TREBALL DE FI DE GRAU

Grau en Estadística

Títol: Extensió de l'algorisme *Minimum Distance Probability* (MDP) i la seva aplicació a l'obtenció de biomarcadors.

Autor: Oriol Prat Vallverdú

Director: Esteban Vegas Lozano

Departament: Estadística, Facultat de Biologia

Universitat de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Facultat d'Economia i Empresa

Facultat de Matemàtiques i Estadística

Treball Final de Grau

Extensió de l'algorisme *Minimum Distance Probability* (MDP) i la seva aplicació a l'obtenció de biomarcadors.

Grau d'Estadística

Autor: Oriol Prat Vallverdú

Director: Esteban Vegas Lozano

Convocatòria: Setembre 2016

Resum

En els últims anys han sorgit noves tecnologies d'alt rendiment que han permès la generació de grans quantitats de dades biològiques. Fet que ha motivat a la interconnexió de la bioinformàtica amb diferentes disciplines. La possible anàlisi d'aquestes dades ha conduït a nous grans reptes en la biologia molecular i les ciències de la salut en general. En aquest context, aquest treball proposa un nou algorisme d'aprenentatge automàtic per a la classificació multinivell, efectuant una aplicació del mètode en una base de dades genòmica per a l'obtenció de biomarcadors. Finalment es proposa una aplicació web interactiva per permetre l'accessibilitat del mètode.

Paraules clau: *MDP*, *dades òmiques*, *aprenentatge automàtic*, *classificació*, *biomarcadors*, *Kernel MDP*, *Shiny*.

Abstract

In recent years there have been new high performance technologies that have allowed the generation of large amounts of biological data. This has led to the interconnection of different disciplines with bioinformatics. The possible analysis of these data led to major new challenges in molecular biology and health sciences in general. In this context, this work proposes a new machine learning algorithm for multiclass classification, making an application method on a database for obtaining genomic biomarkers. Finally proposes an interactive web application to allow the access to the method.

Keywords: *MDP*, *omics data*, *machine learning*, *classification*, *biomarkers*, *Kernel MDP*, *Shiny*.

Índex

Índex de figures

Notacions

x	Valor escalar
$\mathbf{x}$	Vector columna
$\mathbf{x}^T$	Vector transposat
n	Nombre d'observacions
p	Nombre de variables
$\mathcal{X}$	Espai d'entrada o *input space*
$\mathcal{F}$	Espai de característiques o *feature space*
Φ	Funció que transforma $\mathbf{x}$ de $\mathcal{X}$ a $\mathcal{F}$
$\mathcal{H}$	Espai de Hilbert
$k(\mathbf{x}, \mathbf{y})$	Funció kernel
$\#$	Cardinal d'un conjunt

Capítol 1

INTRODUCCIÓ

El present document constitueix la memòria del treball de fi de grau impartit per la Universiat de Barcelona i la Universitat Politècnica de Catalunya. En aquesta primera part introductoria, s'exposarà la justificació, els objectius i la metodologia emprada al llarg d'aquest treball, juntament amb la seva estructura.

1.1 Justificació

La predicció en les ciències de la salut ha esdevingut un factor important per a conèixer quines possibilitats té un individu de patir una malaltia.

En els últims anys, s'ha avançat en el desenvolupament de procediments i millores per a la recol·lecció de dades. Que consegüentment, han fet prosperar en l'evolució de mètodes per a la predicció. Aquests mètodes de predicció, tenen la capacitat de discernir de forma confiada els nivells que pot prendre una variable prèviament incerta.

La integració i ús d'aquests mètodes en tota la informació biològica i clínica pot conduir a grans avenços en el camp mèdic com és la predicció de patir un càncer.

1.2 Objectius

L'objectiu principal que es persegueix en aquest projecte consisteix en l'aprenentage d'un algorisme de classificació per al seu posterior ús en l'aplicació amb dades genètiques. Aquest algorisme és l'anomenat *Minimum Distance Probability* (MDP) que fou proposat per A. Villarroya, Martin Rios, J. Oller, 1995 [6].

A fi d'arribar a aquest objectiu final, s'hauran d'assolir uns seguit d'objectius intermedis al llarg del treball. Els citem a continuació:

- Conèixer la vessant de la classificació desde la perspectiva que ofereix el camp científic conegut com el *machine learning*.
- Comprendre les solucions que ofereixen els mètodes kernel.
- Extendre l'algoritme MDP al món kernel.
- Aprendre a trobar els paràmetres òptims d'un algorisme MDP.
- Aplicar el mètode per a la classificació d'una base de dades real de pacients amb càncer.
- Confeccionar un pàquet d'R per a la divulgació, l'ús i l'aplació col·lectiva del mètode.
- Produir un entorn gràfic interactiu per a una divulgació pràctica no restringida únicament per a professionals de l'estadística.

1.3 Metodologia

Aquest treball ha estat una gran font de coneixements nous que han hagut de ser apresos amb força cura. Ja que gairebé al llarg del grau, no es tracten coneixements d'aprenentatge automàtic. Per tant, per dur a terme aquest projecte ha calgut una extensa recerca bibliogràfica sobre conceptes relacionats amb la classificació en la bioinformàtica, mètodes kernel i programació amb R. La informació consultada pot trobar-se en la bibliografia present al final del treball.

La primera part del projecte s'ha dedicat a l'aprenentatge de l'algorisme des del punt de vista teòric. Tota la informació ha estat extreta directament de l'article original del mètode [6].

Un cop assolida la seva formulació, s'ha procedit a implementar-lo amb el llenguatge R. I després s'ha passat a la confecció del mètode MDP amb la terminologia del paquet `caret` per a poder trobar o *tunejar* els paràmetres òptims d'un model concret.

Posteriorment, s'ha aplicat el mètode per a un seguit de bases de dades, en concret genètiques. I l'última part ha estat dedicada a la seva divulgació. Per mitjà d'un *package* perquè pugui ser utilitzat pels estadístics que treballen amb R. I una aplicació web interactiva per aquell personal que necessiti eines de predicció, sense ser necessàriament professionals en el camp de l'estadística.

1.4 Estructura

La memòria d'aquest treball consta de set capítols, incloent-hi aquest, la introducció. En cadascun dels capítols es respon a objectius específics que s'han esmentat amb anterioritat.

En el segon capítol s'exposa informació bàsica per a posteriorment comprendre les aplicacions pràctiques. Presentant la idea en què es basen els principis de la biologia molecular, i quins són els elements que poden tractar-se de manera informatitzada. A més, s'exemplifiquen alguns dels inconvenients que suposa el tractament de dades en la bioinformàtica.

En el tercer capítol s'exposa la definició de la classificació, així com dels mètodes que es disposa per avaluar com d'encertades són unes prediccions. En la segona part, s'exposen els mètodes no lineals coneguts com a kernels. Els quals poden permetre en determinades situacions, millors classificacions, gràcies a projectar els individus en altres espais de dimensions superiors.

En el quart capítol, es presenta l'explicació detallada del mètode MDP, tant des d'una perspectiva general com empírica. També es dóna una visió d'alguns pros i contres que poden trobar-se a la pràctica. A més, es dóna una aplicació del mètode amb dos exemples extrets de l'article original [6], on s'aprecia la correcta reproducció dels resultats de l'article.

En el cinquè capítol es troba l'aplicació de l'algorisme en la sel·lució de variables predictores de caire genètic (biomarcadors). On s'apliquen mètodes per a *tunejar* els paràmetres òptims. La base de dades tracta amb pacients amb càncer de colon.

En el sisè capítol es fa una explicació de com s'ha dut a terme la implementació i divulgació del mètode. La primera part consta del paquet d'R, mentre que la segona fa referència a la creació d'aplicacions amb el paquet d'R anomenat Shiny.

Capítol 2

NOCIONS SOBRE BIOINFORMÀTICA

Aquest capítol aportarà unes nocions bàsiques per a comprendre la terminologia i els conceptes biològics que aniran apareixent al llarg del treball.

2.1 Introducció a la Biologia Molecular

2.1.1 El genoma eucariòtic

Cada cèl·lula somàtica[1] eucariota conté essencialment el mateix material genètic, és a dir, tota la informació que es coneix com el genoma. Aproximadament entre les dècades dels anys 80 i 90, s'impartiren importants col·laboracions internacionals, per a seqüenciar el genoma humà complet. Aquest fet, proporcionaria un enorme ventall d'aplicacions, que obriria el coneixement de les bases genètiques per al desenvolupament humà i de les malalties. Tot i això, avui en dia encara és necessari comprendre com s'organitza el genoma humà i com desxifrar el significat de la gran part de la seqüència del seu àcid desoxiribonucleic (ADN).

Algunes parts del genoma contenen instruccions genètiques que la cèl·lula utilitza constantment, altres en canvi, només resulten útils quan s'està en situacions extremes, i algunes fins i tot poden no ser usades. Degut a la immensa quantitat d'informació genètica que conté una cèl·lula, resulta essencial conèixer com es guarda i recupera la informació genètica per a comprendre el funcionament del genoma eucariòtic.

L'ADN conté el material genètic amb totes les instruccions necessàries. El codi genètic contingut en l'ADN està compost per quatre "lletres" o bases: l'adenina (A), la guanina (G), la citosina (C) i la timina (T). Un gen és una petita porció de l'ADN necessària per a generar un producte funcional. Aproximadament el 2 % del genoma humà codifica

[1] Les cèl·lules somàtiques són aquelles que conformen el creixement dels teixits i òrgans dels individus pluricel·lulars.

instruccions per sintetitzar proteïnes. Els gens semblem estar concentrats en àrees a l'atzar al llarg del genoma, amb grans extensions d'ADN no codificant entre ells.

Dits gens, es poden dividir en quatre categories segons la seva funció: uns 5000 gens estan implicats en el manteniment del genoma, quasi 5000 amb la transducció dels senyals i 4000 amb funcions bioquímiques generals; la major part, 9000 gens, intervenen en altres activitats. Les seqüències repetides d'ADN repetitives no codifiquen proteïnes i constitueixen almenys el 50% del genoma humà. Tot i que semblen no tenir funcions directes, són de gran importància per a l'estructura i la dinàmica dels cromosomes.

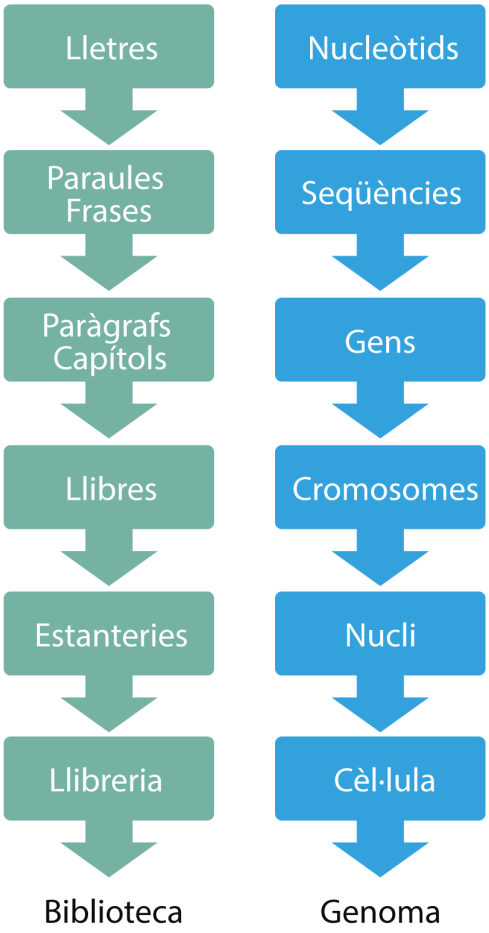


Figura 2.1: Idea intuïtiva comparant els elements que contenen el genoma amb els d'una estanteria.

2.1.2 El dogma central de la biologia molecular

Aquest és el principi general que intenta explicar, quins processos efectua l'ADN contingut en el nucli de les cèl·lules, per a poder transmetre la seva informació per a la creació de les proteïnes. Seguidament, farem un breu resum del seu procés.

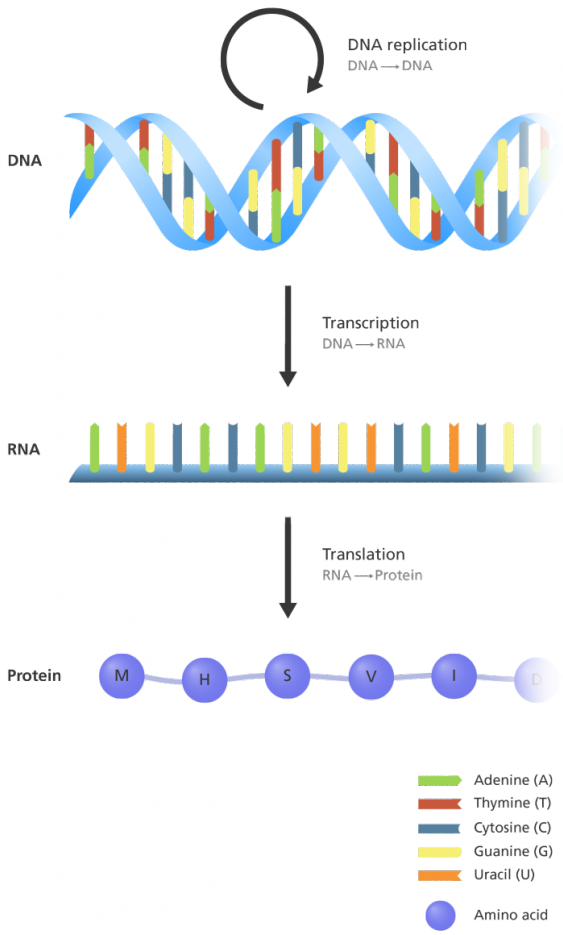


Figura 2.2: Representació gràfica simplificada del que són les tres etapes del dogma central. Extreta de [18]

L'ADN és una molècula, una aglomeració d'àtoms. Aquets àtoms es combinen per crear la forma de doble hèlix.

Els aminoàcids són petites substàncies químiques. Sovint se les nombra com els "maons de la vida". N'hi ha al voltant d'una vintena. Cadascun amb la seva forma única. Es poden ajuntar per a formar unes partícules més grans, les proteïnes.

Els aminoàcids formen les proteïnes, les proteïnes conjuntament amb altres productes químics es combinen per formar les cèl·lules. Les cèl·lules formen els teixits, els quals formen els òrgans i aquets com que funcionen de forma conjunta constitueixen els éssers vius.

Hi ha milions de tipus de proteïnes, han de presentar la forma perfecta per poder funcionar. És aquí on entra en joc l'ADN. La funció més compresa que efectua l'ADN és la d'indicar als aminoàcids com alinear-se i organitzar-se en formes específiques per construir proteïnes que funcionin. Teòricament, si les proteïnes es formen en el moment adequat i en el lloc correcte, les altres formacions com les cèl·lules i els òrgans s'efectuaran amb correctesa.

L'ADN es composa de quatre tipus de productes químics. Són aquets els **nucleòtids**, els

quals es combinen entre si. La seqüència en conté milions. Aquesta cadena en forma de doble hèlix, està enroscada i roman dins del nucli de les cèl·lules. Els aminoàcids, en canvi, resideixen en el citoplasma, fora del nucli. Amb tres nucleòtids es forma un aminoàcid. Així com els nucleòtids són les unitats que formen l'ADN, els aminoàcids són les unitats que formen la proteïna.

Per ajudar a l'ADN a interactuar amb el citoplasma i crear les proteïnes, certs productes químics específics dins del nucli, efectuen còpies parcials del codi de l'ADN. Com ho és l'ARN, que és similar a l'ADN però no té forma de dues tires sinó d'una sola tira. La petita mida i forma de l'ARN li permet ser capaç de passar a través de petits porus en el nucli cap al citoplasma. I dins la boca d'una altra partícula anomenada ribosoma. Els ribosomes poden anomenar-se intuïtivament com les "màquines" que construeixen les proteïnes. Llegeixen els nucleòtids de l'ARN de tres en tres, recullen aminoàcids del seu voltant i els enllacen entre si per crear una cadena, d'acord amb el codi de l'ARN. A mesura que la cadena creix, es doblega, es plega i s'engalla a ella mateixa per a formar una proteïna de forma perfecta. A tres nucleòtids se'ls denomina triplet o codó. Cada joc de tres nucleòtids del codi de l'ARN li diu al ribosoma quina de les varietats d'aminoàcids cal afegir a la cadena de proteïna. Per exemple, CAA li diu al ribosoma que reculli una glutamina, AGU diu que reculli una serina i així successivament. Una vegada, la proteïna finalitza el seu procés de construcció, es poden efectuar diversos passos, un d'ells podria ser formar un nou tipus de cèl·lula.

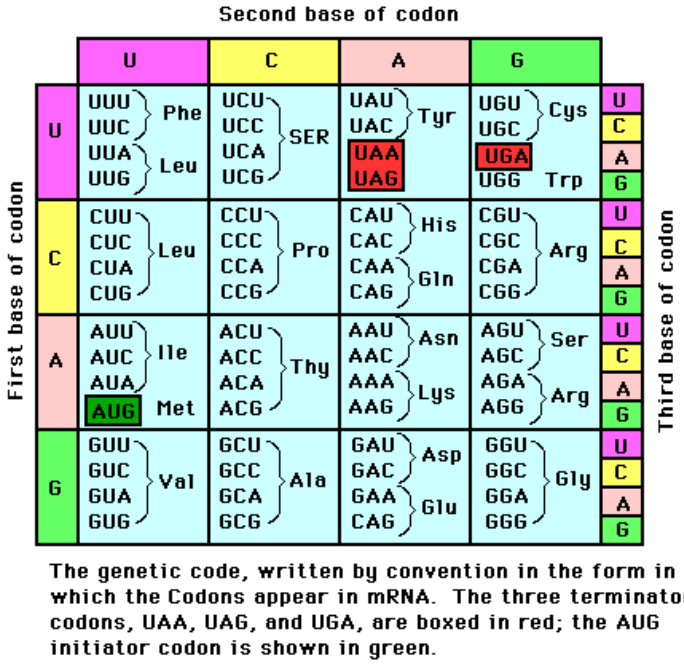


Figura 2.3: Taula amb els nucleòtids de l'ARN i els respectius aminoàcids que es produeixen. Imatge extreta de [27]

2.2 Bioinformàtica

La bioinformàtica pot ser definida com l'aplicació de mètodes matemàtics, estadístics i computacionals de forma conjunta amb l'objectiu de resoldre problemes biològics. Utilitza les tecnologies de la informació per emmagatzemar, recuperar i analitzar dades biològiques. Les principals àrees d'activitat amb que treballa poden resumir-se en tres vessants:

- La creació, l'emmagatzematge i la gestió de dades biològiques.
- El desenvolupament d'eines o mètodes d'anàlisi per a determinar les interrelacions entre les dades.
- I l'anàlisi i la interpretació de les dades biològiques.

El conjunt de dades amb què la bioinformàtica treballa s'anomenen dades òmiques.

Les dades òmiques són totes aquelles que poden analitzar-se en la biologia molecular. Cadascun dels passos de proliferació genètica fins a arribar al fenotip que s'hagin mesurat numèricament, poden tractar-se com a dades òmiques. Aquest conjunt de passos que integren les dades òmiques, poden dividir-se en diversos subcamps com són:

Genòmica. Utilitza la seqüenciació de tecnologies per estudiar el genoma. Les dades que usa són les seqüencies d'ADN.

Proteòmica. Estudia a gran escala les proteïnes. Utilitza estructures de proteïnes en tres dimensions.

Transcriptòmica. Estudia el conjunt de molècules d'ARN en una població cel·lular. Examina nivells d'expressió de l'ARNm. Una de les tecnologies que utilitza són els *microarrays*.

Metabolòmica. Cerca la determinació dels nivells de metabòlits al metaboloma i dels seus canvis en el temps com a conseqüència d'estímuls.

2.2.1 Tecnologies d'alt rendiment

S'entén com a tecnologies d'alt rendiment (*high-throughput techniques*) a tot el conjunt de tècniques computacionals que intenten atorgar millores en els processos experimentals. El tractament de dades òmiques sovintment necessita l'ús de processos automatitzats que requereixen elevats costos computacionals. El que intenta és millorar la capacitat del treball amb dades òmiques tot mantenint una alta sensibilitat i especificitat. Una de les propostes més usuals és la computació paral·lela.

2.2.2 Problemàtiques en el tractament de dades òmiques

A continuació exposem alguns dels problemes o inconvenients que podem trobar-nos a l'analitzar dades de caire òmic.

- **Comparacions múltiples**: En el camp òmic, sovintment disposem d'individus amb milers de variables, fet que incrementa la probabilitat de patir errors de tipus I. I per tant, que quedi immergit l'efecte de diverses variables. Per tant, aquest fet, no és un problema computacional sinó estadístic. Possibles solucions són l'ús de mètodes heurístics per controlar les proporcions esperades de falsos positius. I a la vegada, intentant maximitzar la potència estadística.

- **Gran dimensionalitat de les dades:** Aquest és un problema compartit per diversos camps d'aplicació de l'estadística. La gran dimensionalitat en les dades fa que no puguin ser visualitzades gràficament i per tant sigui més complexa la seva interpretació conjunta. En aquest problema hi acudeixen mètodes per a la reducció de la dimensionalitat, que busquen mantenir la màxima variabilitat en les dades.

- **El paradigma de la n-petita i la p-gran**: La teoria estadística està definida amb assumpcions del tipus $p < N$ i $N \to \infty$. El fet que no es compleixin, pot fer trontollar les estimacions. Una de les causes és deguda al fet que les formulacions estadístiques estan basades amb eines d'àlgebra lineal. És un fet molt freqüent que el trobem per exemple en els experiments amb *microarrays*, on per a un individu disposem de milers de gens (variables). Solucions a aquests problemes són l'ús d'eines estadístiques que controlin i maximitzin la sensibilitat i l'especificitat.

- **Soroll en les dades biològiques d'alt rendiment**: Com tot instrument de mesura, posseeix error, aquest fet també es troba en la informació biològica. S'ha de tenir força present, ja que molts tractaments òmics presenten una gran semblança entre els individus, i un possible error pot ja permetre una conclusió diferencial fictícia. Per tant, tant l'ús de mètodes per a la cerca de possibles errors en la depuració, com d'eines de qualitat en el preprocés són bones alternatives per a fer front a aquests tipus d'errors.

- **Integració de fonts d'informació múltiples i heterogènies**: Un últim inconvent és el de disposar d'eines per a la integració de fonts de diversos camps biològics. La heterogeneïtat en les mesures i formats que presenten aquestes dades dificulta una possible anàlisi conjunta. Aquest és un problema en què es treballa en l'actualitat. La possibilitat d'unificar diferents fonts òmiques permetria el que es coneix com la medecina personaliztada. On per mitjà de diverses anàlisis mèdiques, un pacient podria ser diagnosticat de forma automatitzada.

Capítol 3

CLASSIFICACIÓ I MÈTODES KERNEL

Al llarg d'aquest treball, tractarem l'algorisme MDP i s'exposarà una visió detallada de la seva fórmulació i aplicació en els pròxims capítols.
És per això, que donarem ara una introducció, o idea intuitiva, del que és el paradigma de la classificació, juntament amb els mètodes kernel. Que seran útils per a extendre el MDP al món kernel.

3.1 Concepte de classificació

La classificació consisteix en la predicció d'una variable aleatòria y (anomenada resposta o *output*) a partir d'un conjunt de variables aleatòries $\{\mathbf{x}_i\}_{i=1}^{d}$ (anomenades predictors). Donada una mostra d'individus $S = \{(x_1, y_1) \ldots, (x_n, y_n)\}$ on $x_i = \begin{bmatrix} x_{i1} \\ \vdots \\ x_{id} \end{bmatrix} \in \mathcal{X} \subset \mathbb{R}^d$. $\mathcal{X}$ és el conjunt que conté a tots els individus o punts, que són un conjunt de d característiques mesurades. I y_i és un possible nivell categòric del conjunt finit $\mathcal{Y}$. Els elements de $\mathcal{Y}$, també s'anomenen classes.
Un regla de classificació o també anomenat classificador, serà una funció $h\colon \mathcal{X} \to \mathcal{Y}$.
Per tant, quan observem un nou individu x, predirem a partir de $h(x)$ a quina determinada classe de $\mathcal{Y}$ pertany.

La classificació supervisada, parteix d'una mostra de vectors de $\mathcal{X}$, on cada individu d'aquesta mostra va associat amb la seva vertadera etiqueta de $\mathcal{Y}$. Aquesta mostra, se l'anomena dades d'entrenament (*training data*). Per tant, el classificador h haurà de llegir aquest conjunt de dades que tenen una variable resposta coneguda, per tal de crear un regla o patró de distinció entre les classes de $\mathcal{Y}$. Un cop finalitzat aquest procés, el classificador romandrà entrenat. No obstant, caldrà observar en quina mesura. És per això que hauran d'aplicar-se tècniques de validació, les quals ens donaran una visió més coherent sobre el grau d'objectivitat amb que prediu el classificador.

Una vegada es disposa del classificador validat, podrem introduir un *input* d'individus nous, els quals el classificador h ens retornarà en quina etiqueta o classe és més encertat que un nou individu hi pertanyi.

3.2 Tècniques de validació

La validació creuada (*cross validation*) és un procediment per a l'obtenció d'una estimació de l'error d'un classificador entrenat. La seva aplicació permet informar del rendiment esperat que tindrà un classificador a l'hora de predir nous individus.
Una de les raons per les quals necessitem un mètode de validació és per la desconeixença de la robustesa del mètode. El fet d'haver aconseguit una bona precisió a partir d'unes dades d'entrenament no és una condició suficient. Per exemple, si disposéssim d'una quantitat més nombrosa de dades d'entrenament, podria passar que un mètode poc robust no es veiés influït pels individus rars. Pel simple fet que fossin representants en una petita part, i quedés el seu efecte anul·lat gràcies a la quantitat major restant d'individus poc problemàtics. Per tant, en fer subdivisions de les nostres dades d'entrenament, ens trobaríem probablement en subdivisions formades amb aquells individus amb què el mètode hi tenia més dificultat. Seria allà on classificaria amb més error. És per això, que la validació ens atorga una mesura de l'error a vegades menys optimista, però segur que més realista.

- **k-fold cross validation**: Consisteix en una divisió de les dades d'entrenament amb k parts. Cadascuna d'aquestes parts s'anomena plec (fold). Si el total de dades és n, un plec tindrà una mida de n/k. $k-1$ plecs s'utilitzen com a dades d'entrenament i el plec restant, s'utilitza com a dades de validació. La part que fa de validació, en el següent pas, s'inclou en la mostra d'entrenament. I un plec de mida n/k que ha estat dins de les dades d'entrenament passa a ser de validació. Aquest procés és aplicat k vegades. És a dir, tots els plecs hauran sigut dades d'avaluació una vegada. Per a cadascun d'aquestes k estimacions, es calcularà una mesura de l'error, que n'és la mitjana. I constituirà l'error de la validació creuada.
  La selecció de la partició k, dependrà de l'investigador. Un criteri força acceptat és definir $k = 10$. Cal a dir, l'estimació de l'error serà un esdeveniment aleatori, ja que la partició inicial (agrupació dels individus en plecs) dependrà d'una assignació aleatòria. És per això que sovintment, al efectuar validacions s'apliquen repeticions, per a reduir possibles biaixos.

- **Leave one out cross validation (LOOCV)**: Aquest és un cas particular de k-fold cross validation. Per a $k = n$. Tots els individus són predits per mitjà de la resta. Per tant, aquesta estimació és determinista. És a dir, obtindrem sempre la mateixa estimació de l'error, ja que al ser $k = n$ no podran haver diferentes maneres de regionalitzar els plecs. Cal a dir, que té com a inconvenient que presenta una

elevada variància. Ja que les particions només es diferencien per una observació. I per tant, les estimacions seran molt similars. És a dir, es produirà un solapament entre classificadors.

- **Bootstrap**: Consisteix en aplicar el popular mètode de remostreig per a la selecció de les particions. On cada plec serà una mostra aleatòria amb repetició, d'una determinada mida inferior a n. En cada mostra, els individus no inclosos seran per a la mostra de *test*. Aquest procediment serà efectuat B vegades. La mida de cada mostra B no té perquè ser constant.

3.3 Mètodes kernel

Sovint, poden presentar-se patrons no lineals entre els individus que volem classificar, els quals fan que es pugui trontollar la precisió del classificador. Els mètodes kernel, s'ocupen d'aquest problema. Són extensions no lineals per a algorismes de classificació lineal, que integren la capacitat de transformar els individus a un altre espai d'alta dimensió on si es permet una millor separació entre classes. Per a que tècniques lineals puguin estendre's al món kernel, caldrà que pugui aplicar-se entre individus la idea que veurem més endavant coneguda com el *kernel trick*.

3.3.1 Idea general

Sovint volem captar patrons no lineals en les dades. Tant la regressió no lineal com la classificació no lineal. L'objectiu dels mètodes kernels és *mapejar* l'espai inicial $\mathcal{X}$ en un espai de dimensió superior $\mathcal{F}$ on poden aconseguir-se ajustos lineals. No obstant, efectuar aquests *mapejos* pot significar un elevat cost computacional. Aquest problema pot ser minvat, gràcies al *kernel trick*. Aquesta transformació es defineix com:

$$\begin{aligned} \Phi\colon \mathcal{X} &\to \mathcal{F} \\ x &\mapsto \Phi(x) \end{aligned}$$

Calcular aquestes projeccions $\Phi(x)$ pot resultar molt costós. Mostrem dos exemples, per a visualitzar el pròposit que volen duu a terme aquestes aplicacions Φ. Tal i com es mostra en la següent figura, disposem d'un conjunt d'individus provinents de dues classes. (Vermells i verds). Els quals, voldriem que poguessin ser separats linealment. Romanen en un espai d'una dimensió, on no estan diferenciats linealment. (Si ho estarien amb una línia vertical).

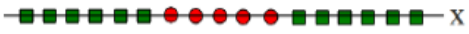


Figura 3.1: Individus de dues classes, mesurats a partir d'una única característica x, [22]

Definint, l'aplicació:

$$\Phi : \mathcal{R} \to \mathcal{R}^2$$
$$x \mapsto \{x, x^2\}$$

On sense cap pèrdua de generalitat podem assumir que el conjunt $\mathcal{R}$ és $\mathbb{R}$. Podem ara, representar els individus en el nou espai de característiques generat per Φ.

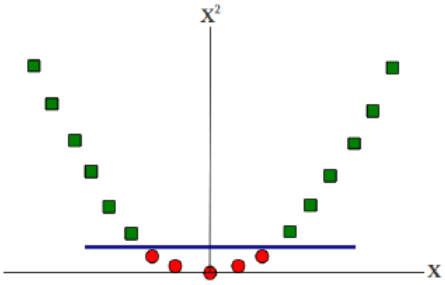


Figura 3.2: Projecció dels mateixos individus en el nou espai generat per Φ, [22]

De fet cada mesura x per individu d'aquest nou espai en forma de pla no és res més que una derivació de la representació inicial, no obstant, ens permet projectar les dades de forma linealment separable.

Un altre atre atractiu exemple és el que es mostra en la següent il·lustració.

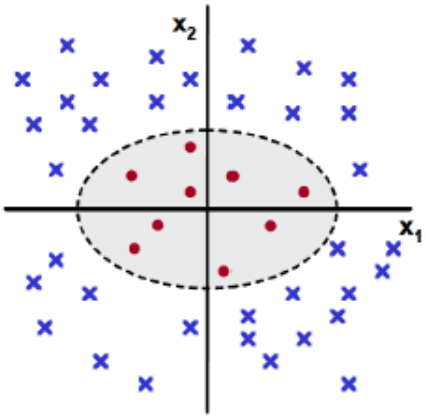


Figura 3.3: Individus de dues classes en un mapa bidimensional, [22]

Que representa, un conjunt d'individus provinents de dues classes, els quals se'ls ha mesurat dues característiques, x_1 i x_2. Novament, s'aprecia un patró no lineal, que no permet una separació adequada per mitjà d'un hiperplà. La següent aplicació:

$$\begin{aligned} \Phi\colon \mathcal{R}^2 &\to \mathcal{R}^3 \\ x &\mapsto \{x_1^2, \sqrt{2}x_1x_2, x_2^2\} \end{aligned}$$

On ara cada indiviu té tres característiques, totes derivades de les dues inicials. Representant la transformació de Φ a un nou espai de característiques en tres dimensions, observem:

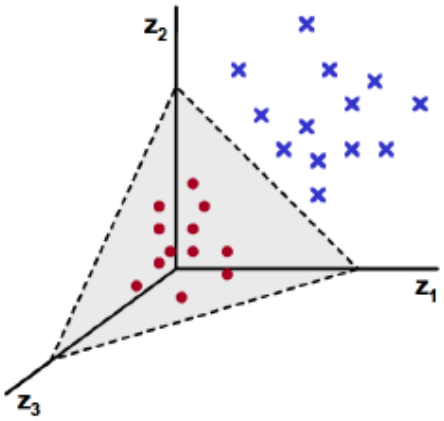


Figura 3.4: Individus de dues classes en el nou mapa tridimensional generat per Φ, [22]

Contemplem, com haver projectat els punts per mitjà de Φ a l'espai, ens ha proporcionat una perfecta separació lineal.

Fins ara hem tractat transformacions a partir d'individus amb dimensions petites (una i dues dimensions). El problema però, sorgeix quan tenim el cas habitual, individus pertanyents a d dimensions. On la transformació Φ ja no pot ser visualitzada gràficament, i en alguns casos, ni tan sols computada. No obstant, si calculem el producte escalar entre dos individus x i z com $\Phi(x)^T\Phi(z)$. Estem efectuant una mesura de similitud entre dos individus sense haver d'obtenir les imatges de Φ. A part, tot i que no calculem les projeccions individuals Φ per a cadascún dels individus, aconseguim tenir mesures de semblança entre ells cosa que ja és suficient per a poder definir l'espai de característiques. És aquesta la gran idea del *kernel trick*.

3.3.2 Definició formal de funció kernel

Donada l'aplicació $k\colon \mathcal{X} \times \mathcal{X} \longrightarrow \mathcal{R}$[1] i els individus $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathcal{X}$. La matriu K amb dimensió $n \times n$ s'anomena matriu Kernel[2] de k respecte als individus $\mathbf{x}_1, \ldots, \mathbf{x}_n$.

$$\mathrm{K}=\begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_n, \mathbf{x}_1) & \cdots & k(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

Sempre i quan compleixi dues condicions: que sigui simètrica i definida positiva. Amb aquesta definició, ja pot exposar-se el resultat que es conèix com el teorema de Mercer.

[Teorema de Mercer] Per a qualsevol funció kernel k, sobre un espai d'entrada no buit $\mathcal{X}$, existeix un $\mathbb{R}$-Espai de Hilbert[3] i un mapeig $\Phi\colon \mathcal{X} \longrightarrow \mathcal{H}$ tal que $\forall\, \mathbf{x}, \mathbf{z} \in \mathcal{X}$

$$k(\mathbf{x}, \mathbf{z}) := \langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle_{\mathcal{H}}$$

El teorema de Mercer ens diu que en condicions bastant generals, qualsevol aplicació kernel simètrica i definida positiva pot representar-se com a un producte intern en un espai de Hilbert. Dit d'una altra manera, que les mesures de similitud en una matriu kernel poden considerar-se com a productes escalars en un espai Hilbert (on la dimensió pot esdevenir molt gran, fins i tot infinita). Per tant, podrem treballar només amb productes interns els quals permetran conectar-nos a l'espai de Hilbert definit implícitament. Aquest espai, és el que anomenarem com espai de característiques o *feature space*. Cal a tenir present, que per a una matriu kernel que no complís alguna de les dues propietats (simetria i definida positiva) dit teorema ja no seria aplicable.

Com ja havíem comentat, el *kernel trick* és l'ús de funcions kernels sense haver de definir explícitament la funció en l'espai de característiques. I ens diu que ja no cal idear una transformació a l'espai de característiques, ja que es defineix de manera implícita amb la mesura de semblança que estableix la funció kernel. Vist amb més detall, una matriu kernel ha de ser:

- **Simètrica:** significa que $K_{ij} = K_{ji}$ que és el mateix que $K = K^T$. Aquesta condició ha de ser totalment vàlida, per construcció d'un producte escalar. Ja que de no ser així, implicaria que $K_{ij} = \langle \Phi(i), \Phi(j) \rangle \neq \langle \Phi(j), \Phi(i) \rangle = K_{ji}$ que no pot ser vàlid, ja que un producte intern és commutatiu. Gràficament la distància entre dos vector $\mathbf{x}, \mathbf{z}, \in \mathcal{X}$ hauria de ser commutativa, almenys per a una correcta representació de l'espai de característiques.

- **Definida positiva:** és a dir, la matriu kernel K de dimensió $n \times n$, ha de complir que $a^T K a \geq 0$. Per a tot $a \in \mathcal{R}^n$. En cas de no restringir-se aquesta propietat,

[1] pot assumir-se $\mathcal{R} = \mathbb{R}$

[2] també anomenada matriu de Gram.

[3] Un espai de Hilbert és un espai complet de producte intern.

podrien produir-se distàncies quadrades negatives i per tant, no podrie aplicar-se la desigualtat triangular. De ser aquest cas, l'espai de característiques no podria ser interpretat com un espai de Hilbert.

Afegim que les funcions kernel són mesures de similitud i no s'han de confondre amb distàncies. Ja que no estan construïdes amb restriccions per a que compleixin els axiomes d'una mètrica.

Exemples

A partir de la següent funció kernel:
$k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^T\mathbf{z}$

$$K = \begin{bmatrix} \mathbf{x}_1^T\mathbf{x}_1 & \cdots & \mathbf{x}_1^T\mathbf{x}_n \\ \vdots & \ddots & \vdots \\ \mathbf{x}_1^T\mathbf{x}_1 & \cdots & \mathbf{x}_n^T\mathbf{x}_n \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1^T \\ \vdots \\ \mathbf{x}_n^T \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 & \cdots & \mathbf{x}_n \end{bmatrix}$$

La propietat de simetria ja es comproba trivialment, pel fet de que la funció kernel és únicament el producte escalar. Pel que fa a la definició de la matriu, caldria veure's si $a^TKa \geq 0$. Que es comproba:

$$a^TKa = \begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix} K \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} = (a_1\mathbf{x}_1^T + \ldots + a_n\mathbf{x}_n^T)(a_1\mathbf{x}_1 + \ldots + a_n\mathbf{x}_n) =$$

$$= (a_1\mathbf{x}_1 + \ldots + a_n\mathbf{x}_n)^T(a_1\mathbf{x}_1 + \ldots + a_n\mathbf{x}_n) = \|a_1\mathbf{x}_1 + \ldots + a_n\mathbf{x}_n\| \geq 0 \ , \forall a$$

3.3.3 *Reproducing* kernel

Sigui $\mathcal{R}^{\mathcal{R}^d}$ el conjunt que conté totes les funcions que van de $\mathcal{R}^d$ a $\mathcal{R}$ és a dir:

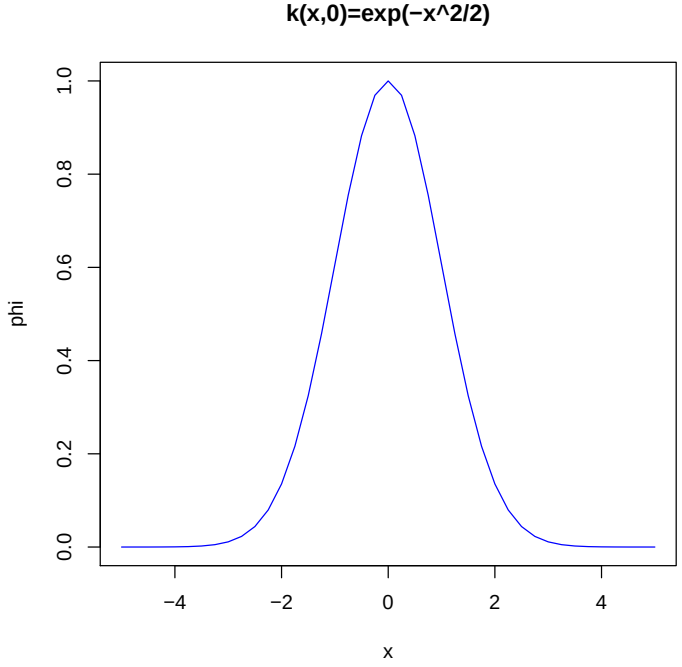
$$\mathcal{R}^{\mathcal{R}^d} = \{f \colon \mathcal{R}^d \to \mathcal{R}\}$$

Aquest conjunt el que farà serà agrupar totes les funcions que dependran únicament d'un individu per a l'espai d'entrada. Un individu que pertanyerà a $\mathcal{R}^d = \mathcal{X}$. Remarcar que les funcions kernel definides anteriorment, són del tipus $k \colon \mathcal{R}^d \times \mathcal{R}^d \to \mathcal{R}$. Seguidament exemplifiquem alguns casos amb diferentes funcions kernel.

Donada la funció kernel $k(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{(x-z)^2}{2}\right)$ i considerant que està definida sota el cos dels reals $\mathcal{R} = \mathbb{R}$, podem transformar-la en una funció univariant. Fixant-li un dels dos paràmetres. Si en fixem un a zero ens quedem amb:

$$\Phi_0(x) = k(\mathbf{x}, 0) = \exp\left(-\frac{x^2}{2}\right)$$

La qual ara passe a estar definida com a $\Phi_0(x)\colon \mathbb{R} \to \mathbb{R}$ que pertany al conjunt $\mathcal{R}^{\mathcal{R}}$.
Per tant, com $x \in \mathbb{R}$, gràcies a Φ, hem aconseguit transformar a x que és un punt unidimensional, en $\Phi(x) \in \mathbb{R}^2$ que ja és una distribució. Podem observar en el següent gràfic $\Phi(x)$.

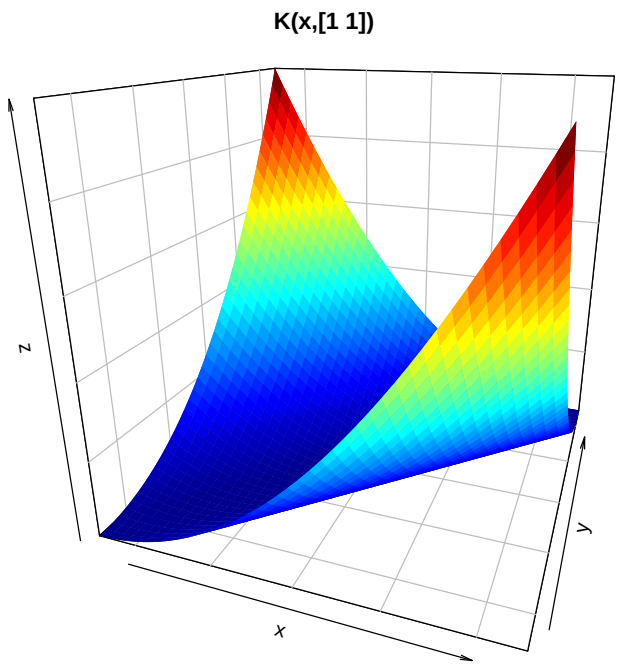


Suposant que $k\colon \mathcal{R}^d \times \mathcal{R}^d \to \mathcal{R}$ és un kernel. El mapa kernel de reproducció (*Reproducing kernel map*) és una funció

$$\Phi\colon \mathcal{R}^d \to \mathcal{R}^{\mathcal{R}^d} \text{ tal que } \Phi(\mathbf{z}) = \phi_{\mathbf{z}}(\cdot) = k(\cdot, \mathbf{z})$$

El punt $\cdot$ indica un argument faltant, i la funció passa a dependre només de $\mathbf{z}$.

Exemplifiquem un altre cas per a vectors o individus que provinguin de l'espai d'entrada $\mathbb{R}^2$. Per mitjà d'una funció Φ enviarem els punts o vectors a un espai de tres dimensions. Sigui $k\colon \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ definida com $k(\mathbf{x}, \mathbf{z}) = (x_1 z_1 + x_2 z_2)^2$. Podem crear una aplicació Φ com una modificació de k. És a dir: $\phi_{[1\ 1]^T}(\mathbf{x}) = k(\mathbf{x}, [1\ 1]^T) = (x_1 + x_2)^2$. El que hem fet és fixar l'individu $\mathbf{z} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. Que pertany a $\mathbb{R}^2$, i ara, per mitjà de Φ, transforma a $\mathbf{z}$ en una superfície. Podem observar-ho a continuació:



On per a qualsevol $\mathbf{x} \in \mathbb{R}^2$ la seva imatge per mitjà de $\phi_{[1\ 1]^T}(\mathbf{x})$ romandrà en la superfície mostrada. De fet, aquesta corba, la formen el conjunt de totes les imatges dels vectors de $\mathbb{R}^2$.

El que exposarem a continuació són les propietats que ens permetran la *kernelització* del MDP.

Donada una funció kernel $k(\mathbf{x}, \mathbf{z})$ sota les condicions del teorema de Mercer. Amb espai d'entrada $\mathcal{X} = \mathbb{R}^n$. Per a qualsevol funció kernel, existirà una aplicació Φ a un espai de característiques $\mathcal{F}$, satisfent que $\langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle$. És a dir, la funció kernel podrà ser utilitzada per a avaluar un producte intern en l'espai $\mathcal{F}$.
Considerant el conjunt de funcions $\mathcal{X}^{\mathcal{X}^d} = \{k(\cdot, \mathbf{x}) \mid \mathbf{x} \in \mathcal{X}\}$. L'espai lineal de funcions que genera a aquest conjunt de funcions és únic i sempre pot ser completat amb un espai de Hilbert. La propietat transcendental d'aquests espais de Hilbert (que hem anat veient de forma exemplificada), és la propietat de reproducció (*reproducing kernel*). La definim a continuació.

Sigui $\mathcal{F}$ un espai de Hilbert de funcions $f \colon \mathcal{X} \to \mathbb{R}$ definit en un conjunt no buit $\mathcal{X}$. Una funció $k \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ és anomenada *reproducing kernel* de $\mathcal{F}$ si satisfà:

- $\forall \mathbf{x} \in \mathcal{X},\ k(\cdot, \mathbf{x}) \in \mathcal{F}$.
- $\forall \mathbf{x} \in \mathcal{X}, \forall f \in \mathcal{F},\ \langle k(\cdot, \mathbf{x}), f \rangle_{\mathcal{H}} = f(\mathbf{x})$

Un espai de característiques $\mathcal{F}$, serà un *reproducing kernel Hilbert space* (RKHS) si i només si té un *reproducing* kernel. A més, serà únic. Aquesta propietat serà crucial ja que permetrà donar una definició de distància.
Per tant, la distància entre dos vectors $\mathbf{x}$ i $\mathbf{z}$ de $\mathcal{F}$ en el RKHS, es calcularà com:

$$d(\mathbf{x}, \mathbf{z}) = \|\Phi(\mathbf{x}) - \Phi(\mathbf{z})\| = \sqrt{k(\mathbf{x}, \mathbf{x}) + k(\mathbf{z}, \mathbf{z}) - 2k(\mathbf{x}, \mathbf{z})} \tag{3.1}$$

Aquesta serà la manera de mesurar la distància entre dos individus en la versió kernel MDP. I dependrà de la funció kernel utilitzada.

Capítol 4

L'ALGORISME MDP

Aquest capítol contempla tota l'explicació de l'algorisme d'aquest projecte, el MDP. Es presenta la formulació del mètode, la justificació i metodologia del seu ús i alguns exemples pràctics d'aplicació.

4.1 Idea general

En aquest capítol es proposarà un algorisme de classificació per a l'assignació d'individus entre un dels grups o poblacions de partida. La idea d'aquest mètode serà efectuar una divisió sobre l'espai mostral d'individus, basant-nos amb una funció de distància. La regla de Bayes es veurà immersa en l'algoritme com a mesura d'assignació dels individus a les poblacions. També proporcionarà per a cada individu, una mesura de confiança en l'assignació.

Com que a la literatura ja trobem una quantitat força elevada de mètodes de classificació, exposarem algunes de les raons que serviran com a motivació per a l'aprenentatge d'aquest mètode.

Mètode personalitzat

La vessant clàssica de l'anàlisi discriminant parteix d'una determinada població Ω i un conjunt de subpoblacions $(E_1, ..., E_k)$ que constitueixen una partició d'Ω. I el que es busca és, assignar un individu $w \in \Omega$, a una de les classes $E_1, ..., E_k$ a partir de les mesures observades d'un conjunt de variables $\mathbf{X}$.

Tal i com s'esmenta a l'article [6], els mètodes més populars d'anàlisi discriminant són l'anàlisi discriminant lineal (LDA) i l'anàlisi discriminant quadràtic (QDA). Els quals s'han millorat per a tractar coneixement previ a priori. Encara que amb assumpcions

requerides de normalitat. Un altre dels possibles inconvenients d'alguns mètodes de discriminació és la no adaptació del mètode per al cas particular d'estudi. Com pot ser el cas d'individus amb variables mixtes que poden fer perdre l'eficiència en les prediccions. El MDP en canvi, presenta la possibilitat de la pròpia selecció de la distància a utilitzar, on l'experimentador podrà escollir la més oportuna funció del seu cas d'estudi. D'igual manera, podrà elegir la distribució a priori, que trobi més adient, per a les poblacions. El MDP doncs, no depèn d'una distància específica. I no necessita el compliment d'hipòtesis prèvies de normalitat.

Mesura de la confiança en l'assignació

El fet que cada individu disposi d'una mesura de qualitat en la regió que ha estat inclòs, permetrà a l'investigador conèixer en quin grau de confiança s'ha assignat un individu. I a la vegada, valorar el grau d'objectivitat en la classificació.

Ús del teorema de Bayes sense inconvenients

Suposem que volem classificar un individu w sense tenir informació observada $X(w) \in \mathbb{R}^p$, però sí probabilitats a priori per a cadascuna de les subpoblacions $(1, \ldots, k)$ conegudes o estimables a partir d'una mostra controlada. De ser aquest cas, el millor criteri seria assignar w a la subpoblació a priori més probable. És a dir, w seria assignat a E_i si i només si $\pi_i = \text{màx}\{\pi_1, \ldots, \pi_k\}$. Aquest, seria el criteri òptim d'assignació per al cas més ingenu. No obstant, el més corrent serà trobar-nos en situacions amb valors observats $X(w)$ coneguts. Llavors un criteri escaient serà assignar w a la subpoblació a posteriori més probable. És a dir, disposant d'un possible conjunt de valors observats $\{r_1, \ldots, r_n\}$ per a cadascún dels n individus, el teorema de Bayes podrà ser utilitzat per a calcular $P(E_i|r_j)$. I per tant, assignar-li a l'individu w amb valor observat r_j la classe E_i que maximitzi la probabilitat a posteriori. Formalment, s'escull E_i si $P(E_i|r_j) = \text{màx}\, P(E_i|r_j)$ per a $i = 1, \ldots, k$.

A la pràctica, l'estimació de $P(r_j|E_i)$ no mostra problemes pràctics quan la mostra és suficientment gran. És a dir, sempre i quan ens trobem davant d'un gran nombre de possibles mesures r d'individus. Serà llavors, quan podrem encara, assumir un model estadístic per a les subpoblacions E_i. Les $P(r_j|E_i)$. Aquest és el problema comú que trobem en la literatura també anomenat com la *maledicció de la dimensionalitat*.

El MDP evita aquest problema gràcies a la construcció d'una partició $\mathscr{R} = \{R_1, \ldots, R_k\}$ sobre l'espai $\mathbb{R}^p$ amb tantes subregions com subpoblacions. Aquesta particició es durà a terme per mitjà d'una funció de distància.
Seguidament exposarem una explicació detallada del mètode.

4.2 Explicació

Primerament, exposarem el desenvolupament del mètode de forma general. Aquesta primera part serà per a comprendre i a la vegada tenir una visió matemàtica sobre quin és el problema a abordar i què s'està intentant fer. Una vegada transmesa la idea, procedirem a exposar i formular el mètode per al cas empíric, que és quan es té una mostra controlada.

4.2.1 Cas general

Denotem a la població global Ω i a $E_1, \ldots, E_k$ al conjunt de subpoblacions que divideixen aquest espai mostral Ω. Considerem el vector $\mathbf{X}$ de $\mathbb{R}^p$, que representa els valors observats per a un individu $\omega \in \Omega$. A més, suposem que disposem d'una distància adequada d, establerta per l'experimentador, acertada per a tractar correctament amb la naturalesa que presenti el vector aleatori $\mathbf{X}$. Cal afegir, que ara estem presentant el cas general, és a dir, per a almenys tots els individus que formen una determinada població. O fins i tot, per a una població $\mathbb{R}^p$.
Ara ens centrarem en la construcció de la partició $\mathscr{R} = \{R_1, \ldots, R_k\}$ sobre l'espai $\mathbb{R}^p$.
Denotem com a $M \subseteq \mathbb{R}^p$ al conjunt que contindrà totes les imatges de cadascún dels individus de la població. I indiquem com a $\mathbf{x}\colon \Omega \to M$ a la funció bijectiva que ens enviarà cada individu ω al seu respectiu vector de característiques r.

Considerem un punt $r \in M$ i una mostra aleatòria, $\mathbf{w}_\lambda$, de mida $k\lambda$ (sent $\lambda \in \mathbb{N} \setminus \{0\}$ un paràmetre arbitrariament escollit). S'obté agafant λ elements de cada subpoblació.

$$\mathbf{w}_\lambda = (w_{11}, \ldots, w_{1\lambda}, \ldots, w_{k1}, \ldots, w_{k\lambda}); (w_{11}, \ldots, w_{1\lambda}) \in E_1^\lambda, \ldots, (w_{k1}, \ldots, w_{k\lambda}) \in E_k^\lambda \tag{4.1}$$

Per a cada element, $r \in M$, podem definir el conjunt $S_i^\lambda(r)$ compost de totes les possibles mostres obtingues en (4.1), en les quals el mínim de les distàncies entre r i els $k\lambda$ valors observats obtinguts de $\mathbf{w}_\lambda$ correspon a un dels λ de E_i inclosos en $\mathbf{w}_\lambda$. Formalment:

$$S_i^\lambda(r) = \Big\{\mathbf{w}_\lambda \in E_1^\lambda \times \ldots \times E_k^\lambda \mid \exists \beta \in \{1, ..., \lambda\} \text{ amb } \min\{d(r, X(w_{11})), \ldots, d(r, X(w_{k\lambda}))\}\Big\} = d(r, X(w_{i\beta}))\ i = 1 \ldots, k \tag{4.2}$$

De moment encara no ens afanyem en estimar quantes mostres possibles tindrem. Ens interessarà en el moment en que disposem d'una mostra controlada. Per exemple, per a conèixer la grandària amb la que estem tractant.

Cada $\mathbf{w}_\lambda$ tindrà $k\lambda$ individus els quals els hi correspondrà $k\lambda$ punts imatge. Per tant, per a cada $\mathbf{w}_\lambda$ hi haurà un punt imatge que és el més proper a r. De totes les $\mathbf{w}_\lambda$ mostres possibles, per a algunes, el punt més proper serà la imatge d'un individu que pertany a una determinada subpoblació E_i i per a altres una altra subpoblació E_j. Doncs bé, el

conjunt $S_i^\lambda(r)$ agruparà mostres, i contindrà totes aquelles en que el punt més proper a r, sigui la imatge d'un element de E_i.

En l'improbable cas d'un emptat, és a dir, succeís que en una determinada mostra $\mathbf{w}_\lambda$, la distància mínima entre r i els punts $\mathbf{x}(w_{11}), \ldots, \mathbf{x}(w_{k\lambda})$ fos la mateixa tant per a un individu de E_i com per a un de E_j. Llavors hauríem d'assignar $\mathbf{w}_\lambda$ ja fos a $S_i^\lambda(r)$ o $S_j^\lambda(r)$ per mitjà d'una assignació aleatòria. De forma anàloga, efectuariament també una assignació aleatòria si ens trobéssim amb empats en diverses classes.

A continuació es defineixen les quantitats $Q(S_1^\lambda(r)), \ldots, Q(S_k^\lambda(r))$ que seran els elements determinants de la partició $\mathscr{R}_\lambda$. $Q(S_i^\lambda(r))$ simplement denota la probabilitat del conjunt $S_i^\lambda(r)$. És a dir, la probabilitat que un punt $r \in M$ estigui més proper a E_i que no pas a cap altra classe E_j. Aquest concepte pot ser interpretat com la probabilitat de que si agafessim una mostra a l'atzar $\mathbf{w}_\lambda$, definida com en (4.1), d'entre totes les mostres possibles, aquesta mostra fos una de les que pertany a $S_i^\lambda(r)$. Dit d'una altra manera, és la proporció de totes les mostres $\mathbf{w}_\lambda$, per les quals el mínim de totes les distàncies entre r i els elements de $\mathbf{w}_\lambda$, és assolit per un element de E_i.
Afegim que no hem de fer abús del llenguatge, i interpretar a $Q(S_i^\lambda(r))$ com la probabiltat de que r pertanyi a S_i^λ. Estem agrupant mostres en el conjunt S_i^λ per a un r en funció d'una mesura de la llunyania entre els individus, posant especial èmfasi amb com d'aprop està un individu respecte la resta d'individus d'una subpoblació. I no com una selecció binària de si l'individu pertany o no pertany al conjunt S_i^λ.

Repassem breument el que fins ara hem anat construint. Per a cada punt r de M, tenim un total de k conjunts. Que són els $S_1^\lambda(r), \ldots, S_k^\lambda(r)$. I a la vegada, també tenim per a ell, les probabilitats d'aquests k conjunts, que són les $Q(S_1^\lambda(r)), \ldots, Q(S_k^\lambda(r))$. Ara ja sí, tenim els elements necessaris per a efectuar la partició de M.

Denotem a la partitció com a $\mathscr{R}_\lambda = \{R_1^\lambda, \ldots, R_k^\lambda\}$. Que haurà depès del paràmetre λ que considerva la mida de selecció mostral. I per suposat, de la idònia distància d assumida al inici. Es defineix doncs de la següent manera:

$$R_i^\lambda = \{r \in M \mid Q(S_i^\lambda(r)) > Q(S_j^\lambda(r)),\ j = 1, \ldots, k;\ j \neq i\}\ i = 1, \ldots, k \qquad (4.3)$$

On R_i^λ representa al conjunt de resultats pels quals la probabilitat de ser més proper a E_i que a una altra possible classe E_j és màxima. Per tant, un punt pertany a la regió R_i^λ, si mirant quin és el punt més proper a r de cada mostra, resulta que aquest punt és més vegades el d'una mostra de E_i que el de qualsevol altra mostra. És a dir, es defineix la partició de manera que la regió R_i^λ està formada per tots aquells punts que, agafant una mostra $\mathbf{w}_\lambda$ a l'atzar, la probabilitat que el punt més proper a r sigui un element de la

classe E_i, és més gran que la probabilitat que sigui un element de qualsevol altra classe. Novament, en el possible i a l'hora improbable cas d'un empat, $Q(S_i^\lambda(r)) = Q(S_j^\lambda(r))$, empraríem una elecció aleatòria.
Cal remarcar que la partició $\mathscr{R}_\lambda$ de M automàticament indueix una partició sobre l'espai mostral Ω, ja que tindrem que per a cada r li correspondrà un únic individu $\mathbf{x}^{-1}(r) = v$, sent $v \in \Omega$. Remarquem que és totalment viable que un individu w de E_i, no vagi a parar a R_i^λ sinó a qualsevol altra regió R_j^λ, $i \neq j$. De fet, és una qüestió que depèn en gran mesura de com s'efectuà la partició inicial d'Ω. Aquest aspecte l'abordarem més tard en els comentaris del mètode 4.3.

Per mitjà del teorema de Bayes, podrem calcular les probabilitats a posteriori de la següent forma:

$$P(E_j|R_i^\lambda) = \frac{P(R_i^\lambda|E_j) \cdot \pi_j}{\sum_{\alpha=1}^{k} P(R_i^\lambda|E_\alpha) \cdot \pi_\alpha} \tag{4.4}$$

I un cop calculades, podrem efectuar unes assignacions de les nostres regions $R_i^\lambda, i = 1, \ldots, k$ a les subpoblacions $E_1, \ldots, E_k$. Aquesta regla de classificació, l'efectuarem de la següent manera:
Assignarem la regió R_i^λ a la subpoblació E_j si i només si:

$$P(E_j|R_i^\lambda) = \text{màx}_j\{P(E_1|R_i^\lambda), \ldots, P(E_k|R_i^\lambda)\} \tag{4.5}$$

Cal remarcar que el mètode MDP no només ens proporciona una regla d'assignació, sinó que les probabilitats $Q(S_i^\lambda(r))$ $(i = 1, \ldots, k)$ poden ser interpretades com a mesures del grau de confiança en la inclusió d'un resultat determinat r a una regió R_i^λ. A més, els conjunts $Q(S_i^\lambda(r))$, seran invariants sota canvis en la funció de distància. El qual ens atorgaran una mesura fixada del grau d'acord que presenta haver agrupat un determinat r a una determinada regió R_i^λ. I per tant, proporcionarà informació extra per al investigador. Com per exemple, per a un exploració d'individus crítics.

4.2.2 Cas empíric

A la pràctica acostumarem a tenir una mostra controlada G amb N individus coneguts que provindran de les k possibles subpoblacions $(E_1, \ldots, E_k)$:

$$\{e_{11}, \ldots, e_{1n_1}\} \in E_1, \ldots, \{e_{k1}, \ldots, e_{kn_k}\} \in E_k,$$

sent $N = n_1 + \ldots + n_k$. A fi d'estimar les probabilitats $Q(S_i^\lambda(r))$, podem considerar dos possibles casos depenent de la distribució de $\mathbf{x}$ donada E_i:

(a) Quan la distribució de $\mathbf{x}$ donada E_i és coneguda, excepte per a alguns paràmetres

desconeguts. De ser aquest cas, podem estimar els paràmetres desconeguts de la mostra controlada i procedir com s'ha descrit anteriorment, substituint els paràmetres desconeguts per les seves estimacions.

(b) La distribució és desconeguda. En aquest cas, la forma de construir la regió $\mathscr{R}_\lambda$ no és immediata degut que les probabilitats Q, no poden ser calculades directament. Ja que per a calcular-les, s'ha de trobar les distàncies de qualsevol punt r a cada element de la població (tots anirien sortint conforme s'anessin efectuant totes les mostres $\mathbf{w}_\lambda$), però no es coneix els punts imatge de tots. No obstant, pot considerar-se una forma d'estimar les probabilitats, que permetran llavors definir una regió $\mathscr{R}_\lambda$ estimada. A continuació, mostrarem els passos per a aplicar el mètode d'avant d'aquesta situació.

Donada la mostra controlada d'individus G, hem d'agafar totes les possibles mostres de G amb mida $k\lambda$ individus. Tindrem n_i individus per a cada classe $i = 1, \ldots, k$. Amb una classe i podrem representar un total de $\binom{n_i}{\lambda}$ combinacions. Per tant, el producte cartesià de tots els nombres combinatoris, serà el nombre total de mostres possibles a efectuar. $E_1 \times \ldots \times E_k$ serà: $N_t = \prod_{i=1}^{k} \binom{n_i}{\lambda}$. Ara sí, podrem acotar l'interval de possibles valors per al paràmetre λ. Com el paràmetre λ en una mostra és constant, és a dir, hem d'agafar un nombre equitatiu d'individus per a cada classe. Com a màxim haurà de ser igual a la subpoblació amb menys individus. Formalment, podem anotar el conjunt total de mostres que poden efectuar-se com:

$$\mathbf{w}_\alpha^\lambda = \{(e_{11}^\alpha, \ldots, e_{1\lambda}^\alpha) \in E_1^\lambda, \ldots, (e_{k1}^\alpha \ldots, e_{k\lambda}^\alpha) \in E_k^\lambda\}\ \alpha = 1, \ldots, N_t \tag{4.6}$$

Per a cada individu $w \in G$ amb $\mathbf{x}(w) = r$ i per cada $\mathbf{w}_\alpha^\lambda (\alpha = 1, \ldots, N_t)$ hem de calcular les distàncies $d(\mathbf{x}(w), \mathbf{x}(e_{11}^\alpha)), \ldots, d(\mathbf{x}(w), \mathbf{x}(e_{k\lambda}^\alpha))$ per tal d'obtenir:

$$\begin{aligned}\hat{q}(s_i^\lambda(w)) = \frac{1}{N_t} \#\{\mathbf{w}_\lambda^\alpha \in E_1^\lambda \times \ldots \times E_k^\lambda \mid \exists\, \beta \in \{1, \ldots, \lambda\} \text{ amb} \\ \min\{d(\mathbf{x}(w), \mathbf{x}(e_{11}^\alpha)), \ldots, d(\mathbf{x}(w), \mathbf{x}(e_{k\lambda}^\alpha))\} = d(\mathbf{x}(w), \mathbf{x}(e_{i\beta}^\alpha))\}\ \alpha = 1, \ldots, N_t\end{aligned} \tag{4.7}$$

On # simbolitza el cardinal del conjunt. Així que $\hat{q}(s_i^\lambda(w))$ és la proporció del nombre de vegades en que el mínim de les distàncies entre w i els individus de $\mathbf{w}_\alpha$ és aconseguida per un element de E_i. La notació E_i^λ fa referència al fet que, al generar les mostres, seran un total de λ individus agafats d'una classe E_i.
Clarament $\hat{q}(s_i^\lambda(w))$ és l'estimació de $Q(s_i^\lambda(w))$. No obstant, aquesta probabilitat pot resultar computacionalment molt costosa d'estimar. S'obretot com major sigui el paràmetre λ escollit. I a la pràctica amb una mostra no necessàriament nombrosa ja tindriem problemes per a computar grans quantitats de mostres. Per exemple per a un cas amb

dues classes amb 50 i 30 individus respectivament per a E_1 i E_2, ja tindriem un total $\binom{50}{3} \cdot \binom{30}{3} = 79576000$ mostres. Per tant, la formulació de (4.7) és per a que disposem d'una idea teòrica del que volem fer, però necessitem una estimació alternativa i equivalent que soporti el que ens trobarem a la pràctica.

Necessitarem per tant, calcular les distàncies entre un individu w respecte la resta d'individus continguts en G. Aquest vector resultant de mida N, ($\mathbf{D} = d_1, \ldots, d_N$) s'ordena de menor a major ($\mathbf{D}^* = d_{(1)}, \ldots, d_{(N)}$). De fet, només caldria ordenar-lo, si estiguessim fent ús del mètode de forma manual. Però, com els càlculs seran efectuats per un ordinador, no caldrà guardar el vector ordenat $\mathbf{D}^*$. Ja que si $d_{(h)} = d(\mathbf{x}(w), \mathbf{x}(e_{ij}))$ en serà suficient amb conèixer que l'individu e_{ij} associat amb $d_{(h)}$ que prové de E_i. Així doncs, l'estimació de les probabilitats podrà efectuar-se com:

$$\hat{q}(s_i^{\lambda}(w)) = \frac{1}{N_t} \sum_{\beta=1}^{n_i} \left(\prod_{\gamma=1}^{k} \binom{a_\gamma^{(\beta)}}{\lambda - \delta_{i\gamma}} \underset{\gamma}{\text{mentre}} \ a_\gamma^{(\beta)} \geq \lambda - \delta_{i\gamma} \right) \quad i = 1, \ldots, k, \tag{4.8}$$

on $\delta_{i\gamma}$ simbolitza la delta de Kronecker, $\delta_{i\gamma} = \begin{cases} 0 & \text{if } i \neq \gamma, \\ 1 & \text{if } i = \gamma. \end{cases}$, $a_\gamma^{(\beta)}$ representa el nombre d'individus de E_γ ($\gamma = 1, \ldots, k$) situats a la dreta del β-èssim individu de E_i ($\beta = 1, \ldots, n_i$) en el vector $\mathbf{D}^*$ i $\binom{a_\gamma^{(\beta)}}{\lambda - \delta_{i\gamma}}$ el nombre combinatori usual.
El nombre total de mostres N_t és el mateix per a qualsevol i. Donat un individu w, haurem d'efectuar el càlcul per a les k subpoblacions. Per a una subpoblació donada i, el sumatori serà recorregut pel paràmetre β. El qual, anirà des del primer individu, fins a l'últim d'aquesta classe i. Per a un β donat, haurem de calcular el productori de nombres combinatoris. Que es recorrerà amb el paràmetre γ. Per a cada γ, haurem de mirar quin és el total de distàncies respecte w que són majors a la distància que es troba entre l'individu w i l'individu β. Aquesta quantitat de mostres majors, la recollirà $a_\gamma^{(\beta)}$. I se li aplicarà el nombre combinatori en funció de λ i de si està coincidint que estem calculant el nombre combinatori per a una determinada classe γ, i a la vegada, estem fent l'estimació per a una mateixa classe i.
També recordar, que la restricció que ens trobarem al calcular el nombre combinatori, ha de complir que $a_\gamma^{(\beta)} \geq \lambda - \delta_{i\gamma}$ ja que de no ser així estariem davant d'una expressió matemàticament incorrecta. A més, cal a tenir en compte, que haurà de verificar-se per a tot γ, és a dir, que donat el nombre total k d'elements que ha de recórrer γ, si existeix almenys un γ el qual no compleix la condició $a_\gamma^{(\beta)} \geq \lambda - \delta_{i\gamma}$, tot el productori valdrà zero. Encara que pugui semblar xocant, si per exemple, aprofitessim per al productori els nombres combinatoris sí calculables, llavors estaríem considerant que els no calculables valen 1, ja que l'1 és l'element neutre del producte. I per tant, s'estaria suposant que el fet de no tenir elements a la dreta, és a dir, 0 sobre 1 és 1 ($\binom{0}{1} = 1$), el qual seria

matemàticament incorrecte.

Una vegada realitzades les estimacions de les probabilitats, els elements $\hat{R}_i^\lambda$ de la partició $\hat{\mathscr{R}}_\lambda$ (estimació de $\mathscr{R}_\lambda$) inclouran tots els individus de G per als quals:

$$\hat{R}_i^\lambda = \{w \in G \mid \hat{q}(s_i^\lambda(w)) > \hat{q}(s_j^\lambda(w)),\ j = 1, \ldots, k;\ j \neq i\} \quad i = 1, \ldots, k. \tag{4.9}$$

I per tant, ja haurem aconseguit regionalitzar G. Per consegüent, gràcies a (4.9) ja és pot calcular la matriu de confusió, que és la intersecció entre les subpoblacions de G i les regions de $\hat{\mathscr{R}}_\lambda$, és a dir, $\hat{R}_i^\lambda \cap E_j\,, \forall\, i, j = 1, \ldots, k$. I podrem calcular les probabilitats a posteriori com:

$$P(E_j|\hat{R}_i^\lambda) = \frac{P(\hat{R}_i^\lambda|E_j) \cdot \pi_j}{\sum_{h=1}^k P(\hat{R}_i^\lambda|E_h) \cdot \pi_h}. \tag{4.10}$$

Un cop disposem d'elles, podrem efectuar l'assignació de les regions $\hat{R}_1^\lambda, \ldots, \hat{R}_k^\lambda$ a les $E_1, \ldots, E_k$ subpoblacions. Els individus inclosos en $\hat{R}_i^\lambda$ seran assignats a la subpoblació E_j si i només si:

$$P(E_j|\hat{R}_i^\lambda) = \text{màx}\{P(E_1|\hat{R}_i^\lambda), \ldots, P(E_k|\hat{R}_i^\lambda)\}. \tag{4.11}$$

On aquí $P(\hat{R}_i^\lambda|E_j)$ pot ser estimat com:

$$\hat{P}(\hat{R}_i^\lambda|E_h) = \frac{\#\{w \in G \mid w \in (\hat{R}_i^\lambda \cap E_h)\}}{n_h} \quad i, h = 1, \ldots, k, \tag{4.12}$$

Si l'experimentador no disposa d'informació addicional sobre la probabilitat a priori d'una determinada subpoblació h, dita probabilitat pot ser estimada com: $\hat{\pi}_h = n_h/N$. O anàlogament $\hat{\pi}_h = 1/k$ si es que es suposa equiprobabilitat entre classes.

A l'hora d'estimar $P(E_j|\hat{R}_i^\lambda)\, i, j = 1, \ldots, k$, si considerem la probabilitat a priori $\hat{\pi}_i$, $i = 1, \ldots, k$ com a $\hat{\pi}_i = \frac{n_i}{N}$. Utilitzant la regla de Bayes (4.10) i l'estimació proposada en (4.12) veiem que:

$$P(E_j|\hat{R}_i^\lambda) = \frac{\frac{\#R_i \cap E_j}{n_j}\frac{n_j}{N}}{\sum_{h=1}^k P(\hat{R}_i^\lambda|E_h)} = \frac{\frac{\#R_i \cap E_j}{n_j}\frac{n_j}{N}}{\frac{\#R_i \cap E_1}{n_1}\frac{n_1}{N} + \ldots + \frac{\#R_i \cap E_k}{n_k}\frac{n_k}{N}} =$$

$$= \frac{\frac{\#R_i \cap E_j}{N}}{\frac{\#R_i \cap E_1}{N} + \ldots + \frac{\#R_i \cap E_k}{N}} = \frac{\frac{1}{N}(\#R_i \cap E_j)}{\frac{1}{N}\left(\#R_i \cap E_1 + \ldots + \#R_i \cap E_k\right)} = \frac{\#R_i \cap E_j}{\sum_{h=1}^k R_i}$$

El qual resulta ser una forma molt atractiva d'estimar les $P(E_j|\hat{R}_i^\lambda)$, ja que només cal mirar del total d'una determinada regió $\hat{R}_i$, quants individus coincideixen en pertanyer a E_j. No obstant, també es podria tractar dites probabilitats a priori a partir de classificacions

d'estudis anteriors. Una altra possibilitat, seria un cas en el qual no necessàriament es disposes d'informació d'una mostra controlada, sinó de la població total en un fet anterior. On en la propera classificació, a partir d'una mostra, ja es podrien utilitzar les probabilitats a priori poblacionals en lloc de les mostrals. De ser aquest cas, les distribucions a priori poblacionals ens aportarien coneixement addicional al de la mostra.

4.2.3 Predicció

Fins ara hem considerat tots els passos i fonaments necessaris per a classificar els individus d'una mostra controlada G. En la terminologia del *machine learning*, el que hem fet és construir una matriu de confusió per a un seguit d'individus d'entrenament (*train data*). El que farem ara, serà classificar nous individus, és a dir, utilitzarem el classificador MDP per a donar prediccions als individus que romanen en la mostra d'avaluació (*test data*).

En l'equació (4.11) hem assignat cadascuna de les $R_1, \ldots, R_k$ que han regionalitzat G amb la vertadera divisió, les subpoblacions, $E_1, \ldots, E_k$. Per tant, aquest criteri definit, és el nostre classificador. La idea és que tots els nous individus que s'assignin a una regió R_i, seran classificats en aquella subpoblació E_j que R_i assigni, gràcies a (4.11). A vegades coincidirà que $R_i \to E_i, \ldots, R_j \to E_j$, i a vegades no. Quan coincideixin, indicarà que s'ha construït un conjunt de subparticions, que han captat un patró de característiques similars als de les classes originals.

Donada una assignació $R_i \to E_j$ $i, j = 1, \ldots, k$. Obtinguda a partir de (4.11). Considerem un nou individu $w \notin G$. Per a que w pugui classificar-se en una subpoblació, caldrà que s'efectuïn els següents passos.

1. Calcular totes les distàncies entre el nou element w i cadascun dels elements de la mostra controlada.
2. Posar les distàncies en ordre creixent.
3. Calcular les $\hat{q}(s_i^\lambda(w)), i = 1, \ldots, k$ amb la fórmula dels números combinatoris (4.8).
4. Agafar el $\hat{q}(s_i^\lambda(w))$ més gran.
5. Assignar-li a w una regió $\hat{R}_i^\lambda$.

I després, farem servir l'assignació de $\hat{R}_i^\lambda$ a E_j que ja teniem. L'element w, per tant, quedarà automàticament classificat. Com s'observa, no caldrà calcular una nova matriu de distàncies. Ja que, aquesta era necessària per a fer la classificació dels elements de G en els $\hat{R}_i^\lambda$ i després efectuar l'assignació $R_i \to E_j$. Per tant, un cop entrenat el MDP,

l'assignació $R_i \to E_j$ serà el classificador que mantindrà l'entropia de les dades d'entrenament. I l'únic que s'estarà fent, serà incloure les dades d'entrenament per a avalua'ls-hi en quina regió hi pertanyen; i finalment aplica'ls-hi l'assignació $R_i \to E_j$. Que es mantindrà intacta en tot moment.
Afegim, que tots els que van a parar a la mateixa $\hat{R}_i^{\lambda}$, automàticament, quedaran classificats a la mateixa E_j.

4.3 Consideracions

Tipus de distàncies

- **Euclidiana**: Aquesta distància és adequada per a mesurar individus amb variables numèriques.
- **Gower**: És la millor opció per al cas mixte. És a dir, per a vectors d'individus mesurats amb variables numèriques i categòriques.
- **Mahalanobis**: Tot i que està considerada per al tractament de variables numèriques, pot donar bons resultats per al cas mixte. No obstant, té l'inconvenient que utilitza la matriu inversa de covariàncies barrejada. I això fa que només sigui aplicable per a casos petits (poques variables). Ja que a la pràctica, amb centenars o milers de variables, com per exemple en bases de dades genòmiques, sempre es trobaran combinacions lineals de variables. I això farà, que la matriu esdivingui singular (no invertible), i per tant, no pugui ser aplicada.
- **Gaussian Kernel**. Es defineix com:

$$k(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{2\sigma}\right)$$

  on σ és un paràmetre a fixar per l'usuari. Com més assemblades siguin les subpoblacions més petit s'haurà de fixar. Ja que de no ser així, podria mesurar el conjunt de distàncies de forma invariant. A la pràctica estimacions del estil 1e−10 poden ser freqüents.
  Fem notar que perquè qualsevol funció kernel pugui tractar-se com a distància, cal aplicar el resultat de (3.1).
- **Laplacian kernel**. Ve definit com:

$$k(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{\sigma}\right)$$

És de la mateixa família exponencial que el Gaussià. Sent aquest més sensible a canvis en el paràmetre σ.

- **Polynomial kernel**. Es defineix com:

$$k(\mathbf{x}, \mathbf{z}) = (\alpha \cdot \langle \mathbf{x}, \mathbf{z} \rangle + c)^d$$

On d és el grau del polinomi, α és el pendent, i la $c \geq 0$ el terme constant.

Selecció de la mida mostral

El mètode MDP_λ depèn del paràmetre λ que recordem que estava fitat per a $\{1, \ldots, c\}$, sent c el nombre d'individus de la classe més petita. La milor elecció de λ serà en funció de com de separades estiguin les subpoblacions. Per a distribucions solapades, petits valors de λ donaran bones classificacions. Ja que al haver barreja entre classes, caldran mostres més petites per a avaluar amb més detall les diferències entre individus. En canvi, per a poblaciones perfectament separades, no tindrà sentit efectuar mostres amb comparacions tant petites, ja que les diferències estaran senyalades i per tant efectuant mostres de mida superior en serà suficient per a regionalitzar els individus.

Mala assignació

L'assignació de regions R_i a respectives subpoblacions E_j no funcionarà quan estiguem davant el cas en què tinguem un seguit d'individus provinents d'una subpoblació i però amb un seguit de característiques mesurades per a una altra subpoblació j. És a dir, gràficament, seran individus que romandran propers, però en canvi no provindran de la mateixa classe. Ja que si un individu w pertanyent a un grup E_i presenta un seguit de característiques d'una altra classe E_j al calcular $S_i^\lambda(r)$, (sent $\mathbf{x}(w) = r$), succeirà que $S_i^\lambda(r)$, serà a la força menor que $S_j^\lambda(r)$, ja que el fet que r romangui més a la vora dels altres punts de E_j que de E_i ja farà que $S_j^\lambda(r)$ contingui més quantitat de mostres. I per tant, l'estimació de $Q(S_j^\lambda(r))$ serà més gran que la de $Q(S_i^\lambda(r))$. Aquest fet, causarà que R_j^λ inclogui a r. I donarà lloc, a una mala classificació dels individus, és a dir, individus afegits a una regió R_h^λ sent originaris d'una subpoblació diferent de h.

Classes no balancejades

Encara que l'elecció de la distribució a priori dependrà de l'experimentador, cal destacar que és recomanable no utilitzar distribucions a priori com a freqüències mostrals $\frac{n_i}{N}$, en el cas de disposar de subpoblacions que no romanguin balancejades. Ja que de ser aquest cas, es suposa donar-li un pes addicional al conjunt de classes que continguin major evidència empírica. I probablement provocarà que les regions amb poca evidència mostral R_h siguin

assignades a subpoblacions $E_j,\ h \neq j$ amb alta evidència mostral. Ho exemplifiquem amb el cas més simple, el de dues classes.

Disposant d'una mostra amb dues subpoblacions E_1 i E_2, si es definissin les probabilitats a priori com $\hat{\pi}_1 = \frac{n_1}{N}$ i $\hat{\pi}_2 = \frac{n_2}{N}$. Sent la diferència de proporcions entre $\hat{\pi}_1$ i $\hat{\pi}_2$ no balancejada. Llavors al tractar-se d'una mostra no balancejada i a sobre donar-li una ponderació a la classe més nombrosa, faria que tots els individus de la classe petita, fossin assignats a la classe més nombrosa.
En efecte, al generar-se la matriu de confusió,

$$\begin{bmatrix} \#R_1 \cap E_1 & \#R_1 \cap E_2 \\ \#R_2 \cap E_1 & \#R_2 \cap E_2 \end{bmatrix}$$

les probabilitats a posteriori seran estimades com

$$P(E_i|R_j) = \frac{\#R_j \cap E_i}{\#R_j \cap E_i + \#R_j \cap E_j}$$

En cas que $\#R_1 \cap E_1 > \#R_1 \cap E_2$, com és d'esperar, R_1 s'assignarà a E_1. Ja que, s'estarà complint que $P(E_1|R_1) > P(E_2|R_1)$.
Assumint que E_2 ha estat correctament agrupat en les regions R_1, R_2.
I si $\#R_1 \cap E_2 < \#R_2 \cap E_2$, és a dir, es complís que s'han agrupat més individus a R_2 que no pas a R_1. Però tot i així, si $\#R_2 \cap E_1 > \#R_2 \cap E_2$ implicarà que $P(E_1|R_2) > P(E_2|R_2)$. I per tant, R_2 quedaria assignada a E_1 quan realment presentava major similitud amb E_2.

Això pot repercutir de forma general en les prediccions i consegüentment en les validacions.
Pot observar-se per a un cas com el que s'ha exemplificat. Ocorreria la següent classificació.

$$\begin{bmatrix} \#R_1 \cap E_1 & \#R_1 \cap E_2 \\ 0 & 0 \end{bmatrix}$$

La mesura observada d'avaluació de la precisió ve deifinida com:

$$\frac{\#R_1 \cap E_1 + \#R_2 \cap E_2}{N}$$

en cas que $\frac{\#R_1 \cap E_1}{N}$ fos força gran, ens atorgaria, per tant, una bona precisió en l'estimació. Sent realment una predicció incoherent, ja que efectua una assignació invariant. És per a això, que en tot moment, (al validar models) caldrà fixar-se en una altra mesura, el que es coneix com el coeficient de Kappa. Ja que aquest sí que és capaç de reconèixer o ser sensible a aquest succés. Mostrem doncs, com aquest coeficient reconeix el problema.

Cal calcular les precisions esperades esp_1 i esp_2:

$$\text{esp}_1 = \frac{(\#R_1 \cap E_1 + \#R_2 \cap E_1) \cdot (\#R_1 \cap E_1 + \#R_1 \cap E_2)}{N} = \frac{\#R_1 \cap E_1 \cdot N}{N} = \#R_1 \cap E_1$$

$$\text{esp}_2 = \frac{(\#R_2 \cap E_2 + \#R_1 \cap E_2) \cdot (\#R_2 \cap E_2 + \#R_2 \cap E_1)}{N} = \frac{(\#R_1 \cap E_2) \cdot 0}{N} = 0$$

La precisió esperada global es calcula com:

$$\text{esp} = \frac{\text{esp}_1 + \text{esp}_2}{N} = \frac{\#R_1 \cap E_1}{N} = \text{obs}$$

I el coefcient de Kappa, com es comproba, resulta ser nul:

$$K = \frac{\text{obs} - \text{esp}}{1 - \text{esp}} = 0$$

Remarquem que aquest fet és una condicció suficient a que el coeficient Kappa sigui nul, però no necesària.
El mateix efecte succeiria per a una precisió esperada superior a la observada, obtenint llavors un coeficient negatiu. És per a això que caldrà fixar-se dit coeficient en les estimacions com a estrictament positiu.

4.4 La funció `mdp` amb R

En aquest apartat exposem els conceptes necessaris per a poder cridar amb R les funcions implementades.

funció `mdp`

Depent dels següents paràmetres:

- **Class**: Aquest és el paràmetre que especifica la classe o variable dependent.
- **data**: És la matriu de dades, o el conjunt de variables predictores.
- **lambda**: Mida de la selecció mostral.
- **distance**: És la funció de distància a utilitzar. Pot escollir-se entre la distància euclidiana, la distància de Gower i la distància de Mahalanobis.
- **prior**: És la distribució a priori a assumir. Pot escollir-se `"1/k"` per a equiprobabilitat entre classes, `"ni/N"` per a utilitzar les freqüències mostrals i un vector numèric en cas que se'n volgués introduir una particular.

- **kernel**: És la funció kernel a utilitzar. Pot escollir-se entre tres funcions kernel. El kernel Gaussià, s'especifica com a `g.s=0.1`, per exemple si es vol $\sigma = 0.1$. El kernel Laplacià, s'epecifica com a `l.s=0.1`, per exemple si es vol $\sigma = 0.1$. I el kernel polinòmic, s'especifica com a `p.d=2`, per exemple si es vol amb grau $d = 2$.

- **info.pred**: Paràmetre per indicar si volem el model per a validació. En cas d'especificar TRUE ens guardarà la informació pertinent per a poder utilitzar la funció de predicció `predict.mdp`. Si simplement volem el model per a classificar uns individus, deixant el paràmetre com a *missing* ja n'és suficient.

funció `predict.mdp`

Depent dels següents paràmetres:

- **modelFit**: Fa referència a un objecte de tipus `mdp`. És a dir, és el paràmetre per a especificar el nostre model entrenat.

- **newdata**: Son les dades d'avaluació (*test data*).

4.5 Exemples

Il·lustrarem alguns exemples d'aplicació del mètode. Tots dos estan extrets de l'article del mètode [6]. I per tant, el que farem en part, serà reproduir els resultats de l'article.

4.5.1 Cetoacidosis diabética

Aquest primer exemple, tracta amb pacients amb cetoacidosi diabètica. La cetoacidosi és una de les causes més freqüent de mort entre els pacients diabètics, de manera que emprà un pronòstic a temps a l'inici d'una crisi podria ser un factor important. Es varen recollir dades durant 6 anys recopilant informació sobre 4 variables. S'aconseguí enregistrar un total de 106 pacients amb cetoacidosi ingressats a la unitat de cures intensives de l'Hospital Clínic de Barcelona. Dels 106 pacients, 93 d'ells varen sobreviure, mentre que 13 moriren. Constituiran la mostra controlada per a exposar el MDP. Duess variables eren contínues, l' *edat* i la *glucèmia*, que és el nivell de glucosa en sang. Les altres dues variables foren categòriques (dicotòmiques), el *diagnòstic prèvi de diabetis* (sí = 1 o no = 0) i l'*estat del coma* (sí = 1 o no = 0). Poden consultar-se a la (Taula 1) de l'article Angel Villarroya (1995) [6].

Considerarem dues subpoblacions: E_1, composta pels pacients que varen sobreviure i E_2, pels que moriren. Es trià la distància de Mahalanobis. S'escolleix, i a l'hora per simplicitat, el paràmetre $\lambda = 1$.

Escollim el primer individu de E_1 per a observar els passos del mètode.
El denotem com a e_1 i consta del vector de valors observats $(23, 374, 1, 1)$.
Primer, cal calcular la distància de e_1 amb tots els individus. Un cop calculada, i tenint en compte que disposem de dues classes i $\lambda = 1$, per a cada mostra, haurem de seleccionar un individu de cada grup. El nombre total de mostres a calcular és $N_t = \binom{93}{1}\binom{13}{1} = 1209$. Per a obtenir $\hat{q}(s_i^{\lambda}(w))$ tant partint de l'estimació de la probabilitat de (4.7) com de la de (4.8) n'hi haurà prou amb comptar el nombre de vegades en que la distància de e_1 amb un individu de E_1 és menor que amb un individu de E_2. El resultat obtingut és un total de 1088 mostres. I com estem en un cas amb dues subpoblacions, de fet ja no caldrà calcular $s_2^1(e_1)$ ja que serà el seu complementari $N_t - s_1^1(e_1) = 1209 - 1088 = 121$. El càlcul a emprar per mitjà de (4.8) és:

$$\widehat{q}(s_1^1(e_1)) = \frac{1}{1209}\sum_{\beta=1}^{93}\left[\binom{a_1^{(1)}}{1-1}\binom{a_2^{(1)}}{1}\right] = \frac{1}{1209}\sum_{\beta=1}^{93}\binom{a_2^{(1)}}{1} = \frac{1088}{1209} = 0.89992$$

I d'igual manera, per a E_2:

$$\widehat{q}(s_2^1(e_1)) = \frac{1}{1209}\sum_{\beta=94}^{106}\left[\binom{a_1^{(1)}}{1}\binom{a_2^{(1)}}{1-1}\right] = \frac{1}{1209}\sum_{\beta=94}^{106}\binom{a_1^{(1)}}{1} = \frac{121}{1209} = 0.10008$$

Aquest procés l'haurem hagut de realitzar per als 105 individus restants. Mostrem una taula 4.1 amb les estimacions de les $s_i^1(w)$ i les probabilitats $\hat{q}(s_i^1(w))$ per als 13 primers individus de E_1 i per als 13 individus de E_2 .

L'individu 7, per exemple, presenta una probabilitat $\hat{q}(s_1^1(w))$ molt alta, per tant, ens indica que presenta unes característiques típiques dels individus que formen E_1. A diferència de per exemple l'individu 94 de E_2 que tot i que té una probabilitat més alta a $\hat{q}(s_2^1(w))$ que a $\hat{q}(s_1^1(w))$, ho fa amb només un avantatge de 0.01. Per tant, és un cas d'un individu que estaria molt proper a la interacció entre ambdues subpoblacions.

Amb el conjunt de probabilitats, s'ha regionalitzat la mostra amb les R_i^1. La seva interacció amb E_i és el que mostra la següent soritda d'R.

```
> model = mdp(Class=dades$class,data=dades[,-1],lambda=1,
+             distance="mahalanobis")
> model$Confusion_matrix
```

```
     [,1] [,2]
[1,]   89    3
[2,]    4   10
```

On les columnes representen E_1 i E_2. I les files R_1 i R_2. Obtenim la matriu de confusió exposada en l'article del mètode. A part, aquest és el model amb una precisió més

Taula 4.1: Estimacions de les $s_i^1(w)$ i les $q(s_i^1(w))$ per a 26 individus

Individu	$s_1^1(w)$	$s_2^1(w)$	$\hat{q}(s_1^1(w))$	$\hat{q}(s_2^1(w))$
1	1088	121	0.8999173	0.10008271
2	983	226	0.8130687	0.18693135
3	893	316	0.7386270	0.26137304
4	719	490	0.5947064	0.40529363
5	752	457	0.6220017	0.37799835
6	772	437	0.6385443	0.36145575
7	1100	109	0.9098428	0.09015715
8	788	421	0.6517783	0.34822167
9	588	621	0.4863524	0.51364764
10	1057	152	0.8742763	0.12572374
11	1082	127	0.8949545	0.10504549
12	892	317	0.7377998	0.26220017
13	1043	166	0.8626964	0.13730356
94	598	611	0.4946237	0.50537634
95	596	613	0.4929694	0.50703060
96	493	716	0.4077750	0.59222498
97	594	615	0.4913151	0.50868486
98	1013	196	0.8378825	0.16211745
99	757	452	0.6261373	0.37386270
100	541	668	0.4474773	0.55252275
101	519	690	0.4292804	0.57071960
102	605	604	0.5004136	0.49958644
103	585	624	0.4838710	0.51612903
104	421	788	0.3482217	0.65177833
105	549	660	0.4540943	0.54590571
106	505	704	0.4177006	0.58229942

elevada. Respecte altres valors de λ i altres funcions de distància. En l'article es selecciona les freqüenciès mostrals com a distribucions a priori. És a dir $\hat{\pi}_1 = \frac{93}{106} = 0.8774$ i $\hat{\pi}_2 = \frac{13}{106} = 0.1226$. Aquest és el cas en que les probabilitats a posteriori per a una classe, podran estimar-se com la proporció d'individus de la regió que intersecten amb la classe. Aquestes probabilitats són: $P(E_1|\hat{R}_1^1) = \frac{89}{92} = 0.9674$ i $P(E_2|\hat{R}_2^1) = \frac{10}{14} = 0.7143$. I venen a ser màximes per a cadascuna de les regions.

```
> model$prob_post
```

```
          [,1]      [,2]
[1,] 0.9673913 0.0326087
[2,] 0.2857143 0.7142857
```

Com $P(E_1|\hat{R}_1^1)$ és el màxim per a $\hat{R}_1^1$ i $P(E_2|\hat{R}_1^1)$ és el màxim per a $\hat{R}_2^1$. El classificador, assignarà aquells nous individus que es classifiquin a $\hat{R}_1^1$ a E_1. I aquells nous individus que es classifiquin a $\hat{R}_2^1$ a E_2.

L'últim pas faltant és validar el model. Efectuarem els 4 models que s'exposaren en l'article. Per a $\lambda = \{1, 2\}$, i en funció de la distribució a priori. "1/k" o "ni/N". Finalment calculem l'error de validació com s'efectuà en l'article [6].

```
Minimum Distance Probability

106 samples
  4 predictor
  2 classes: '1', '2'

No pre-processing
Resampling: Leave-One-Out Cross-Validation
Summary of sample sizes: 105, 105, 105, 105, 105, 105, ...
Resampling results across tuning parameters:

  lambda  prior  Accuracy   Kappa
  1       1/k    0.8867925   0.39370829
  1       ni/N   0.8867925   0.39370829
  2       1/k    0.8867925   0.58404186
  2       ni/N   0.8867925   0.58404186
  3       1/k    0.8396226   0.47888953
  3       ni/N   0.8490566  -0.04820766

Tuning parameter 'distance' was held constant at a value of mahalanobis
Accuracy was used to select the optimal model using  the largest value.
The final values used for the model were lambda = 1, prior = 1/k and distance
 = mahalanobis.
```

S'observa a partir de $\lambda = 3$ un coeficient de Kappa negatiu, el que indica que s'han produït assignacions constants. Això comença a passar a mesura que s'augmenta el paràmetre λ, fet provocat per utilitzar les freqüències mostrals com a distribucions a priori, i no tenir les subpoblacions balancejades.

4.5.2 Tres espècies de Chaetocnema

Els Chaetocnema són un gènere d'escarabats. El *dataset* consta de tres espècies: Ch.concinna, Ch.heptapotamica i Ch.heikertingeri. I es disposa de 21, 31 i 22 mascles per cadascuna de les tres espècies respectivament.

Es seleccionaren tres variables per a la classificació. tars1=Amplada de la primera articulació del primer tars mesurat en micres, tars2= Amplada de la segona articulació

del primer tars en micres, aede2= l'angle frontal de la aedeagus (1 unitat = 7,5 graus). Es consideren les primeres variables com a contínues, mentre que la tercera s'estableix com a categòrica. Mostrem com són les dades i la matriu de confusió obtinguda.

```
> head(Data)
```

```
  width1 width2 angle class
1    191    131    15     1
2    185    134    13     1
3    200    137    14     1
4    173    127    16     1
5    171    118    13     1
6    160    118    15     1
```

```
> mod.ch = mdp(Class = Data$class, data = Data[,-4],
+              distance = "mahalanobis", lambda = 1)
> mod.ch$Confusion_matrix
```

```
     [,1] [,2] [,3]
[1,]   21    4    0
[2,]    0   27    0
[3,]    0    0   22
```

S'aconsegueix una bona classificació amb una taxa d'error del 5%. Considerant les probabilitats a priori com a freqüències observades, que és com s'assumí en l'article.

```
> mod.ch$prob_post
```

```
     [,1] [,2] [,3]
[1,] 0.84 0.16    0
[2,] 0.00 1.00    0
[3,] 0.00 0.00    1
```

Que és manualment: $P(\text{Ch. concin.} \mid \hat{R}_1^1) = \frac{21}{25} = 0.84$, $P(\text{Ch. concin.} \mid \hat{R}_2^1) = \frac{27}{27} = 1$ i $P(\text{Ch. concin.} \mid \hat{R}_3^1) = \frac{22}{22} = 1$. I per tant totes les dades d'avaluació incloses a $\hat{R}_1^1$, $\hat{R}_2^1$ i $\hat{R}_3^1$, seran assignades a E_1, E_2 i E_3 respectivament.
Consegüentment mostrem la validació de les dades per a $\lambda = 1, 2, 3$. Ho efectuem per mitjà de LOOCV, tal com es realitzà en l'article.

```
Minimum Distance Probability
```

```
74 samples
```

```
 3 predictor
 3 classes: '1', '2', '3'

No pre-processing
Resampling: Leave-One-Out Cross-Validation
Summary of sample sizes: 73, 73, 73, 73, 73, 73, ...
Resampling results across tuning parameters:

  lambda  prior  Accuracy   Kappa
  1       1/k    0.9459459  0.9184573
  1       ni/N   0.9459459  0.9184573
  2       1/k    0.9459459  0.9184573
  2       ni/N   0.9459459  0.9184573
  3       1/k    0.9324324  0.8977901
  3       ni/N   0.9324324  0.8977901

Tuning parameter 'distance' was held constant at a value of mahalanobis
Accuracy was used to select the optimal model using  the largest value.
The final values used for the model were lambda = 1, prior = 1/k and distance
 = mahalanobis.
```

S'aprecia un baix error de validació.

Capítol 5

APLICACIÓ DEL MDP A L'OBTENCIÓ DE BIOMARCADORS

En aquest capítol es presenta l'aplicació de l'algorisme d'aquest projecte per a un conjunt de dades òmiques. El que es busca serà l'obtenció d'un biomarcador genètic.

5.1 Descripció de la base de dades

Alon et al. (1999) [17], presentà un *dataset* amb nivells d'expressió gènica de 40 pacients amb tumor i de 22 pacients amb teixits normals del còlon. Les dades foren extretes a partir de 6500 gens humans obtinguts amb una sèrie d'oligonucleòtids d'*Affymetrix*. El *dataset* inclou l'expressió gènica dels 2000 gens amb els que s'obtingueren major intensitat mínima, a través de 62 teixits.
Per a cada mesura d'intensitat d'un gen, es derivà al voltant de 20 parells de característiques que corresponien amb el gen en el xip, mitjançant l'ús d'un procés de filtrat. Les dades que disposem no han estat prèviament normalitzades.
L'objectiu serà classificar els teixits cancerosos dels que no ho són.
Escollirem els 45 primers individus com a dades d'entrenament i els 17 últims com a individus d'avaluació.
Començarem estimant el mètode MDP per a alguns valors lambda. Després validarem el model segons el valor lambda i la funció de distància escollida. Calcularem, a més, la *kernelització* del mètode. I ho compararem amb un algorisme de classificació. El K-veïns més propers (*K-nearest neighbors*).
Aquesta primera etapa serà útil per a elaborar l'últim pas, l'obtenció d'un biomarcador.

5.1.1 Visió de la precisió en les dades

En aquest apartat, durem a terme un procés seqüencial per a veure com es comporten les expressions dels gens en funció dels teixits de colon. Ens permetrà conèixer quins són

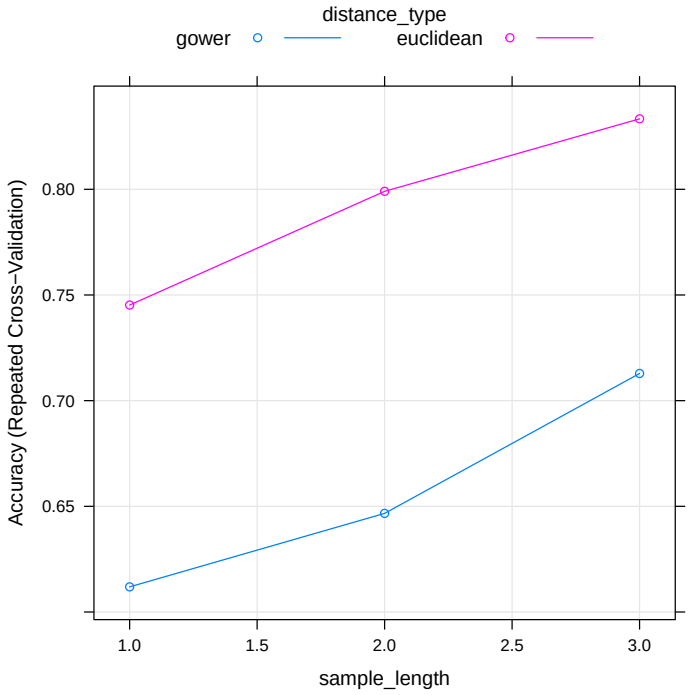
els paràmetres interessants a escollir per a després poder fer la cerca del biomarcador. Fet que requereix una computació intensiva, i una selecció paramètrica pot fer reduir un elevat cost.

Cas MDP

- **Class**: Variable reposta dicotòmica indicant "t" pacient amb tumor i "n" pacient sense.
- **data**: Els 62 pacients (files) i les 2000 expressions dels gens (columnes).
- **lambda**: Mida de selecció mostral $\lambda = 1$.
- **distance**: Distància euclidiana i distància de Gower. No treballarem amb la distància de Mahalanobis, ja que no seria capaç computar-la, degut a la singularitat de la matriu al ser de dimensió 2000.
- **prior**: Establim una distribució a priori uniforme degut al no balanceig entre les classes.
  A continuació mostrarem el mètode MDP *tunejat* per mitjà de 10-cross validation (amb repeticions de 5).

```
Minimum Distance Probability

  62 samples
2000 predictors
   2 classes: 'n', 't'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 56, 56, 56, 56, 55, 56, ...
Resampling results across tuning parameters:

  lambda  distance   Accuracy   Kappa
  1       gower      0.6119048  0.2300185
  1       euclidean  0.7452381  0.5187242
  2       gower      0.6466667  0.2620728
  2       euclidean  0.7990476  0.5339463
  3       gower      0.7128571  0.4025031
  3       euclidean  0.8333333  0.6027856
```

```
Accuracy was used to select the optimal model using  the largest value.
The final values used for the model were lambda = 3 and distance = euclidean.
```

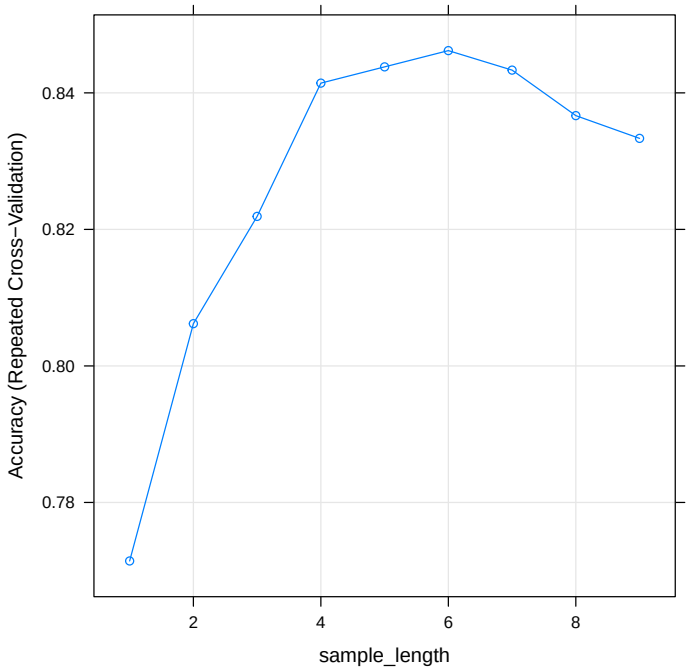


Apreciem una precisió major emprant la distància euclidiana. El paràmetre lambda (eix horitzontal) presenta una variació semblant en ambdues distàncies. Per tant, apliquem novament el MDP utlitzant únicament la distància euclidiana. Fins que la funció que mesura l'*accuracy* arribi al màxim.

La següent sortida i el següeny gràfic mostren la precisió del MDP amb distància euclidiana segons $\lambda = \{1, \ldots, 9\}$.

```
Minimum Distance Probability

  62 samples
2000 predictors
   2 classes: 'n', 't'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 55, 56, 56, 56, 56, 56, ...
Resampling results across tuning parameters:

  lambda  Accuracy   Kappa
  1       0.7714286  0.5487338
```

```
2       0.8061905  0.5431613
3       0.8219048  0.5691743
4       0.8414286  0.6293089
5       0.8438095  0.6375910
6       0.8461905  0.6453810
7       0.8433333  0.6398013
8       0.8366667  0.6280394
9       0.8333333  0.6213727

Tuning parameter 'distance' was held constant at a value of euclidean
Accuracy was used to select the optimal model using  the largest value.
The final values used for the model were lambda = 6 and distance = euclidean.
```



I s'aconsegueix una precisió màxima de 0.8462 amb $\text{MDP}_{\lambda=6}$.

Cas KMDP

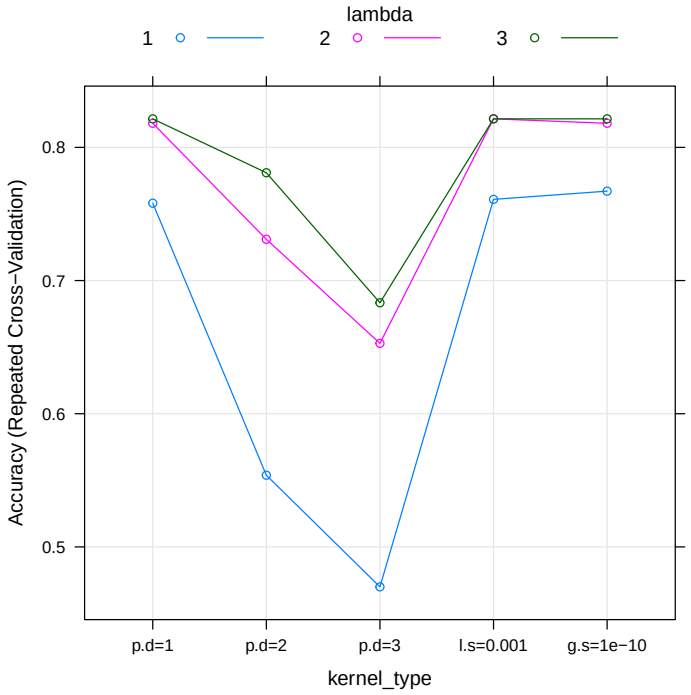
- **Class**: Variable reposta dicotòmica indicant "t" pacient amb tumor i "n" pacient sense.
- **data**: Els 62 pacients (files) i les 2000 expressions dels gens (columnes).
- **lambda**: Mida de selecció mostral $\lambda = \{1, 2, 3\}$.
- **kernel**: Presentem tres funcions kernel. La polinòmica (polynomial), la Gaussiana (gaussian) i la Laplaciana (laplacian).

- **polynomial**: Grau del poliomi $d = \{1, 2, 3\}$. Té l'etiqueta `p.d`.
- **gaussian**: Depèn d'un paràmetre σ. S'ha observat prèviament la matriu de distàncies i no ha començat a ser una mesura de similitud fins a partir de 1e−10. Té l'etiqueta `g.s`.
- **laplacian**: Depèn d'un paràmetre σ. A començat a ser una mesura correcta de similitud a partir de 0.001. Té l'etiqueta `l.s`.
- **prior**: Distribució a priori uniforme.

Mostrem doncs, els càlculs de forma numèrica i gràfica.

```
Minimum Distance Probability

  62 samples
2000 predictors
   2 classes: 'n', 't'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 56, 56, 56, 56, 55, 56, ...
Resampling results across tuning parameters:

  lambda  kernel     Accuracy   Kappa
  1       p.d=1      0.7580952  0.5380939
  1       p.d=2      0.5538095  0.2232421
  1       p.d=3      0.4700000  0.1067796
  1       l.s=0.001  0.7609524  0.5422451
  1       g.s=1e-10  0.7671429  0.5542736
  2       p.d=1      0.8180952  0.5657928
  2       p.d=2      0.7309524  0.3685927
  2       p.d=3      0.6528571  0.2268142
  2       l.s=0.001  0.8214286  0.5724595
  2       g.s=1e-10  0.8180952  0.5657928
  3       p.d=1      0.8214286  0.5590287
  3       p.d=2      0.7809524  0.4577546
  3       p.d=3      0.6833333  0.2712615
  3       l.s=0.001  0.8214286  0.5590287
  3       g.s=1e-10  0.8214286  0.5590287
```

```
Accuracy was used to select the optimal model using  the largest value.
The final values used for the model were lambda = 2 and kernel = l.s=0.001.
```

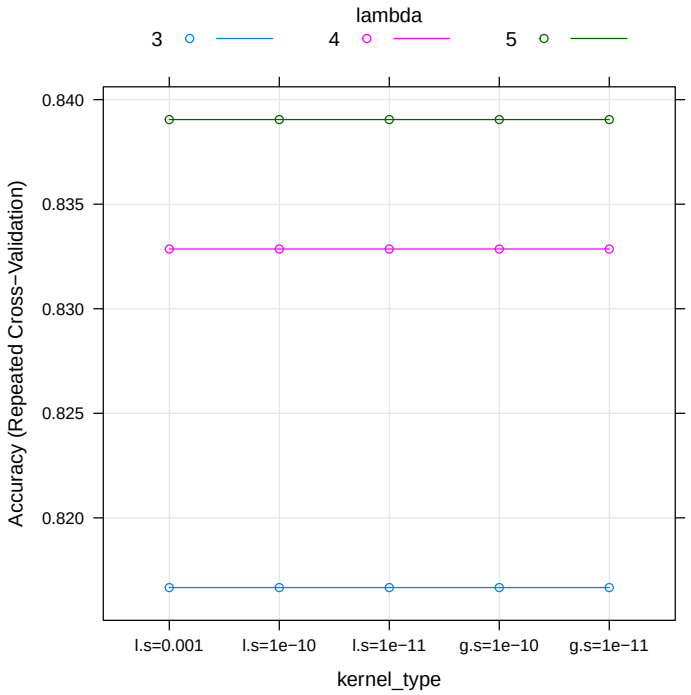


Observem l'augment en la precisió per als valors de λ. El kernel polinòmic no té un bon ajust. Per tant caldrà eludir-lo. A més, augmenta l'error al augmentar el grau del polinomi. La precisió augmenta al augmentar el valor de λ. Per tant, caldrà seguir provant valors restringint per a $\lambda = \{3, 4, 5\}$. I utilitzant el kernel Gaussià i Laplacià. Provem també els paràmetres $\sigma = \{\text{e}{-10}, \text{e}{-11}\}$ per a la funció kernel Laplaciana.

```
Minimum Distance Probability

  62 samples
2000 predictors
   2 classes: 'n', 't'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 55, 56, 56, 56, 56, 55, ...
Resampling results across tuning parameters:

  lambda  kernel     Accuracy   Kappa
  3       l.s=0.001  0.8166667  0.5552239
  3       l.s=1e-10  0.8166667  0.5552239
  3       l.s=1e-11  0.8166667  0.5552239
```

```
3       g.s=1e-10  0.8166667  0.5552239
3       g.s=1e-11  0.8166667  0.5552239
4       l.s=0.001  0.8328571  0.6077191
4       l.s=1e-10  0.8328571  0.6077191
4       l.s=1e-11  0.8328571  0.6077191
4       g.s=1e-10  0.8328571  0.6077191
4       g.s=1e-11  0.8328571  0.6077191
5       l.s=0.001  0.8390476  0.6219941
5       l.s=1e-10  0.8390476  0.6219941
5       l.s=1e-11  0.8390476  0.6219941
5       g.s=1e-10  0.8390476  0.6219941
5       g.s=1e-11  0.8390476  0.6219941

Accuracy was used to select the optimal model using  the largest value.
The final values used for the model were lambda = 5 and kernel = l.s=0.001.
```

Gràficament:



Aconseguim una millora constant depenent de la funció kernel emprada. On l'error de validació disminueix al augmentar el paràmetre lambda. Els diferents valors per a σ no presenten diferències. Podrem doncs, fixar-ne un. La precisió tendeix per igual tant considerant una funció Gaussiana com Laplaciana. Provarem valors superiors a $\lambda = 6$ per a conèixer en quin valor tendeix la funció de l'error.

```
> entreno.kmd12
```

```
Minimum Distance Probability

  62 samples
2000 predictors
   2 classes: 'n', 't'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 56, 56, 56, 56, 56, 55, ...
Resampling results across tuning parameters:

  lambda  kernel     Accuracy   Kappa
  6       l.s=1e-10  0.8376190  0.6381666
  6       g.s=1e-10  0.8376190  0.6381666
  7       l.s=1e-10  0.8419048  0.6506044
  7       g.s=1e-10  0.8419048  0.6506044
  8       l.s=1e-10  0.8323810  0.6368217
  8       g.s=1e-10  0.8323810  0.6368217

Accuracy was used to select the optimal model using  the largest value.
The final values used for the model were lambda = 7 and kernel = l.s=1e-10.
```

La precisó deixa d'augmentar a partir de $\lambda = 8$. Per tant l'òptim obtingut és per a $\lambda = 7$.

Efectuem doncs el mateix procés per al mètode Knn. Com depèn d'un paràmetre també anomenat λ, realitzarem una exploració per als 8 primers valors, $\lambda = \{1, \ldots, 8\}$.

```
> knnFit <- train(x= alon$x[1:45,], y = alon$y[1:45], method = "knn",
+                 trControl = fitControl, tuneGrid=expand.grid(.k=1:8))

k-Nearest Neighbors

  62 samples
2000 predictors
   2 classes: 'n', 't'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 56, 56, 55, 56, 56, 56, ...
Resampling results across tuning parameters:
```

```
k  Accuracy   Kappa
1  0.7914286  0.5186510
2  0.8242857  0.5980618
3  0.8428571  0.6207152
4  0.7871429  0.4896019
5  0.8290476  0.5760107
6  0.8166667  0.5503426
7  0.7752381  0.4271918
8  0.7757143  0.4269800

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was k = 3.
```

La millor estimació del Knn és per a $\lambda = 3$, amb una taxa de 0.8423 superior a la taxa 0.8419 del KMDP per a $\lambda = 7$. Però inferior al 0.8462 del $\text{MDP}_{\lambda=6}$.

5.1.2 Cerca d'un biomarcador

El que farem a continuació, serà l'efectuació d'un remostreig per a l'obtenció d'un biomarcador. En aquest treball, un biomarcador, de form pràctica i simplificada, serà un conjunt de gens (variables biològiques) amb la capacitat de classificar individus en funció d'una variable resposta, en aquest cas, si es pateix o no càncer.

Gràcies a la informació prèvia analitzada, no haurem d'estimar un gran nombre de varietats de models, ja que recentment, ha pogut apreciar-se, quins són els paràmetres més fluïts per a modelar les dades d'entrenament.

Efectuarem el $\text{MDP}_{\lambda=6}$, ja que el paràmetre òptim ha estat $\lambda = 6$. I el mètode MDP *kernelitzat*, el fixarem a $\text{KMDP}_{\lambda=7}$ ja que ha estat el paràmetre λ òptim. Establim una funció kernel Gaussiana amb $\sigma = 1\text{e}{-10}$, però de fet, seria indiferent l'ús d'un kernel Laplacià amb $\sigma = 1\text{e}{-10}$, que com ja hem vist, proporcionava uns resultats similars.
Haurem d'escollir dos valors arbitraris: n i x.
La n serà el conjunt de mostres que efectuarem. I la x, la mida o nombre de variables que contindrà cadascuna de les n mostres. El nombre total de mostres serà de $\binom{n}{x}$. S'utilitzarà una validació creuada (*cross-validation*) amb $k = 10$.

Afegir que aquest procediment no serà massa confortable degut a l'alt cost computacional que suposa. El procés serà efectuat per mitjà d'un processador CORE i5, que s'estima una estància d'execució al voltant de 3h.

Bucle de selecció MDP

- **n**: Total de mostres, n=100.
- **x**: Mida de les mostres, x=5.
- **Class**: Variable reposta dicotòmica indicant "t" pacient amb tumor i "n" pacient sense.
- **data**: 45 pacients (files) i les 2000 expressions gèniques (columnes).
- **lambda**: Mida de selecció mostral $\lambda = \{3, 4, 5, 6, 7\}$.
- **distance**: Distància euclidiana. Aportava milllors resultats que la de Gower.
- **prior**: `"1/k"` Distribució a priori uniforme.

Amb aquets paràmetres per al mètode MDP, guardarem seqüencialment cadascuna de les 100 estimacions. I finalment escollirem aquella que maximitzi la precisió. Seguidament ho exemplifiquem per a donar una idea més aclaridora.
Donada una mostra de 5 gens, s'estimaran 5 models MDP, ja que, hem definit que es provi per a 5 valors λ, des de 3 fins a 7. Per a cadascuna d'aquestes 5 estimacions, s'escollirà el model que minimitzi la taxa d'error. És a dir, partint d'un conjunt determinat de variables, s'elegeixen els hiperparàmetres que proporcionen millor ajust. Aquest procés, serà realitzat per a 100 conjunts de variables diferents. Llavors, disposarem dels 100 millors models per a la centena de conjunts de variables. El que maximitzi la taxa de precisió serà el biomarcador.
Mostrem la sortida amb la informació del model òptim. Recull els paràmetres i les variables escollides.

```
> MDP.n100.l1

$i
[1] 80

$vb
[1]  493  343 1985  943  873

$maxim
[1] 0.9047619

$lambda
[1] 5 6 7
```

```
$distance
[1] euclidean euclidean euclidean
Levels: euclidean
```

El millor model ha estat seleccionat en la 80a mostra. On l'òptim s'ha aconseguit per a $\lambda = \{5, 6, 7\}$. Podem imprimir la informació de la precisió obtinguda.

```
> MDP.cv10.n100[[80]]

Minimum Distance Probability

62 samples
 5 predictor
 2 classes: 'n', 't'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 56, 56, 56, 56, 55, 55, ...
Resampling results across tuning parameters:

  lambda  Accuracy   Kappa
  3       0.8880952  0.7570414
  4       0.8880952  0.7570414
  5       0.9047619  0.7998986
  6       0.9047619  0.7998986
  7       0.9047619  0.7998986

Tuning parameter 'distance' was held constant at a value of euclidean
Accuracy was used to select the optimal model using  the largest value.
The final values used for the model were lambda = 5 and distance = euclidean.
```

De la mateixa manera, procedim a realitzar els mateixos passos per al KMDP.

Bucle de selecció KMDP

- **n**: 100 mostres.
- **x**: Conjunts de 5 variables.
- **Class**: Variable reposta dicotòmica indicant "t" pacient amb tumor i "n" pacient sense.

- **data**: 45 pacients (files) i les 2000 expressions gèniques (columnes).
- **lambda**: Mida de selecció mostral $\lambda = 7$.
- **gaussian**: Depèn del paràmetre σ. Fixat a 1e−10.
- **prior**: "1/k" Distribució a priori uniforme.

Provarem el mateix conjunt de mostres generat per al MDP. Per a poder apreciar els resultats d'una forma més homogènia. A continuació, exposem la informació obtinguda per a el millor model.

```
> KMDP.95.100.1k1

$i
[1] 80

$vb
[1]  493  343 1985  943  873

$maxim
[1] 0.955

$lambda
[1] 7

$kernel
[1] g.s=1e-10
Levels: g.s=1e-10
```

I la respectiva informació referent a la precisió.

```
> KMDP.95.100.1[[80]]

Minimum Distance Probability

45 samples
 5 predictor
 2 classes: 'n', 't'

No pre-processing
Resampling: Cross-Validated (10 fold)
```

```
Summary of sample sizes: 40, 41, 41, 40, 41, 41, ...
Resampling results:

  Accuracy  Kappa
  0.955     0.9115385

Tuning parameter 'lambda' was held constant at a value of 7
Tuning
 parameter 'kernel' was held constant at a value of g.s=1e-10
```

Ambdós mètodes han seleccionat el mateix conjunt de variables. La precisió ha estat força semblant tot i que una mica més elevada per al KMDP. A continuació, validarem novament els dos mètodes, per mitjà de validació amb $k = 10$ i amb 10 repeticions. Restringim el MDP als valors λ òptims $\{5, 6, 7\}$. I el KMDP amb $\lambda = 7$.

```
> BoMDP

Minimum Distance Probability

45 samples
 5 predictor
 2 classes: 'n', 't'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 40, 41, 41, 41, 40, 41, ...
Resampling results across tuning parameters:

  lambda  Accuracy  Kappa
  5       0.9770    0.9546154
  6       0.9750    0.9507692
  7       0.9545    0.9103846

Tuning parameter 'distance' was held constant at a value of euclidean
Accuracy was used to select the optimal model using  the largest value.
The final values used for the model were lambda = 5 and distance = euclidean.
```

```
> BoKMDP

Minimum Distance Probability
```

```
45 samples
 5 predictor
 2 classes: 'n', 't'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 41, 40, 40, 41, 40, 41, ...
Resampling results:

  Accuracy  Kappa
  0.9535    0.9106264

Tuning parameter 'lambda' was held constant at a value of 7
Tuning
 parameter 'kernel' was held constant at a value of g.s=1e-10
```

S'aprecia en aquestes últimes validacions una lleugera millora amb el $\text{MDP}_{\lambda=5}$ i el $\text{MDP}_{\lambda=6}$, respecte al $\text{KMDP}_{\lambda=7}$.

Seguidament, recalculem models amb subconjunts del biomarcador escollit per a observar en quant baixa la precisió. Agrupats amb triplets, tindrem un total de $\binom{5}{3} = 10$ models a validar. I d'igual forma amb combinacions en parelles, tindrem un total de $\binom{5}{2} = 10$ models a validar. Apliquem novament validació creuada $k = 10$ i amb 10 repeticions.
Definim les combinacions de tres.

```
> mo1=rbind(c(493,343,1985),c(493,343,943),c(493,343,873),
+           c(343,1985,943),c(873,1985,493),c(343,943,873),
+           c(1985,943,873),c(873,943,493),c(873,493,1985),
+           c(873,1985,343))
```

Mostrem les tres combinacions de 3 amb major precisió.

```
> l.n3[[1]]

Minimum Distance Probability

45 samples
 3 predictor
 2 classes: 'n', 't'

No pre-processing
```

```
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 40, 41, 40, 40, 41, 41, ...
Resampling results:

  Accuracy  Kappa
  0.971     0.9430769

Tuning parameter 'lambda' was held constant at a value of 6
Tuning
 parameter 'distance' was held constant at a value of euclidean
```

> l.n3[[2]]

```
Minimum Distance Probability

45 samples
 3 predictor
 2 classes: 'n', 't'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 40, 40, 41, 41, 40, 40, ...
Resampling results:

  Accuracy  Kappa
  0.9475    0.8973077

Tuning parameter 'lambda' was held constant at a value of 6
Tuning
 parameter 'distance' was held constant at a value of euclidean
```

> l.n3[[3]]

```
Minimum Distance Probability

45 samples
 3 predictor
 2 classes: 'n', 't'

No pre-processing
```

```
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 41, 40, 41, 40, 41, 41, ...
Resampling results:

  Accuracy  Kappa
  0.9795    0.9596154

Tuning parameter 'lambda' was held constant at a value of 6
Tuning
 parameter 'distance' was held constant at a value of euclidean
```

La combinació de tres variables òptima és la formada pels gens: $\{343, 493, 873\}$. I s'obté una precisió lleugerament superior a l'obtinguda amb el conjunt original format pels 5 gens $\{343, 493, 873, 943, 1985\}$.
D'igual forma ho fem per a les combinacions de dos.

```
> mo2=rbind(c(493,343),c(493,1985),c(493,943),c(493,873),c(1985,943),
+           c(1985,873),c(943,873),c(873,343),c(343,943),c(343,1985))
```

I mostrem les precisions màximes per a aquestes combinacions de dos.

```
> l.n2[[1]]

Minimum Distance Probability

45 samples
 2 predictor
 2 classes: 'n', 't'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 41, 40, 41, 41, 40, 41, ...
Resampling results:

  Accuracy  Kappa
  0.965     0.9308392

Tuning parameter 'lambda' was held constant at a value of 6
Tuning
 parameter 'distance' was held constant at a value of euclidean

> l.n2[[2]]
```

```
Minimum Distance Probability

45 samples
 2 predictor
 2 classes: 'n', 't'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 41, 41, 41, 40, 40, 41, ...
Resampling results:

  Accuracy  Kappa
  0.889     0.77899

Tuning parameter 'lambda' was held constant at a value of 6
Tuning
 parameter 'distance' was held constant at a value of euclidean
```

> l.n2[[3]]

```
Minimum Distance Probability

45 samples
 2 predictor
 2 classes: 'n', 't'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 41, 41, 40, 40, 40, 41, ...
Resampling results:

  Accuracy  Kappa
  0.879     0.745039

Tuning parameter 'lambda' was held constant at a value of 6
Tuning
 parameter 'distance' was held constant at a value of euclidean
```

Les mostres formades per combinacions de dos en dos presenten taxes d'error superiors. Per tant, el conjunt de gens $\{343, 493, 873\}$ és el que proporciona una millor precisió en la classificació, i pot ser una alternativa simplificada del biomarcador seleccionat.

Finalment estimem les prediccions per a les dades de *test*. On també considerem el model entrenat amb les 2000 expresions gèniques. Per a tenir una referència del grau de precisió en la classificació.
Model entrenat amb el biomarcador:

```
> a1=mdp(Class = alon$y[1:45], data = alon$x[1:45,c(493,343,1985,873)],
+        lambda = 5, distance = "euclidean",info.pred = TRUE)
> CM=table(predict.mdp(a1,alon$x[46:62,]),alon$y[46:62])
> CM

   n t
n 8 3
t 1 5
```

Model entrenat amb totes les variables:

```
> a=mdp(Class = alon$y[1:45], data = alon$x[1:45,],
+        lambda = 5, distance = "euclidean",info.pred = TRUE)
> CM.full=table(predict.mdp(a,alon$x[46:62,]),alon$y[46:62])
> CM.full

   n t
n 5 1
t 2 9
```

Que ve a ser una precisió del 0.76 respecte a una del 0.82 que és l'obtinguda amb els 2000 gens.

5.1.3 Resultats

S'ha pogut aplicar amb èxit, el MDP i la seva respectiva *kernelització*, el KMDP. On s'han efectuat un seguit de passos prèvis per conèixer les millors condicions paramètriques dels models. En la fase inicial, $\text{MDP}_{\lambda=6}$ ha estat el model que ha aportat millors resultats tot i que amb precisió força similar al $\text{KMDP}_{\lambda=7}$. Gràcies a aquest pas, s'ha pogut conèixer la millora de les precisions per a valors λ superiors, fet que ha permès discernir les opcions a parametritzar en la part de selecció del biomarcador.
En la última part, ha estat exposat un mètode iteratiu per a la selecció de variables genètiques predictores, on s'ha aconseguit localitzar un possible biomarcador, format pel conjunt de gens $\{343, 493, 873, 943, 1985\}$. A més, s'ha assajat una validació més profunda per mijà de 10 repeticions 10-*cross validation* on ha pogut visualitzar-se una opció reduïda del biomarcador, el conjunt $\{343, 493, 873\}$.
Tant el biomarcador complert com el reduït han efectuat una classificació d'un individu

més erroni d'entre els 17 individus de la mostra d'avaluació, respecte a la predicció del model entrenat amb el conjunt total de variables. Que representa ser una variació de l'error del 6%.

Capítol 6

IMPLEMENTACIÓ I DIVULGACIÓ DEL KERNEL MDP

En aquest capítol es produeix la divulgació i implementació del mètode MDP, a través d'un paquet d'R i d'un seguit d'aplicacions web interactives.

6.1 Creació del paquet MDP

Requisits

Treballarem amb el software RStudio. És preferible utilitzar aquest software, ja que disposa d'una plataforma molt orientada a la creació de paquets, treball amb projectes, aplicacions amb Shiny etc. RStudio disposa d'una interfície gràfica especialment enfocada a la compilació de codi per a treballar amb paquets. Per a la creació, necessitarem el paquet devtools. També pot ser útil utilitzar el paquet roxygen2 per a la respectiva generació de la documentació del paquet. En la següent comanda mostrem la instal·lació d'aquets paquets. En cas que ja els tinguéssim instalůlats, només es carregaran.

```
> if("devtools" %in% rownames(installed.packages()) == FALSE){
+   install.packages("devtools")
+ } else {
+     library(devtools)
+   }
> if("roxygen2" %in% rownames(installed.packages()) == FALSE){
+   install.packages("roxygen2")
+ } else {
+     library(roxygen2)
+   }
>
```

Definició de la ubicació del paquet

És important definir el nostre directori de treball. En aquest cas el definirem en la biblioteca Documents. Ho fem creant una simple carpeta, que anomenem **paquet**. En aquesta carpeta, arxivarem el paquet amb els seus corresponents arxius. Per tant, aquesta carpeta serà el directori de treball d'R.

```
> setwd("C:/Users/Oriol/Documents/paquet")
```

Ara a R tenim com a directori la carpeta **paquet**. Cal recordar que de moment estem treballant en una sessió d'R on hem definit un directori de treball. Després ens anirem en una altra sessió o entorn de projecte on treballarem els aspectes del paquet.

Creació del paquet

Al nostre paquet l'anomenarem MDP. Per a crear-lo, es pot fer de diverses maneres, no obstant ho farem per mitjà de la funció `create` del package devtools.

```
> create("MDP")
```

Al aplicar la sentència anterior, ja disposem d'un paquet buit. Aquest paquet anomenat MDP, romandrà dins la carpeta **paquet**. Podem observar-ho en la pestanya "Files" de RStudio, que indica els arxius que tenim en el directori.

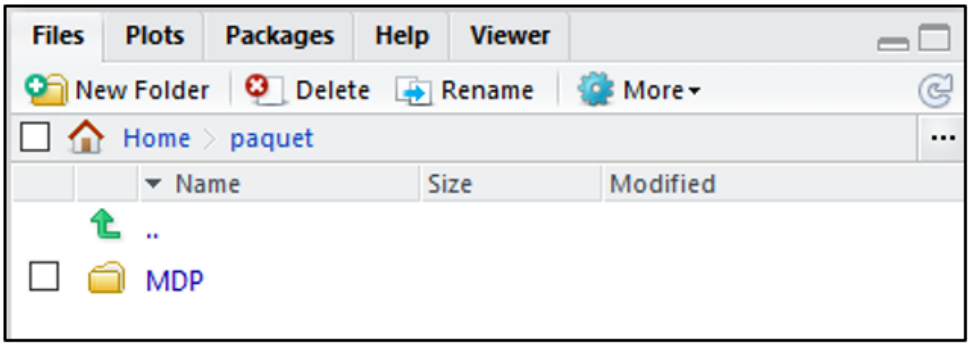


Figura 6.1: Sessió RStudio amb el paquet buit creat

Si entrem en el paquet veurem: uns fitxer anomenats DESCRIPTION i NAMESPACE que comentarem més endavant i una carpeta anomenada R. Dins d'aquesta carpeta R, es situarà un script d'extensió .R que incorporarà les funcions del paquet. Podem ja obrir un nou script en blanc mateix i desar-lo dins d'aquesta carpeta R. En el nostre cas, anomenem a aquest script: `mdp`.R. És a dir, estarà dins la ruta "`C:/Users/Oriol/Documents/paquet/MDP/R/mdp.R`".

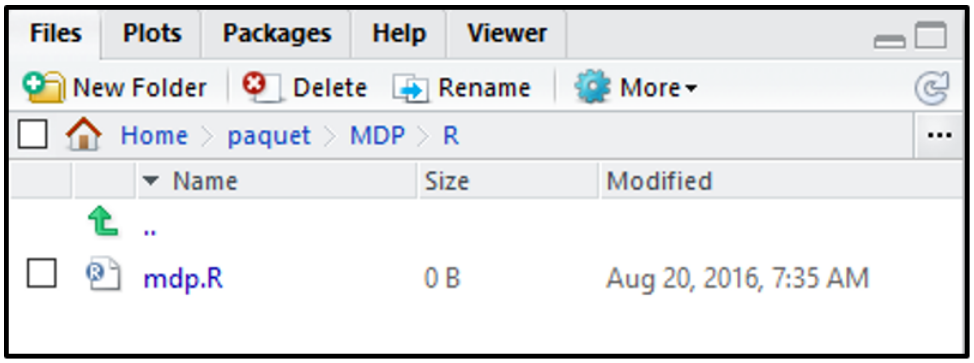


Figura 6.2: Sessió RStudio amb la funció `mdp` creada

Sessió de treball amb el paquet

Ara ja podem obrir una sessió per anar construint internament el paquet. Això es fa per mitjà d'una nova sessió que incorpora RStudio, que l'anomena Projecte on es permet construir les funcions que el paquet incorporarà entre d'altres aspectes. Per tant, ara ens desplaçarem al Projecte del nostre paquet i ja treballarem d'ara endavant en aquesta nova sessió. Per accedir-hi, ho podem fer de dues maneres:

1. Observant a l'extrem superior dret la pestanya de la plataforma d'RStudio:

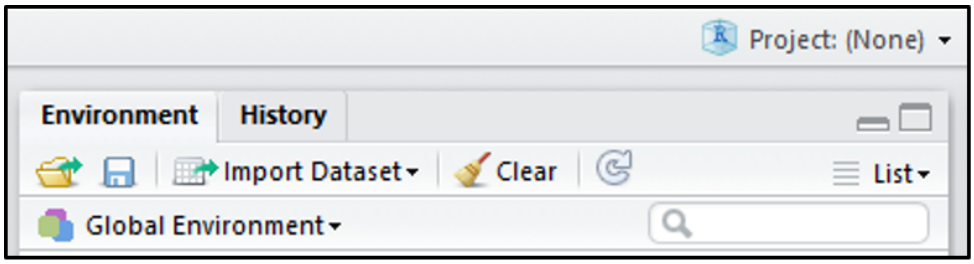


Figura 6.3: Project d'R encara no assignat

Ens indica Project: (None). Si li fem un clic ens apareixeran les següents opcions:

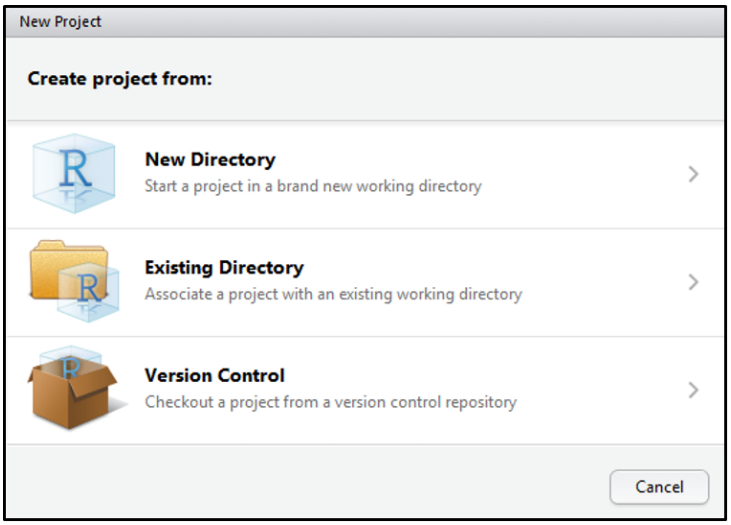


Figura 6.4: Pas intermedi 1

Com nosaltres ja hem definit el nostre directori de treball, marquem la opció "Existing Directory". D'altra banda marcaríem la primera opció "New Directory". Un cop marcat create *project>Existing Directory*, ens apareixerà:

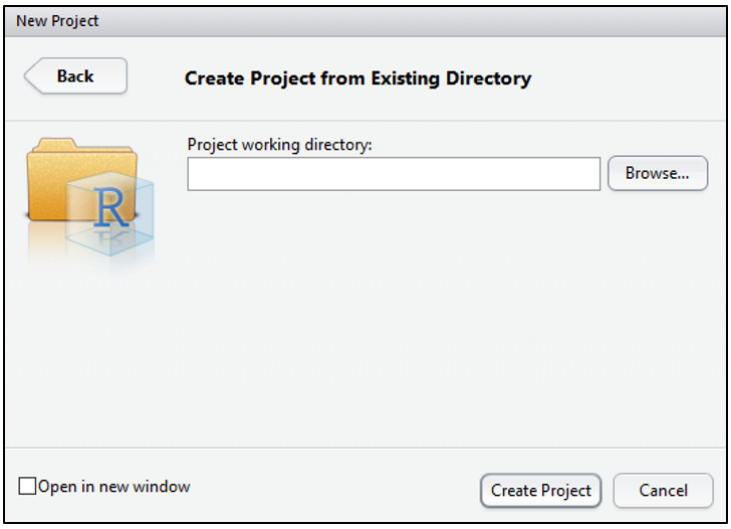


Figura 6.5: Pas intermedi 2

Aquí li haurem d'especificar la ruta on roman el nostre paquet. És a dir: "`C:/Users/Oriol/Documents/pac/funcio`".
Al pressionar "Create Project" se'ns carrega la nova sessió desitjada. En aquesta sessió, treballarem per a anar modificant el paquet: ficant-li descripcions, implementar les funcions etc. Podem observar a l'extrem dret superior com indica que estem en una sessió del paquet MDP.

2. L'altra alternativa molt més ràpida, consisteix en d'obrir la nostra sessió de treball clicant el projecte del paquet creat, que té extensió `.Rproj`. És a dir clicar a MDP`.Rproj` com es mostra en la figura de sota.

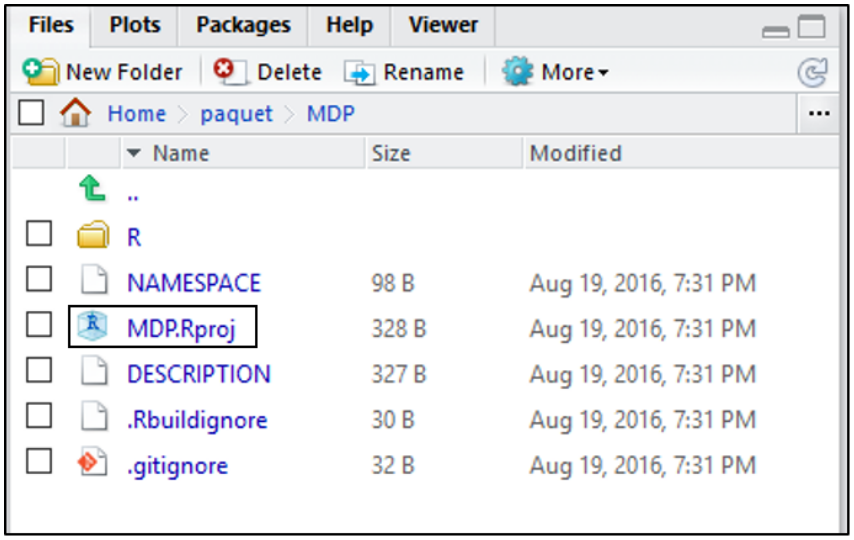


Figura 6.6: Selecció del nostre Projecte

Tant en un manera com en l'altra el resultat és traslladar-se ja a la sessió Projecte del nostre paquet. Seguidament, podem modificar la configuració per mitjà del paquet roxigen2. Anant a la pestanya *Built>Configure Build Tools*.

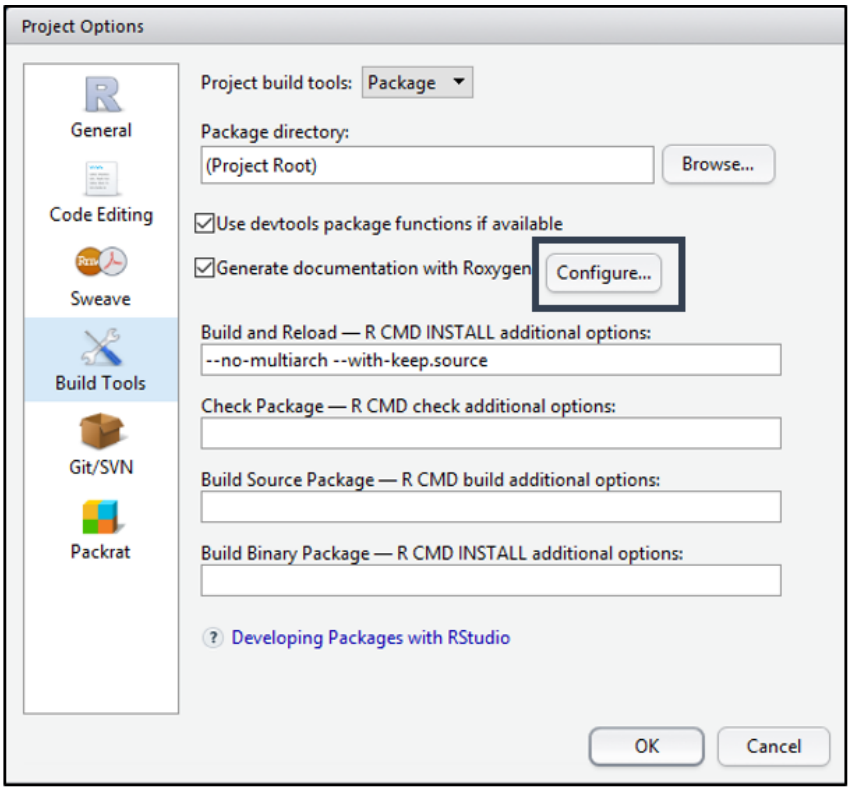


Figura 6.7: Selecció de les opcions de roxigen2

En la opció "Configure", podem incloure totes les opcions que roxigen2 ens permet per a amenitzar la forma de documentar el paquet. *Built>Configure Build Tools*.

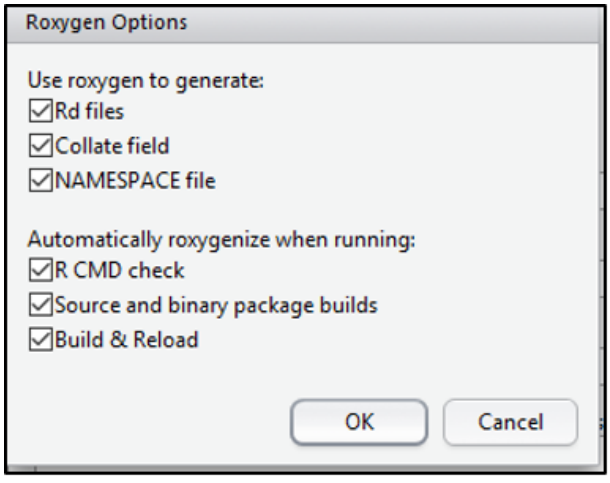


Figura 6.8: Opcions de roxigen2

Dins la carpeta MDP, allà romandran tots els arxius pertinents al paquet. En un fitxer anomenat DESCRIPTION hi contindrà la descripció del paquet. Aquest arxiu es crea per defecte quan hem creat el paquet. Si per qualsevol motiu l'haguéssim eliminat el podem tornar a crear per mitjà de la comanda:

```
> load_all()
```

Si li fem clic obtindrem la descripció que tenim del paquet per defecte:

```
Package: MDP
Title: Minimum Distance Probability
Version: 0.0.0.9000
Authors@R: person("First", "Last", email = "first.last@example.com", role =
    c("aut", "cre"))
Description: What the package does (one paragraph).
Depends: R (>= 3.1.3)
License: What license is it under?
LazyData: true
```

I podem adaptar-ho sengos la nostra conveniència:

```
Package: MDP
Title: Minimum Distance Probability
Version: 0.1
Author: Oriol Prat <oprat@outlook.com>
Maintainer: Oriol Prat <oprat@outlook.com>
Description: Classification algorithm.
Depends: R (>= 3.1.3)
License: UB
LazyData: true
```

Posteriorment executem la opció "BuildReload". Llavors haurem construït el paquet. Ara ja sí el podrem observar en la nostra llista de paquets.

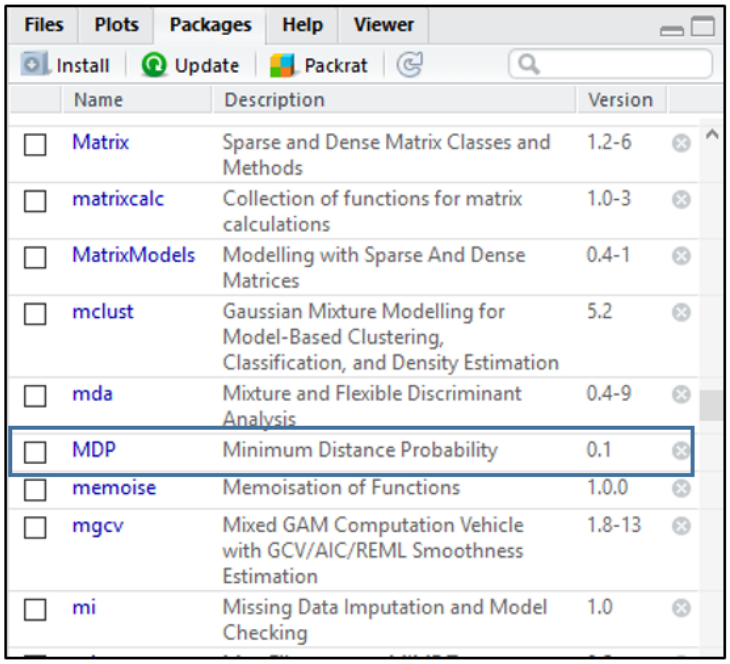


Figura 6.9: Llista de paquets instal·lats

Creació de funcions

Les funcions es creen dins d'un script `.R` hostejat en:
"`C:/Users/Oriol/Documents/paque/MDP/R`".
Dins l'script s'escriu el codi de la funció i la seva definició. Seguidament, implementarem les funcions que acompanyaran al paquet. N'hem de crear dues: `mdp` i `predict.mdp`. Les definim en l'script `mdp.R`. Podem fer-ho de dues maneres. Una forma és identificant cada sentència per mitjà del símbol `"#'"`. El primer comentari codificat amb #′ es correspon al títol de la funció en la descripció. El segon es deixa en blanc, i el tercer comentari és la descripció de la funció. El símbol `@` crida alguna acció que fa referència a la definició de la funció. Per exemple, `@param` és una forma de parametritzar el paràmetre d'entrada per a la funció. I `@return` s'especifica per a mostrar la classe de l'objecte que la funció retorna.
Però trobem més convenient per al nostre cas, treballar per a definir el caràcter de la funció per mitjà dels arxius amb extensió "`.Rd`" (*Rd documentation file*). A l'hora de grabar les funcions s'ha de considerar dos tipus d'arxius:

- Arxius `.Rd`. Són els arxius que contindran tota la informació de les funcions que incorpori el paquet. Bàsicament la informació que tots trobem al `help` quan volem conèixer la correcta sintàxis d'una funció. Aquest conjunt d'arxius d'extensió `.Rd` seran emmagatzemats dins de la carpeta man del directori de treball MDP.

- Arxius `.R`. Són aquells que incorporaran el codi de les nostres funcions. Pot escollir-se tant crear guions `.R` per a cada funció que tinguem o ficar-les juntes en un

mateix *script* d'extensió `.R`. És aconsellable introduir en un mateix guió aquelles que radiquen per a la mateixa utilitat. Com per exemple, per al nostre cas (la funció d'entrenament `mdp` i la de predicció `predict.mdp`) aniran en l'interior d'un mateix guió `.R`. I col·locarem per a guions separats, aquelles que siguin per a utilitats diferents. Aquest conjunt de guions `.R` es desaran dins la carpeta R del directori de treball MDP.

I finalment cal anar al arxiu NAMESPACE ubicat al directori MDP. Aquest arxiu, ha d'incloure la comanda d'exportació de les funcions que incorpori el paquet. Amb la següent sintaxi:

```
export(mdp)
export(predict.mdp)
```

Un cop s'ha acabat de definir i implementar les funcions del paquet, passem a compilar l'script per mitjà del botó BuildReload. Si no s'ha comès cap error en la sintaxi, la consola de la pestanya Build ens indicarà una lectura exitosa de les funcions en el paquet.

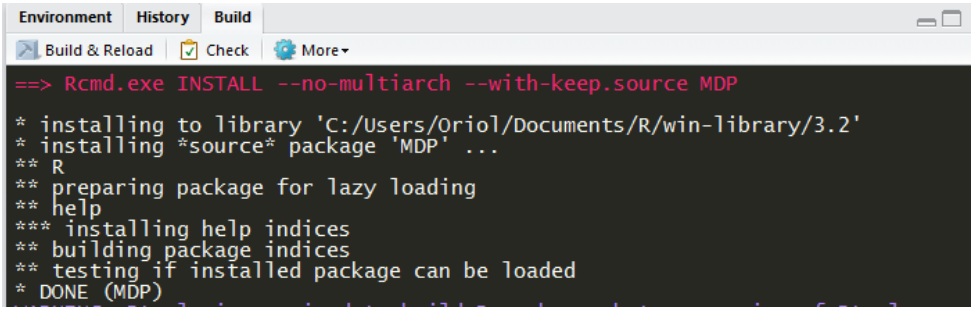


Figura 6.10: Sortida de la consola per al paquet creat

Altres consideracions

Cridant les funcions des del Help o per mitjà del símbol ? en la consola, podrem accedir a la descripció de cadascuna d'elles.

D'igual forma, fent clic en el nostre paquet MDP en l'apartat on estan totes les llibreries. Trobarem un enllaç anomenat Description file. Clicant-lo obtindrem la descripció que havíem definit del paquet.

La opció **Built**, per mitjà del CMD llegeix la sintaxi de la construcció o actualització del paquet i ens informa si la lectura ha estat exitosa, és a dir, mostra els possibles errors sintàctics que s'hagin pogut cometre. D'altra banda, la opció **Chek**, comprova automàticament el codi per a problemes comuns. És essencial de cara a seguir unes pautes homogènies en cas de pujar el paquet a CRAN. Podem verificar el nostre paquet, clicant en Check.

La primera verificació almenys ens indicarà un error en la creació d'un fitxer amb format LaTeX. Per a corregir-ho, haurem d'especificar en Build>Configure Global Options: Que no ens verifica la creació d'un manual. Li especifiquem afegint-hi "–no-manual".

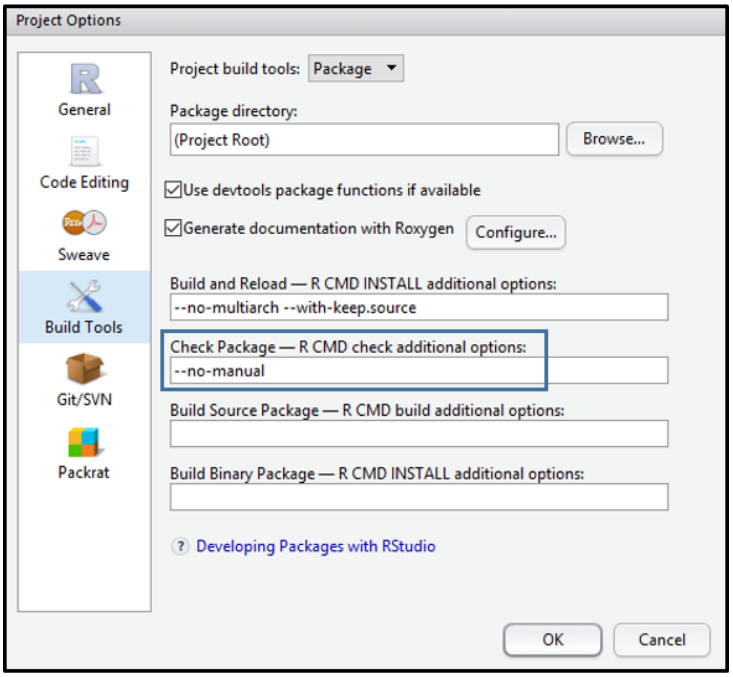


Figura 6.11: Especificació per a que accepti que no disposem de manual per al nostre paquet

Si hi ha algun factor faltant, ens avisarà per la consola. Llavors haurem de revisar les especificacions que ens digui a modificar. Una vegada verifiqui tot el paquet exitosament ja estarà a punt per a compartir-lo.

Exportació del paquet

Per a poder guardar el paquet amb una format llegible posteriorment. El podem exportar amb format `source` i format `bin`. En format `source` ens crearà una còpia del paquet amb extensió .tar.gz , mentre que el format `bin` ens ho exportarà amb format .zip. En aquest exemple, s'exportarà amb el format `source`. Hem de clicar a *Build Source Package*. La següent figura de la consola, ens permet veure que l'exportació s'ha efectuat amb èxit i ens indica la ruta on s'ha creat la còpia del paquet.
Una vegada descarregat, podrem instar-lo amb les següents comandes.

```
> paquet<-"C:/Users/Oriol/Documents/paquet/MDP_0.1.zip"
> install.packages(paquet, repos = NULL, type="source")
```

6.2 Presentació interactiva dels resultats

En aquest apartat exposarem una forma senzilla i pràctica de calcular i avaluar el mètode MDP. S'intentarà dissenyar dues aplicacions. La primera serà per aplicar el mètode MDP i KMDP. És a dir, partint d'un conjunt de dades importades, podrem seleccionar els paràmetres del nostre interès per a posteriorment calcular el mètode.
La segona aplicació, d'igual forma, partirà d'un conjunt de dades d'entrada, i un seguit de paràmetres per a *tunejar*. I respectivament ens generarà la validació dels models.

Aquestes aplicacions es dissenyen per mitjà del paquet d'R Shiny. Se'n aquest, l'encarregat de generar l'aplicació.
Caldran dues funcions, les anomenades `ui` i `server`. Però a mesura que es van omplint de codi pot resultar farragós treballar-hi, és per això, que poden definir-se ambdós *scripts* per separat. Exposem quin paper realitzen aquestes dues funcions:

- L'arxiu `ui.R`. Conté l'esquelet de l'aplicació. És on s'hauran d'especificar: els títols, comentaris, botons o pestanyes que formaran l'aplicació. En el nostre cas, especificarem en aquest guió la selecció de l'arxiu que es vol pujar i les opcions (paràmetres) del model a escollir.

- L'arxiu `server.R`. Disposa de tots els procediments necessaris per a que pugui resoldre's l'aplicació. Per al nostre cas, serà en aquest guió, on s'emmagatzemarà el codi de les nostres funcions de predicció.

6.2.1 MDP per a classificació

Aquesta aplicació és la que efetuarà una classificació tant pel MDP com pel KMDP. S'hi pot accedir a través del següent enllaç.
`https://prat.shinyapps.io/mdp_classification/`.
A continuació mostrarem un exemple per a que es comprengui el seu funcionament.

Utilitzarem les dades extretes de [28] sobre 83 pacients amb 4 diferents tipus de tumors cancerígens. Es disposen un total de 2300 gens de cèl·lules tumorals. Carreguem les dades, on es visualitza un `head` de la base de dades.

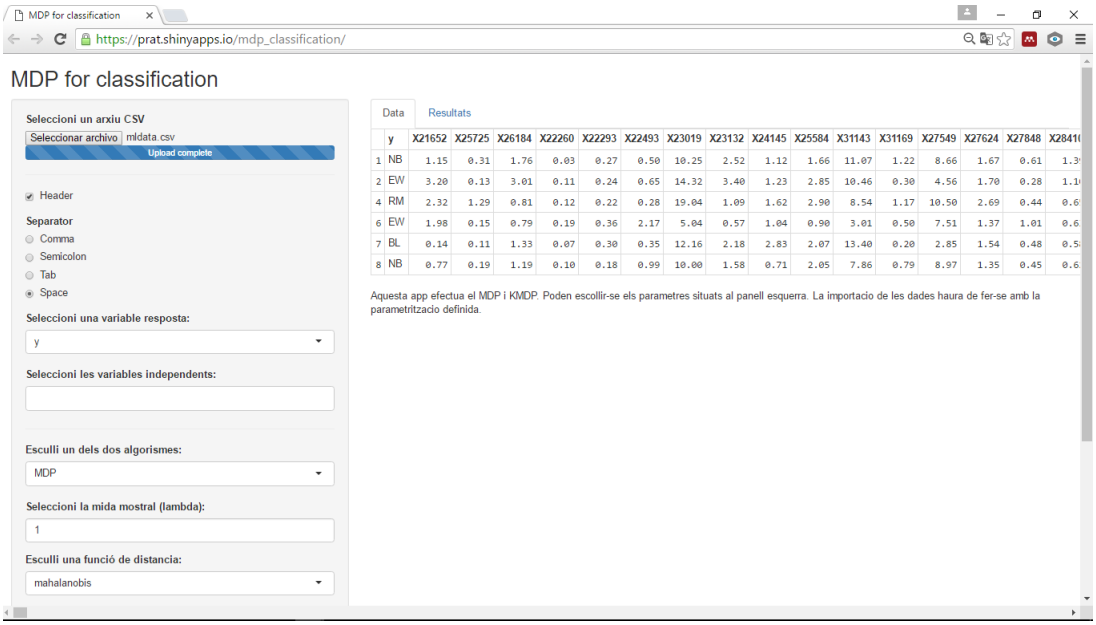


Figura 6.12: Panell web

Seleccionem la variable resposta i les explicatives. Shiny deixa un màxim de selecció de 62 variables, per tant caldrà escollir l'opció que les recull totes, les 2300.

Figura 6.13: Selecció de variables

Tenim una varietat de parametritzacions a escollir. Per a aquest exemple agafarem les que mostrem a continuació.

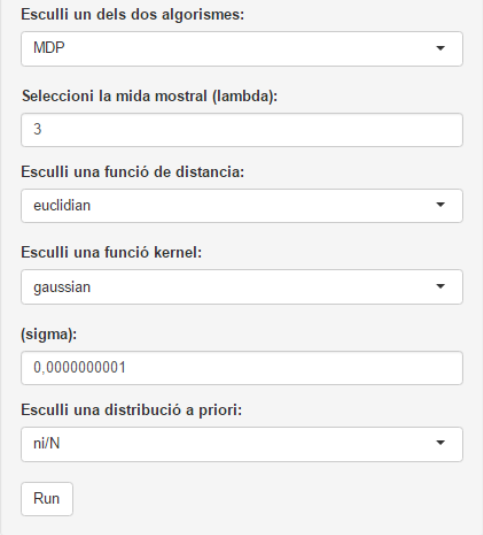


Figura 6.14: Selecció de paràmetres

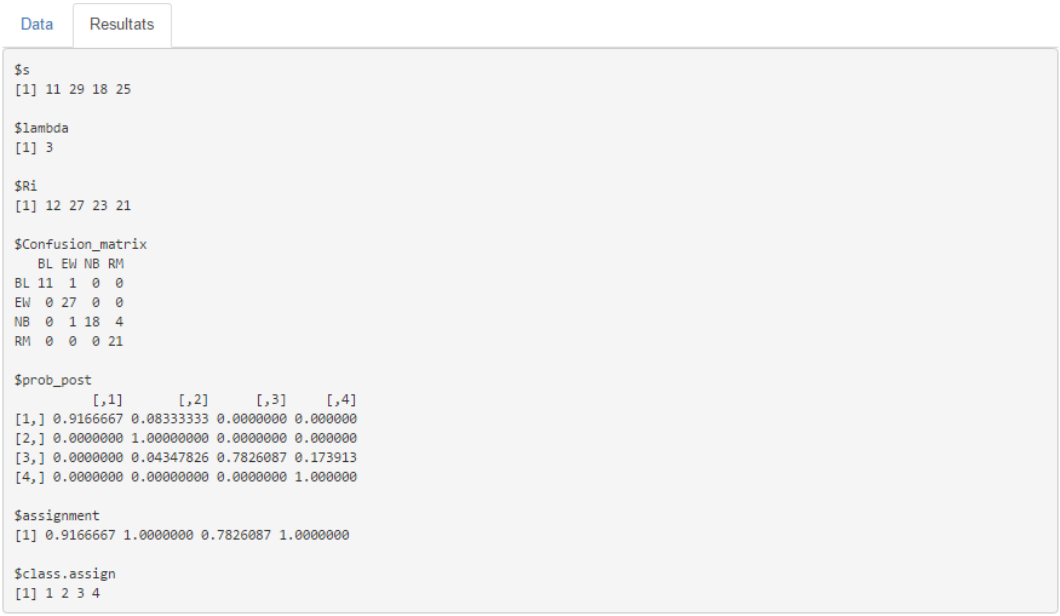


Figura 6.15: Resultats MDP

Finalment la pestanya de resultats ens genera una sortida de la funció `mdp`. On `s` és una taula de freqüències, `lambda` el paràmetre λ escollit i `Ri` la quantitat d'individus agrupats per regió. S'observa la matriu de confusió, i s'obté una taxa d'error del 7 %. També es mostra les probabilitats a posteriori, `assignment` són les probabilitats màximes i `class.assign` les subpoblacions assignades per a cada regió.

6.2.2 MDP per a validació

D'igual forma exposarem unes dades per a la selecció d'un model òptim. Pot accedir-s'hi a través de `https://prat.shinyapps.io/mdp_validation/`.

Carreguem les dades de cetoacidosis diabética 4.5.1. Caldrà escollir-se els paràmetres que es trobin covenients. Escollim validació creuada. Si es volen escollir repeticions caldrà seleccionar-se *"repeatedcv"* en la pestanya de selecció del mètode d'avaluació.

Aplicació MDP

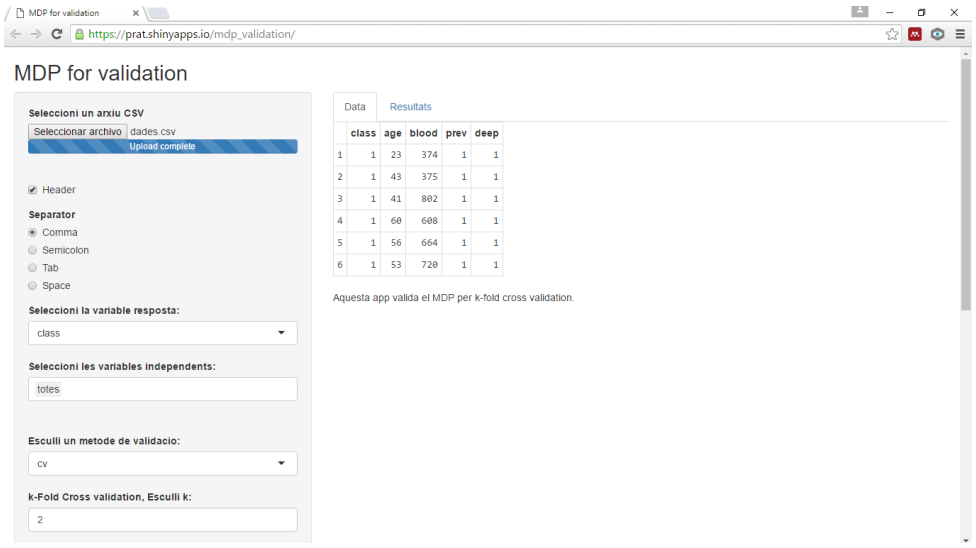


Figura 6.16: Panell de l'aplicació

Escollim les tres distàncies disponibles. I tres paràmetres λ.

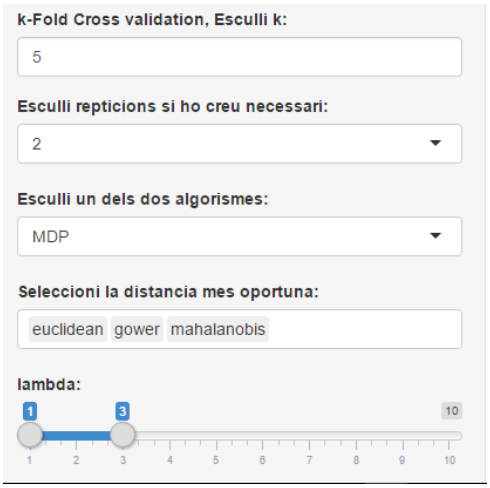


Figura 6.17: Especificacions dels paràmetres a *tunejar*

I finalment la distribució a priori, que la seleccionem com a equiprobable.

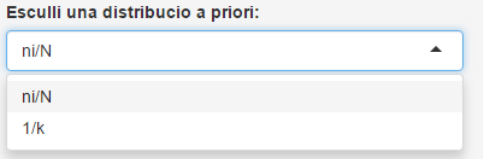


Figura 6.18: Selecció de l'a priori

Aconseguim un model òptim amb distància de Mahalanobis amb $\lambda = 1$. Que és la millor opció que ja s'havia exposat en 4.5.1.

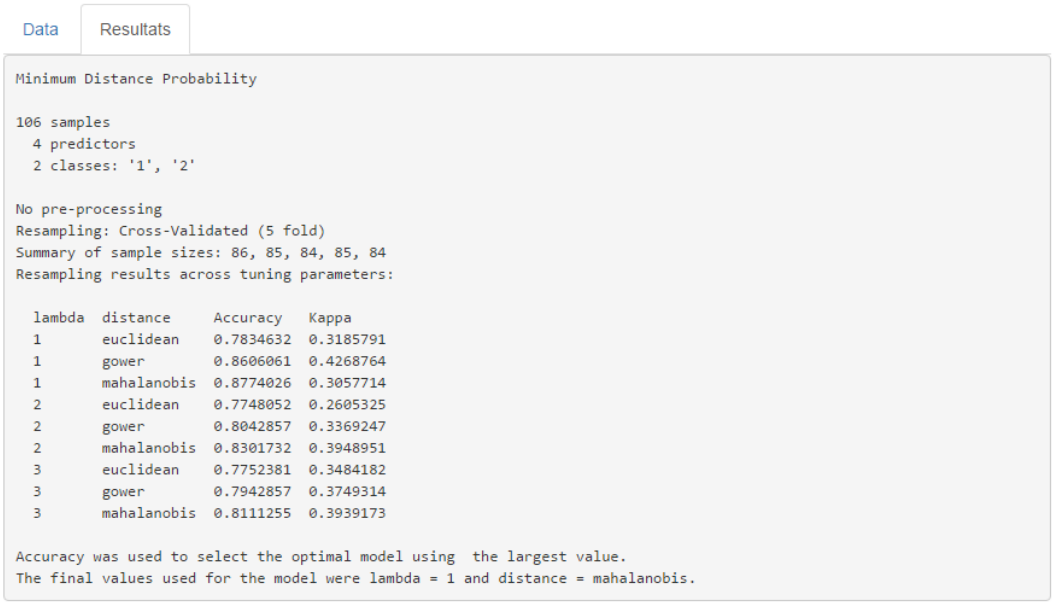


Figura 6.19: Generació dels resultats

Aplicació KMDP

Per a les mateixes dades, establim un seguit de models kernel. Seleccionem vectors de paràmetres σ a computar.

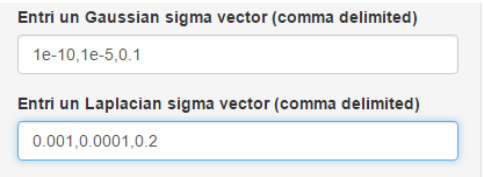


Figura 6.20: Parametrització kernel

S'aconsegueix un model òptim amb precisió de 0.79. Tenint una funció kernel Gaussiana. I $\lambda = 1$.

Data Resultats

```
Minimum Distance Probability

106 samples
  4 predictors
  2 classes: '1', '2'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 85, 85, 85, 85, 84
Resampling results across tuning parameters:

  lambda  kernel     Accuracy   Kappa
  1       p.d=1      0.7831169  0.3077660
  1       p.d=2      0.7826840  0.2367229
  1       g.s=1e-10  0.7831169  0.3077660
  1       g.s=1e-05  0.7831169  0.3077660
  1       g.s=0.1    0.7926407  0.2573062
  1       l.s=0.001  0.7831169  0.3077660
  1       l.s=1e-04  0.7831169  0.3077660
  1       l.s=0.2    0.7354978  0.1435512
  2       p.d=1      0.7744589  0.3182679
  2       p.d=2      0.7554113  0.1993161
  2       g.s=1e-10  0.7744589  0.3182679
  2       g.s=1e-05  0.7744589  0.3182679
  2       g.s=0.1    0.7549784  0.2171855
  2       l.s=0.001  0.7744589  0.3182679
  2       l.s=1e-04  0.7744589  0.3182679
  2       l.s=0.2    0.7649351  0.2326991
  3       p.d=1      0.7649351  0.2904763
  3       p.d=2      0.7077922  0.2154392
  3       g.s=1e-10  0.7649351  0.2904763
  3       g.s=1e-05  0.7649351  0.2904763
  3       g.s=0.1    0.7740260  0.1003394
  3       l.s=0.001  0.7649351  0.2904763
  3       l.s=1e-04  0.7649351  0.2904763
  3       l.s=0.2    0.7649351  0.2904763

Accuracy was used to select the optimal model using  the largest value.
The final values used for the model were lambda = 1 and kernel = g.s=0.1.
```

Figura 6.21: Resultats kernel

Capítol 7

CONCLUSIONS

7.1 Resultats obtinguts

El present treball s'ha realitzat amb l'objectiu principal d'aplicar l'estadística en el camp de la bioinformàtica.
Aquest objectiu s'ha plasmat des de l'inici d'aquesta memòria en el capítol 2. On s'han exposat alguns dels conceptes bàsics de la biologia mol·lecular, que han servit de bona base per a treballar de forma concisa al llarg del treball. A més, s'han exposat els principals problemes i, a l'hora, reptes que presenten les dades òmiques. Els quals, s'han vist clarament reflectits en les posteriors parts del treball.

En el tercer capítol s'han exposat els mètodes kernel, que són una molt bona proposta en situacions complexes on els mètodes lineals no poden donar solucions. A més, s'ha especificat una contundent característica que disposen per a calcular distàncies. En el quart capítol s'ha intentat explicar de la forma més detallada possible, la formulació del *Minimum Distance Probability* perquè pugui ser compresa i divulgada. També s'han analitzat algunes consideracions pràctiques que val la pena tenir en ment a l'hora d'utilitzar-se. S'ha implementat el mètode i exemplificat amb casos pràctics. A més, s'ha definit el mètode amb la sintaxi del paquet caret, per a poder trobar el model òptim a partir d'un nombre donat de paràmetres.

En el cinquè capítol, s'ha efectuat la cerca d'un biomarcador en una base de dades amb pacients amb càncer de colon. S'ha obtingut un conjunt de 5 gens capaços de predir amb una alta precisió per mitjà 10-*cross validation*. D'aquests 5 gens s'ha pogut observar que amb només 3 d'ells ja es guardava la mateixa capacitat predictiva. Provat amb 10-repeticions de 10-*cross validation*. L'execució del mètode MDP entrenat amb el biomarcador ha conduït a una taxa d'error del 5.9 % superior, respecte de la taxa d'error del MDP entrenat amb els 2000 gens. Finalment, l'últim objectiu ha estat subministrar

el mètode MDP tant als usuaris específics que treballen amb R, com a tot el públic, per mitjà d'una aplicació web interactiva.

Com a conclusió final cal afegir que el mètode MDP per a classificació proporciona tan bons resultats de predicció com els mètodes de classificació ja existents. El fet que disposi d'una varietat paramètrica com l'extensió al món kernel fa que pugui aconseguir-se una forçosa millora en les prediccions.

7.2 Possibles extencions de la memòria

Un cop acabat el projecte i revisant fins a on s'ha arribat, sorgeixen algunes extensions que es podrien realitzar a partir d'aquest treball:

- **Cost computacional de l'algorisme**: El tractament de l'ordre de la complexitat d'un algorisme és una camp que no s'ha estudiat al llarg del grau, fet que ha ajudat a la no consideració d'aquest a l'hora de programar. És per a això, que una futura dedicació focalitzada en optimitzar les funcions implementades, permetria una major rapidesa en l'execució i per tant un millor ús del mètode.

- **Selecció del paràmetre λ òptim**: En tot el treball s'ha fet ús de l'heurística *grid search* per a l'optimització del paràmetre λ i la resta de paràmetres. No obstant, una adaptació específica d'un algorisme d'optimització com pot ser el Nelder-Mead per als paràmetres numèrics, podria donar òptims més precisos que proporcionarien un millor entrenament del mètode.

- **Paràmetres del algorisme**: Al llarg del treball només s'han suposat tres funcions kernel, però n'existeix una gran varietat. Estudiar més a fons criteris de selecció de funcions o simplement integrar un ventall major, enriquiria més el mètode.

- **Alternatives de selecció**: En efectuar la recerca de biomarcadors s'ha hagut dedicar força temps d'execució. Sent aquest un dels inconvenients trobats al treballar amb dades d'alt rendiment. Per tant, una implantació de millores, com són la computació paral·lela o el clustering, podrien acomodar la recerca i fer-la més ràpida i eficient.

- **Sobredispersió en els biomarcadors**: La precisió aconseguida per al biomarcador ha estat forçosament més elevada que l'aconseguida per al conjunt total de variables. No obstant, en la predició amb les dades de *test* ha disminuït connsiderablement la predicció. Per tant, l'ús de mesures per tractar de forma més acurada la sobredispersió podria ser un proper repte a posteriori.

7.3 Valoració personal

La realització d'aquest treball ha estat una etapa que m'ha requerit un gran esforç i dedicació. A l'inici, tot i la gran motivació que tenia per a endinsar-m'hi, veia els objectius a l'horitzó com uns reptes difícils d'assolir. Per tant, vull transmetre que estic molt content per tot el treball realitzat durant aquest període que ha fet enriquir-me de coneixements des de molts vessants, que eren a l'inici desconeguts.

El que m'ha captivat més ha estat l'aprenentatge autònom a l'hora de resoldre problemes tant teòrics com pràctics. També en instruir-me en la lectura d'articles científics, ja que a l'inici estava molt poc entrenat i vaig haver d'invertir-hi força dedicació.

La capacitat de poder comprendre i implementar un mètode científic m'ha permès valorar i observar tota la feina feta que he hagut de dedicar per arribar a aquest objectiu.

Per últim vull afegir que m'he sentit molt afortunat d'haver pogut participar en un treball com el que aquest ha estat. I per tant, vull mostrar la total gratitud a l'Esteban Vegas que n'ha estat el tutor, el qual m'ha assessorat i m'ha ajudat sempre al llarg de totes les dificultats que han anat sorgint. Ja que sense ell, no hagués estat possible. També vull agrair el suport i recolzament que he tingut per part de família i amics.

Bibliografia

[1] J. Hoefkens, *Towards unbiased biomarker discovery,* Drug Discov. World, vol. 11, no. 2, pp. 19-24, 2010.

[2] M. Lin, B. Fan, J. C. S. Lui, and D. M. Chiu, *Stochastic analysis of file-swarming systems,* Perform. Eval., 2007.

[3] MATLAB, *Prediction of tumor class from gene expression data using bagged decision trees.*

[4] L. M. Merino González and E. Santos Aláez, *Álgebra lineal: con métodos elementales. Madrid: Thomson, 2006.*

[5] R. Spang and F. Markowetz, *Microarray Analysis Classification by SVM and PAM.*

[6] A. Villarroya, M. Ríos, and J. M. Oller, *Discriminant analysis algorithm based on a distance function and on a Bayesian decision.,* Biometrics, vol. 51, no. 3, pp. 908-919, 1995.

[7] J. Witten, Hastie, Tibshirani, G. James, D. Witten, T. Hastie, and R. Tibshirani, *Springer Texts in Statistics An Introduction to Statistical Learning.*

[8] *Practising Microarray Data Analysis Using R amp; BioConductor.*

[9] O. Chapelle and A. Zien, *Semi-Supervised Classification by Low Density Separation.*

[10] H. Daumé Iii, *From Zero to Reproducing Kernel Hilbert Spaces in Twelve Pages or Less,* 2004.

[11] M. Kuhn, *Predictive Modeling with R and the caret Package.*

[12] A. Sánchez, *New Challenges in Bioinformatics: Integrative Analysis of Omics Data.*

[13] D. Sejdinovic, A. Gretton, B. Sriperumbudur, and K. Fukumizu, *Hypothesis Testing Using Pairwise Distances and Associated Kernels,* 2010.

[14] D. Sejdinovic and A. Gretton, *Foundations of Reproducing Kernel Hilbert Spaces Advanced Topics in Machine Learning,* 2013.

[15] K. Teknomo, *Linear Discriminant Analysis (LDA) Numerical Example.* [Online]. Available: http://people.revoledu.com/kardi/tutorial/LDA/NumericalExample.html.

[16] A. Von Heydebreck, *Statistical tests for differential gene expression.*

[17] *Colon cancer data set of Alon et al. 1999.* [Online]. Available: http://www.stats.uwo.ca/faculty/aim/2015/9850/microarrays/FitMArray/chm/Alon.html.

[18] *What is the 'Central Dogma'? | Facts | yourgenome.org.* [Online]. Available: http://www.yourgenome.org/facts/what-is-the-central-dogma.

[19] *kernel functions for machine learning applications,* 2010.

[20] G. Karp, Biología celular y molecular. McGraw-Hill, 2009.

[21] S. V. Nalini Chandar, Biología molecular y celular. Lippincott Williams and Wilkis. Wolters Kluwer Health, 2012.

[22] P. Rai, *Kernel Methods and Nonlinear Classification slides,* vol. 2011, 2011.

[23] J. Shawe-Taylor and N. Cristianini, Kernel Methods for Pattern Analysis, vol. 47. 2004.

[24] Riba, L. *Integració de diferents fonts de dades òmiques i visualització de les variables originals mitjançant tècniques de Machine Learning*, Vegas, E., 2014.

[25] Cartanyà, A. *Classificació de proteïnes segons on es troben a la cèl·lula mitjançant tècniques de Machine Learning,*, Vegas, E., 2015.

[26] Reproducing Kernel Hilbert Space I: Basic Definitions. 2011.

[27] Astrobiologia UPC [Online]. Available: http://www.dmae.upm.es/Astrobiologi/Curso_online_UPC/capitulo9/.

[28] Khan. *Classification and diagnostic prediction of cancers using gen expression profiling and ANN* [Online]. Available: http://research.nhgri.nih.gov/microarray/Supplement/.

[29] M. Núñez, A. Villarroya, and J. M. Oller, *Minimum Distance Probability Discriminant Analysis for Mixed Variables,* Biometrics, vol. 59, no. 2, pp. 248-253, 2003.

[30] R. Gentleman, Bioinformatics and Computacional Biology Solutions Using R and Bioconductor.

[31] P. P. Sinha, Bioinformatics with R Cookbook.

Annex A

CODI R

A.1 Capítol 4: Funcions implementades amb R

A.1.1 mdp

```
> mdp<-function(Class,data,lambda,distance=NULL,prior=NULL,info.pred=NULL
+               ,kernel=NULL){
+   input=mget(names(formals()),sys.frame(sys.nframe()))
+   if(is.numeric(Class)){Class=as.numeric(Class); lev=FALSE
+   } else { lev=TRUE
+   if(!is.factor(Class)){
+     Class=as.matrix(Class)
+     Class=as.factor(Class)
+   }
+   cm.levels=list(levels(Class),levels(Class))
+   Class=as.numeric(Class)
+   }
+   data = as.matrix(data)
+   k=nrow(table(Class))
+   K.laplacedot=NULL; K.polydot=NULL; K.rbfdot=NULL
+   C.pool=NULL
+   if(missing(distance) && missing(kernel)){distance="mahalanobis"}
+
+   if(!is.null(distance)){
+     if(distance=="mahalanobis"){
+       D=matrix(data=0,nrow=dim(data)[1],ncol=dim(data)[1])
+       Data=data.frame(data); Data$class=Class
+       d=c()
+       L.cov=vector("list",k)
```

```
+       for(cl in 1:k){
+         L.cov[[cl]]=cov(subset(Data,Data$class==cl,select = -c(class)))
+       }
+       C.pool=matrix(0,nrow=dim(data)[2],ncol=dim(data)[2])
+       for (i in 1:ncol(data)){
+         for(j in 1:ncol(data)){
+           for(cl in 1:k){
+ C.pool[i,j]=C.pool[i,j]+L.cov[[cl]][i,j]*(as.numeric(table(Class)[cl])-1)/(dim(dat
+           }
+         }
+       }
+
+       for(i in 1:dim(data)[1]){
+         d=c()
+         for(j in 1:dim(data)[1]){
+           d=c(d,mahalanobis(data[i,], data[j,],C.pool))
+           D[i,j]=d[j]
+         }
+       }
+
+       } else if(distance=="euclidean"){
+       D=matrix(data=NA,nrow=dim(data)[1],ncol=dim(data)[1])
+       d=c()
+       for(i in 1:dim(data)[1]){
+         d=c()
+         for(j in 1:dim(data)[1]){
+           d=c(d,as.numeric(dist(rbind(data[i,],data[j,]))))
+           D[i,j]=d[j]
+         }
+       }
+
+       } else if(distance=="gower"){
+       if(!require(cluster)){
+         install.packages("cluster")
+         library(cluster)
+       }
+       D=as.matrix(daisy(data, metric = "gower"))
+     }
+   }
```

```
+
+   if(!is.null(kernel)){
+     if(!require(kernlab)){
+       install.packages("kernlab")
+       library(kernlab)
+     }
+     D=matrix(data=NA,nrow=dim(data)[1],ncol=dim(data)[1])
+     d=c()
+     if(substr(kernel, 1, 1)=="g"){
+       sigma=as.numeric(substr(kernel, 5, nchar(kernel)))
+
+       K.rbfdot<-function(x,y){
+         rbf <- rbfdot(sigma = sigma)
+         dist = 2*(1-rbf(x,y))
+         sqrt(dist)
+       }
+       for(i in 1:dim(data)[1]){
+         d=c()
+         for(j in 1:dim(data)[1]){
+           d=c(d,K.rbfdot(data[i,],data[j,]))
+           D[i,j]=d[j]
+         }
+       }
+     } else if(substr(kernel, 1, 1)=="p"){
+       degree=as.numeric(substr(kernel, 5, nchar(kernel)))
+       K.polydot<-function(x,y){
+         lap <- polydot(degree = degree, scale = 1, offset = 1)
+         dist = lap(x,x)+lap(y,y)-2*lap(x,y)
+         sqrt(dist)
+       }
+       for(i in 1:dim(data)[1]){
+         d=c()
+         for(j in 1:dim(data)[1]){
+           d=c(d,K.polydot(data[i,],data[j,]))
+           D[i,j]=d[j]
+         }
+       }
+     } else if(substr(kernel, 1, 1)=="l"){
+       sigma=as.numeric(substr(kernel, 5, nchar(kernel)))
```

```
+
+       K.laplacedot<-function(x,y){
+         lap <- laplacedot(sigma = sigma)
+         dist = 2*(1-lap(x,y))
+         sqrt(dist)
+       }
+       for(i in 1:dim(data)[1]){
+         d=c()
+         for(j in 1:dim(data)[1]){
+           d=c(d,K.laplacedot(data[i,],data[j,]))
+           D[i,j]=d[j]
+         }
+       }
+     }
+   }
+
+   comb = function(n, x) {
+     return(factorial(n) / (factorial(x) * factorial(n-x)))
+   }
+
+   ind<-function(x,j){
+     delta=0;
+     if(x%in%g[,j]){delta=1}
+     delta
+   }
+
+   h=matrix(data=0,nrow=dim(data)[1],ncol=k)
+   g=matrix(data=0,nrow=dim(data)[1],ncol=k)
+   s<-rep(0,k)
+   for(c in 1:k){
+     for(i in 1:dim(data)[1]){
+       if(Class[i]==c){g[i,c]=i; s[c]<-s[c]+1}
+     }
+   }
+   #S = prod(comb(s,lambda))
+
+   row=1
+   R=rep(0,k)
+   ER=rep(0,k)
```

```
+   mat=matrix(data=0,nrow=k,ncol=k)
+   while(row<=(dim(h)[1])){
+     h=matrix(data=0,nrow=dim(data)[1],ncol=k)
+     col=1
+     while(col<=k){
+       for(i in 1:dim(h)[1]){
+         for(j in which(g[,col]!=0)){
+           if(D[row,i]<D[row,j]){h[i,col]=h[i,col]+1}
+         }
+       }
+       col<-col+1
+     }
+
+     prod=0
+     vec.q<-rep(0,k)
+     col=1
+
+     while(col<=k){
+       for(i in which(g[,col]!=0)){
+         vec.k<-rep(0,k)
+         for(j in 1:dim(h)[2]){
+           if(h[i,j]>=lambda-ind(i,j)){vec.k[j]=comb(h[i,j],lambda-ind(i,j))}
+         }
+
+         if(sum(vec.k==0)==0){
+           prod = prod+prod(vec.k)
+         }
+
+       }
+       vec.q[col]=prod; prod=0
+       col<-col+1
+     }
+
+     j=which.max(vec.q)
+     if(sum(vec.q[j]==vec.q[-j],na.rm = T)!=0){
+       w = which(vec.q[j]==vec.q)
+       j=sample(w,1)
+     }
+
```

```
+     R[j]=R[j]+1
+
+     if(row%in%g[,j]){ER[j]=ER[j]+1}
+     else{
+       for(f in (1:k)[-j]){
+         if(row%in%g[,f]){mat[j,f]=mat[j,f]+1}
+       }
+     }
+
+     row<-row+1
+   }
+   cm=diag(ER)+mat
+   if(lev){dimnames(cm)=cm.levels}
+   if(missing(prior)){prior="ni/N"}
+
+   if(class(prior)=="character"){
+
+     if(prior=="ni/N"){
+       pi=as.numeric(table(Class))/dim(data)[1]
+     } else if(prior=="1/k"){pi=rep(1/k,k)}
+
+     pER=matrix(ncol=k,nrow=k)
+     pRE=matrix(ncol=k,nrow=k)
+     for(i in 1:k){
+       for(j in 1:k){
+         pRE[i,j]=cm[i,j]/sum(cm[,j])
+       }
+     }
+     R.scale = numeric(k)
+     for(i in 1:k){
+       for(j in 1:k){
+         R.scale[i] = R.scale[i] + pRE[i,j]*pi[j]
+       }
+     }
+     for(i in 1:k){
+       for(j in 1:k){
+         pER[i,j]=(pRE[i,j]*pi[j])/R.scale[i]
+       }
+     }
```

```
+       assignment = numeric(k)
+       class.assign = numeric(k)
+       for(i in 1:k){
+         assignment[i] = max(pER[i,])
+         class.assign[i]=which.max(pER[i,])
+       }
+     }
+
+   if(class(prior)=="numeric"){
+     pER=matrix(ncol=k,nrow=k)
+     pRE=matrix(ncol=k,nrow=k)
+     for(i in 1:k){
+       for(j in 1:k){
+         pRE[i,j]=cm[i,j]/sum(cm[,j])
+         pRE[i,j]=pRE[i,j]*prior[j]
+       }
+     }
+
+     for(i in 1:k){
+       for(j in 1:k){
+         pER[i,j]=pRE[i,j]/sum(pRE[i,])
+       }
+     }
+     assignment = numeric(k)
+     class.assign = numeric(k)
+     for(i in 1:k){
+       assignment[i] = max(pER[i,])
+       class.assign[i]=which.max(pER[i,])
+     }
+   }
+
+
+
+   if(missing(info.pred)||info.pred=="no" ||info.pred==FALSE){
+     return(list(s=s,lambda=lambda,Ri=R,Confusion_matrix=cm,
+                 prob_post=pER,assignment=assignment,
+                 class.assign=class.assign))
+
+   } else if(lev){return(list(input=input,D=D,k=k,g=g,comb=comb,ind=ind,
```

```
+                       K.rbfdot=K.rbfdot,K.laplacedot=K.laplacedot,K.polydot=K.poly
+                       C.pool=C.pool,Ri=R,Confusion_matrix=cm,
+                       cm.levels=cm.levels,assignment=assignment,class.assign=class
+   } else{
+     return(list(input=input,D=D,k=k,g=g,comb=comb,ind=ind,
+                 K.rbfdot=K.rbfdot,K.laplacedot=K.laplacedot,K.polydot=K.polydot,
+                 C.pool=C.pool,Ri=R,Confusion_matrix=cm,assignment=assignment,
+                 class.assign=class.assign))
+     }
+
+ }
```

A.1.2 predict.mdp

```
> predict.mdp<-function(modelFit,newdata){
+   data=as.matrix(modelFit$input[["data"]])
+   newdata=as.matrix(newdata)
+   k=modelFit$k
+   g=modelFit$g
+   D=modelFit$D
+   lambda=modelFit$input[["lambda"]]
+   comb=modelFit$comb
+   ind=modelFit$ind
+   class.assign=modelFit$class.assign
+   start=dim(D)[1]
+   vec.new=(start+1):(start+dim(newdata)[1])
+   C.pool=modelFit$C.pool
+
+   if(is.null(modelFit$input[["distance"]]) && is.null(modelFit$input[["kernel"]]))
+     modelFit$input[["distance"]]="mahalanobis"
+   } else if(!is.null(modelFit$input[["distance"]])){
+     distance=modelFit$input[["distance"]]
+     if(distance=="mahalanobis"){
+       for(i in 1:dim(newdata)[1]){
+         d=c()
+         D=cbind(D,c(0))
+         D=rbind(D,c(0))
+         for(j in 1:dim(data)[1]){
+           d=c(d,mahalanobis(newdata[i,], data[j,],C.pool))
```

```
+           D[vec.new[i],j]=d[j]
+         }
+       }
+     } else if(distance=="euclidean"){
+       for(i in 1:dim(newdata)[1]){
+         d=c()
+         D=cbind(D,c(0))
+         D=rbind(D,c(0))
+         for(j in 1:dim(data)[1]){
+           d=c(d,as.numeric(dist(rbind(newdata[i,], data[j,]))))
+           D[vec.new[i],j]=d[j]
+         }
+       }
+     } else if(distance=="gower"){
+       if(!require(cluster)){
+         install.packages("cluster")
+         library(cluster)
+       }
+       data2=rbind(data,newdata)
+       D=as.matrix(daisy(data2, metric = "gower"))
+     }
+   } else if(!is.null(modelFit$input[["kernel"]])){
+     if(!require(kernlab)){
+       install.packages("kernlab")}
+     kernel=modelFit$input[["kernel"]]
+     if(substr(kernel, 1, 1)=="g"){
+       sigma=as.numeric(substr(kernel, 5, nchar(kernel)))
+       K.rbfdot<-function(x,y){
+         rbf <- rbfdot(sigma = sigma)
+         dist = 2*(1-rbf(x,y))
+         sqrt(dist)
+       }
+       for(i in 1:dim(newdata)[1]){
+         d=c()
+         D=cbind(D,c(0))
+         D=rbind(D,c(0))
+         for(j in 1:dim(data)[1]){
+           d=c(d,K.rbfdot(newdata[i,], data[j,]))
+           D[vec.new[i],j]=d[j]
```

```
+         }
+       }
+     } else if(substr(kernel, 1, 1)=="l"){
+       sigma=as.numeric(substr(kernel, 5, nchar(kernel)))
+       K.laplacedot<-function(x,y){
+         lap <- laplacedot(sigma = sigma)
+         dist = 2*(1-lap(x,y))
+         sqrt(dist)
+       }
+       for(i in 1:dim(newdata)[1]){
+         d=c()
+         D=cbind(D,c(0))
+         D=rbind(D,c(0))
+         for(j in 1:dim(data)[1]){
+           d=c(d,K.laplacedot(newdata[i,], data[j,]))
+           D[vec.new[i],j]=d[j]
+         }
+       }
+
+     } else if(substr(kernel, 1, 1)=="p"){
+       degree=as.numeric(substr(kernel, 5, nchar(kernel)))
+       K.polydot<-function(x,y){
+         lap <- polydot(degree = degree, scale = 1, offset = 1)
+         dist = lap(x,x)+lap(y,y)-2*lap(x,y)
+         sqrt(dist)
+       }
+       for(i in 1:dim(newdata)[1]){
+         d=c()
+         D=cbind(D,c(0))
+         D=rbind(D,c(0))
+         for(j in 1:dim(data)[1]){
+           d=c(d,K.polydot(newdata[i,], data[j,]))
+           D[vec.new[i],j]=d[j]
+         }
+       }
+
+     }
+   }
+
```

```
+   r=numeric(dim(newdata)[1])
+   R=rep(0,k)
+   ER=rep(0,k)
+   mat=matrix(data=0,nrow=k,ncol=k)
+   t=1
+   for(row in vec.new){
+     h=matrix(data=0,nrow=start,ncol=k)
+     col=1
+     while(col<=k){
+       for(i in 1:dim(h)[1]){
+         for(j in which(g[,col]!=0)){
+           if(D[row,i]<D[row,j]){h[i,col]=h[i,col]+1}
+         }
+       }
+       col<-col+1
+     }
+
+     prod=0
+     vec.q<-rep(0,k)
+     col=1
+
+     while(col<=k){
+       for(i in which(g[,col]!=0)){
+         vec.k<-rep(0,k)
+         for(j in 1:dim(h)[2]){
+           if(h[i,j]>=lambda-ind(i,j)){vec.k[j]=comb(h[i,j],lambda-ind(i,j))}
+         }
+
+         if(sum(vec.k==0)==0){
+           prod = prod+prod(vec.k)
+         }
+       }
+       vec.q[col]=prod; prod=0
+       col<-col+1
+     }
+
+     j=which.max(vec.q)
+     if(sum(vec.q[j]==vec.q[-j],na.rm = T)!=0){
+       w = which(vec.q[j]==vec.q)
```

```
+       j=sample(w,1)
+     }
+
+     r[t]=class.assign[j]
+     t<-t+1
+   }
+   if(!is.null(modelFit$cm.levels)){
+     n=length(modelFit$cm.levels[[1]])
+     for(i in 1:n){
+       r[which(r==i)]<-modelFit$cm.levels[[1]][i]
+     }
+     r=factor(r)
+   }
+   return(r)
+ }
>
```

A.2 Capítol 5: *tuning parameters* amb R

A.2.1 tune.mdp

```
> tune.mdp<-function(algorithm, Class, data, validation.method,k.fold,repeats=5,
+                    degree=1:2, sigma.g=NULL,sigma.l=NULL, lambda){
+   library(caret)
+   my_mod <- list(type = "Classification",
+                  library = "stats",
+                  loop = NULL)
+
+   my_mod$label <- "Minimum Distance Probability"
+
+   my_mod$sort <- NULL
+   my_mod$levels<-NULL
+
+   fitControl <- trainControl(method = validation.method,number = k.fold,
+                              repeats = repeats)
+
+
+
+   if(algorithm=="MDP"){
```

```
+     my_mod$parameters <- data.frame(
+       parameter = c("lambda","distance"),
+       class = c("numeric", "character"),
+       label = c("sample_length","distance_type"))
+
+     ## The grid Element
+
+     my_mod$grid <- function(x, y, len = NULL){expand.grid(lambda=lambda,
+     distance=c("euclidean"))}
+     mygrid<-expand.grid(lambda=lambda,distance=c("euclidean","gower"))
+
+
+
+     ## The fit Element
+
+     my_mod$fit <- function(x, y, wts, param, lev, last, classProbs, ...){
+       mdp(Class=y,data=x,lambda=param$lambda,prior="1/k",
+           distance=as.character(param$distance),info.pred ="yes")
+     }
+
+     ## The predict Element
+
+     mdpPred <- function(modelFit, newdata, preProc = NULL, submodels = NULL)
+       predict.mdp(modelFit, newdata)
+     my_mod$predict <- mdpPred
+
+     mdpProb <- function(modelFit, newdata, preProc = NULL, submodels = NULL)
+       predict.mdp(modelFit, newdata)
+     my_mod$prob <- mdpPred
+     entreno=train(x=data, y = factor(Class),method = my_mod,trControl = fitControl
+     entreno
+
+   } else {
+     my_mod$parameters <- data.frame(
+       parameter = c("lambda","kernel"),
+       class = c("numeric", "character"),
+       label = c("sample_length", "kernel_type"))
+
+     ## The grid Element
```

```
+     vec.degree  = paste0("p.d=",degree)
+     vec.sigma.l = paste0("l.s=",sigma.l)
+     vec.sigma.g = paste0("g.s=",sigma.g)
+
+     my_mod$grid <- function(x, y, len = NULL){
+
+       expand.grid(lambda=lambda,kernel=c(vec.degree,vec.sigma.g,vec.sigma.l))}
+     mygrid<-expand.grid(lambda=lambda,kernel=c(vec.degree,vec.sigma.g,vec.sigma.l)
+
+
+     ## The fit Element
+
+     my_mod$fit <- function(x, y, wts, param, lev, last, classProbs, ...){
+       mdp(Class=y,data=x,lambda=param$lambda,prior="1/k",
+           kernel=as.character(param$kernel),info.pred ="yes")
+     }
+
+     ## The predict Element
+
+     mdpPred <- function(modelFit, newdata, preProc = NULL, submodels = NULL)
+       predict.mdp(modelFit, newdata)
+     my_mod$predict <- mdpPred
+
+     mdpProb <- function(modelFit, newdata, preProc = NULL, submodels = NULL)
+       predict.mdp(modelFit, newdata)
+     my_mod$prob <- mdpPred
+     entreno=train(x=data, y = factor(Class),method = my_mod,trControl = fitControl
+                   tuneGrid = mygrid)
+     entreno
+     }
+
+ }
>

> x=5
> n=100
> vb.n = dim(alon$x)[2]
> set.seed(33)
> mostres=matrix(ncol = x, nrow = n)
```

```
> for(i in 1:n){
+   mostres[i,]=sample(1:vb.n, size=x, replace = F)
+ }
> l=vector("list",n)
> for(i in 1:n){
+   l[[i]]=tune.mdp(algorithm = "MDP",Class = alon$y[1:45],
+                   data = alon$x[1:45,mostres[i,]],
+                   validation.method = "cv",k.fold = 10,
+                   lambda = 3:7)
+ }
> l1=vector("list",1)
> maxim=0
> for(i in 1:n){
+   if(max(l[[i]]$results$Accuracy[l[[i]]$results$Kappa>0]) > maxim){
+     maxim=max(l[[i]]$results$Accuracy[l[[i]]$results$Kappa>0])
+     lambda=l[[i]]$results$lambda[l[[i]]$results$Accuracy==maxim]
+     distance=l[[i]]$results$distance[l[[i]]$results$Accuracy==maxim]
+     l1=list(i=i,vb=mostres[i,],maxim=maxim, lambda=lambda,
+             distance=distance)
+   }
+ }
> l.k=vector("list",n)
> for(i in 1:n){
+   l.k[[i]]=tune.mdp(algorithm = "KMDP",Class = alon$y[1:45],
+                     data = alon$x[1:45,mostres[i,]],
+                     validation.method = "cv",k.fold = 10
+                     ,sigma.g = c(1e-10),lambda = 7)
+ }
> lk1=vector("list",1)
> maxim=0
> for(i in 1:n){
+   if(max(l.k[[i]]$results$Accuracy[l.k[[i]]$results$Kappa>0]) > maxim){
+     maxim=max(l.k[[i]]$results$Accuracy[l.k[[i]]$results$Kappa>0])
+     lambda=l.k[[i]]$results$lambda[l.k[[i]]$results$Accuracy==maxim]
+     kernel=l.k[[i]]$results$kernel[l.k[[i]]$results$Accuracy==maxim]
+     lk1=list(maxim=maxim, lambda=lambda, kernel=kernel)
+   }
+ }
> mdp1=mdp(Class = alon$y, data = alon$x, lambda = 1,
```

```
+          distance = "euclidean",prior = "1/k")
> mdp2=mdp(Class = alon$y, data = alon$x, lambda = 2,
+          distance = "euclidean",prior = "1/k")
> mdp3=mdp(Class = alon$y, data = alon$x, lambda = 3,
+          distance = "euclidean",prior = "1/k")
> entreno.kmdp=train(x=alon$x, y = factor(alon$y),
+                    method = my_mod,trControl = fitControl,
+                    tuneGrid = mygrid)
> fitControl <- trainControl(method = "repeatedcv",number = 10,
+                            repeats = 5)
> train=1:45
> test=46:62
> sigma=0.0000000001
> tune.mdp(algorithm = "MDP",Class = alon$y,data = alon$x,
+          validation.method = "cv",k.fold = 10,lambda = 1:3)
> knn1 <- knn3(alon$x,alon$y, k = 1)
> knnFit <- train(x= alon$x, y = alon$y, method = "knn",
+                 trControl = fitControl,
+                 tuneGrid=expand.grid(.k=1:8))
> plot(knnFit)
> plot(entreno)
>

> mostra=mostres[80,]
> sample(mostra,size = 2,replace = F)
> mo1=rbind(c(493,343,1985),c(493,343,943),c(493,343,873),
+           c(343,1985,943),c(873,1985,493),
+           c(343,943,873),c(1985,943,873),c(873,943,493),
+           c(873,493,1985),c(873,1985,343))
> mo2=rbind(c(493,343),c(493,1985),c(493,943),c(493,873),
+           c(1985,943),c(1985,873),
+           c(943,873),c(873,343),c(343,943),c(343,1985))
> l.n3=vector("list",10)
> for(i in 1:10){
+   l.n3[[i]]=tune.mdp(algorithm = "MDP",Class = alon$y[1:45],
+            data = alon$x[1:45,mo1[i,]]
+            ,validation.method = "repeatedcv",
+            k.fold = 10,repeats=10,lambda = 5)
+ }
```

```
> l.n2=vector("list",10)
> for(i in 1:10){
+   l.n2[[i]]=tune.mdp(algorithm = "MDP",Class = alon$y[1:45],
+                   data = alon$x[1:45,mo2[i,]],
+                   validation.method = "repeatedcv",
+                   k.fold = 10,repeats=10,lambda = 5)
+ }
> tune.mdp(algorithm = "MDP",Class = alon$y[1:45],
+          data = alon$x[1:45,c(343,493)],validation.method = "cv",
+          k.fold = 10,lambda = 7)
> tune.mdp(algorithm = "MDP",Class = alon$y[1:45],
+          data = alon$x[1:45,343],validation.method = "cv",
+          k.fold = 10,lambda = 7)
> inter=mdp(Class = alon$y[1:45], data = alon$x[1:45,343],
+           distance = "euclidean",info.pred = T,lambda = 7,
+           prior = "1/k")
> predict.mdp(inter,alon$x[46:62])
>
```

A.3 Capítol 6: Aplicació amb *Shiny*

A.3.1 App 1

```
> # ui.R
> library(shiny)
> shinyUI(fluidPage(
+   titlePanel("MDP for classification"),
+   sidebarLayout(
+     sidebarPanel(
+       fileInput('file1', 'Seleccioni un arxiu CSV',
+                 accept=c('text/csv',
+                          'text/comma-separated-values,text/plain',
+                          '.csv')),
+
+       tags$hr(),
+       checkboxInput(inputId = 'header', label = 'Header', value = FALSE),
+       radioButtons(inputId = 'sep',label = 'Separator',
+                    choices = c(Comma=',',Semicolon=';',
+                                Tab='\t',Space=''),selected = ','),
```

```
+
+       uiOutput("dependent"),
+       uiOutput("independents"),
+
+
+       tags$hr(),
+       selectInput("algorithm", "Esculli un dels dos algorismes:",
+                   choices = c("MDP", "KMDP")),
+       numericInput("lambda", "Seleccioni la mida mostral (lambda):",
+                    min = 1,max = 5, 1),
+
+       selectInput("distance", "Esculli una funció de distancia:",
+                   choices = c("mahalanobis", "euclidian", "gower")),
+       selectInput("kernel", "Esculli una funció kernel:",
+                   choices = c("gaussian", "polynomial", "laplacian")),
+       numericInput("sigma", "(sigma):",1e-10,min = 1e-10,max = 3, step=1e-9),
+
+       selectInput("prior", "Esculli una distribució a priori:",
+                   choices = c("ni/N", "1/k")),
+       uiOutput('ui.action') # instead of conditionalPanel
+     ),
+     mainPanel(
+       uiOutput("tb"),
+       p("Aquesta app efectua el MDP i KMDP.
+         Poden escollir-se els parametres situats al panell esquerra.
+         La importacio de les dades haura de fer-se amb la parametritzacio definida
+
+     )
+   )
+ ))
>

> # App1 .server
> # server.R
> library(shiny)
> shinyServer(function(input, output) {
+
+   filedata <- reactive({
+     infile <- input$file1
```

```
+     if (is.null(infile)){
+       return(NULL)
+     }
+     read.csv(infile$datapath, header = input$header,
+              sep = input$sep)
+   })
+
+
+   output$sum<-renderTable({
+     if(is.null(filedata())){return ()}
+     summary(filedata())
+   })
+
+   output$dependent <- renderUI({
+     df <- filedata()
+     if (is.null(df)) return(NULL)
+     items=names(df)
+     names(items)=items
+     selectInput("dependent","Seleccioni una variable resposta:",items)
+   })
+   output$nrows <- reactive({
+     nrow(filedata())
+   })
+
+   output$independents <- renderUI({
+     df <- filedata()
+     if (is.null(df)) return(NULL)
+     items=names(df)
+     names(items)=items
+     items=items[!items==input$dependent]
+     items=c("totes",items)
+     selectInput("independents","Seleccioni les variables independents:",
+                 items,multiple=TRUE)
+   })
+
+   output$contents <- renderPrint({
+     input$action
+ mdp<-function(Class,data,lambda,distance=NULL,prior=NULL,info.pred=NULL,
+               kernel=NULL,degree=NULL,sigma=NULL){
```

```
+       input=mget(names(formals()),sys.frame(sys.nframe()))
+       if(is.numeric(Class)){Class=as.numeric(Class); lev=FALSE
+       } else { lev=TRUE
+       if(!is.factor(Class)){
+         Class=as.matrix(Class)
+         Class=as.factor(Class)
+       }
+       cm.levels=list(levels(Class),levels(Class))
+       Class=as.numeric(Class)
+       }
+       data = as.matrix(data)
+       k=nrow(table(Class))
+       K.laplacedot=NULL; K.polydot=NULL; K.rbfdot=NULL
+       C.pool=NULL
+       if(missing(distance) && missing(kernel)){distance="mahalanobis"}
+
+       if(!is.null(distance)){
+         if(distance=="mahalanobis"){
+           D=matrix(data=0,nrow=dim(data)[1],ncol=dim(data)[1])
+           Data=data.frame(data); Data$class=Class
+           d=c()
+           L.cov=vector("list",k)
+           for(cl in 1:k){
+             L.cov[[cl]]=cov(subset(Data,Data$class==cl,select = -c(class)))
+           }
+           C.pool=matrix(0,nrow=dim(data)[2],ncol=dim(data)[2])
+           for (i in 1:ncol(data)){
+             for(j in 1:ncol(data)){
+               for(cl in 1:k){
+ C.pool[i,j]=C.pool[i,j]+L.cov[[cl]][i,j]*(as.numeric(table(Class)[cl])-1)/(dim(dat
+               }
+             }
+           }
+
+           for(i in 1:dim(data)[1]){
+             d=c()
+             for(j in 1:dim(data)[1]){
+               d=c(d,mahalanobis(data[i,], data[j,],C.pool))
+               D[i,j]=d[j]
```

```
+             }
+           }
+
+         } else if(distance=="euclidian"){
+           D=matrix(data=NA,nrow=dim(data)[1],ncol=dim(data)[1])
+           d=c()
+           for(i in 1:dim(data)[1]){
+             d=c()
+             for(j in 1:dim(data)[1]){
+               d=c(d,as.numeric(dist(rbind(data[i,],data[j,]))))
+               D[i,j]=d[j]
+             }
+           }
+
+         } else if(distance=="gower"){
+           if(!require(cluster)){
+             install.packages("cluster")
+             library(cluster)
+           }
+           D=as.matrix(daisy(data, metric = "gower"))
+         }
+       }
+
+       if(!is.null(kernel)){
+         if(!require(kernlab)){
+           install.packages("kernlab")
+           library(kernlab)
+         }
+         D=matrix(data=NA,nrow=dim(data)[1],ncol=dim(data)[1])
+         d=c()
+         if(kernel=="gaussian"){
+           K.rbfdot<-function(x,y){
+             rbf <- rbfdot(sigma = sigma)
+             dist = 2*(1-rbf(x,y))
+             sqrt(dist)
+           }
+           for(i in 1:dim(data)[1]){
+             d=c()
+             for(j in 1:dim(data)[1]){
```

```
+              d=c(d,K.rbfdot(data[i,],data[j,]))
+              D[i,j]=d[j]
+            }
+          }
+        } else if(kernel=="polynomial"){
+          K.polydot<-function(x,y){
+            lap <- polydot(degree = degree, scale = 1, offset = 1)
+            dist = lap(x,x)+lap(y,y)-2*lap(x,y)
+            sqrt(dist)
+          }
+          for(i in 1:dim(data)[1]){
+            d=c()
+            for(j in 1:dim(data)[1]){
+              d=c(d,K.polydot(data[i,],data[j,]))
+              D[i,j]=d[j]
+            }
+          }
+        } else if(kernel=="laplacian"){
+          K.laplacedot<-function(x,y){
+            lap <- laplacedot(sigma = sigma)
+            dist = 2*(1-lap(x,y))
+            sqrt(dist)
+          }
+          for(i in 1:dim(data)[1]){
+            d=c()
+            for(j in 1:dim(data)[1]){
+              d=c(d,K.laplacedot(data[i,],data[j,]))
+              D[i,j]=d[j]
+            }
+          }
+        }
+
+       comb = function(n, x) {
+         return(factorial(n) / (factorial(x) * factorial(n-x)))
+       }
+
+       ind<-function(x,j){
+         delta=0;
```

```
+         if(x%in%g[,j]){delta=1}
+         delta
+       }
+
+       h=matrix(data=0,nrow=dim(data)[1],ncol=k)
+       g=matrix(data=0,nrow=dim(data)[1],ncol=k)
+       s<-rep(0,k)
+       for(c in 1:k){
+         for(i in 1:dim(data)[1]){
+           if(Class[i]==c){g[i,c]=i; s[c]<-s[c]+1}
+         }
+       }
+       #S = prod(comb(s,lambda))
+
+       row=1
+       R=rep(0,k)
+       ER=rep(0,k)
+       mat=matrix(data=0,nrow=k,ncol=k)
+       while(row<=(dim(h)[1])){
+         h=matrix(data=0,nrow=dim(data)[1],ncol=k)
+         col=1
+         while(col<=k){
+           for(i in 1:dim(h)[1]){
+             for(j in which(g[,col]!=0)){
+               if(D[row,i]<D[row,j]){h[i,col]=h[i,col]+1}
+             }
+           }
+           col<-col+1
+         }
+
+         prod=0
+         vec.q<-rep(0,k)
+         col=1
+
+         while(col<=k){
+           for(i in which(g[,col]!=0)){
+             vec.k<-rep(0,k)
+             for(j in 1:dim(h)[2]){
+               if(h[i,j]>=lambda-ind(i,j)){vec.k[j]=comb(h[i,j],lambda-ind(i,j))}
```

```
+             }
+
+             if(sum(vec.k==0)==0){
+               #if(sum(vec.k==0)>0){vec.k[which(vec.k==0)]=1}
+               prod = prod+prod(vec.k)
+             }
+
+           }
+           vec.q[col]=prod; prod=0
+           col<-col+1
+         }
+
+         j=which.max(vec.q)
+         if(sum(vec.q[j]==vec.q[-j],na.rm = T)!=0){
+           w = which(vec.q[j]==vec.q)
+           j=sample(w,1)
+         }
+
+         R[j]=R[j]+1
+
+         if(row%in%g[,j]){ER[j]=ER[j]+1}
+         else{
+           for(f in (1:k)[-j]){
+             if(row%in%g[,f]){mat[j,f]=mat[j,f]+1}
+           }
+         }
+
+         row<-row+1
+       }
+       cm=diag(ER)+mat
+       if(lev){dimnames(cm)=cm.levels}
+       if(missing(prior)){prior="ni/N"}
+
+       if(class(prior)=="character"){
+
+         if(prior=="ni/N"){
+           pi=as.numeric(table(Class))/dim(data)[1]
+         } else if(prior=="1/k"){pi=rep(1/k,k)}
+
```

```
+         pER=matrix(ncol=k,nrow=k)
+         pRE=matrix(ncol=k,nrow=k)
+         for(i in 1:k){
+           for(j in 1:k){
+             pRE[i,j]=cm[i,j]/sum(cm[,j])
+           }
+         }
+         R.scale = numeric(k)
+         for(i in 1:k){
+           for(j in 1:k){
+             R.scale[i] = R.scale[i] + pRE[i,j]*pi[j]
+           }
+         }
+         for(i in 1:k){
+           for(j in 1:k){
+             pER[i,j]=(pRE[i,j]*pi[j])/R.scale[i]
+           }
+         }
+         assignment = numeric(k)
+         class.assign = numeric(k)
+         for(i in 1:k){
+           assignment[i] = max(pER[i,])
+           class.assign[i]=which.max(pER[i,])
+         }
+       }
+
+       if(class(prior)=="numeric"){
+         pER=matrix(ncol=k,nrow=k)
+         pRE=matrix(ncol=k,nrow=k)
+         for(i in 1:k){
+           for(j in 1:k){
+             pRE[i,j]=cm[i,j]/sum(cm[,j])
+             pRE[i,j]=pRE[i,j]*prior[j]
+           }
+         }
+
+         for(i in 1:k){
+           for(j in 1:k){
+             pER[i,j]=pRE[i,j]/sum(pRE[i,])
```

```
+           }
+         }
+         assignment = numeric(k)
+         class.assign = numeric(k)
+         for(i in 1:k){
+           assignment[i] = max(pER[i,])
+           class.assign[i]=which.max(pER[i,])
+         }
+       }
+
+
+
+       if(missing(info.pred)||info.pred=="no"){
+         return(list(input=input,s=s,lambda=lambda,Ri=R,
+                     Confusion_matrix=cm,prob_post=pER,
+                     assignment=assignment,class.assign=class.assign))
+
+       } else {return(list(input=input,D=D,k=k,g=g,comb=comb,ind=ind,
+                           K.rbfdot=K.rbfdot,K.laplacedot=K.laplacedot,K.polydot=K.
+                           C.pool=C.pool,Ri=R,Confusion_matrix=cm,
+                           assignment=assignment,class.assign=class.assign))}
+
+     }
+
+     isolate({
+       df <- filedata()
+       if (is.null(df)) return(NULL)
+       if(input$algorithm=="MDP"){
+         if(input$independents=="totes"){
+           m=mdp(Class = df[,input$dependent],data = df[,!colnames(df)%in%input$dep
+                 lambda = input$lambda,distance = input$distance, prior = input$pri
+           m[-c(1)]
+         } else{
+           m=mdp(Class = df[,input$dependent],data = df[,input$independents],
+                 lambda = input$lambda,distance = input$distance, prior = input$pri
+           m[-c(1)]
+         }
+       } else if(input$algorithm=="KMDP"){
+         if(input$independents=="totes"){
```

```
+           m=mdp(Class = df[,input$dependent],data = df[,!colnames(df)%in%input$dep
+                 lambda = input$lambda,kernel=input$kernel,sigma =input$sigma , pri
+           m[-c(1)]
+         } else{
+           m=mdp(Class = df[,input$dependent],data = df[,input$independents],
+                 lambda = input$lambda,kernel=input$kernel,sigma =input$sigma, pric
+           m[-c(1)]
+         }
+       }
+
+
+
+     })
+
+   })
+
+   output$summary <- renderPrint({
+     summary(filedata())
+   })
+   output$ui.action <- renderUI({
+     if (is.null(input$file1)) return()
+     actionButton("action", "Run")
+   })
+   output$table <- renderTable({
+     if(is.null(filedata())){return ()}
+     head(filedata())
+   })
+   output$tb<-renderUI({
+
+       tabsetPanel(tabPanel("Data",tableOutput("table")),
+                   tabPanel("Resultats",verbatimTextOutput("contents")))
+   })
+ })
```

A.3.2 App 2

```
> # ui.R
> library(shiny)
> shinyUI(fluidPage(
```

```
+   titlePanel("MDP for validation"),
+   sidebarLayout(
+     sidebarPanel(
+       fileInput('file1', 'Seleccioni un arxiu CSV',
+                 accept=c('text/csv',
+                          'text/comma-separated-values,text/plain',
+                          '.csv')),
+
+       tags$hr(),
+       checkboxInput(inputId = 'header', label = 'Header', value = FALSE),
+       radioButtons(inputId = 'sep',label = 'Separator',
+                    choices = c(Comma=',',Semicolon=';',Tab='\t',Space=''),selected
+
+       uiOutput("dependent"),
+       uiOutput("independents"),
+
+
+       tags$hr(),
+       selectInput("validation.method", "Esculli un metode de validacio:",
+                   choices = c("cv", "repeatedcv","LOOCV")),
+       numericInput("k.fold", "k-Fold Cross validation, Esculli k:",min = 2,max = 1
+       selectInput("Repeats", "Esculli repticions si ho creu necessari:",
+                   choices = c(2, 5, 10)),
+       selectInput("algorithm", "Esculli un dels dos algorismes:",
+                   choices = c("MDP", "KMDP")),
+       uiOutput("distances"),
+       sliderInput("lambda", "lambda:",
+                   min = 1, max = 10, value = c(1,2)),
+       textInput('sigma.g', 'Entri un Gaussian sigma vector (comma delimited)', "1e
+       textInput('sigma.l', 'Entri un Laplacian sigma vector (comma delimited)', "1
+
+       selectInput("prior", "Esculli una distribucio a priori:",
+                   choices = c("ni/N", "1/k")),
+       uiOutput('ui.action') # instead of conditionalPanel
+     ),
+     mainPanel(
+       uiOutput("tb"),
+       p("Aquesta app valida el MDP per k-fold cross validation.")
+
```

```
+     )
+   )
+ ))
>

> # server.R
> library(shiny)
> shinyServer(function(input, output) {
+
+   filedata <- reactive({
+     infile <- input$file1
+     if (is.null(infile)){
+       return(NULL)
+     }
+     read.csv(infile$datapath, header = input$header, sep = input$sep)
+   })
+
+
+   output$sum<-renderTable({
+     if(is.null(filedata())){return ()}
+     summary(filedata())
+   })
+
+   output$dependent <- renderUI({
+     df <- filedata()
+     if (is.null(df)) return(NULL)
+     items=names(df)
+     names(items)=items
+     selectInput("dependent","Seleccioni la variable resposta:",items)
+   })
+   output$distances <- renderUI({
+     di=c("euclidean", "gower", "mahalanobis")
+     selectInput("distances","Seleccioni la distancia mes oportuna:",
+                 di,multiple=TRUE)
+   })
+   output$nrows <- reactive({
+     nrow(filedata())
+   })
+
```

```
+   output$independents <- renderUI({
+     df <- filedata()
+     if (is.null(df)) return(NULL)
+     items=names(df)
+     names(items)=items
+     items=items[!items==input$dependent]
+     items=c("totes",items)
+     selectInput("independents","Seleccioni les variables independents:",items,mult
+   })
+
+   output$contents <- renderPrint({
+     input$action
+     if(!require(e1071)){
+       install.packages("e1071")
+       library(e1071)
+     }
+     mdp<-function(Class,data,lambda,distance=NULL,prior=NULL,info.pred=NULL,kernel
+       input=mget(names(formals()),sys.frame(sys.nframe()))
+       if(is.numeric(Class)){Class=as.numeric(Class); lev=FALSE
+       } else { lev=TRUE
+       if(!is.factor(Class)){
+         Class=as.matrix(Class)
+         Class=as.factor(Class)
+       }
+       cm.levels=list(levels(Class),levels(Class))
+       Class=as.numeric(Class)
+       }
+       data = as.matrix(data)
+       k=nrow(table(Class))
+       K.laplacedot=NULL; K.polydot=NULL; K.rbfdot=NULL
+       C.pool=NULL
+       if(missing(distance) && missing(kernel)){distance="mahalanobis"}
+
+       if(!is.null(distance)){
+         if(distance=="mahalanobis"){
+           D=matrix(data=0,nrow=dim(data)[1],ncol=dim(data)[1])
+           Data=data.frame(data); Data$class=Class
+           d=c()
+           L.cov=vector("list",k)
```

```
+           for(cl in 1:k){
+             L.cov[[cl]]=cov(subset(Data,Data$class==cl,select = -c(class)))
+           }
+           C.pool=matrix(0,nrow=dim(data)[2],ncol=dim(data)[2])
+           for (i in 1:ncol(data)){
+             for(j in 1:ncol(data)){
+               for(cl in 1:k){
+ C.pool[i,j]=C.pool[i,j]+L.cov[[cl]][i,j]*(as.numeric(table(Class)[cl])-1)/(dim(dat
+               }
+             }
+           }
+
+           for(i in 1:dim(data)[1]){
+             d=c()
+             for(j in 1:dim(data)[1]){
+               d=c(d,mahalanobis(data[i,], data[j,],C.pool))
+               D[i,j]=d[j]
+             }
+           }
+
+         } else if(distance=="euclidean"){
+           D=matrix(data=NA,nrow=dim(data)[1],ncol=dim(data)[1])
+           d=c()
+           for(i in 1:dim(data)[1]){
+             d=c()
+             for(j in 1:dim(data)[1]){
+               d=c(d,as.numeric(dist(rbind(data[i,],data[j,]))))
+               D[i,j]=d[j]
+             }
+           }
+
+         } else if(distance=="gower"){
+           if(!require(cluster)){
+             install.packages("cluster")
+             library(cluster)
+           }
+           D=as.matrix(daisy(data, metric = "gower"))
+         }
+       }
```

```
+
+       if(!is.null(kernel)){
+         if(!require(kernlab)){
+           install.packages("kernlab")
+           library(kernlab)
+         }
+         D=matrix(data=NA,nrow=dim(data)[1],ncol=dim(data)[1])
+         d=c()
+         if(substr(kernel, 1, 1)=="g"){
+           sigma=as.numeric(substr(kernel, 5, nchar(kernel)))
+
+           K.rbfdot<-function(x,y){
+             rbf <- rbfdot(sigma = sigma)
+             dist = 2*(1-rbf(x,y))
+             sqrt(dist)
+           }
+           for(i in 1:dim(data)[1]){
+             d=c()
+             for(j in 1:dim(data)[1]){
+               d=c(d,K.rbfdot(data[i,],data[j,]))
+               D[i,j]=d[j]
+             }
+           }
+         } else if(substr(kernel, 1, 1)=="p"){
+           degree=as.numeric(substr(kernel, 5, nchar(kernel)))
+           K.polydot<-function(x,y){
+             lap <- polydot(degree = degree, scale = 1, offset = 1)
+             dist = lap(x,x)+lap(y,y)-2*lap(x,y)
+             sqrt(dist)
+           }
+           for(i in 1:dim(data)[1]){
+             d=c()
+             for(j in 1:dim(data)[1]){
+               d=c(d,K.polydot(data[i,],data[j,]))
+               D[i,j]=d[j]
+             }
+           }
+         } else if(substr(kernel, 1, 1)=="l"){
+           sigma=as.numeric(substr(kernel, 5, nchar(kernel)))
```

```
+
+           K.laplacedot<-function(x,y){
+             lap <- laplacedot(sigma = sigma)
+             dist = 2*(1-lap(x,y))
+             sqrt(dist)
+           }
+           for(i in 1:dim(data)[1]){
+             d=c()
+             for(j in 1:dim(data)[1]){
+               d=c(d,K.laplacedot(data[i,],data[j,]))
+               D[i,j]=d[j]
+             }
+           }
+         }
+       }
+
+       comb = function(n, x) {
+         return(factorial(n) / (factorial(x) * factorial(n-x)))
+       }
+
+       ind<-function(x,j){
+         delta=0;
+         if(x%in%g[,j]){delta=1}
+         delta
+       }
+
+       h=matrix(data=0,nrow=dim(data)[1],ncol=k)
+       g=matrix(data=0,nrow=dim(data)[1],ncol=k)
+       s<-rep(0,k)
+       for(c in 1:k){
+         for(i in 1:dim(data)[1]){
+           if(Class[i]==c){g[i,c]=i; s[c]<-s[c]+1}
+         }
+       }
+       #S = prod(comb(s,lambda))
+
+       row=1
+       R=rep(0,k)
+       ER=rep(0,k)
```

```
+       mat=matrix(data=0,nrow=k,ncol=k)
+       while(row<=(dim(h)[1])){
+         h=matrix(data=0,nrow=dim(data)[1],ncol=k)
+         col=1
+         while(col<=k){
+           for(i in 1:dim(h)[1]){
+             for(j in which(g[,col]!=0)){
+               if(D[row,i]<D[row,j]){h[i,col]=h[i,col]+1}
+             }
+           }
+           col<-col+1
+         }
+
+         prod=0
+         vec.q<-rep(0,k)
+         col=1
+
+         while(col<=k){
+           for(i in which(g[,col]!=0)){
+             vec.k<-rep(0,k)
+             for(j in 1:dim(h)[2]){
+               if(h[i,j]>=lambda-ind(i,j)){vec.k[j]=comb(h[i,j],lambda-ind(i,j))}
+             }
+
+             if(sum(vec.k==0)==0){
+               prod = prod+prod(vec.k)
+             }
+
+           }
+           vec.q[col]=prod; prod=0
+           col<-col+1
+         }
+
+         j=which.max(vec.q)
+         if(sum(vec.q[j]==vec.q[-j],na.rm = T)!=0){
+           w = which(vec.q[j]==vec.q)
+           j=sample(w,1)
+         }
+
```

```
+         R[j]=R[j]+1
+
+         if(row%in%g[,j]){ER[j]=ER[j]+1}
+         else{
+           for(f in (1:k)[-j]){
+             if(row%in%g[,f]){mat[j,f]=mat[j,f]+1}
+           }
+         }
+
+         row<-row+1
+       }
+       cm=diag(ER)+mat
+       if(lev){dimnames(cm)=cm.levels}
+       if(missing(prior)){prior="ni/N"}
+
+       if(class(prior)=="character"){
+
+         if(prior=="ni/N"){
+           pi=as.numeric(table(Class))/dim(data)[1]
+         } else if(prior=="1/k"){pi=rep(1/k,k)}
+
+         pER=matrix(ncol=k,nrow=k)
+         pRE=matrix(ncol=k,nrow=k)
+         for(i in 1:k){
+           for(j in 1:k){
+             pRE[i,j]=cm[i,j]/sum(cm[,j])
+           }
+         }
+         R.scale = numeric(k)
+         for(i in 1:k){
+           for(j in 1:k){
+             R.scale[i] = R.scale[i] + pRE[i,j]*pi[j]
+           }
+         }
+         for(i in 1:k){
+           for(j in 1:k){
+             pER[i,j]=(pRE[i,j]*pi[j])/R.scale[i]
+           }
+         }
```

```
+         assignment = numeric(k)
+         class.assign = numeric(k)
+         for(i in 1:k){
+           assignment[i] = max(pER[i,])
+           class.assign[i]=which.max(pER[i,])
+         }
+       }
+
+       if(class(prior)=="numeric"){
+         pER=matrix(ncol=k,nrow=k)
+         pRE=matrix(ncol=k,nrow=k)
+         for(i in 1:k){
+           for(j in 1:k){
+             pRE[i,j]=cm[i,j]/sum(cm[,j])
+             pRE[i,j]=pRE[i,j]*prior[j]
+           }
+         }
+
+         for(i in 1:k){
+           for(j in 1:k){
+             pER[i,j]=pRE[i,j]/sum(pRE[i,])
+           }
+         }
+         assignment = numeric(k)
+         class.assign = numeric(k)
+         for(i in 1:k){
+           assignment[i] = max(pER[i,])
+           class.assign[i]=which.max(pER[i,])
+         }
+       }
+
+
+
+       if(missing(info.pred)||info.pred=="no"){
+         return(list(s=s,lambda=lambda,Ri=R,Confusion_matrix=cm,
+                     prob_post=pER,assignment=assignment,class.assign=class.assign)
+
+       } else if(lev){return(list(input=input,D=D,k=k,g=g,comb=comb,ind=ind,
+                                  K.rbfdot=K.rbfdot,K.laplacedot=K.laplacedot,
```

```
+                                   K.polydot=K.polydot,
+                                   C.pool=C.pool,Ri=R,Confusion_matrix=cm,
+                                   cm.levels=cm.levels,assignment=assignment,
+                                   class.assign=class.assign))
+       } else{
+         return(list(input=input,D=D,k=k,g=g,comb=comb,ind=ind,
+                     K.rbfdot=K.rbfdot,K.laplacedot=K.laplacedot
+                     ,K.polydot=K.polydot,
+                     C.pool=C.pool,Ri=R,Confusion_matrix=cm,assignment=assignment,
+                     class.assign=class.assign))
+       }
+
+     }
+     predict.mdp<-function(modelFit,newdata){
+       data=as.matrix(modelFit$input[["data"]])
+       newdata=as.matrix(newdata)
+       k=modelFit$k
+       g=modelFit$g
+       D=modelFit$D
+       lambda=modelFit$input[["lambda"]]
+       comb=modelFit$comb
+       ind=modelFit$ind
+       class.assign=modelFit$class.assign
+       start=dim(D)[1]
+       vec.new=(start+1):(start+dim(newdata)[1])
+       C.pool=modelFit$C.pool
+
+       if(is.null(modelFit$input[["distance"]]) && is.null(modelFit$input[["kernel"
+         modelFit$input[["distance"]]="mahalanobis"
+       } else if(!is.null(modelFit$input[["distance"]])){
+         distance=modelFit$input[["distance"]]
+         if(distance=="mahalanobis"){
+           for(i in 1:dim(newdata)[1]){
+             d=c()
+             D=cbind(D,c(0))
+             D=rbind(D,c(0))
+             for(j in 1:dim(data)[1]){
+               d=c(d,mahalanobis(newdata[i,], data[j,],C.pool))
+               D[vec.new[i],j]=d[j]
```

```
+             }
+           }
+         } else if(distance=="euclidean"){
+           for(i in 1:dim(newdata)[1]){
+             d=c()
+             D=cbind(D,c(0))
+             D=rbind(D,c(0))
+             for(j in 1:dim(data)[1]){
+               d=c(d,as.numeric(dist(rbind(newdata[i,], data[j,]))))
+               D[vec.new[i],j]=d[j]
+             }
+           }
+         } else if(distance=="gower"){
+           if(!require(cluster)){
+             install.packages("cluster")
+             library(cluster)
+           }
+           data2=rbind(data,newdata)
+           D=as.matrix(daisy(data2, metric = "gower"))
+         }
+       } else if(!is.null(modelFit$input[["kernel"]])){
+         if(!require(kernlab)){
+           install.packages("kernlab")}
+         kernel=modelFit$input[["kernel"]]
+         if(substr(kernel, 1, 1)=="g"){
+           sigma=as.numeric(substr(kernel, 5, nchar(kernel)))
+           K.rbfdot<-function(x,y){
+             rbf <- rbfdot(sigma = sigma)
+             dist = 2*(1-rbf(x,y))
+             sqrt(dist)
+           }
+           for(i in 1:dim(newdata)[1]){
+             d=c()
+             D=cbind(D,c(0))
+             D=rbind(D,c(0))
+             for(j in 1:dim(data)[1]){
+               d=c(d,K.rbfdot(newdata[i,], data[j,]))
+               D[vec.new[i],j]=d[j]
+             }
```

```
+           }
+         } else if(substr(kernel, 1, 1)=="l"){
+           sigma=as.numeric(substr(kernel, 5, nchar(kernel)))
+           K.laplacedot<-function(x,y){
+             lap <- laplacedot(sigma = sigma)
+             dist = 2*(1-lap(x,y))
+             sqrt(dist)
+           }
+           for(i in 1:dim(newdata)[1]){
+             d=c()
+             D=cbind(D,c(0))
+             D=rbind(D,c(0))
+             for(j in 1:dim(data)[1]){
+               d=c(d,K.laplacedot(newdata[i,], data[j,]))
+               D[vec.new[i],j]=d[j]
+             }
+           }
+
+         } else if(substr(kernel, 1, 1)=="p"){
+           degree=as.numeric(substr(kernel, 5, nchar(kernel)))
+           K.polydot<-function(x,y){
+             lap <- polydot(degree = degree, scale = 1, offset = 1)
+             dist = lap(x,x)+lap(y,y)-2*lap(x,y)
+             sqrt(dist)
+           }
+           for(i in 1:dim(newdata)[1]){
+             d=c()
+             D=cbind(D,c(0))
+             D=rbind(D,c(0))
+             for(j in 1:dim(data)[1]){
+               d=c(d,K.polydot(newdata[i,], data[j,]))
+               D[vec.new[i],j]=d[j]
+             }
+           }
+
+         }
+       }
+
+       r=numeric(dim(newdata)[1])
```

```
+       R=rep(0,k)
+       ER=rep(0,k)
+       mat=matrix(data=0,nrow=k,ncol=k)
+       t=1
+       for(row in vec.new){
+         h=matrix(data=0,nrow=start,ncol=k)
+         col=1
+         while(col<=k){
+           for(i in 1:dim(h)[1]){
+             for(j in which(g[,col]!=0)){
+               if(D[row,i]<D[row,j]){h[i,col]=h[i,col]+1}
+             }
+           }
+           col<-col+1
+         }
+
+         prod=0
+         vec.q<-rep(0,k)
+         col=1
+
+         while(col<=k){
+           for(i in which(g[,col]!=0)){
+             vec.k<-rep(0,k)
+             for(j in 1:dim(h)[2]){
+               if(h[i,j]>=lambda-ind(i,j)){vec.k[j]=comb(h[i,j],lambda-ind(i,j))}
+             }
+
+             if(sum(vec.k==0)==0){
+               prod = prod+prod(vec.k)
+             }
+           }
+           vec.q[col]=prod; prod=0
+           col<-col+1
+         }
+
+         j=which.max(vec.q)
+         if(sum(vec.q[j]==vec.q[-j],na.rm = T)!=0){
+           w = which(vec.q[j]==vec.q)
+           j=sample(w,1)
```

```
+         }
+
+         r[t]=class.assign[j]
+         t<-t+1
+       }
+       if(!is.null(modelFit$cm.levels)){
+         n=length(modelFit$cm.levels[[1]])
+         for(i in 1:n){
+           r[which(r==i)]<-modelFit$cm.levels[[1]][i]
+         }
+         r=factor(r)
+       }
+       return(r)
+     }
+     tune.mdp<-function(algorithm, Class, data, validation.method,k.fold,repeats=5,
+                        degree=1:2, sigma.g=1e-10,sigma.l=1e-10,distance=c("euclide
+                        lambda){
+       library(caret)
+       my_mod <- list(type = "Classification",
+                      library = "stats",
+                      loop = NULL)
+
+       my_mod$label <- "Minimum Distance Probability"
+
+       my_mod$sort <- NULL
+       my_mod$levels<-NULL
+
+       if(validation.method=="LOOCV"){
+         fitControl <- trainControl(method = "LOOCV")
+
+       } else {
+         fitControl <- trainControl(method = validation.method,number = k.fold,
+                                    repeats = repeats)
+
+       }
+
+
+
+       if(algorithm=="MDP"){
```

```
+         my_mod$parameters <- data.frame(
+           parameter = c("lambda","distance"),
+           class = c("numeric", "character"),
+           label = c("sample_length","distance_type"))
+
+         ## The grid Element
+         if("mahalanobis"%in% distance){
+           my_mod$grid <- function(x, y, len = NULL){expand.grid(lambda=lambda,
+                                       distance=c("euclidean","gower","mahalanobi
+           mygrid<-expand.grid(lambda=lambda,distance=c("euclidean","gower","mahala
+         } else {
+           my_mod$grid <- function(x, y, len = NULL){expand.grid(lambda=lambda,
+                                            distance=c("euclidean,gower"))}
+           mygrid<-expand.grid(lambda=lambda,distance=c("euclidean","gower"))
+         }
+
+         ## The fit Element
+
+         my_mod$fit <- function(x, y, wts, param, lev, last, classProbs, ...){
+           mdp(Class=y,data=x,lambda=param$lambda,prior="1/k",
+               distance=as.character(param$distance),info.pred ="yes")
+         }
+
+         ## The predict Element
+
+         mdpPred <- function(modelFit, newdata, preProc = NULL, submodels = NULL)
+           predict.mdp(modelFit, newdata)
+         my_mod$predict <- mdpPred
+
+         mdpProb <- function(modelFit, newdata, preProc = NULL, submodels = NULL)
+           predict.mdp(modelFit, newdata)
+         my_mod$prob <- mdpPred
+         entreno=train(x=data, y = factor(Class),method = my_mod,trControl = fitCon
+         entreno
+
+       } else {
+         my_mod$parameters <- data.frame(
+           parameter = c("lambda","kernel"),
+           class = c("numeric", "character"),
```

```
+           label = c("sample_length", "kernel_type"))
+
+         ## The grid Element
+         vec.degree  = paste0("p.d=",degree)
+         vec.sigma.l = paste0("l.s=",sigma.l)
+         vec.sigma.g = paste0("g.s=",sigma.g)
+
+         my_mod$grid <- function(x, y, len = NULL){
+
+           expand.grid(lambda=lambda,kernel=c(vec.degree,vec.sigma.g,vec.sigma.l))}
+         mygrid<-expand.grid(lambda=lambda,kernel=c(vec.degree,vec.sigma.g,vec.sigm
+
+
+         ## The fit Element
+
+         my_mod$fit <- function(x, y, wts, param, lev, last, classProbs, ...){
+           mdp(Class=y,data=x,lambda=param$lambda,prior="1/k",
+               kernel=as.character(param$kernel),info.pred ="yes")
+         }
+
+         ## The predict Element
+
+         mdpPred <- function(modelFit, newdata, preProc = NULL, submodels = NULL)
+           predict.mdp(modelFit, newdata)
+         my_mod$predict <- mdpPred
+
+         mdpProb <- function(modelFit, newdata, preProc = NULL, submodels = NULL)
+           predict.mdp(modelFit, newdata)
+         my_mod$prob <- mdpPred
+         entreno=train(x=data, y = factor(Class),method = my_mod,trControl = fitCon
+
+         entreno
+       }
+
+
+
+     }
+
+
```

```
+     isolate({
+       df <- filedata()
+       if (is.null(df)) return(NULL)
+       if(input$independents=="totes"){
+         tune.mdp(algorithm = input$algorithm, Class=df[,input$dependent],
+                  data = df[,!colnames(df)%in%input$dependent], k.fold=input$k.fold
+                  validation.method =input$validation.method, distance = input$dist
+                  sigma.g=as.numeric(unlist(strsplit(input$sigma.g,","))),
+                  sigma.l=as.numeric(unlist(strsplit(input$sigma.l,","))),
+                  repeats=input$Repeats,lambda = input$lambda[1]:input$lambda[2])
+       } else {
+         tune.mdp(algorithm = input$algorithm, Class=df[,input$dependent],data = di
+                  k.fold=input$k.fold,
+                  validation.method =input$validation.method, distance = input$dist
+                  sigma.g=input$sigma.g[1]:input$sigma.g[2],sigma.l=input$sigma.l[1
+                  repeats=input$Repeats,lambda = input$lambda[1]:input$lambda[2])
+       }
+
+     })
+   })
+
+   output$summary <- renderPrint({
+     summary(filedata())
+   })
+   output$ui.action <- renderUI({
+     if (is.null(input$file1)) return()
+     actionButton("action", "Run")
+   })
+   output$table <- renderTable({
+     if(is.null(filedata())){return ()}
+     head(filedata())
+   })
+   output$tb<-renderUI({
+
+     tabsetPanel(tabPanel("Data",tableOutput("table")),
+                 tabPanel("Resultats",
+                          verbatimTextOutput("contents")))
+   })
+ })
```

>