp\{-(\beta_0+\beta^t x)\}} = \frac{\exp\{\beta_0+\beta^t x\}}{1+\exp\{\beta_0+\beta^t x\}}. \tag{1.2}$$

Na Figura 1.1 pódese ver a forma desta función. Aprécianse dúas asíntotas horizontais en 0 e 1, o cal significa que a probabilidade tende a 0 para valores baixos de $\beta_0+\beta^t x$, e a 1 para valores altos. Ademais, o maior efecto prodúcese cando a probabilidade está en valores intermedios, como habitualmente acontece en moitos fenómenos reais.

Á hora de clasificar unha nova observación, escolleremos o grupo para o cal a probabilidade *a posteriori* sexa maior. É dicir, clasificaremos no grupo 1 se

$$\mathbb{P}\big(1 \,|\, x\big) > \mathbb{P}\big(0 \,|\, x\big) \Rightarrow \frac{\mathbb{P}\big(1 \,|\, x\big)}{\mathbb{P}\big(0 \,|\, x\big)} > 1$$

$$\Rightarrow \log \frac{\mathbb{P}\big(1 \,|\, x\big)}{\mathbb{P}\big(0 \,|\, x\big)} = \beta_0+\beta^t x > 0,$$

e no grupo 0 en caso contrario. Nótese que, en definitiva, temos unha regra de clasificación linear:

$$\hat{G}(x) = \begin{cases} 1, & \text{se } \beta_0+\beta^t x > 0, \\ 0, & \text{se } \beta_0+\beta^t x \leq 0. \end{cases}$$

Nota: No ámbito de variables continuas, non é preciso prestar especial atención ó caso de que $\beta_0+\beta^t x = 0$; simplemente asígnase un grupo calquera. Sen embargo, se hai variables discretas é preciso calibrar con que probabilidade se asigna a cada grupo nese caso, de xeito que se garanta un erro de clasificación mínimo.

Axuste do modelo por máxima verosimilitude

Comezamos cun pouco de notación. Xuntamos os parámetros do modelo nun único elemento, $\theta = (\beta_0, \beta) = (\beta_0, \beta_1, \ldots, \beta_p)$. Para abreviar, chamamos $q(x_i, \theta) = \mathbb{P}(G = 1 \,|\, X = x_i\,; \theta)$, a probabilidade a posteriori do grupo 1, supoñendo que G segue un modelo loxístico cos parámetros θ. Finalmente, ó vector coa observación i-ésima engadímoslle un 1 diante, para ter conta do intercepto: $x_i^* = (1, x_{i(1)}, \ldots, x_{i(p)})^t$. Ademais, para cada observación, definimos a variable y_i, que vale 1 se pertence ó grupo 1, e 0 se pertence ó grupo 2.

Dados os parámetros θ, a log-verosimilitude é

$$\ell(\theta) = \log\left[\prod_{i=1}^{n}\left[y_i\mathbb{P}\big(1\,|\,x_i\big) + (1-y_i)\mathbb{P}\big(0\,|\,x_i\big)\right]\right] =$$

$$\sum_{i=1}^{n}\left[y_i\theta^t x_i^* - \log\left(1+\exp\{\theta^t x_i^*\}\right)\right]$$

(as contas detállanse no Apéndice A.2). Para atopar o seu máximo, cómpre derivar esta expresión respecto de cada compoñente de θ. Procedemos por partes, derivando separadamente cada un dos sumandos:

$$\frac{\partial}{\partial\theta_k}\big(y_i\theta^t x_i^*\big) = y_i x_{i(k)}^*,$$

$$\frac{\partial}{\partial\theta_k}\log\left(1+\exp\{\theta^t x_i^*\}\right) = \frac{1}{1+\exp\{\theta^t x_i^*\}}\frac{\partial}{\partial\theta_k}\big(1+\exp\{\theta^t x_i^*\}\big) =$$

$$x_{i(k)}^*\frac{\exp\{\theta^t x_i^*\}}{1+\exp\{\theta^t x_i^*\}} = x_{i(k)}^* q(x_i,\theta).$$

Nótese que $x_{i(k)}^*$ refírese ó k-ésimo elemento do vector x_i^*, que é 1 se $k=1$, ou a compoñenete $k-1$ do vector x_i noutro caso. Se xuntamos estes resultados, temos a derivada da log-verosimilitude respecto de θ_k,

$$\frac{\partial\ell(\theta)}{\partial\theta_k} = \sum_{i=1}^{n}\Big[y_i x_{i(k)}^* - x_{i(k)}^* q(x_i,\theta)\Big] = \sum_{i=1}^{n} x_{i(k)}^*\big(y_i - q(x_i,\theta)\big).$$

Movendo k entre 1 e $p+1$ obtemos precisamente esta cantidade de ecuacións, as cales conforman o grandiente da log-verosimilitude,

$$\frac{\partial\ell(\theta)}{\partial\theta} = \sum_{i=1}^{n} x_i^*\big(y_i - q(x_i,\theta)\big). \tag{1.3}$$

Os parámetros que maximizan a log-verosimilitude (e en consecuencia a verosimilitude) atópanse entre as raíces das súas derivadas (ou son precisamente estas raíces, se son únicas). Igualando (1.3) a cero, obtemos un sistema non linear de $p+1$ ecuacións:

$$\begin{aligned}
\sum_{i=1}^{n}\big(y_i - q(x_i,\theta)\big) &= 0,\\
\sum_{i=1}^{n} x_{i(1)}\big(y_i - q(x_i,\theta)\big) &= 0,\\
&\vdots\\
\sum_{i=1}^{n} x_{i(p)}\big(y_i - q(x_i,\theta)\big) &= 0.
\end{aligned} \tag{1.4}$$

Nótese que a primeira destas ecuacións implica que $\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} q(x_i,\theta)$, é dicir, que a cantidade esperada de elementos da clase 1 (termo da dereita), coincide coa cantidade observada (esquerda).

O sistema (1.4) non se pode resolver alxebraicamente, polo que é preciso recorrer a aproximacións numéricas. O método máis usado é o *Iterative Reweighted Least Squares* (IRLS), que consiste simplemente en aplicar o método de Newton-Raphson. Partindo dun vector de parámetros θ^{old}, obtense un novo iterante do seguinte xeito:

$$\theta^{new} = \theta^{old} - \left[\frac{\partial^2\ell(\theta^{old})}{\partial\theta\partial\theta^t}\right]^{-1}\frac{\partial\ell(\theta^{old})}{\partial\theta}. \tag{1.5}$$

Este proceso repítese iterativamente ata que se satisfai algún test de parada (sobre a diferenza relativa entre dous interantes consecutivos e/ou sobre a norma da función avaliada no iterante). Cómpre comentar que este algoritmo non ten a converxencia garantida para calquera vector de parámetros inicial.

Como se observa en (1.5), é preciso coñecer a matriz Hessiana. No Apéndice A.2 demóstrase que esta vén dada por

$$\frac{\partial^2 \ell(\theta)}{\partial\theta\partial\theta^t} = \sum_{k=1}^{n} -q(x_k,\theta)\big(1-q(x_k,\theta)\big)\, x_k^*\, x_k^{*t}. \tag{1.6}$$

Esta expresión, xunto con (1.3) e (1.5), permiten aproximar os parámetros do modelo de regresión loxística que maximizan a verosimilitude. Mais é posible entender mellor o funcionameno do método, e de paso o seu nome, se empregamos notación matricial. Sexan

$$X = \begin{pmatrix} 1 & x_{1(1)} & \dots & x_{1(p)} \\ \vdots & \vdots & \ddots & \dots \\ 1 & x_{n(1)} & \dots & x_{n(p)} \end{pmatrix}, \quad y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \quad q = \begin{pmatrix} q(x_1,\theta) \\ \vdots \\ q(x_n,\theta) \end{pmatrix},$$

$$W = \begin{pmatrix} q(x_1,\theta)\big(1-q(x_1,\theta)\big) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & q(x_n,\theta)\big(1-q(x_n,\theta)\big) \end{pmatrix}$$

(cómpre non confundir a matriz X recén definida, coa variable aleatoria da cal proceden as obsevacións x_i). Así, o gradiente (1.3) queda como

$$\frac{\partial \ell(\theta)}{\partial\theta} = X^t\big(y-q\big),$$

e a matriz Hessiana (1.6) como

$$\frac{\partial^2 \ell(\theta)}{\partial\theta\partial\theta^t} = -X^t W X.$$

O paso dado en cada iteración do método de Newton pódese reescribir entón do seguinte xeito:

$$\theta^{new} = \theta^{old} + \big(X^t W X\big)^{-1} X^t\big(y-q\big).$$

Chamando $z = X\theta^{old} + W^{-1}(y-q)$ e facendo algunhas contas chégase a

$$\theta^{new} = \big(X^t W X\big)^{-1} X^t W z.$$

É dicir, cada iteración pode ser vista como un axuste ponderado por mínimos cadrados sobre a variable resposta z, as veces chamada *resposta axustada*. Este é o motivo polo que este método recibe o nome de *Iteratively Reweighted Least Squares*.

1.2.2. Regresión loxística con varios grupos

No caso de ter máis de dous grupos, digamos K, o modelo loxístico segue a ser aplicable. A idea é a mesma que no caso anteiror: considerar as *log-odds* como funcións lineais. Para iso, é preciso escoller un grupo de referencia. Nós, no presente traballo, tomamos como referencia o último dos grupos.

Así, para cada grupo $i = 1, \dots, K-1$, temos os parámetros β_{i0} e β_i, e modelamos o logaritmo da súa *odds* respecto do grupo K como

$$\log \frac{\mathbb{P}\big(G = i \,|\, X = x\big)}{\mathbb{P}\big(G = K \,|\, X = x\big)} = \beta_{i0} + \beta_i^t x. \tag{1.7}$$

Despexando, temos

$$\mathbb{P}\big(i \,|\, x\big) = \mathbb{P}\big(K \,|\, x\big) \exp\big\{\beta_{i0} + \beta_i^t x\big\}, \tag{1.8}$$

e con esta expresión podemos sumar a probabilidade de todos os grupos e igualar a 1,

$$1 = \sum_{j=1}^{K} \mathbb{P}\big(j \,|\, x\big) =$$

$$\sum_{j=1}^{K-1} \mathbb{P}\big(K \,|\, x\big) \exp\big\{\beta_{j0} + \beta_j^t x\big\} + \mathbb{P}\big(K \,|\, x\big) \Rightarrow$$

$$\mathbb{P}\big(K \,|\, x\big) \left(1 + \sum_{j=1}^{K-1} \exp\big\{\beta_{j0} + \beta_j^t x\big\}\right) = 1 \Rightarrow$$

$$\mathbb{P}\big(K \,|\, x\big) = \frac{1}{1 + \sum_{j=1}^{K-1} \exp\big\{\beta_{j0} + \beta_j^t x\big\}}. \tag{1.9}$$

Con isto podemos reescribir (1.8),

$$\mathbb{P}\big(i \,|\, x\big) = \frac{\exp\big\{\beta_{i0} + \beta_i^t x\big\}}{1 + \sum_{j=1}^{K-1} \exp\big\{\beta_{j0} + \beta_j^t x\big\}}, \; i = 1, \dots, K-1. \tag{1.10}$$

Seguindo coa idea de clasificar segundo o grupo que teña maior probabilidade *a posteriori*, das propias ecuacións (1.7) despréndese que a separación entre a clase K e calquera outra vén dada por unha ecuación linear. De feito, isto acontece para calquera parella de clases i e j:

$$\log \frac{\mathbb{P}\big(i \,|\, x\big)}{\mathbb{P}\big(j \,|\, x\big)} = \log \mathbb{P}\big(i \,|\, x\big) - \log \mathbb{P}\big(j \,|\, x\big) =$$

$$\log \mathbb{P}\big(i \,|\, x\big) - \log \mathbb{P}\big(K \,|\, x\big) + \log \mathbb{P}\big(K \,|\, x\big) - \log \mathbb{P}\big(j \,|\, x\big) =$$

$$\log \frac{\mathbb{P}\big(i \,|\, x\big)}{\mathbb{P}\big(K \,|\, x\big)} - \log \frac{\mathbb{P}\big(j \,|\, x\big)}{\mathbb{P}\big(K \,|\, x\big)} = \beta_{i0} + \beta_i^t x - \beta_{j0} - \beta_j^t x,$$

é dicir,

$$\log \frac{\mathbb{P}\big(i \,|\, x\big)}{\mathbb{P}\big(j \,|\, x\big)} = \big(\beta_{i0} - \beta_{j0}\big) + \big(\beta_i^t - \beta_j^t\big) x.$$

As probabilidades *a posteriori* para cada grupo, dadas polas ecuacións (1.9) e (1.10), teñen todas un denominador común. Polo tanto, a maior delas será a que teña maior numerador. Estes son $\exp\{\beta_{i0} + \beta_i^t x\}$ para os $K-1$ primeiros grupos, e 1 para o K-ésimo. A clase que maximice estas cantidades será a mesma que a que maximice os seus logaritmos (pois o logaritmo é unha función monótona crecente), de xeito que temos a seguinte regra de clasificación:

$$\hat{G}(x) = \operatorname{argmax} \begin{pmatrix} \beta_{10} + \beta_1^t x \\ \vdots \\ \beta_{K-1,0} + \beta_{K-1}^t x \\ 0 \end{pmatrix}.$$

Nótese que nesta expresión interveñen $K-1$ funcións lineais.

1.3. K - Veciños Máis Próximos

O algoritmo de K veciños máis próximos (KNN, K - *Nearest Neighbors*), para clasificación, consiste en asignar a cada nova observación a clase máis frecuente entre as K observacións máis próximas a ela. É preciso ter coidado para non confundir esta K co número de grupos, que neste traballo tamén denotamos por K.

En principio, o único parámetro axustable que ten é o número de veciños próximos a considerar, K. Sen embargo, é posible considerar unha versión do algoritmo máis complexa, na cal se empregue unha función núcleo para ponderar esas K observacións segundo a distancia á que estean (vén sendo unha regresión tipo núcleo). Neste caso, débese ter en conta tamén a función núcleo que se escolle.

Por outra banda, este algoritmo require do concepto de distancia. En principio, independentemente da natureza dos datos, con que sexamos capaces de definir unha distancia entre as observacións, xa podemos aplicar o algoritmo KNN.

1.4. Random Forest

Leo Breiman, no seu coñecido paper de 2001 [3], define o *Random Forest* (RF) como un método consistente dunha colección de clasificadores de tipo árbore, $\{h(x,\Theta_k),\, k=1,...\}$, onde os $\{\Theta_k\}$ son vectores aleatorios independentes e idénticamente distribuídos, e cada árbore emite un voto dada a entrada x.

Sen embargo, usualmente na bibliografía emprégase o nome de Random Forest para referirse a un caso particular da anterior definición, o chamado *Forest-RI* (de *Random Input*). Neste esquema, cada vez que se pretende separar un nodo dunha árbore, escóllense aleatoriamente m variables de entre as p existentes, e só se teñen estas en conta para realizar a partición.

Antes de entrar no desenvolvemento do Forest-RI, presentamos un dos modelos máis coñecidos de árbore de clasificación, o CART. Aínda que no contexto do Random Forest a construción das árbores difire lixeiramente deste esquema, consideramos conveniente presentar aquí o modelo clásico, co proceso de poda incluído.

A continuación, expoñemos un dos principais inconvenientes das árbores: a súa alta variabilidade. Isto motiva o desenvolvemento de técnicas *ensemble*, entre elas, o Random Forest.

Despois de desenvolver o Forest-RI, explicamos unha metodoloxía para a estimación do erro de predición cando se traballa con remostras bootstrap, a estimación Out of Bag. A súa natureza é moi similar á validación cruzada, pero require dunha cantidade de cálculos menor. Á parte do erro de predición, esta técnica tamén permite medir a importancia relativa de cada variable.

1.4.1. Árbores de clasificación

Os métodos de clasificación de tipo árbore particionan o espazo das observacións nunha serie de rectángulos, asignando a cada un unha certa clase (ou un valor constante no caso de seren empregados para regresión). Nesta sección centramos a nosa atención nun tipo concreto de árbores: o CART (*Classification and Regression Trees*).

Na parte superior da árbore sitúase un nodo, que representa a todo o espazo das observacións, $\mathbb{R}^p$. En xeral, un nodo m ten asociada unha rexión de $\mathbb{R}^p$ que denotamos por R_m. Os seus nodos fillos terán asociadas as rexións

$$R_m^- = \{x \in R_m / x_j \leq s_m\} \quad \text{e} \quad R_m^+ = \{x \in R_m / x_j > s_m\},$$

que consisten na partición da rexión do nodo pai respecto dunha variable x_j nun punto s_m. Denotamos por N_m o número de observacións que se sitúan en R_m, é dicir,

$$N_m = \#\{x_i \in R_m,\ i=1,...,n\}.$$

Partindo do nodo raíz, pódese expandir a árbore tanto como se queira. Usualmente, adóptanse criterios de parada como un certo número de observacións por nodo ou a ganancia, nos termos en que esta se defina, por unha nova partición. Finalmente, un proceso moi importante (aínda que non se aplica nos Random Forests) é a poda, mediante a cal se eliminan certos nodos, simplificando así a árbore resultante. Este procedemento permite eliminar o problema de sobreaxuste asociado a unha árbore demasiado grande.

Construción da árbore completa

Un concepto importante á hora de buscar a mellor partición sobre unha variable, é o de impureza dun nodo. Considérase que un nodo é máis puro canto máis predominante sexa unha clase, mentras que será máis impuro se hai presentes varias clases en proporcións similares. Evidentemente, o obxectivo é obter nodos o máis puros posible. Para cuantificar este concepto, introducimos a función de impureza, que denotamos por $Q(m)$. Existen diversas maneiras de definila:

- Índice de Gini:
$$Q(m) = \sum_{k=1}^{K} \sum_{l=1, l\neq k}^{K} p(k|m)\, p(l|m).$$
onde
$$p(k|m) = \frac{\#\{x_i \in R_m / y_i = k\}}{N_m}$$
é a proporción de observacións de R_m que están no grupo k.

- Deviance:
$$Q(m) = -\sum_{k=1}^{K} p(k|m) \log p(k|m).$$

- Erro de clasificación:
$$Q(m) = \frac{1}{N_m} \sum_{x_i \in R_m} I\left(y_i \neq k(m)\right),$$
sendo $k(m)$ a clase máis frecuente no nodo m.

Polo xeral, o algoritmo CART emprega o índice de Gini, mais os outros tamén poden ser perfectamente empregados para a construción das árbores. Un inconveniente do erro de clasificación é a súa non derivabilidade, o cal dificulta a optimización cando este é empregado.

Á hora de particionar un certo nodo m, emprégase o incremento de impureza como medida para determinar a mellor variable j sobre a que efectuar a separación, así como o punto de corte, s. Denotamos os nodos fillos por $m^-(j,s)$ e $m^+(j,s)$, indicando a dependencia respecto de j e s. Os números de observacións que corresponden a cada un serán, respectivamente, $N_{r^-(j,s)}$ e $N_{r^+(j,s)}$. O incremento de impureza vén definido pola diferenza entre a impureza do nodo pai, e a suma das impurezas dos fillos, ponderados segundo o número de observacións:

$$\Delta Q(m,j,s) = Q(m) - \left[N_{m^-(j,s)}\, Q\left(m^-(j,s)\right) + N_{m^+(j,s)}\, Q\left(m^+(j,s)\right)\right]. \tag{1.11}$$

Para cada variable j, calcúlase o s que maximiza dito incremento,

$$\hat{s}_j = \underset{s\in\mathbb{R}}{\operatorname{argmax}}\ \Delta Q(m,j,s).$$

A continuación, compáranse os incrementos obtidos para cada variable con $\hat{s}_j$, e escóllese aquela con maior valor,

$$\hat{j} = \underset{j=1,\ldots,p}{\operatorname{argmax}}\ \Delta Q(m,j,\hat{s}_j).$$

Os nodos fillos que finalmente se construirán serán $m^-(\hat{j},\hat{s}_j)$ e $m^+(\hat{j},\hat{s}_j)$. Nótese que a maximización do incremento de impureza é equivalente á minimización do substraendo de (1.11), pois o sumando $Q(m)$ é constante e non depende de s nin de j.

Proceso de poda

Para desenvolver esta sección, é preciso ampliar o concepto de impureza a unha árbore. Primeiramente, denotamos o conxunto dos nodos terminais dunha árbore T por $\mathrm{Ter}(T)$. A súa impureza definímola como

$$Q(T) = \sum_{m \in \mathrm{Ter}(T)} N_m \, Q(m).$$

A idea da poda é obter unha árbore máis pequena que a orixinal, de xeito que se reduza o problema do sobreaxuste, facéndose ó mesmo tempo máis interpretable. Sen embargo, para iso non se pode empregar soamente o criterio da impureza, pois este só favorece árbores grandes. É necesario incluír un termo que teña conta do tamaño da árbore, de xeito que se equilibre co anterior. Defínese a complexidade dunha árbore T como o número de nodos terminais que ten,

$$C(T) = \#\,\mathrm{Ter}(T).$$

Xuntando a impureza coa complexidade, multiplicada por un factor α, constrúese o chamado criterio *cost-complexity*:

$$C_\alpha(T) = Q(T) + \alpha C(T).$$

Referímonos por T_α á subárbore da árbore orixinal, T_0, que minimiza $C_\alpha(T)$. O método usado para a súa obtención coñécese como *weakest link pruning*, que de seguido desenvolvemos.

Dado un nodo m, definimos a árbore podada polo nodo m como T'_m. Isto quere dicir que dito nodo se converte en terminal, agrupando todas as observacións que quedaban por debaixo del. Á parte podada, considerada como unha árbore con nodo raíz m, denotámola por B_m (de *branch*, rama). É doado percatarse de que a impureza da árbore resultado de podar non é máis que a impureza da árbore orixinal, menos a suma das impurezas dos nodos terminais que "colgan" do m, máis a impureza deste mesmo nodo, que pasa a ser terminal:

$$Q(T'_m) = Q(T_0) + N_m \, Q(m) - \sum_{l \in \mathrm{Ter}(B_m)} N_l \, Q(l).$$

Polo propio concepto de impureza, tense sempre que $Q(T'_m) \geq Q(T_0)$. Por outra banda, o número de nodos terminais redúcese en $\#\mathrm{Ter}(B_m) - 1$ ó aplicar a poda; na mesma cantidade redúcese a complexidade:

$$C(T'_m) = C(T_0) + 1 - \#\mathrm{Ter}(B_m).$$

O método do *weakest link pruning* consiste en localizar o nodo m para o cal o cociente

$$\frac{Q(m) - \sum_{l \in \mathrm{Ter}(B_m)} N_l \, Q(l)}{\#\mathrm{Ter}(B_m) - 1}$$

é menor (enlace máis débil), e podalo. A nova árbore podada, T^1, pasaría a xogar o papel de T_0, e repetiríase o proceso ata chegar ó nodo raíz. Deste xeito obtense unha sucesión finita de subárbores $T_0 \supset T^1 \supset \cdots \supset T^b$. Pódese probar que a árbore T_α que minimiza $C_\alpha(T)$ está contida na devandita sucesión (para estas afirmacións, en [7] referénciase a [2] e [17]). A selección do parámetro α faise por validación cruzada k-fold, xeralmente empregando $k = 10$.

1.4.2. Problema da inestabilidade

Debido a súa natureza xerárquica, as árbores presentan unha alta variabilidade: modificando lixeiramente a mostra, pódese obter un resultado totalmente diferente. Se unha partición dun nodo na parte superior da árbore cambia, esta modificación vai ser arrastrada ata ó final, modificando toda a rama que colga de dito nodo.

Dende logo, este comportamento non é desexable, e limita a fiabilidade e interpretabilidade que se poida extraer das árbores de clasificación. Para aliviar este problema, xurdiron distintos algoritmos, entre os que podemos citar o *bagging* e o *random forest*.

Ademais, existe outra técnica, coñecida como *boosting*, que conseguiu, a partir de árbores de clasificación, obter precisións realmente altas. Esta técnica non entra nos obxectivos deste traballo, mais cómpre citala, pois moitos dos métodos de clasificación son comparados precisamente con respecto o boosting. Destaca o algoritmo AdaBoost, proposto por Freund e Schapire en 1999 [6].

O bagging (*bootstrap aggregating*) foi proposto en 1996 por Leo Breiman. A idea, en liñas xerais, consiste en *promediar* unha serie de árbores, de xeito que se reduza a súa varianza. Máis en detalle, constrúense B réplicas bootstrap do conxunto de adestramento orixinal, e con cada unha estímase unha árbore de clasificación. Finalmente, para unha nova entrada, todas as árbores votan, e o resultado é a clase máis votada.

Un sinxelo símil cunha variable aleatoria unidimensional permite ver que, aínda que deste xeito se pode reducir a varianza, a correlación entre as distintas árbores limita este efecto. A varianza da media de B variables aleatorias independentes e identicamente distribuídas, con varianza σ^2, é σ^2/B. Sen embargo, se estas presentan unha certa correlación positiva ρ entre elas, a varianza resultante é

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2.$$

Deste xeito, por moito que aumente B, o termo $\rho\sigma^2$ non se ve afectado, mantendo unha varianza considerable. O obxectivo co que se plantexa o Random Forest é, precisamente, o de diminuír a correlación entre as árbores, tratando ó mesmo tempo de non aumentar en gran medida a varianza de cada unha.

1.4.3. Forest-RI

A definición dada por Breiman para os Random Forests é moi ampla, mais na bibliografía adóitase chamar así a un caso particular, o Forest-RI (*Random Input*). Este caracterízase por ser unha colección de árbores, formadas a partir de remostras bootstrap, nas cales para o proceso de separación dos nodos só se consideran algunhas variables, seleccionadas aleatoriamente de cada vez. Esta aleatorización é a que reduce a correlación entre ás árbores. Cantas menos variables se consideren á hora de particionar os nodo, menor será a correlación, así como a precisión de cada árbore. Sen embargo, experimentalmente obsérvase que, aínda con moi poucas variables, a perda de precisión das árbores non é especialmente daniña para o resultado final [7].

Ademais, as árbores nunca se podan; constrúense ata acadar un certo número de observacións por nodo, e déixanse así. Nalgúns casos, dito número pode chegar a ser un, é dicir, esténdese a árbore ata o máximo posible.

A continuación detállase o algoritmo:

1. Dende $b = 1$ ata B:
   a) Faise unha réplica bootstrap Z^*, de igual tamaño que a mostra orixinal, n.
   b) Constrúese unha árbore T_b empregando Z^*, como se indica a continuación: Á hora de realizar a separación en cada nodo, selecciónanse aleatoriamente m variables das p orixinais. Búscase entre elas a que resulta nunha mellor separación, en termos da función de impureza que se queira, e créanse os dous nodos fillos. Repítese este proceso ata acadar un tamaño de nodo igual ou inferior a n_{max}.
2. A clasificación dunha nova observación x realízase mediante o voto maioritario das B árbores así construídas.

Os valores recomendados para o número de variables selecciondas, m, e o tamaño mínimo de nodo, n_{max}, son respectivamente, segundo o propio Breiman, a parte enteira de $\sqrt{p}$, e 1. Sen embargo, en [7] amósanse algúns exemplos nos que outros valores dan mellores resultados.

1.4.4. Estimacións Out of Bag

O concepto das estimacións Out of Bag é moi similar ó da validación cruzada, e pode ser empregado cando se traballa con métodos bootstrap. Á hora de calcular o erro de predición para unha certa observación x_i, empréganse unicamente ás árbores correspondentes ás remostras bootstrap nas cales non aparece x_i. Deste xeito, estas predicións non están influenciadas pola propia observación.

Á diferenza da validación cruzada, este procedemento non implica recalcular o clasificador cun novo conxunto de variables, o cal pode reducir en gran medida o tempo de cómputo.

Á parte da estimación do erro de xeneralización, tamén se pode aplicar o mesmo concepto para medir a importancia dunha certa variable x_j. Primeiramente, calcúlase o erro como se indicou previamente. A continuación, modifícanse aleatoriamente valores de x_j, en todas as observacións de todas as remostras, e vólvese a calcular o erro do mesmo xeito. O incremento porcentual que se produce no erro tras alterar deste xeito a variable x_j é unha medida da súa importancia no modelo. Facendo isto para todas as variables, pódese ver cales son as máis influíntes. Cómpre destacar que os resultados de aquí obtidos non permiten determinar o efecto da non existencia dunha variable, pois se se recalculase o modelo sen ela, outra podería ocupar o seu lugar (se estivese altamente correlada coa primeira, por exemplo).

1.5. Máquina de Vectores de soporte

Adicamos esta sección ó estudo das máquinas de vectores de soporte (en inglés *Support Vector Machine*, SVM). Comezamos presentando uns resultados sobre optimización, as condicións de Karush-Kuhn-Tucker, as cales nos permitirán entender a idea sobre a que repousa esta técnica de clasificación. De seguido, para chegar á definición formal de SVM, procedemos en tres etapas, seguindo a estrutura e nomenclatura empregada en [7]. Primeiramente expoñemos o problema do hiperplano de separación óptimo para dúas clases, cando estas son separables. A continuación, ampliamos a formulación anterior para o caso de clases non separables, definindo o *clasificador de vectores de soporte*. Na Sección 1.5.4 facemos un inciso para introducir a expansión de bases, un paradigma que se pode empregar con outras moitas técnicas de clasificación, pero que resulta imprescindible neste ámbito. Finalmente desenvolvemos a teoría da máquina de vectores de soporte.

1.5.1. Condicións de Karush-Kuhn-Tucker

Nesta sección presentamos as condicións de Karush-Kuhn-Tucker (KKT), que nos van permitir caracterizar os óptimos dun problema de optimización convexa. Posteriormente isto será de utilidade no desenvolvemento das máquinas de vectores de soporte. Os resultados aquí expostos están baseados, principalmente, en [10] e [1].

Consideremos o problema de optimización convexa dado por

$$\begin{cases} \text{máx} & f(x) \\ \text{s.a.} & g_i(x) \leq 0, \quad i = 1, \dots, m, \end{cases} \tag{1.12}$$

onde $x \in \mathbb{R}^n$, e $f, g_1, \dots, g_m$ son funcións convexas definidas en $\mathbb{R}^n$. Supoñamos que existe un x_0 para o cal se verifica que $g_i(x_0) < 0$ para toda g_i non linear (*condición de regularidade de Slater*).

Aínda que non vai intervir directamente nesta exposición, definimos o *Lagranxiano* asociado ó problema (1.12) como a aplicación $L : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ dada por

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x).$$

Referímonos ás compoñentes do vector $\lambda \in \mathbb{R}^m$ como *multiplicadores de Lagrange*.

Se supoñemos que as funcións $f, g_1, \dots, g_m$ son continuamente diferenciables, o óptimo do problema anterior (1.12) vén caracterizado polas condicións de Karush-Kuhn-Tucker. Un punto $\bar{x} \in \mathbb{R}^n$ é unha solución óptima se e só se existe un $\bar{\lambda} \in \mathbb{R}^m$, con $\bar{\lambda}_i \geq 0$, $i = 1, \dots, m$, tal que

$$\begin{aligned} &\text{(i)} \quad 0 = \nabla f(\bar{x}) + \sum_{i=1}^{m} \bar{\lambda}_i \nabla g_i(\bar{x}), \\ &\text{(ii)} \quad 0 = \bar{\lambda}_i g_i(\bar{x}), \quad i = 1, \dots, m. \end{aligned} \tag{1.13}$$

Nótese que a primeira das condicións equivale a que a derivada do Lagranxiano respecto do vector x sexa 0 en $(\bar{x}, \bar{\lambda})$, é dicir,

$$0 = \frac{\partial}{\partial x} L(\bar{x}, \bar{\lambda}).$$

1.5.2. Hiperplano separador óptimo

Un hiperplano Π no espazo $\mathbb{R}^p$ vén definido polo conxunto de puntos $x \in \mathbb{R}^p$ que verifican a ecuación

$$\beta_0 + \beta^t x = 0, \tag{1.14}$$

onde $\beta_0 \in \mathbb{R}$ e $\beta \in \mathbb{R}^p$. Cómpre mencionar que o que fan todas as técnicas de clasficación lineais é, en realidade, definir un hiperplano, e asignar unha clase a cada lado deste (en caso de ter unicamente dúas clases). Cada técnica busca o hiperplano óptimo segundo un certo criterio: supoñendo que os grupos seguen distribucións normais no caso do LDA, supoñendo o modelo loxístico para a regresión loxística, etc. No ámbito das SVM, o criterio de optimalidade que consideramos é a distancia mínima que deixa Π a todos os puntos.

A dirección do vector normal a Π vén dada por β, de xeito que o vector normal unitario é

$$n = \frac{\beta}{||\beta||}.$$

Sexa x_0 un punto de Π arbitrario. Dado outro punto calquera $x \in \mathbb{R}^p$, a súa distancia a Π é $d = |(x - x_0) \cdot n|$, ou sexa, a proxección do vector que une os dous puntos sobre a recta perpendicular ó hiperplano. Na Figura 1.2 móstrase o caso bidimensional. Tense que

$$d = |(x - x_0) \cdot n| = \left| (x - x_0) \cdot \frac{\beta}{||\beta||} \right| = \frac{|\beta^t x - \beta^t x_0|}{||\beta||}.$$

Dado que $x_0 \in \Pi$, x_0 debe verificar (1.14), e polo tanto $-\beta^t x_0 = \beta_0$. Séguese entón que

$$d = \frac{|\beta_0 + \beta^t x|}{||\beta||}.$$

Para cada observación x_i, definimos a cantidade y_i, que vale 1 se x_i pertence ó grupo 1, e -1 se pertence ó grupo 2. Sen perda de xeneralidade, podemos supoñer (ou máis ben impoñer) que o grupo 1 está situado na parte "positiva" do hiperplano, é dicir, que todos os seus puntos están no conxunto

$$\left\{ x \in \mathbb{R}^p \,/\, \beta_0 + \beta^t x > 0 \right\}.$$

Así, a cantidade

$$y_i \frac{\beta_0 + \beta^t x_i}{||\beta||} \tag{1.15}$$

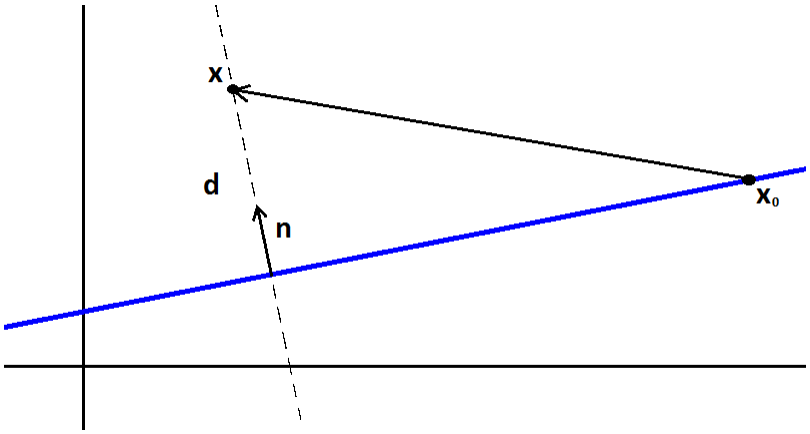

Figura 1.2: Distancia dun punto a unha recta en $\mathbb{R}^2$.

é precisamente a distancia de x_i a Π, pois o factor y_i corrixe o signo cando o outro termo é negativo.

O obxectivo é contruír un hiperplano que maximice a mínima distancia a todos os puntos, que denotamos por M. En linguaxe matemática, isto é

$$\begin{cases} \max\limits_{\beta,\beta_0} & M, \\ \text{s.a.} & y_i \dfrac{\beta_0 + \beta^t x_i}{||\beta||} \geq M, \quad i = 1, \dots, n. \end{cases} \tag{1.16}$$

Aquí é moi importante a hipótese sobre a separabilidade das dúas clases, pois se non se dese, o conxunto factible deste problema sería vacío. O hiperplano resultante deixará unha marxe de tamaño M ata o punto máis próximo por calquera dos dous lado, habendo polo tanto unha distancia mínima de $2M$ entre os puntos das dúas clases. Intuitivamente, cabe esperar que canto máis grande sexa esta marxe, menor erro haberá á hora de clasificar novas observacións.

As restricións presentes en (1.16) son equivalentes a

$$y_i\big(\beta_0 + \beta^t x_i\big) \geq M||\beta||, \quad i = 1, \dots, n.$$

Ademais, a norma de β non xoga ningún papel na determinación do hiperplano de separación, pois se β_0 e β satisfán as restrición anteriores, calquera múltiplo destes o farán tamén:

$$y_i \frac{(\alpha\beta_0) + (\alpha\beta^t)x_i}{||\alpha\beta||} = y_i \frac{\alpha\big(\beta_0 + \beta^t x_i\big)}{\alpha||\beta||} = y_i \frac{\beta_0 + \beta^t x_i}{||\beta||}.$$

Polo tanto, podemos fixar $||\beta||$ en calquera valor arbitrario, e o hiperplano obtido na solución óptima seguirá a ser o mesmo. Tomando $||\beta|| = 1/M$, o problema (1.16) tradúcese en

$$\begin{cases} \max\limits_{\beta,\beta_0} & \dfrac{1}{||\beta||}, \\ \text{s.a.} & y_i\big(\beta_0 + \beta^t x_i\big) \geq 1, \quad i = 1, \dots, n, \end{cases}$$

ou equivalentemente,

$$\begin{cases} \min\limits_{\beta,\beta_0} & ||\beta||^2, \\ \text{s.a.} & y_i\big(\beta_0 + \beta^t x_i\big) \geq 1, \quad i = 1, \dots, n. \end{cases} \tag{1.17}$$

Neste último problema de minimización, a función obxectivo é convexa e as restricións afíns, logo trátase de programación convexa. Por concordancia coa notación de [7], denotamos por α ós multiplicadores de Lagrange. O Lagranxiano vén dado por

$$L(\beta_0, \beta, \alpha) = ||\beta||^2 + \sum_{i=1}^{n} \alpha_i\big(1 - y_i\big(\beta_0 + \beta^t x_i\big)\big),$$

e en aplicación das condicións KKT (1.13), temos

$$\begin{aligned} &\text{(i)} \quad 0 = \frac{\partial L(\beta_0, \beta, \alpha)}{\partial(\beta_0, \beta^t)}, \\ &\text{(ii)} \quad 0 = \alpha_i\big(1 - y_i\big(\beta_0 + \beta^t x_i\big)\big), \quad i = 1, \dots, n. \end{aligned} \tag{1.18}$$

Procedemos por partes para obter a derivada do Lagranxiano. Primeiramente,

$$\frac{\partial}{\partial \beta_k}||\beta||^2 = \begin{cases} 0 & \text{se } k = 0, \\ 2\beta_k & \text{se } k \neq 0, \end{cases}$$

e analogamente

$$\frac{\partial}{\partial \beta_k}\beta^t x_i = \begin{cases} 0 & \text{se } k = 0, \\ x_{i(k)} & \text{se } k \neq 0. \end{cases}$$

Entón,

$$\frac{\partial L(\beta_0, \beta, \alpha)}{\partial \beta_0} = -\sum_{i=1}^{n} \alpha_i y_i,$$

$$\frac{\partial L(\beta_0, \beta, \alpha)}{\partial \beta_k} = 2\beta_k - \sum_{i=1}^{n} \alpha_i y_i x_{i(k)}, \quad k = 1, \dots, p.$$

Igualando a cero, obtemos a seguinte serie de ecuacións:

$$\sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\beta_k = \frac{1}{2}\sum_{i=1}^{n} \alpha_i y_i x_{i(k)}, \quad k = 1, \dots, p. \tag{1.19}$$

En canto á condición (ii) de (1.18), esta implica que, para cada observación x_i, ou ben o multiplicador de Lagrange é cero, ou ben $y_i(\beta_0 + \beta^t x_i) = 1$. Se lembramos a expresión da distancia dun punto ó hiperplano, (1.15), e a formulación do problema (1.17), dámonos de conta de que esta última situación dáse cando x_i está precisamente a unha distancia M de Π. É dicir, que α_i é distinto de cero unicamente para os puntos que caen na fronteira da marxe de separación.

Este feito é de vital importancia para entender o funcionamento das máquinas de sorpote vectorial. Se agora nos fixamos na expresión dos coeficientes do hiperplano, (1.19), podemos comprobar que só interveñen as observacións para as cales $\alpha_i \neq 0$. En conclusión, o hiperplano de separación óptimo vén

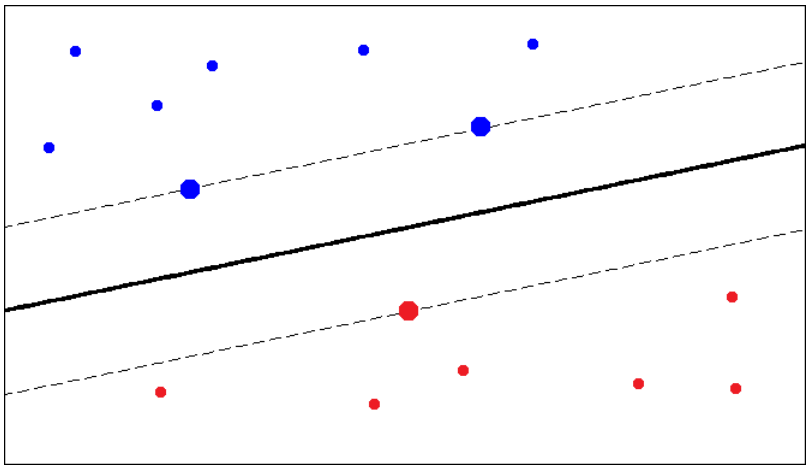

Figura 1.3: Hiperplano separador en $\mathbb{R}^2$. Liña negra grosa: hiperplano separador. Liñas discontinuas: límites das marxes. Puntos grosos: vectores de soporte.

determinado soamente polas observacións que están na fronteira da súa marxe, ás cales nos referimos como *vectores de soporte*.

Na Figura 1.3 amósase un exemplo no que se pon de manifesto este feito. É evidente que, aínda que se suprima calquera dos puntos que non caen sobre a liña discontinua, a recta de separación vai permanecer igual.

Unha vez que o problema de optimización (1.17) está resolto e o hiperplano separador determinado, a regra de clasificación vén dada polo signo da función

$$f(x) = \beta_0 + \beta^t x. \tag{1.20}$$

1.5.3. Clasificador de vectores de soporte

No caso de que os dous grupos non sexan separables, o problema (1.17) é non factible. Haberá osbervacións no lado incorrecto do hiperplano separador. Pensando nas restricións do problema orixinal, (1.16), que son en termos de distancias e resultan máis intuitivas, isto quere dicir que, para algún i,

$$y_i \frac{\beta_0 + \beta^t x_i}{||\beta||} < 0.$$

A introdución de variables de folgura nas restricións permite ter conta destes incumprimentos. A forma nas que estas quedan é a seguinte:

$$y_i \frac{\beta_0 + \beta^t x_i}{||\beta||} \geq M(1 - \xi_i), \quad \xi_i \geq 0, \quad i = 1, \dots, n.$$

Tomando, ó igual que antes, $||\beta|| = 1/M$, eliminamos a multiplicidade de solucións, resultando

$$y_i \left(\beta_0 + \beta^t x_i\right) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, \dots, n.$$

Por outra banda, a suma das variables de folgura incorpórase á función obxectivo, xunto cun parámetro de custo, C. O problema finalmente queda como

$$\begin{cases} \min\limits_{\beta,\beta_0,\xi} & ||\beta||^2 + C\sum\limits_{i=1}^{n}\xi_i, \\ \text{s.a.} & y_i\left(\beta_0 + \beta^t x_i\right) \geq 1 - \xi_i, \quad i = 1, \dots, n, \\ & \xi_i \geq 0, \quad i = 1, \dots, n. \end{cases} \tag{1.21}$$

Seguindo a nomenclatura de [7], referímonos a este problema como *clasificador de vectores de soporte*.

Cómpre notar que, se unha observación está mal clasificada, entón $\xi_i > 1$. Polo tanto, se limitamos a suma do vector ξ por unha constante K enteira, estamos restrixindo o número máximo de incumprimentos a K. De feito, en [7] introducen inicialmente o clasificador de vectores de soporte sen considerar a suma de ξ na función obxectivo, senón coa restrición adicional

$$\sum_{i=1}^{n}\xi_i \leq K.$$

Sen embargo, esta formulación fai que o problema non sexa factible no caso de que as dúas clases estean o suficientemente mesturadas.

Razoando de igual xeito que para o hiperplano separador óptimo, o Lagranxiano do problema (1.21) vén dado por

$$L(\beta_0, \beta, \alpha, \mu) = ||\beta||^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i\left(1 - \xi_i - y_i\left(\beta_0 + \beta^t x_i\right)\right) - \sum_{i=1}^{n}\xi_i,$$

onde α é o vector de multiplicadores de Lagrange asociados á primeira tanda de restricións, e cada μ_i é o multiplicador asociado á restrición $\xi_i \geq 0$. A aplicación das condicións KKT permite obter a forma da solución, que é análoga á do hiperplano separador óptimo.

A primeira destas é

$$0 = \nabla L(\beta_0, \beta, \alpha, \mu) = \frac{\partial}{\partial(\beta, \beta_0, \xi)} L(\beta_0, \beta, \alpha, \mu).$$

Esta expresión descomponse en $p+1+n$ ecuacións, unha para cada derivada parcial. Comezando polas de β, temos

$$0 = 2\beta_k - \sum_{i=1}^{n}\alpha_i x_{i(k)}, \ k = 1, \dots, p \quad \Rightarrow \quad \beta = \frac{1}{2}\sum_{i=1}^{n}\alpha_i x_i. \tag{1.22}$$

De seguido, a derivada respecto a β_0 implica

$$0 = \sum_{i=1}^{n}\alpha_i y_i.$$

Finalmente, derivando respecto a ξ, resulta

$$0 = C - \alpha_k - \mu_k \quad \Rightarrow \quad C = \alpha_k + \mu_k, \quad k = 1, \dots, n.$$

A segunda condición de Karush-Kuhn-Tucker permite obter as seguintes igualdades:

$$0 = \alpha_i\left(1 - \xi_i - y_i\left(\beta_0 + \beta^t x_i\right)\right), \quad i = 1, \dots, n,$$

$$0 = \mu_i \xi_i, \quad i = 1, \dots, n.$$

Nótese que α_i pode ser distinto de cero só se $1-\xi_i = y_i(\beta_0+\beta^t x_i)$, é dicir, se a restrición correspondente se verifica cunha igualdade. Xa que $\xi_i \geq 0$, tense que neste caso $y_i(\beta_0+\beta^t x_i) \leq 1$, o cal quere dicir que a observación x_i se atopa na marxe de separación, ou incluso fóra dela no lado incorrecto do hiperplano. Tendo en conta isto, así como a expresión de β, (1.22), chegamos a unha conclusión análoga á do caso do hiperplano separador óptimo: o clasificador de vectores de soporte só depende das observacións que non quedan perfectamente clasificadas. Evidentemente, a elección do parámetro C determinará qué observacións son estas.

1.5.4. Expansión de bases

A idea fundamental da expansión de bases é a de aumentar o número de variables explicativas, mediante a inclusión de transformacións das variables orixinais. En xeral, se dispoñemos de $X_1, \dots, X_p$, podemos aplicar unha transformación $h : \mathbb{R}^p \to \mathbb{R}^q$ para obter $X' = h(X)$, é dicir, $X'_1, \dots, X'_q$. Estas novas variables son as que se empregarían no axuste de calquera modelo.

Vexamos un caso particular. Supoñamos que temos as variables X_1 e X_2. A partir delas definimos $X_3 = X_1^2$, $X_4 = X_2^2$ e $X_5 = X_1X_2$. Se facemos un axuste linear sobre o novo conxunto de variables $(X_1, X_2, X_3, X_4, X_5)$, obtemos un modelo que é cadrático sobre X_1 e X_2. De feito, o modelo de regresión xeral obtense mediante a *expansión* do conxunto de variables orixinal.

Como vemos, unha importante aplicación desta técnica é a obtención de modelos non lineais, sen máis que axustar un modelo linear sobre unha base expandida.

1.5.5. Máquina de vectores de soporte

A máquina de vectores de soporte non é máis que unha extensión do clasificador de vectores de soporte mediante unha expansión de bases. Mediante a consideración de certas expansións determinadas, é posible aumentar considerablemente a dimensión do problema, obtendo a non linealidade, sen requerir un excesivo custo computacional.

Consideremos unha aplicación $h : \mathbb{R}^p \to \mathbb{R}^q$, que leva as p variables orixinais nas q que conforman a nova base. Considerando os vectores $h(x_i)$, $i = 1, \dots, n$, como observacións, o hiperplano definido polo clasificador de vectores de soporte, (1.22), queda como

$$\beta = \frac{1}{2}\sum_{i=1}^{n} \alpha_i h(x_i).$$

Substituíndo isto na función de clasificación (1.20), obtemos

$$f\big(h(x)\big) = \beta_0 + \beta^t h(x) = \beta_0 + \left(\frac{1}{2}\sum_{i=1}^{n} \alpha_i h(x_i)\right)^t h(x) =$$

$$\beta_0 + \frac{1}{2}\sum_{i=1}^{n} \alpha_i h(x_i)^t h(x) = \beta_0 + \frac{1}{2}\sum_{i=1}^{n} \alpha_i \big(h(x_i) \cdot h(x)\big).$$

Como se pode ver, a regra de clasificación só depende das novas variables a través dos seus produtos interiores. Polo tanto, é posible non definir a transformación h en si, senón soamente os ditos produtos. Isto faise a través do que se coñece como función núcleo,

$$K(x_1, x_2) = h(x_1) \cdot h(x_2).$$

O papel do parámetro de custo C apréciase máis claramente ó considerar a expansión de bases, xa que nun espazo expandido, habitualmente é posible conseguir unha separación perfecta entre as clases. Este sobreaxuste, aínda que reduciría o erro de adestramento a cero, levaría a erros de predición moito maiores. É polo tanto necesario regular a cantidade C de xeito que non se produza o denominado *overfit*. Usualmente, o tipo de núcleo utilizado é o Gaussiano, e inclúe un parámetro axustable extra, o seu ancho de banda, que denotamos por σ.

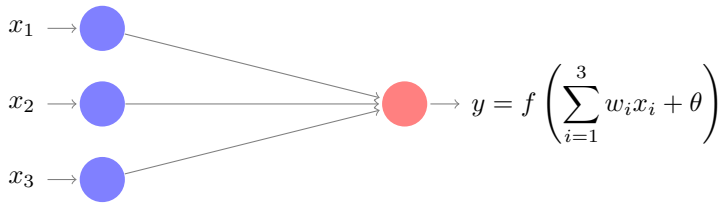

Figura 1.4: Esquema do funcionamento dunha neurona de McCulloch-Pitts.

1.6. Redes Neuronais Artificiais

O campo das Redes Neuronais Artificiais (RNA, ou ANN, do inglés *Artificial Neural Network*) engloba unha gran variedade de modelos, todos eles baseados no funcionamento das redes de neuronas biolóxicas, é dicir, nos cerebros dos animais.

Os primeiros traballos nesta disciplina datan da década dos 40, cando Warren McCulloch e Walter Pitts desenvolveron o modelo que leva o seu nome para o comportamento dunha neurona. A chamada neurona de McCulloch-Pitts, representada na Figura 1.4, recibe como entrada o sinal de varias neuronas e realiza unha suma ponderada dos seus valores. Denotamos por x_i e w_i cada unha das entradas e dos pesos, respectivamente. A idea fundamental é que a neurona se activa cando esta suma sobrepasa un certo valor. Para modelar isto, introdúcese un parámetro chamado limiar, θ, e suponse que o valor de excitación da neurona vén dado por

$$y = \phi\left(\sum_{i=1}^{n} w_i x_i + \theta\right).$$

A transformación ϕ coñécese como función de activación. Se esta toma soamente dous valores constantes, temos precisamente unha neurona que se activa ó acadar un certo valor. Se pola contra é continua, a saída da neurona dependerá gradualmente das entradas. É común empregar a función sigmoidal.

Mediante o uso de este tipo de células, é posible contruír unha gran cantidade de arquitecturas diferentes. Xa que este traballo non pretende centrarse unicamente nas redes neuronais artificiais, só abordaremos un caso concreto, o das redes Perceptrón. Este modelo, sendo un dos máis sinxelos, goza dun extenso uso, e serviu de inspiración para moitos outros.

1.6.1. Perceptrón simple

En 1957, o psicólogo Frank Rosenblatt presentou o modelo de rede neuronal coñecido como Perceptrón. Este consiste dunha serie de células de McCulloch-Pitts unidas a unhas entradas. O conxunto das entradas coñécese como *capa de entrada*, mentras que nos referimos ás neuronas como *capa de saída* (ver Figura 1.5). Na seguinte sección, cando estudemos o Perceptrón Multicapa, veremos como esta nomenclatura cobra máis sentido.

Evidentemente, o tamaño da capa de entrada será a dimensión do problema e, no ámbito da clasificación, o tamaño da de saída será o número de grupos, pois cada neurona modela a probabilidade de pertenza a unha clase.

Se denotamos por w_{ij} o peso da conexión entre a entrada i e a neurona j, e por θ_j o limiar desta última, a saída da rede vén dada por

$$y_j = \phi\left(\sum_{i=1}^{p} w_{ij} x_i + \theta_j\right), \quad j = 1, \ldots, K, \tag{1.23}$$

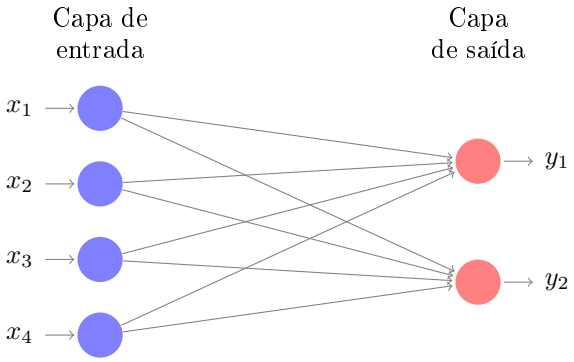

Figura 1.5: Perceptrón simple con catro entradas e dúas saídas.

onde p é o número de entradas, e K o de grupos. O argumento de ϕ denotámolo por η_j,

$$\eta_j = \sum_{i=1}^{p} w_{ij} x_i + \theta_j.$$

De igual xeito que no caso da regresión loxística, clasificaremos unha observación x no grupo para o cal y_j é maior. Xa que a función de activación ϕ é a mesma en todos os casos, se supoñemos que esta é monótona crecente, tense que o anterior equivale a asignar o grupo para o cal se maximiza o seu argumento, ou sexa,

$$\hat{G}(x) = \operatorname{argmax} \begin{pmatrix} \eta_1 \\ \vdots \\ \eta_K \end{pmatrix}. \tag{1.24}$$

A expresión (1.24) pon de manifesto o carácter linear do Perceptrón. De feito, se ollamos a forma da súa saída, (1.23), é doado percatarse de que se trata simplemente dun GLM (*Generalized Linear Model*), onde η é o predictor linear, e ϕ^{-1} a función de enlace.

Este feito evidentemente implica que hai problemas que non son resolubles con esta técnica, como a función lóxica XOR (é un problema bidimensional non separable por unha recta), o cal supuxo un estancamento no desenvolvemento do campo das redes de neuronas artificiais. Non sería ata a chegada do Perceptrón multicapa cando este inconveniente se solventaría, devolvendo o pulo á investigación neste ámbito.

1.6.2. Perceptrón multicapa

O Perceptrón multicapa (MLP, *Multi-Layer Perceptron*) caracterízase por ter as neuronas agrupadas en diferentes niveis. A maiores das capas de entrada e saída, incorpora outras intermedias, coñecidas como *capas ocultas*. Na Figura 1.6 pódese ver o esquema dun Perceptrón multicapa cunha capa oculta.

Polo xeral, cada capa intermedia está totalmente conectada ás contiguas, é dicir, cada unha das súas neuronas recibe como entrada a todas as da capa anterior, e a súa saída é propagada a todas as da seguinte. Neste caso falamos de conectividade total. Sen embargo, existen implementacións nas que se eliminan certas conexións. Diminuíndo o número de parámetros axustables (cada conexión leva asociado un peso), é posible obter un modelo máis eficaz para algún problema en concreto.

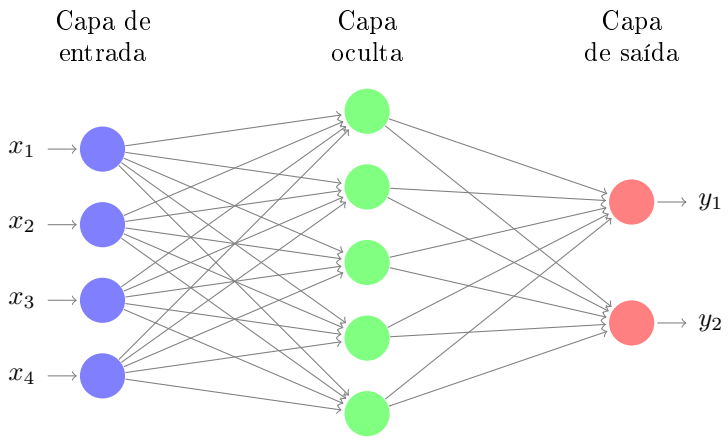

Figura 1.6: Perceptrón multicapa con unha capa oculta.

O feito de que as neuronas de saída non vexan directamente a entrada, senón que esta sexa transformada a través das capas ocultas, con independencia de cantas sexan estas (sempre que haxa alomenos unha), dota ó Perceptrón multicapa da capacidade de resolver problemas non lineais. Diferentes autores demostraron que o Perceptrón multicapa é un aproximador universal, é dicir, que pode aproximar calquera función continua sobre un conxunto compacto de $\mathbb{R}^n$ [8]. Isto permite, por exemplo, resolver o problema da función lóxica XOR.

Antes de comezar co seu desenvolvemento formal, é conveniente introducir algo de notación. Chamaremos C ó número total de capas (sen contar a de entrada, que será a capa 0). A cantidade de neuronas presentes na capa d será representada por m_d. O peso da conexión entre a neurona i da capa $d-1$ e a j da d, será w_{ij}^d. Ampliando esta notación, chamaremos w_{0j}^d ó limiar da neurona j da capa d (deste xeito, estamos a consideralo como unha entrada máis). O valor de activación desta mesma neurona virá dado por z_j^d. Para poder obter unha expresión compacta deste valor, como veremos a continuación, definimos tamén $z_0^d = 1$, para calquera d.

En principio, a función de activación pode ser distinta en cada capa (habitualmente, nas ocultas ten unha forma, e na de saída outra), de xeito que denotamos por ϕ_d a correspondente á capa d.

$$z_j^d = \phi_d \left(\sum_{k=0}^{m_{d-1}} w_{kj}^d z_k^{d-1} \right) \tag{1.25}$$

Chamamos η_j^d á entrada que recibe a neurona j da capa d, que é a suma ponderada das saídas da capa anterior, xunto co limiar,

$$\eta_j^d = \sum_{k=0}^{m_{d-1}} w_{kj}^d z_k^{d-1}.$$

Así, resulta a seguinte expresión para o valor de activación desta mesma neurona,

$$z_j^d = \phi_d \left(\eta_j^d \right).$$

De aquí dedúcense un par de resultados triviais, que máis adiante serán de utilidade:

$$\frac{\partial z_j^d}{\partial \eta_j^d} = \phi_d' \left(\eta_j^d \right), \quad \frac{\partial \eta_j^d}{\partial w_{ij}^d} = z_i^{d-1}.$$

Nótese que, para que a última igualdade sexa certa, ambas variables η_j^d e w_{ij}^d deben estar referidas á mesma capa d. Noutro caso, é necesario considerar a derivada de todos os sumandos presentes no sumatorio que define η_j^d.

Para entender o mecanismo de propagación dos sinais de entrada que ten lugar no Perceptrón multicapa, consideremos unha rede con p entradas, unha capa oculta de q neuronas, e unha saída de tamaño K (na Figura 1.6 móstrase un caso particular). A compoñente k-ésima da saída da rede será

$$\begin{aligned} y_k = & \quad \phi_2 \left(\sum_{j=1}^{q} w_{jk}^2 z_j^1 + w_{0k}^2 \right) \\ = & \quad \phi_2 \left(\sum_{j=1}^{q} w_{jk}^2 \, \phi_1 \left(\sum_{i=1}^{p} w_{ij}^1 x_i + w_{0j}^1 \right) + w_{0k}^2 \right). \end{aligned} \tag{1.26}$$

Esta expresión pon de manifesto que, en resumidas contas, o proceso que ten lugar na capa oculta non é máis que unha expansión de bases. A capa de saída, en lugar de ver as variables orixinais, ve unha transformación dunha combinación linear delas. Se definimos a función $h : \mathbb{R}^p \to \mathbb{R}^q$ dada por

$$h : X \rightsquigarrow h(X) = (z_1^1, \dots, z_q^1),$$

resulta que a saída (1.26) é a mesma que a dun Perceptrón simple sobre as variables $h_1(X), \dots, h_q(X)$,

$$y_k = \phi_2 \left(\sum_{j=1}^{q} w_{jk}^2 h_j(X) + w_{0k}^2 \right).$$

No caso de haber máis dunha capa oculta, é fácil ver que cada unha suporía unha nova expansión de bases.

En conclusión, tendo en conta o comentado na sección anterior sobre a similitude do Perceptrón simple cos GLM, podemos afirmar que o Perceptrón multicapa é equivalente a un GLM sobre un conxunto de variables expandido. Este tipo de modelos coñécense, en inglés, como *Projection Pursuit Regression model* (*PPR*). No caso de ter varias capas ocultas, todas elas en conxunto poden ser representadas como unha única expansión de bases.

Sen embargo, cómpre destacar unha propiedade importante do Perceptrón: este, a través do algoritmo de adestramento que presentamos na seguinte sección, calcula implicitamente a expansión de bases (ou sexa, a función h). Baseándose nos datos dispoñibles e nun certo criterio de erro, este algoritmo busca a transformación máis axeitada, dentro dunha familia que vén determinada pola súa estrutura (o número de capas, o número de neuronas de cada capa, e a función de activación). Polo tanto, a determinación da expansión de bases non é algo que teña que facer o usuario.

1.6.3. Algoritmo de Retropropagación

O algoritmo de Retropropagación (*Backpropagation*, en inglés) permitiu, como comentabamos nas seccións anteriores, o paso do Perceptrón simple ó multicapa. Isto fixo posible a resolución de problema non lineais, e supuxo un pulo á investigación no campo das redes neuronais. A súa popularización veu, principalmente, da man dos psicólogos David Rumelhart e James McClelland, cando o incluíron no libro *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, en 1986.

Este algoritmo non é máis que o método do gradiente aplicado á estrutura particular do Perceptrón multicapa. Denotemos por ξ o conxunto de todos os parámetros da rede (limiares máis pesos das conexións). Dado un iterante inicial ξ^1, defínese a sucesión

$$\xi^{i+1} = \xi^i - \alpha_i \nabla E\left(\xi^i\right), \quad i \in \mathbb{N}, \tag{1.27}$$

onde α_i é un parámetro que regula a velocidade de converxencia (taxa de aprendizaxe), e $E(\xi^i)$ o erro cometido pola rede para os parámetros ξ^i. Este método converxe a un mínimo local da función de erro. A cuestión de se este mínimo é global, moi importante neste ámbito, depende en gran medida do iterante inicial ξ^1.

O gradiente do erro ten a seguinte forma:

$$\nabla E\left(\xi^i\right) = \begin{pmatrix} \dfrac{\partial E\left(\xi^i\right)}{\partial \xi_1} \\ \\ ... \\ \\ \dfrac{\partial E\left(\xi^i\right)}{\partial \xi_M} \end{pmatrix},$$

sendo M o número total de parámetros. Polo tanto, a ecuación vectorial (1.27) transfórmase, para cada compoñente, en

$$\xi_j^{i+1} = \xi_j^i - \alpha_i \frac{\partial E\left(\xi^i\right)}{\partial \xi_j}, \quad j = 1, ..., M.$$

É dicir, cada parámetro actualízase a través da derivada parcial do erro respecto a dito parámetro. Por outra banda, en todo o que segue, asumiremos que o erro que consideramos é o erro cadrático medio. Así, este pódese descompoñer como a dobre suma dos erros cometidos para cada observación e compoñente,

$$E(\xi^i) = \frac{1}{n} \sum_{l=1}^{n} \sum_{k=1}^{K} \big(g_k(x_l) - y_k(x_l)\big)^2,$$

onde $g_k(x_l)$ é 1 se a observación pertence á clase k, e 0 noutro caso, e $y_k(x_l)$ é o valor proporcionado pola neurona k-ésima da capa de saída para x_l. Podemos denotar o erro cadrático para cada observación por

$$e_l^i = \sum_{k=1}^{K} \big(g_k(x_l) - y_k(x_l)\big)^2,$$

resultando

$$E(\xi^i) = \frac{1}{n} \sum_{l=1}^{n} e_l^i$$

e

$$\frac{\partial E\left(\xi^i\right)}{\partial \xi_j} = \frac{1}{n} \sum_{l=1}^{n} \frac{\partial e_l^i}{\partial \xi_j}.$$

Deste xeito, cada paso do método do gradiente antes descrito require o coñecemento da derivada, con respecto a calquera parámetro, do erro cadrático cometido para unha soa observación.

A partir deste punto, comezan a aparecer cantidade de subíndices, e faise complicado mantelos todos nas contas. Por claridade, é preferible esquecerse dalgunhas referencias, sen que isto teña que inducir a ningún erro. En particular, obviaremos índices i e l que determinan, respectivamente, o iterante ξ^i e a observación x_l. De feito, estas mesmas letras aparecerán máis adiante con distinto significado. Polo tanto, estudaremos simplemente as derivadas do erro cadrático para unha observación calquera, que denotaremos por e.

O conxunto dos parámetros da rede pode descompoñerse en dous grupos: os pesos das conexións que chegan á capa de saída (capa C), xunto cos seus limiares, e o resto de pesos e limiares *ocultos*. No caso dun dos parámetros do primeiro grupo, a derivada respecto deste é doada de calcular, pois só unha das saídas se ve afectada por el ($\partial y_k / \partial w_{ij}^C = 0$ se $k \neq j$),

$$\frac{\partial e}{\partial w_{ij}^C} = 2 \sum_{k=1}^{K} \big(y_k - g_k\big) \frac{\partial y_k}{\partial w_{ij}^C} = 2\big(y_j - g_j\big) \frac{\partial y_j}{\partial w_{ij}^C}.$$

Tendo en conta a función de activación da neurona j da capa de saída, (1.25), resulta

$$\frac{\partial e}{\partial w_{ij}^C} = 2\big(y_j - g_j\big) \phi_C' \left(\sum_{l=0}^{m_{C-1}} w_{lj}^C z_l^{C-1} \right) z_i^{C-1}.$$

No caso de ter que derivar o erro respecto dun peso entre capas ocultas, os cálculos tórnanse un pouco máis tediosos. De feito, hai que ser especialmente coidadoso coa notación, pois rapidamente aparecen cantidade de subíndices, e faise necesario prescindir dalgúns (como comentamos anteriormente, non estamos a indicar o iterante ξ^i de que se trata).

Denotaremos a entrada que recibe unha neurona por η, que é a suma ponderada dos valores de activación das neuronas da capa anterior. No caso xeral da capa d e a neurona i, esta cantidade é

$$\eta_i^d = \sum_{l=0}^{m_{d-1}} w_{li}^d z_l^{d-1}. \tag{1.28}$$

Así, para calquera neurona, o seu valor de activación será

$$z_i^d = \phi_d(\eta_i^d),$$

onde ϕ_d é a función de activación da capa d.

Empregando a regra da cadea, é doado ver que

$$\frac{\partial e}{\partial w_{ij}^d} = \frac{\partial e}{\partial z_j^d}\frac{\partial z_j^d}{\partial \eta_j^d}\frac{\partial \eta_j^d}{\partial w_{ij}^d}. \tag{1.29}$$

O primeiro termo, tamén coa regra da cadea, podémolo reescribir en función dos valores da capa seguinte,

$$\frac{\partial e}{\partial z_j^d} = \sum_{l=0}^{m_{d+1}} \frac{\partial e}{\partial z_l^{d+1}}\frac{\partial z_l^{d+1}}{\partial z_j^d} = \sum_{l=0}^{m_{d+1}} \frac{\partial e}{\partial z_l^{d+1}}\frac{\partial z_l^{d+1}}{\partial \eta_l^{d+1}}\frac{\partial \eta_l^{d+1}}{\partial z_j^d}.$$

De (1.28) dedúcese que $\partial \eta_l^{d+1}/\partial z_j^d = w_{jl}$, logo

$$\frac{\partial e}{\partial z_j^d} = \sum_{l=0}^{m_{d+1}} \frac{\partial e}{\partial z_l^{d+1}}\frac{\partial z_l^{d+1}}{\partial \eta_l^{d+1}} w_{jl}.$$

Definimos

$$\delta_l^{d+1} = \frac{\partial e}{\partial z_l^{d+1}}\frac{\partial z_l^{d+1}}{\partial \eta_l^{d+1}}.$$

Esta cantidade representa, en certo modo, a propagación do erro a partir da neurona l da capa $d+1$, que é do que se *alimenta* a derivada do erro da capa d. Resulta

$$\frac{\partial e}{\partial z_j^d} = \sum_{l=0}^{m_{d+1}} \delta_l^{d+1} w_{jl}$$

Repetindo este razoamento para a capa $d-1$, teriamos que

$$\delta_j^d = \frac{\partial e}{\partial z_j^d}\frac{\partial z_j^d}{\partial \eta_j^d} = \frac{\partial z_j^d}{\partial \eta_j^d}\sum_{l=0}^{m_{d+1}} \delta_l^{d+1} w_{jl}$$

Obsérvase como os δ de cada capa se constrúen recursivamente cos da capa seguinte, o cal permite calcular todas as derivadas do erro de xeito sinxelo. Substituíndo todo isto na expresión (1.29), temos finalmente unha expresión moi compacta para a derivada do erro respecto de calquera peso da rede,

$$\frac{\partial e}{\partial w_{ij}^d} = \delta_j^d \frac{\partial \eta_j^d}{\partial w_{ij}^d} = \delta_j^d z_i^{d-1}.$$

Ademais, débese ter en conta que

$$\frac{\partial z_j^d}{\partial \eta_j^d} = \phi_d'(\eta_j^d),$$

que é a derivada da función de activación da capa d.

Capítulo 2

Clasificación con Datos Funcionais

Entendemos por variable aleatoria funcional aquela que non toma valores numéricos nin categóricos, senón que as súas realizacións son funcións. Cada observación de dita variable é unha función real de variable real. As curvas dadas polas temperaturas medias ó longo do ano nun lugar concreto son un exemplo. Outro podería ser a enerxía absorbida por unha certa substancia en función da lonxitude de onda da radiación que recibe. Aínda que na práctica os datos que se teñen para estas situacións son multivariantes (ou sexa, unha discretización da curva en moitos puntos), é conveniente utilizar un enfoque que teña conta do feito de que os datos son funcións, e non simplemente vectores numéricos.

De igual xeito que no ámbito dos datos multivariantes, a clasificación supervisada con datos funcionais é de gran importancia. Na bibliografía poden atoparse diversas técnicas para este fin. Aínda que neste traballo só utilizamos unha, citamos a continuación algunhas delas:

Unha posibilidade é empregar o concepto de profundidade (tamén existente no caso multivariante). Entre as definicións máis habituais de profundidade destacan a de Fraiman e Muñiz, as *Random Projection Depths* e a *Band Depth*. Esta magnitude escalar representa unha medida do inmersa que está unha certa curva nunha clase. A clasificación realízase simplemente elixindo a clase para a cal a profundidade é maior. No caso de haber só dúas clases, é posible construír o chamado *D-D plot*, que consiste nunha gráfica onde cada eixo se corresponde coa profundidade respecto dunha clase. Este tipo de representación pode achegar información moi interesante.

Outra alternativa consiste en definir unha distancia entre cada par de curvas. Dependendo do tipo de datos cos que se traballe, será conveniente definir unha métrica ou outra. Cómpre citar a de Hausdorff, a do supremo, e a de Lebesgue, se o espazo funcional é $\mathcal{L}^2$. Unha vez definidas as distancias, é posible empregar algúns dos clasificadores multivariantes que só precisan este dato, como o K - Nearest Neighbors, por exemplo.

Por último, outra das técnicas dispoñibles, a cal é precisamente a que empregamos neste traballo, é a representación dos datos nunha base, e o uso dos coeficientes nela con clasificadores multivariantes.

No ámbito dos datos funcionais (e, en xeral, na estatística) non é posible falar da "mellor" técnica. Cada unha presenta mellores resultados nuns casos, e peores noutros. Polo tanto, á hora de estudar a fondo un conxunto de datos en particular, é preciso analizar cal é o mellor enfoque. É máis, aínda tendo claro a metodoloxía a usar, segue a ser preciso determinar outro tipo de parámetros, como o tipo de profundidade, a distancia, ou a familia de bases e o número de elementos delas.

Tamén cómpre comentar que, ás veces, a información relevante non se atopa nas propias curvas orixinais, senón nas súas derivadas. Neste traballo analizaremos un exemplo no que isto acontece.

2.1. Representación de datos funcionais en bases

Un sistema de bases, no ámbito dos datos funcionais, é un conxunto de funcións $\phi_i : \mathbb{R}^p \to \mathbb{R}$, con i pertencente a un conxunto de índices I arbitrario, tais que calquera función $f : \mathbb{R}^p \to \mathbb{R}$ pode ser

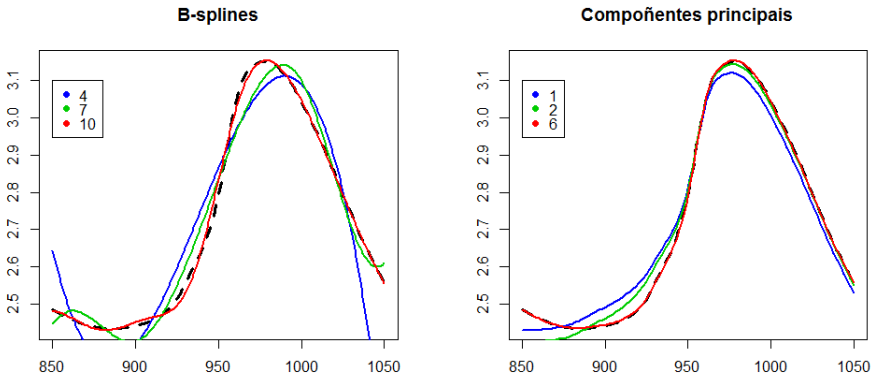

Figura 2.1: Representación dunha curva do conxunto de datos *tecator* en bases, usando B-splines e compoñentes principais. As cantidades indicadas nas lendas son os números de elementos das bases en cada caso. A liña grosa discontinua é a curva orixinal.

aproximada, cun erro tan pequeno como se queira, por unha combinación linear das funcións ϕ_i.

Na práctica, consideramos bases finitas, é dicir, só algúns elementos do conxunto $\{\phi_i\}_{i\in I}$, digamos $B = \{\phi_1, \dots, \phi_s\}$. Unha función f arbitraria será aproximada por

$$c_1\phi_1 + \cdots + c_s\phi_s,$$

e chamaremos a $c_1, \dots, c_s$ os coeficientes de f na base B. Evidentemente, en xeral as aproximacións serán mellores canto máis grande sexa s (aínda que esta relación non é constante: unha certa función pode ser mellor aproximada cunha base de tamaño s que cunha de $s+1$). Na Figura 2.1 pódense ver as aproximacións obtidas, dunha certa curva, con bases de distinto tamaño e tipo.

Unha das vantaxes da representación en bases é pasar dun problema de dimensión infinita (o espazo das funcións), a un de dimensión finita, sendo esta o número de elementos considerados na base. Os coeficientes pasan a identificar calquera función f, e poden ser usados por clasificadores multivariantes.

Unha cuestión de suma importancia é o tipo de base escollido. Existe unha gran variedade, cada un imitando certas propiedades que sabemos que teñen os nosos datos (periodicidade, derivabilidade, etc.). Neste traballo consideramos dous sistemas de bases distintos: os B-splines e as bases de compoñentes principais. A continuación facemos un breve resumo sobre elas, baseándonos principalmente no libro [16], onde se poden atopar os detalles.

Bases de B-splines

As funcións *spline* son unhas das máis usadas con datos non periódicos (noutro caso adóitanse empregar as series de Fourier). Están formadas por polinomios, dun certo grao g, unidos en certos nodos. Nas unións, presérvase a continuidade e a diferenciabilidade ata o grao $g-1$. A orde m dun polinomio, que é o número de constantes necesarias para definilo, é o seu grao máis un, $m = g+1$. Os graos de liberdade no axuste dun spline é a orde dos seus polinomio máis o número de nodos interiores.

Nós consideramos os splines máis habituais, que son os de orde 4. É dicir, son polinomios cúbicos unidos preservando a continuidade e a diferenciabilidade ata grao 2.

Aínda que aquí non entramos nos detalles, existen diferentes sistemas de bases de splines. Son conxuntos de funcións capaces de reproducir calquera spline mediante combinacións lineais. Neste traballo empregamos o sistema máis habitual, as bases de B-splines. Entre outras cualidades, estas son especialmente eficientes computacionalmente.

Bases de compoñentes principais

No caso das compoñentes principais en datos multivariantes, cada compoñente é un vector cos coeficientes da combinación linear das variables de partida. No contexto funcional, as compoñentes principais son funcións $\xi(s)$, e os produtos interiores cos datos $x(s)$ pasan a estar definidos por

$$\int \xi(s)x(s)ds.$$

Defínese a primeira compoñente principal, ξ_1, como a función que maximiza

$$\frac{1}{n}\sum_{i=1}^{n}\left(\int \xi_1(s)x_i(s)ds\right)^2,$$

suxeita a restrición

$$\int \xi_1(s)^2 ds = 1.$$

Para a segunda compoñente principal requírese, analogamente ó caso multivariante, que esta sexa ortogonal á primeira, ou sexa,

$$\int \xi_1(s)\xi_2(s)ds = 0.$$

O mesmo se aplica para calquera das seguintes compoñentes ξ_i, $i = 3, \ldots$, con respecto ás anteriores a ela.

As bases de compoñentes principais teñen a propiedade de depender de cada conxunto de datos, á diferenza doutras como as de B-splines nas que as súas funcións teñen sempre a mesma forma. Ademais, polo xeral recollen a información relevante dunha maneira moito máis eficiente que outras bases. Voltando á Figura 2.1, podemos apreciar como, cun número considerablemente menor de elementos, as aproximacións obtidas con bases de compoñentes principais son claramente mellores que as de B-splines.

Capítulo 3

Deseño Experimental

O deseño dun correcto prodemento experimental é fundamental se se queren inferir uns resultados valiosos. É preciso asegurarse de que todas as bases de datos e todos os modelos son tratados en igualdade de condicións, pois doutro xeito sería imposible realizar un comparación xusta.

O obxecto deste capítulo é precisamente o detalle deses aspectos. Comezamos presentando as bases de datos escollidas. Por un lado, temos nove conxuntos extraídos do respositorio da UCI (*UCI Machine Learning Repository*, [14]) e, por outro, tres conxuntos de datos funcionais, incluídos en varios paquetes de R. Tamén explicamos o código que empregamos para a posta en funcionamento dos clasificadores, que basicamente son diversas funcións de R. A continuación, detallamos a metodoloxía seguida para a sintonización de parámetros, e como estimamos a precisión do modelo finalmente seleccionado.

3.1. Bases de datos

Para poñer en funcionamento os métodos de clasificación expostos no Capítulo 1, escollemos nove conxuntos de datos multivariantes do repositorio da UCI. Estes teñen propiedades moi diferentes entre si: variables continuas e categóricas, distintos número de clases, moitas e poucas observacións, etc. Isto permitirá analizar que situacións son favorables e desfavorables para cada método de clasificación. A continuación presentamos os distintos conxuntos:

- **abalone:** Contén diferentes medidas fisiolóxicas sobre orellas de mar. Orixinalmente, o propósito deste conxunto de datos era predecir a idade a partir do resto de variables. Sen embargo, dado que ese é un problema de regresión, para cinguirnos á clasificación empregamos como variable resposta o sexo (ten tres clases: masculino, feminino, e non adulto).
- **breast-cancer:** Nove variables, moitas delas categóricas, son usadas para predecir a recurrencia ou non do cancro de mama en mulleres.
- **car:** O obxectivo é predicir a "aceptación" dun coche, separada esta en catro posibles valores: inaceptable, aceptable, bo, moi bo. Faise en base a variables categóricas relacionadas co prezo, a capacidade e a seguridade do vehículo.
- **chess-krvkp:** Partidas de xadrez enfrontándose rei e torre contra rei e peón. A variable resposta é se as brancas poden gañar ou non. Todas as variables son categóricas. Unha ten tres valores, e o resto son binarias.
- **heart-cleveland:** Contén certa información sobre pacientes, coa fin de predicir o impacto da enfermidade do corazón, medida esta nunha escala de cinco posibles valores.
- **pima:** Datos físicos e médicos, de mulleres maiores de 21 anos e de ascendencia Pima (un pobo indio americano), para predecir a presenza ou non de diabetes.

BD	Observacións	Clases	Variables (Núm./Cat.)
abalone	4177	3	8 (7/1)
breast-cancer	286	2	9 (1/8)
car	1728	4	6 (0/6)
chess-krvkp	3196	2	36 (0/36)
heart-cleveland	303	5	13 (5/8)
pima	768	2	8 (8/0)
wine	178	3	13 (13/0)
yeast	1484	10	8 (6/2)
zoo	101	7	16 (0/16)

Táboa 3.1: Resumo das bases de datos multivariantes.

- **wine:** Información sobre certas propiedades químicas de tres tipos de viños italianos. O tipo de viño é, precisamente, a variable obxectivo. Segundo din no propio repositorio da UCI, trátase dun problema fácil para a clasificación.
- **yeast:** Inclúe unha serie de variables para a predición do sitio no que se atopan as proteínas en células eucariotas.
- **zoo:** Contén unha serie de variables categóricas para determinar a clase á que pertence cada animal. Son todas binarias a excepción dunha, que ten seis valores.

Na Táboa 3.1 amosamos un resumo dos nove conxuntos de datos, co número de observacións que teñen, a cantidade de clases distintas, e o número de variables, diferenciando entre numéricas e categóricas. É conveniente resaltar que esta separación non sempre é doada de facer, pois nalgúns casos atopámonos con variables que podía tomar, por exemplo, 5 valores diferentes, pertencentes a unha escala cunha orde clara. Optouse por identificar estas situacións como categóricas, pero noutras cun rango maior de valores para a escala, como nun caso de 11 valores, tomouse a variable como numérica.

A parte dos conxuntos multivariantes, tamén empregamos tres bases de datos funcionais, *phoneme*, *medflies* e *medflies*. Estas foron obtidas de diversos paquetes de R. Como se comentou no Capítulo 2, o procedemento que se segue con elas é a súa representación en bases, e os coeficientes resultantes son empregados polos clasificadores multivariantes. De feito, para estudar a dependencia con respecto ó número de elementos das bases, repítese o proceso para unha certa cantidade de valores desta magnitude. En particular, considéranse bases de B-splines de 4, 6, 8, 10, 15, 20, 25, 30, 35, e 40 elementos. No caso das de compoñentes principais, as cantidade elixidas comprenden todos os enteiros entre 1 e 10, ambos inclusive.

No caso de *tecator*, é sabido que a información máis relevante se atopa na segunda derivada. Por iso, tanto os datos orixinais, como as primeiras e segundas derivadas son representadas en bases (nas que corresponda), e as súas coordenadas postas nun vector todas seguidas. Así, os patróns desta base de datos teñen o triplo do tamaño que terían se só considerásemos os orixinais.

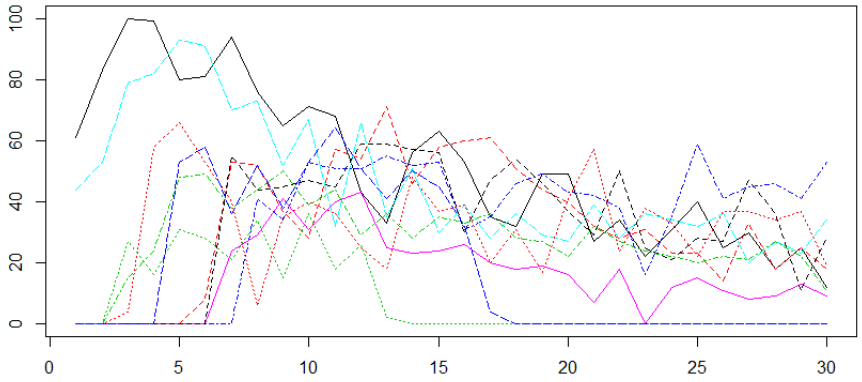

Figura 3.1: Mostra de 10 curvas de *medflies* escollidas ó azar.

De seguido expoñemos os conxuntos de datos funcionais escollidos:

- **medflies:** As funcións representan o número de ovos postos diariamente, durante un período de 30 días, por moscas da froita mediterráneas. Debido a que hai soamente 30 puntos de discretización, este é un caso que ben podería ser tratado directamente como multivariante, obviando a estrutura funcional. Paquete: *ddalpha* [12].
- **phoneme:** Cada curva correspóndese co log-periodograma dun fonema. Trátase de identificar se se trata do sonido "aa" ou "ao". Estas curvas presentan un alto número de oscilacións, de xeito que é preciso un alto número de B-splines para a súa correcta representación. Paquete: *fds* [18].
- **tecator:** Contén os resultados dunha espectrometría realizada a 215 pedazos de carne. Cada punto da discretización das funcións corresponde coa absorbencia rexistrada nunha certa lonxitude de onda. O contido en graxa, que se clasificou como alto se supera 22 e baixo en caso contrario, é o obxectivo da clasificación. Este é un caso no que os mellores resultados se obteñen ó considerar a derivada segunda das funcións, en lugar das curvas orixinais. Paquete: *fda.usc* [5].

Nas Figuras 3.1, 3.2 e 3.3 pódese ver unha selección aleatoria de dez curvas de cada base de datos. Isto serve para facerse unha idea da forma que teñen as curvas en cada caso. Nos datos de *phoneme*, apréciase unha enorme variabilidade, sendo case imposible discernir a simple vista cada curva. As funcións de *medflies* tamén presentan unha alta variabilidade, mais algunhas fanse constantes a cero en certas zonas. Finalmente, os datos de *tecator* son os máis suaves de todos, e presentan un ou dous máximos locais, dependendo de cada curva. A Táboa 3.2 recolle o número de observacións de cada conxunto e os puntos de discretización que teñen.

3.2. Código empregado

Todos os clasificadores que empregamos neste traballo están implementados en distintas librerías de R [21]. Nalgúns casos, en lugar de chamar directamente á función correspondente, usamos por simplicidade a interfaz proporcionada polo paquete *caret* [11] de R, aínda que a sintonización de

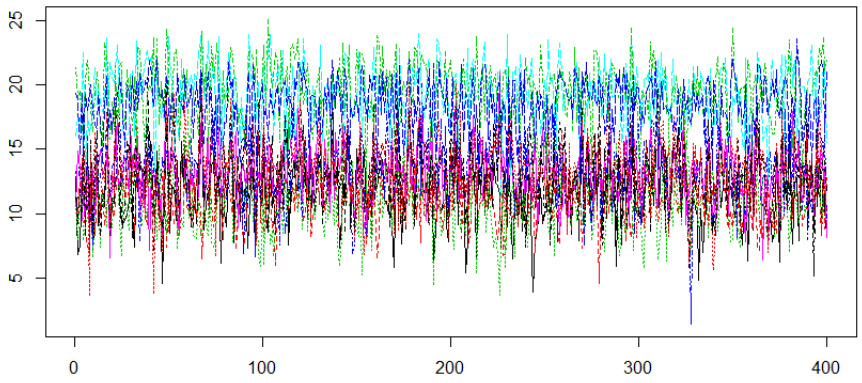

Figura 3.2: Mostra de 10 curvas de *phoneme* escollidas ó azar.

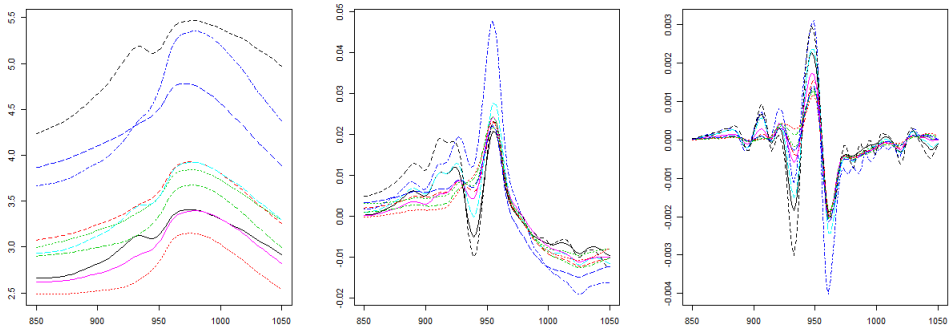

Figura 3.3: Mostra de 10 curvas de *tecator* escollidas ó azar, xunto coas súas derivadas primeira e segunda (de esquerda a dereita).

BD	Número de datos	Puntos de discretización
tecator	215	100
phoneme	300	400
medflies	534	30

Táboa 3.2: Resumo das bases de datos funcionais.

parámetros e a estimación do erro facémolas manualmente (é dicir, co procedemento exposto nas Seccións 3.3 e 3.4).

- **Análise discriminante:** Para a análise discriminante linear e cadrática empregamos as funcións *lda* e *qda*, respectivamente, do paquete *MASS* [20] de R.
- **Regresión loxística:** O axuste do modelo de regresión loxística facémolo utilizando a interfaz *caret*, coa función *multinom* do paquete *nnet* [20].
- **K - veciños máis próximos:** O método KNN executámolo coa función *knn* do paquete *class* [20]. O único parámetro axustable é, evidentemente, o número de veciños.
- **Perceptrón multicapa:** A través da interfaz de *caret*, empregamos a función *mlp* do paquete *RSNNS* [4], que permite entrenar un Perceptrón multicapa. Pódese especificar o número de neuronas da capa oculta.
- **Máquina de vectores de soporte:** Tamén mediante *caret*, chamamos á función *svmRadial* do paquete *kernlab* [9]. Esta permite axustar dous parámetros: o ancho de banda do núcleo, e o parámetro de custo C.
- **Random Forest:** Usamos a función *randomForest*, incluída no paquete homónimo [13]. Esta permite empregar como parámetro axustable o número de variables a seleccionar en cada partición. Aínda que tamén se pode especificar o número máximo de nodos terminais ou o seu tamaño mínimo, decidimos construír totalmente as árbores.

Cómpre comentar que a todos os conxuntos de datos, tanto multivariantes como funcionais, se lles aplica un preproceso que consiste na súa estandarización, é dicir, a cada variable se lle subtrae a súa media, e a continuación divídese pola súa desviación típica.

3.3. Sintonización de parámetros

A maior parte dos modelos de clasificación teñen asociados varios parámetros que determinan o seu funcionamento (o número de neuronas nas redes neuronais, o ancho de banda do núcleo nas máquinas de vectores de soporte, etc.). En función destes, obteranse uns resultados ou outros. O procedemento de determinar os mellores para cada conxunto de datos é o que denominamos "sintonización de parámetros".

Para cada conxunto de datos, facemos unha partición en dous subconxuntos, un de adestramento e outro de test (empregamos unha proporción 50-50, ou sexa, os dous subconxuntos son de igual tamaño). Esta operación repetímola 10 veces. Estas particións fanse respetando ó máximo posible as proporcións de patróns en cada clase.

A continuación, seleccionados uns valores dos parámetros sintonizables, adestramos o modelo e estimamos a súa precisión en cada partición (cos subconxuntos de adestramento e test, respectivamente), e calculamos a súa media. Tras facer isto con toda a colección de valores dos parámetros sintonizables, seleccionamos aquel (ou aqueles, se hai varios parámetros sintonizables) que maior precisión ten.

Evidentemente, este é un procedemento custoso computacionalmente, pois estamos axustando e validando cada modelo unha cantidade de veces igual ó número de particións multiplicado polo número de combinacións dos parámetros. No caso máis extremo, o da máquina de vectores de soporte (que ten dous metaparámetros), esta cantidade resulta ser de $10 \times (5 \times 5) = 250$. E está claro que unha malla de cinco valores para cada parámetros non é un axuste moi fino.

Cómpre comentar que as particións dos conxuntos orixinais constrúense antes de executar os clasificadores, de xeito que aquelas son as mesmas para todos eles.

Os métodos da análise discriminante linear e cadrática, así como a regresión loxística, non precisan da sintonización de ningún parámetro. O resto teñen os parámetros que se indican na Táboa 3.3.

Modelo	Núm. par.	Parámetros	Valores
KNN	1	K	1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25
MLP	1	m_1	10, 20, 30, 40, 50, 60, 70, 80
SVM	2	σ / C	0.2, 0.4, 0.6, 0.8, 1 / 0.4, 0.7, 1, 1.3, 1.6
RF	1	m	1, 2, 3, 4, 5, 6, 7

Táboa 3.3: Parámetros axustables en cada modelo. Os nomes correspóndense cos definidos nas correspondentes seccións do Capítulo 1.

3.4. Estimación da precisión

Unha vez que están determinados os metaparámetros do modelo, cómpre estimar a súa precisión. Certo é que, durante o procedemento de sintonización, xa traballamos con estimacións da precisión. Sen embargo, debido a que estas debían facerse para todas as combinacións de parámetros, non nos podíamos permitir estimar a precisión todo o finamente que se podería desexar. Agora, tendo xa os parámetros determinados, podemos empregar máis recursos nesta tarefa.

Por esta razón, para a estima da precisión final empregamos unha validación cruzada 20-fold (usando o 95 % dos patróns para adestramento e o 5 % para test). Os conxuntos de datos orixinais son divididos en 20 subconxuntos (manendo as proporcións iniciais en cada clase), e adéstrase ós clasificadores con todos eles menos un, que é empregado para a avaliación da precisión. Isto repítese de xeito que se usan todas as combinacións posibles, 20, e calcúlase a media da precisión de cada clasificador. Este será o valor finalmente considerado.

Tanto o proceso de sintonización de parámetros, como a execución e a estimación da precisión foron implementados cun código (*script*) do *Bash* (intérprete de comandos) de Linux (Ubuntu 14.04) nunha máquina virtual (VMware 7.1.2) con CPU a 2.20 GHz (dous núcleos dedicados, de catro) e 2 GB Mb de memoria RAM.

Capítulo 4

Resultados Numéricos

O presente capítulo adícase á exposición e análise dos resultados dos clasificadores descritos no Capítulo 1 sobre as bases de datos, tanto multivariantes como funcionais, amosadas no Capítulo 3, seguindo a metodoloxía exposta no mesmo.

4.1. Datos multivariantes

Comezamos a análise dos resultados ollando a Táboa 4.1. Esta recolle o tempo empregado, en segundos[1], para completar a execución de cada clasificador en cada problema, incluíndo a sintonización de parámetros e a validación cruzada. Salta á vista a gran diferenza que existe nos tempos de execución dos diferentes modelos. Por un lado, aqueles máis sinxelos (análise discriminante de Fisher, regresión loxística e K veciños próximos) non requiren máis duns poucos segundos. Por outra banda, o resto, máis modernos e complexos, requiren cantidades de tempo que nin sequera son comparables coas anteriores. En gran medida, isto é debido á sintonización. Tamén é certo que, dentro deste último grupo, o Random Forest é notablemente máis rápido, mellorando nalgunhas ocasións incluso á regresión loxística.

Na Táboa 4.2 recóllense as precisións medias resultantes da validación cruzada, en tanto por cen. O que chama a atención é, precisamente, que ningún modelo destaca, nin para ben nin para mal (salvando

[1] A precisión coa que se mediron os tempos é, en xeral, dun segundo. Sen embargo, nalgúns casos isto resultaba en tempos de 0 segundos. Para evitar isto, empregouse unha precisión maior para os clasificadores máis rápidos (LDA, QDA e KNN).

Problema	LDA	QDA	RL	KNN	MLP	SVM	RF
abalone	1.3	1.3	13	12.4	693	792	218
breast-cancer	0.6	0.6	10	0.8	102	117	12
car	0.8	–	12	3.0	324	314	43
chess-krvkp	2.6	–	17	26.8	1218	826	288
heart-cleveland	0.7	–	11	0.9	117	171	18
pima	0.7	0.7	11	1.4	172	144	33
wine	0.6	0.6	11	0.8	88	121	8
yeast	0.9	–	19	2.8	366	567	84
zoo	–	–	11	0.8	87	–	6

Táboa 4.1: Tempos de execución, en segundos. Inclúe a sintonización de parámetros e a estimación da precisión mediante validación cruzada. As raias indican un erro.

Problema	LDA	QDA	RL	KNN	MLP	SVM	RF
abalone	63.7	61.8	65.1	62.1	65.4	**66.3**	64.7
breast-cancer	70.4	64.6	71.1	70.4	66.1	69.6	**74.6**
car	81.7	–	83.0	94.2	**99.2**	98.4	98.7
chess-krvkp	95.0	–	97.6	95.6	**99.4**	97.6	99.2
heart-cleveland	**74.7**	–	**74.7**	72.7	71.0	73.7	73.7
pima	78.4	75.9	**78.9**	76.3	76.8	77.9	77.4
wine	97.2	97.8	95.6	92.8	97.2	96.7	**98.3**
yeast	58.9	–	59.3	58.5	58.9	48.0	**62.0**
zoo	–	–	100	100	100	–	100
Media	77.5	75.0	80.6	80.3	81.6	78.6	83.2
Ran. Friedman	4.12	5.75	3.12	5.31	3.50	3.87	2.31

Táboa 4.2: Precisión, en tanto por cen, coa media e o ranking de Friedman. As raias indican un erro.

os casos nos que apareceu algún erro). Tamén se inclúe a media sobre todas as bases de datos, e o ranking de Friedman (pódese atopar información sobre esta proba estatística en [19]). Este último pon de manifesto, claramente, o mellor comportamento do Random Forest.

A pesar de que, polo xeral, os datos non seguen unha distribución normal en cada grupo, a análise discriminante de Fisher obtén uns resultados comparables ós do resto de modelos. Unicamente no caso do conxunto *car*, obsérvase como o LDA e a regresión loxística cometen un erro considerablemente superior ós dos métodos "menos paramétricos". Este conxunto está formado unicamente por variable categóricas, o cal evidentemente non ten nada que ver coa asunción de normalidade.

Outra cuestión de gran importancia é o efecto da existencia de grupos con poucas observacións. Ese é o motivo polo que o QDA falla máis veces das que se executa correctamente. Tanto *car*, como *chess-krvkp*, *heart-cleveland*, *yeast* e *zoo* teñen algunha clase con poucos representantes. Dado que a análise discriminante cadrática require estimar unha matriz de varianzas-covarianzas en cada grupo, o procedemento pode fallar se non hai información suficiente. Pero á parte do QDA, este problema parece afectar tamén, aínda que en menor medida, ó LDA e á SVM. Estes dous clasificadores fallan na base de datos *zoo* (que é a máis extrema en canto a poucos elementos por grupo), e o último tamén se ve seriamente afectado na base de datos *yeast*, pois obtén un resultado claramente inferior ós do resto de clasificadores.

Finalmente, tamén podemos resaltar que o Random Forest é o método que máis veces obtén a maior precisión. En ningún caso opera claramente peor que o resto, independentemente de se os datos son categóricos ou numéricos. Ademais, como comentabamos anteriormente, o tempo consumido na súa execución non é dos máis altos. Por estes motivos, o Random Forest parece ser un dos clasficadores máis robustos.

4.2. Datos funcionais

Debido á gran cantidade de resultados obtidos na clasificación dos datos funcionais (pois considerabamos dous tipos de bases con dez números de elementos distintos), optamos por representalos doutro xeito alternativo ás táboas. As Figuras 4.1, 4.2 e 4.3 amosan as precisións como función do número de elementos da base, sendo cada curva un clasificador. Esta visualización permite detectar tendencias e facer comparacións máis facilmente. En todo caso, no Apéndice B pódense consultar as táboas cos valores numéricos.

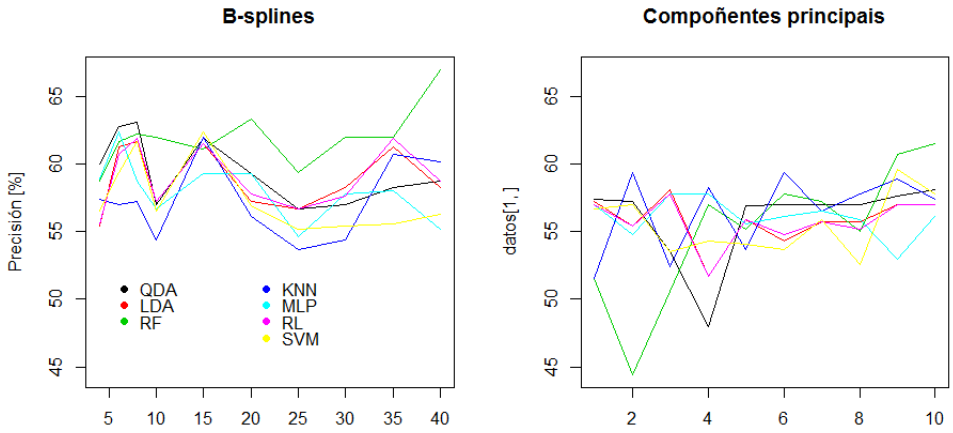

Figura 4.1: Representación das precisións para os datos de *medflies*. Cada liña é un clasificador. Os eixos de abcisas indican o número de elementos das bases.

Na Figura 4.1, correspondente ós datos de *medflies*, obsérvase unha clara dependecia non linear co número de B-splines: practicamente todos os modelos perden precisión con 10 elementos, gáñana con 15, e sofren outra perda entre os 25 e os 30. Isto pode ser debido, simplemente, ós puntos onde caian os nodos dos splines. No tocante ás bases de compoñentes principais, ningún clasificador parece presentar unha tendencia significativa, agás o Random Forest, que comeza sendo o peor, e vai remontando co número de compoñentes ata acadar a máxima precisión.

A Figura 4.2 amosa os resultados dos datos *phoneme* representados en bases de B-splines. Apréciase un aumento continuado da precisión co número de elmentos. Isto é porque as formas destas funcións son altamente complexas e son precisos splines con moitos nodos para a súa representación. Non se inclúe a figura cos resultados para as bases de compoñentes principais, pois só se apreciaba unha característica: todos os modelos comezaban en valores similiares de precisión cunha única compoñente principal (en torno ó 60 %), e acadaban xa o 100 % con dúas, manténdonse constantes a partir de aí.

Por último, as precisións dos datos de *tecator* están recollidas na Figura 4.3. Neste caso, cómpre destacar o feito de que o número de variables é o triplo do número de elementos considerados nas bases, pois inclúense os datos orixinais, e as súas derivadas primeiras e segundas. Por este motivo, os efectos do ruído son moito máis acentuados.

Destaca, en primeiro lugar, a gran caída do QDA no caso das bases de B-splines, ata o punto de fallar a súa execución a partir dos 25 elementos. Isto é debido a que non hai suficientes observacións en cada grupo, en relación co número de variables. Obsérvase tamén unha forte caída na SVM e, máis lixeiramente, no LDA e na regresión loxística. Pola contra, RF, MLP e KNN mantéñense, mentres que a RL ten unha lixeira caída.

En canto as bases de compoñentes principais, apréciase tamén unha caída no QDA e na SVM, aínda que menos pronunciada. É posible que incluso o KNN se estea vendo afectado polo ruído neste caso. Por outra banda, o Random Forest semella ser o máis robusto, manténdose todo o tempo en valores altos de precisión.

Outra característica salientable, en xeral para os tres conxuntos de datos, é a similitude entre o LDA e a regresión loxística. A fin de contas, ambos os dous fan unha partición linear do espazo. Por outra banda, o Perceptrón multicapa e o Random Forest parecen ser os modelos menos afectados polo ruído.

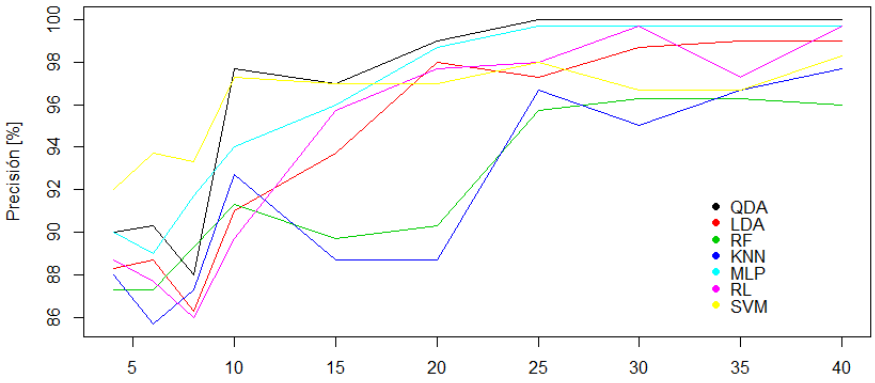

Figura 4.2: Representación das precisión para os datos de *phoneme*, utilizando unha base de B-splines. Cada liña é un clasificador. O eixo de abcisas indica o número de elementos das bases.

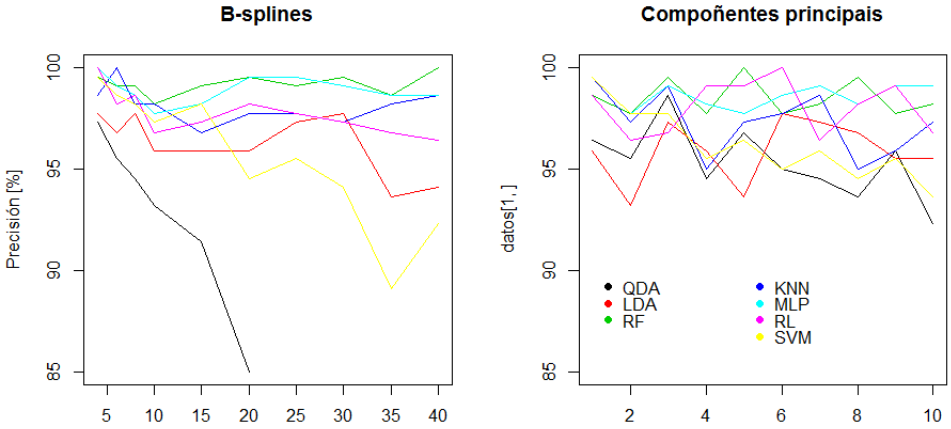

Figura 4.3: Representación das precisión para os datos de *tecator*. Cada liña é un clasificador. Os eixos de abcisas indican o número de elementos das bases.

Conclusión

Se ben o campo da clasificación estatística é moi amplo (e máis ó abranguer o caso funcional), neste traballo tentouse tocar aspectos ben diversos, pero mantendo unha certa liña que fixera de nexo de todos eles. Procurouse, por un lado, non deixar ningún aspecto importante sen citar; por outra banda, todo o comentado garda relación entre si, e forma parte dun único proceso, que consiste na clasificación e discusión de resultados, case dende cero, de conxuntos de datos reais.

Primeiramente, detallouse toda a teoría relativa a sete modelos de clasificación supervisada: A análise discriminante linear e cadrática, a regresión loxística, os K - veciños máis próximos, o Perceptrón multicapa, a Máquina de Vectores de Soporte e o Random Forest. De seguido, explicouse a técnica da representación en bases para datos funcionais, e como deste xeito este poden ser clasificados empregando os anteriores modelos.

Deseñouse un plan experimental homoxéneo para todos os conxuntos de datos e clasificadores, indicando os procedementos para a sintonización de parámetros e a estimación final da precisión. Finalmente, expuxéronse os resultados obtidos, e razoouse sobre o que podía estar acontecendo en cada situación. Entre outras cousas, observouse a alta precisión do LDA, sobre todo tendo en conta a rapidez da súa execución. Tamén destacou o Random Forest como un dos modelos máis robustos.

Apéndice A

Cálculos dos clasificadores multivariantes

Neste apéndice esténdense algúns cálculos correspondentes ó Capítulo 1. En cada unha das seguintes seccións, séguese a notación introducida na parte do traballo onde esta é referenciada.

A.1. QDA

Consideremos o logaritmo do cociente das probabilidades *a posteriori* dos grupos i e j:

$$\log\left(\frac{\mathbb{P}(i\,|\,x)}{\mathbb{P}(j\,|\,x)}\right) = \log\left(\frac{\pi_i}{\pi_j}\right) + \frac{1}{2}\log\left(\frac{|\Sigma_j|}{|\Sigma_i|}\right) +$$
$$+\frac{1}{2}\left[(x-\mu_j)^t\Sigma_j^{-1}(x-\mu_j) - (x-\mu_i)^t\Sigma_i^{-1}(x-\mu_i)\right]$$

Facendo contas sobre o último termo, temos

$$(x-\mu_j)^t\Sigma_j^{-1}(x-\mu_j) - (x-\mu_i)^t\Sigma_i^{-1}(x-\mu_i) =$$
$$= x^t\Sigma_j^{-1}x - x^t\Sigma_j^{-1}\mu_j - \mu_j^t\Sigma_j^{-1}x + \mu_j^t\Sigma_j^{-1}\mu_j -$$
$$-x^t\Sigma_i^{-1}x + x^t\Sigma_i^{-1}\mu_i + \mu_i^t\Sigma_i^{-1}x - \mu_i^t\Sigma_i^{-1}\mu_i =$$
$$x^t(\Sigma_j^{-1} - \Sigma_i^{-1})x + \mu_j^t\Sigma_j^{-1}\mu_j - \mu_i^t\Sigma_i^{-1}\mu_i$$
$$+x^t(\Sigma_i^{-1}\mu_i - \Sigma_j^{-1}\mu_j) + (\mu_i^t\Sigma_i^{-1} - \mu_j\Sigma_j^{-1})x.$$

Xa que a inversa dunha matriz simétrica é tamén simétrica, e que cada un dos sumandos da expresión anterior é un escalar (e polo tanto coincide coa súa trasposta), o último elemento transfórmase en

$$(\mu_i^t\Sigma_i^{-1} - \mu_j\Sigma_j^{-1})x = \left((\mu_i^t\Sigma_i^{-1} - \mu_j\Sigma_j^{-1})x\right)^t =$$
$$x^t(\mu_i^t\Sigma_i^{-1} - \mu_j\Sigma_j^{-1})^t = x^t(\Sigma_i^{-1}\mu_i - \Sigma_j^{-1}\mu_j).$$

Así, o logaritmo da ratio entre as probabilidades *a posteriori* queda como

$$\log\left(\frac{\mathbb{P}(i\,|\,x)}{\mathbb{P}(j\,|\,x)}\right) = \log\left(\frac{\pi_i}{\pi_j}\right) + \frac{1}{2}\log\left(\frac{|\Sigma_j|}{|\Sigma_i|}\right) +$$
$$\frac{1}{2}\left[x^t(\Sigma_j^{-1} - \Sigma_i^{-1})x + \mu_j^t\Sigma_j^{-1}\mu_j - \mu_i^t\Sigma_i^{-1}\mu_i\right] + x^t(\Sigma_i^{-1}\mu_i - \Sigma_j^{-1}\mu_j)$$

Reordenando os termos, tense

$$\log\left(\frac{\mathbb{P}(i\,|\,x)}{\mathbb{P}(j\,|\,x)}\right) = \frac{1}{2}x^t(\Sigma_j^{-1} - \Sigma_i^{-1})x \; + x^t(\Sigma_i^{-1}\mu_i - \Sigma_j^{-1}\mu_j) \; +$$
$$\log\left(\frac{\pi_i}{\pi_j}\right) + \frac{1}{2}\log\left(\frac{|\Sigma_j|}{|\Sigma_i|}\right) + \frac{1}{2}\left(\mu_j^t\Sigma_j^{-1}\mu_j - \mu_i^t\Sigma_i^{-1}\mu_i\right).$$

A.2. Regresión Loxística

Cálculo da log-verosimilitude

Dados os parámetros θ, a log-verosimilitude é

$$\ell(\theta) = \log\left[\prod_{i=1}^{n}\left[y_i\mathbb{P}\big(1\,|\,x_i\big) + (1-y_i)\mathbb{P}\big(0\,|\,x_i\big)\right]\right] =$$

$$\log\left[\prod_{i=1}^{n}\left[y_i q(x_i,\theta) + (1-y_i)\big(1-q(x_i,\theta)\big)\right]\right]$$

Nótese que o que hai dentro do produtorio é a probabilidade de que cada observación pertenza ó grupo que lle corresponde. Se $y_i = 1$, temos a probabilidade de que sexa do grupo 1, e se $y_i = 0$, a do grupo 2. Desfacendo o produtorio en suma de logaritmos, temos

$$\ell(\theta) = \sum_{i=1}^{n}\Big(y_i\log\big(q(x_i,\theta)\big) + (1-y_i)\log\big(1-q(x_i,\theta)\big)\Big).$$

A continuación, desfacemos os distintos sumandos botando man de (1.2), e tendo en conta a notación introducida na Sección 1.2.1:

$$q(x_i,\theta) = \frac{\exp\{\beta_0+\beta^t x_i\}}{1+\exp\{\beta_0+\beta^t x_i\}} = \frac{\exp\{\theta^t x_i^*\}}{1+\exp\{\theta^t x_i^*\}} \Rightarrow$$

$$\log\big(q(x_i,\theta)\big) = \theta^t x_i^* - \log\big(1+\exp\{\theta^t x_i^*\}\big),$$
$$\log\big(1-q(x_i,\theta)\big) = \log\left(\frac{1+\exp\{\theta^t x_i^*\}-\exp\{\theta^t x_i^*\}}{1+\exp\{\theta^t x_i^*\}}\right) =$$
$$-\log\big(1+\exp\{\theta^t x_i^*\}\big).$$

Así resulta

$$\ell(\theta) = \sum_{i=1}^{n}\left[y_i\big(\theta^t x_i^* - \log\big(1+\exp\{\theta^t x_i^*\}\big)\big) - (1-y_i)\log\big(1+\exp\{\theta^t x_i^*\}\big)\right]$$

$$= \sum_{i=1}^{n}\left[y_i\theta^t x_i^* - \log\big(1+\exp\{\theta^t x_i^*\}\big)\right].$$

Expresión da matriz Hessiana

O elemento (i,j) da matriz Hessiana da log-verosimilitude será

$$\frac{\partial^2 \ell(\theta)}{\partial\theta_j \partial\theta_i} = \frac{\partial}{\partial\theta_j}\left(\frac{\partial \ell(\theta)}{\partial\theta_i}\right) = \frac{\partial}{\partial\theta_j}\left(\sum_{k=1}^{n} x^*_{k(i)}\big(y_k - q(x_k,\theta)\big)\right) =$$

$$\frac{\partial}{\partial\theta_j}\left(-\sum_{k=1}^{n} x^*_{k(i)} q(x_k,\theta)\right) = \frac{\partial}{\partial\theta_j}\left(-\sum_{k=1}^{n} x^*_{k(i)} \frac{\exp\{\theta^t x^*_k\}}{1+\exp\{\theta^t x^*_k\}}\right) =$$

$$\sum_{k=1}^{n} -x^*_{k(i)} \frac{\partial}{\partial\theta_j}\left(\frac{\exp\{\theta^t x^*_k\}}{1+\exp\{\theta^t x^*_k\}}\right) =$$

$$\sum_{k=1}^{n} -x^*_{k(i)} \left(\frac{x^*_{k(j)}\exp\{\theta^t x^*_k\}\big(1+\exp\{\theta^t x^*_k\}\big) - x^*_{k(j)}\big(\exp\{\theta^t x^*_k\}\big)^2}{\big(1+\exp\{\theta^t x^*_k\}\big)^2}\right) =$$

$$\sum_{k=1}^{n} -x^*_{k(i)} x^*_{k(j)} \frac{\exp\{\theta^t x^*_k\}}{1+\exp\{\theta^t x^*_k\}}\left(1 - \frac{\exp\{\theta^t x^*_k\}}{1+\exp\{\theta^t x^*_k\}}\right) =$$

$$\sum_{k=1}^{n} -x^*_{k(i)} x^*_{k(j)} q(x_k,\theta)\big(1-q(x_k,\theta)\big) =$$

$$\sum_{k=1}^{n} -q(x_k,\theta)\big(1-q(x_k,\theta)\big)\big(x^*_k\, x^{*t}_k\big)_{ij},$$

onde $(x^*_k\, x^{*t}_k)_{ij}$ representa o elemento da fila i e columna j da matriz resultante de multiplicar o vector columna x^*_k polo vector fila x^{*t}_k. En definitiva, a matriz Hessiana vén dada por

$$\frac{\partial^2 \ell(\theta)}{\partial\theta \partial\theta^t} = \sum_{k=1}^{n} -q(x_k,\theta)\big(1-q(x_k,\theta)\big)\, x^*_k\, x^{*t}_k.$$

Apéndice B

Resultados da clasificación con datos funcionais

Neste apéndice incluímos as Táboas B.1, B.2, B.3, B.4, B.5 e B.6, cos resultados obtidos na clasificación dos datos funcionais.

Clasif.	4	6	8	10	15	20	25	30	35	40
QDA	60.0	62.8	63.1	57.0	61.9	59.3	56.7	57.0	58.3	58.7
LDA	55.4	61.3	61.7	57.2	61.5	57.2	56.7	58.3	61.3	58.3
RF	58.7	61.7	62.2	62.0	61.1	63.3	59.4	62.0	62.0	67.0
KNN	57.4	57.0	57.2	54.4	62.0	56.1	53.7	54.4	60.7	60.2
MLP	58.9	62.4	58.7	56.7	59.3	59.3	54.6	57.8	58.0	55.2
RL	55.6	60.7	61.9	57.2	61.5	57.8	56.7	57.6	61.9	58.7
SVM	56.5	59.4	61.7	56.5	62.4	56.9	55.2	55.4	55.6	56.3

Táboa B.1: Precisións para os datos de *medflies*, representados nunha base de B-splines, co número de elementos indicado nas columnas.

Clasif.	1	2	3	4	5	6	7	8	9	10
QDA	57.4	57.2	53.5	48.0	56.9	57.0	57.0	57.0	57.6	58.1
LDA	57.2	55.4	58.1	51.7	55.9	54.3	55.7	55.7	57.0	57.0
RF	51.5	44.4	50.6	57.0	55.2	57.8	57.2	55.0	60.7	61.5
KNN	51.5	59.4	52.4	58.3	53.7	59.4	56.5	57.8	58.9	57.4
MLP	57.0	54.8	57.8	57.8	55.6	56.1	56.5	55.9	53.0	56.1
RL	57.0	55.4	57.8	51.7	55.9	54.8	55.7	55.2	57.0	57.0
SVM	56.7	57.0	53.5	54.3	54.1	53.7	55.9	52.6	59.6	57.8

Táboa B.2: Precisións para os datos de *medflies*, representados nunha base de compoñentes principais, co número de elementos indicado nas columnas.

Clasif.	4	6	8	10	15	20	25	30	35	40
QDA	90.0	90.3	88.0	97.7	97.0	99.0	100	100	100	100
LDA	88.3	88.7	86.3	91.0	93.7	98.0	97.3	98.7	99.0	99.0
RF	87.3	87.3	89.3	91.3	89.7	90.3	95.7	96.3	96.3	96.0
KNN	88.0	85.7	87.3	92.7	88.7	88.7	96.7	95.0	96.7	97.7
MLP	90.0	89.0	91.7	94.0	96.0	98.7	99.7	99.7	99.7	99.7
RL	88.7	87.7	86.0	89.7	95.7	97.7	98.0	99.7	97.3	99.7
SVM	92.0	93.7	93.3	97.3	97.0	97.0	98.0	96.7	96.7	98.3

Táboa B.3: Precisións para os datos de *phoneme*, representados nunha base de B-splines, co número de elementos indicado nas columnas.

Clasif.	1	2	3	4	5	6	7	8	9	10
QDA	61.0	100	100	100	100	100	100	100	100	100
LDA	61.0	100	100	100	100	100	100	99.7	100	100
RF	57.3	100	99.7	98.7	99.7	100	100	99.3	100	100
KNN	69.3	100	99.7	98.7	99.3	99.7	99.7	99.7	99.7	100
MLP	64.3	100	100	100	100	100	100	100	100	100
RL	61.0	100	100	100	100	100	100	99.7	100	100
SVM	64.3	100	100	100	100	100	100	100	100	100

Táboa B.4: Precisións para os datos de *phoneme*, representados nunha base de compoñentes principais, co número de elementos indicado nas columnas.

Clasif.	4	6	8	10	15	20	25	30	35	40
QDA	97.3	95.5	94.5	93.2	91.4	85.0	–	–	–	–
LDA	97.7	96.8	97.7	95.9	95.9	95.9	97.3	97.7	93.6	94.1
RF	99.5	99.1	99.1	98.2	99.1	99.5	99.1	99.5	98.6	100
KNN	98.6	100	98.2	98.2	96.8	97.7	97.7	97.3	98.2	98.6
MLP	100	99.1	98.6	97.7	98.2	99.5	99.5	99.1	98.6	98.6
RL	100	98.2	98.6	96.8	97.3	98.2	97.7	97.3	96.8	96.4
SVM	99.5	98.6	98.2	97.3	98.2	94.5	95.5	94.1	89.1	92.3

Táboa B.5: Precisións para os datos de *tecator*, representados nunha base de B-splines, co número de elementos indicado nas columnas.

Clasif.	1	2	3	4	5	6	7	8	9	10
QDA	96.4	95.5	98.6	94.5	96.8	95.0	94.5	93.6	95.9	92.3
LDA	95.9	93.2	97.3	95.9	93.6	97.7	97.3	96.8	95.5	95.5
RF	98.6	97.7	99.5	97.7	100	97.7	98.2	99.5	97.7	98.20
KNN	99.5	97.3	99.1	95.0	97.3	97.7	98.6	95.0	95.9	97.3
MLP	99.5	97.7	99.1	98.2	97.7	98.6	99.1	98.2	99.1	99.1
RL	98.6	96.4	96.8	99.1	99.1	100	96.4	98.2	99.1	96.8
SVM	99.5	97.7	97.7	95.5	96.4	95.0	95.9	94.5	95.5	93.6

Táboa B.6: Precisións para os datos de *tecator*, representados nunha base de compoñentes principais, co número de elementos indicado nas columnas.

Bibliografía

[1] Stephen Boyd, Lieven Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[2] Leo Breiman, Jerome Friedman, Charles J. Stone, R. A. Olshen, *Classification and Regression Trees*. Taylor & Francis, 1984.

[3] Leo Breiman, *Random Forests*. Machine Learning 45, 5-32, 2001.

[4] Christoph Bergmeir, Jose M. Benitez, *Neural Networks in R Using the Stuttgart Neural Network Simulator: RSNNS*. Journal of Statistical Software, 46(7), 2012.

[5] Manuel Febrero-Bande, Manuel Oviedo de la Fuente, *Statistical Computing in Functional Data Analysis*. Journal of Statistical Software, 51(4), 1-28, 2012.

[6] Yoav Freund, Robert E. Schapire, *A Short Introduction to Boosting*. Journal of Japanese Society for Artificial Intelligence, 14(5):771-780, 1999.

[7] Trevor Hastie, Robert Tibshirani, Jerome Friedman, *The Elements of Statistical Learning*. Springer, 2009

[8] Pedro Isasi, Inés M. Galván León, *Redes de neuronas artificiales: un enfoque práctico*. Pearson Educación, S.A. Madrid, 2004.

[9] Alexandros Karatzoglou, Alex Smola, Kurt Hornik, Achim Zeileis, *kernlab - An S4 Package for Kernel Methods in R*. Journal of Statistical Software 11(9), 1-20, 2004.

[10] E. de Klerk, C. Roos, T. Terlaky, *Nonlinear Optimization (CO 367)*. Waterloo, 2006.

[11] Max Kuhn, *Classification and Regression Training*. http://CRAN.R-project.org/package=fds, 2015.

[12] Lange, T., Mosler, K. and Mozharovskyi, P., *Fast nonparametric classification based on data depth*. Statistical Papers, 55, 49-69, 2014.

[13] A. Liaw and M. Wiener, *Breiman and Cutler's random forests for classification and regression*. R News 2(3), 18–22, 2002.

[14] M. Lichman, *UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]*. Irvine, CA: University of California, School of Information and Computer Science, 2013.

[15] Wei-Yin Loh, *Classification and regression trees*. Wiley & Sons, Inc. WIREs Data Mining and Knowledge Discovery 1: 14-23, 2011.

[16] J. O. Ramsay, B. W. Silverman, *Functional Data Analysis*. Springer, 2005.

[17] Braian David Ripley, *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[18] Han Lin Shang and Rob J. Hyndman, *Functional Data Sets.* http://CRAN.R-project.org/package=fds, 2013.

[19] Sheskin, D.J., *Handbook of Parametric and Nonparametric Statistical Procedures.* CRC Press, 2006.

[20] Venables, W. N. and Ripley, B. D., *Modern Applied Statistics with S.* Springer, New York, 2002.

[21] R Core Team, *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/, 2014.