Universitat Politècnica de Catalunya

Facultat d'Informàtica de Barcelona

Especialitat de Computació

Disseny, implementació i estudi de funcions de kernel per a vectors de variables no reals

Treball de Final de Grau

Memòria del projecte

Arnau Arqué Martínez

Director: Lluís A. Belanche Muñoz

22 de juny de 2021

Resum

Les funcions de kernel han estat objecte d'estudi durant els últims anys. Se n'han enunciat i desenvolupat per a diferents tipus de variables com ara numèriques, difuses o categòriques, entre d'altres. Tot i així, en la literatura actual no trobem cap funció de kernel que permeti tractar vectors de dades amb atributs de múltiples tipologies. En aquest projecte comencem fent un estat de l'art sobre funcions de kernel per a diferents tipus de dades. Seguidament, enunciem el kernel d'agregació, que soluciona la problemàtica esmentada anteriorment i l'implementem a la llibreria LIBSVM. Finalment, resolem un seguit de problemes de classificació per testar la nova versió del software i el kernel d'agregació, tot obtenint resultats satisfactoris.

Resumen

Las funciones de kernel han sido objeto de estudio durante los últimos años. Se han enunciado para diferentes tipos de variables como, por ejemplo, numéricas, difusas o categóricas, entre otras. Aun así, en la literatura actual no encontramos ninguna función de kernel que permita tratar vectores de datos con atributos de múltiples tipologías. En este proyecto empezamos haciendo un estado del arte sobre funciones de kernel para diferentes tipos de datos. A continuación, enunciamos el kernel de agregación, que soluciona la problemática mencionada anteriormente y lo implementamos en la librería LIBSVM. Para concluir, resolvemos un conjunto de problemas de clasificación para testar la nueva versión del software y el kernel de agregación, obteniendo así unos resultados satisfactorios.

Abstract

Kernel functions have been a subject of study for the past few years. They have been stated for different types of variables such as, for example, numerical, fuzzy or categorical, among others. Even so, in the current literature we do not find any kernel function that allows us to treat vectors of data with attributes of multiple typologies. In this project we started by doing a state of the art on kernel functions for different types of data. Next, we propose the aggregation kernel, which solves the aforementioned problem and we implement it in the LIBSVM library. To conclude, we solve a set of classification problems to test the new version of the software and the aggregation kernel, thus obtaining satisfactory results.

Agraïments

M'agradaria agrair al director d'aquest projecte, en Lluís A. Belanche, per oferir-me l'oportunitat d'estudiar i fer recerca en un tema tan interessant com són les funcions de kernel, les *support vector machines* i, en general, l'aprenentatge automàtic. Gràcies per guiar-me al llarg d'aquests últims mesos i ajudar-me a assolir els objectius que ens havíem proposat.

M'agradaria també agrair a la Biblioteca Rector Gabriel Ferraté la prestació de serveis de documentació i informació que han fet possible el desenvolupament d'aquest projecte.

Finalment, agrair a la meva família l'incondicional suport que m'han atorgat aquests últims mesos. Ha estat un camí llarg i ple d'obstacles, però tot esforç té la seva recompensa. Aquesta n'és la prova. Gràcies de tot cor.

Índex

Índex de figures

Índex de taules

Índex de codis

1 Introducció

El projecte «*Disseny, implementació i estudi de funcions de kernel per a vectors de variables no reals*» s'emmarca en l'àmbit del Treball de Final de Grau del Grau en Enginyeria Informàtica en l'especialitat de Computació. Tot plegat es duu a terme a la Universitat Politècnica de Catalunya i, més concretament, a la Facultat d'Informàtica de Barcelona.

En aquest capítol pretenem oferir un escrit introductori al projecte que tot just volem iniciar. Començarem tractant alguns conceptes previs que considerem adients per a la correcta comprensió del projecte. Seguidament, durem a terme una anàlisi i identificació del problema que ens ocupa i motiu pel qual desenvoluparem aquest projecte. A més a més, justificarem l'elecció de la temàtica del projecte, tot analitzant les possibles solucions existents. Per acabar, establirem els objectius i subobjectius del projecte, així com els requisits necessaris per dur-lo a terme. També esmentarem quins són els actors implicats en el treball. Per acabar, comentarem i analitzarem els possibles obstacles i riscs que es poden donar al llarg de l'etapa de desenvolupament.

1.1 Conceptes previs

En aquest apartat, farem un breu viatge des de l'aparició del concepte d'intel·ligència, passant pel desenvolupament de la intel·ligència artificial i l'aprenentatge automàtic. També explicarem en què consisteixen els mètodes kernel i, més concretament, les *support vector machines*. Acabarem tractant i desenvolupant el concepte de les funcions de kernel.

1.1.1 Breu història del concepte d'intel·ligència

El concepte d'intel·ligència ha estat motiu de discussió i debat al llarg de la història del raonament i pensament crític humà. Tant és així que avui dia encara no s'ha arribat a un consens per establir una definició formal única del terme.

Una de les referències més antigues al concepte d'intel·ligència data del període de la Grècia clàssica (Segles V i VI a.C.). Plató (427-347 a.C.), per exemple, determina que un dels aspectes principals de la intel·ligència és la capacitat per aprendre [1]. A mesura que avancem en el temps sorgeixen nous corrents de pensament i, com és d'esperar, les idees clàssiques comencen a ser desestimades. Arribats a l'Edat Moderna (Segles XV-XVIII d.C.), Kant (1724-1804) defineix la intel·ligència com a una facultat de la cognició i determina que aquesta està composta per l'enteniment, el seny i la cognició [1].

No és difícil adonar-se que com més ens apropem a l'actualitat la definició d'intel·ligència deixa de ser un afer exclusiu de la filosofia. D'aquesta manera comença a estudiar-se des del punt de vista científic, en gran part de la mà de la psicologia.

1.1.2 Intel·ligència Artificial i Aprenentatge Automàtic

L'any 1936 Alan Turing publica «Els nombres computables, amb una aplicació a l'*Entscheidungsproblem*» [2]. En aquest article, el científic britànic demostra que tot problema matemàtic es pot representar mitjançant un algorisme. A més a més, també prova l'existència de màquines (conegudes actualment com a màquines de Turing) amb la capacitat de resoldre

aquests problemes. El mateix Turing es referiria anys després a aquestes màquines com a «màquines intel·ligents» [3].

Des d'aleshores la computació avançà sense fre. Els progressos en la teoria de la computació, juntament amb les millores tecnològiques de l'època facilitaren la creació dels primers ordinadors. No és, però, fins al 1955 que apareix per primer cop el terme d'Intel·ligència Artificial (IA). John McCarthy juntament amb quatre investigadors més publicaren la proposta d'impulsar un projecte de recerca orientat a la IA de cara a l'estiu del 1956 [4]. Els anys següents es consideren l'època daurada de la IA. Els programes i aportacions que sorgeixen eren impensables anys enrere, fet que provocà un augment en la inversió de capital per part de governs i empreses privades.

El 1959, Arthur Samuel encunya el terme *machine learning* al seu article «*Some studies in machine learning using the game of checkers*» [5]. En aquest proposa alguns algorismes de jocs per a ordinadors basats en IA. L'aprenentatge automàtic (AA) ha evolucionat —i encara ho fa— fins a dia d'avui, tot adadptar-se a definicions que descriguin de manera més acurada aquest conjunt d'algorismes.

Avui en dia podem dir que l'AA és l'estudi d'algorismes computacionals fonamentats en evolucionar automàticament a través de l'experiència i l'ús de dades [6]. L'AA es planteja les qüestions de com construïr sistemes computacionals que evolucionin i millorin de manera automàtica a través de l'experiència i quins són els fonaments estadístics, computacionals i teòrics que permeten fer-ho [7].

Podem distingir tres tipologies diferenciades d'AA: l'aprenentatge supervisat (*supervised learning)*, que fa servir dades etiquetades amb el resultat desitjat; l'aprenentatge no supervisat (*unsupervised learning*), que treballa amb dades sense etiquetar i l'aprenentatge per reforç (*reinforcement learning*), en el qual els algorismes exploren diferents accions a dur a terme i intenten maximitzar algun tipus de recompensa.

1.1.3 *Support Vector Machines* i funcions de kernel

Les *Support Vector Machine* (SVM) són un conjunt d'algorismes basats en aprenentatge supervisat que ens permeten solucionar problemes de classificació i regressió mitjançant l'anàlisi d'un conjunt de dades. V. Vapnik i els seus col·laboradors desenvolupen el primer model als laboratoris Bell d'AT&T i publicaren, finalment, els resultats obtinguts l'any 1992 [8].

Les SVM han estat emprades en múltiples àmbits d'estudi amb resultats exitosos. Entre d'altres destaquen l'àmbit de la visió artificial, el reconeixement de caràcters, el processament de llenguatge natural o l'anàlisi de sèries temporals [9].

A l'hora de dur a terme tasques de classificació, les SVM pertanyen al grup dels *classificadors lineals*. És així perquè basen el seu funcionament a separar les dades fent ús de separadors lineals, també anomenats *hiperplans*. La separació de les dades es pot donar en l'espai original d'aquestes o en un espai transformat anomenat *feature space* (espai de característiques). Aquests espais de característiques solen ser d'alta dimensionalitat. És per això que s'hi accedeix de forma impícita fent servir el que s'anomenen *funcions de kernel* [10].

La idea general del funcionament de les SVM es basa a seleccionar un hiperplà que separi les dades de manera que equidisti dels exemples més propers de cada classe. D'aquesta manera s'aconsegueix maximitzar el marge que hi ha a cada costat de l'hiperplà. A l'hora de definir l'hiperplà, únicament es tenen en compte aquelles dades que conformen la frontera d'aquests marges.

Concretament, les mostres que recauen sobre el marge esmentat s'anomenen *support vectors* [10].

Ara bé, la tasca de trobar aquest hiperplà que maximitzi la distància entre les dades més properes d'ambdós costats no és past trivial. Per trobar l'hiperplà òptim serà necessari resoldre un problema d'optimització quadràtica amb restriccions lineals.

Una funció de kernel és una funció definida com $k : X \times X \to \mathbb{R}$. El propòsit d'aquesta funció és associar a cada parell del conjunt d'entrada X un valor real resultant del producte escalar de les imatges d'ambdós elements en un nou espai $\mathcal{F}$ que s'anomena espai de característiques (de l'anglès *feature space*). És a dir, sigui $\Phi : X \to \mathcal{F}$:

$$k(x, z) = \langle \Phi(x), \Phi(z) \rangle = \phi_1(x)\phi_1(z) + \ldots + \phi_m(x)\phi_m(z)$$

On m és la dimensionalitat del *feature space* $\mathcal{F}$ i $\phi_i(\cdot),\ i = 1, ..., m$ conformen el conjunt de funcions base de Φ, de manera que podem expressar $\Phi(\cdot) = \{\phi_1(\cdot), \ldots, \phi_m(\cdot)\}$.

Ara bé, per què estem explicant tot aquest funcionament de les funcions de kernel? Doncs bé, en algunes ocasions, no és possible trobar un hiperplà que separi les dades en la dimensió original d'aquestes. La idea de les funcions de kernel és, doncs, trobar un hiperplà que sigui capaç de separar les mostres en l'espai de característiques amb dimensionalitat superior.

Noti's que estem parlant d'augmentar la dimensionalitat de les dades, però no estem establint cap límit. En alguns casos, fins i tot podríem arribar a accedir a *feature spaces* de dimensionalitat infinita. Com pal·liem amb aquesta problemàtica? Doncs bé, tot plegat pren sentit en saber que el problema d'optimització per trobar els vectors suport que hem comentat anteriorment no depèn de la dimensionalitat de l'espai, sinó de la cardinalitat del conjunt dels vectors suport.

Tot i així, seguim apreciant una problemàtica en tot aquest raonament. Com és possible transformar una dada de dimensió finita en una dada de dimensió infinita? La resposta a aquesta pregunta ens l'ofereix el teorema d'Aronszajn [11]. El teorema d'Aronszajn ens diu que per a tota funció $k : X \times X \to \mathbb{R}$ simètrica i semidefinida positiva, existeix un espai de Hilbert i una funció $\Phi : X \to \mathcal{F}$, tal que:

$$k(x, z) = \langle \Phi(x), \Phi(z) \rangle, \quad \forall x, z \in X$$

D'aquest teorema en podem deduir que no és necessari generar una funció de kernel a partir d'un conjunt de funcions base $\Phi(\cdot)$. Únicament cal trobar una funció que compleixi amb les condicions del teorema, *i.e.* k ha de ser simètrica i semidefinida positiva. Tot plegat significa que per avaluar el resultat d'una funció de kernel no serà necessari visitar el *feature space* i, per tant, no haurem de visitar cap hipotètic espai de dimensionalitat infinita.

Una altra manera d'interpretar les funcions de kernel és des del punt de vista de la similitud. En definitiva, el que pretenem amb l'avaluació d'aquestes funcions és determinar la *similitud* de les dades en qüestió. Aquesta informació ens permet ajustar iterativament l'hiperplà fins a arribar a una solució òptima.

1.2 Identificació del problema

Els mètodes kernel estan essent objecte d'estudi i emprats des de fa anys a causa —entre altres factors— del seu bon rendiment en la resolució de problemes amb *datasets* complexos. Com bé hem explicat en apartats anteriors, els mètodes kernel i, en conseqüència, les *Support Vector Machine*, es fonamenten en el fet d'usar una funció de kernel per valorar la similitud entre les

diferents mostres del *dataset* en qüestió. En les etapes primerenques de l'àmbit d'estudi, n'hi havia prou amb dissenyar funcions de kernel que tractessin amb variables numèriques contínues. Tot i així, els experts no trigaren a adonar-se del potencial d'aquests algorismes i varen començar a crear noves funcions de kernel per tractar conjunts de dades de diferents tipologies [12].

En l'actualitat disposem de múltiples funcions de kernel enfocades a treballar amb diferents tipologies de dades; des de funcions de kernel per a *strings* [13] fins a funcions de kernel per a imatges [14]. Ara bé, què passa quan disposem de diferents tipus de variables en un mateix conjunt de dades? La solució que s'utilitza de manera general passa per, en primer lloc, transformar totes les dades a variables numèriques. Seguidament, es fa servir una funció de kernel ja coneguda com, per exemple, la *Radial Basis Function* (RBF) per generar el model de predicció. Per acabar, es duen a terme les prediccions tenint en compte els resultats de similitud que proporcionen les funcions de kernel sobre les dades transformades. No és difícil adonar-se que aquest procediment sembla poc acurat amb relació a conservar les característiques intrínseques de cada tipus de dada. Aplicant aquest tipus de transformacions, podem estar perdent precisió a l'hora d'avaluar la similitud entre les mostres.

Tenint en compte el que hem comentat al paràgraf anterior ens podem fer la següent pregunta: no hi ha alguna manera d'evitar transformar les dades? No podríem generar una funció de kernel que sigui capaç d'avaluar la similitud entre mostres formades per diferents tipus de variables? La resposta és afirmativa.

El kernel que estudiarem i dissenyarem en aquest projecte s'anomena *kernel d'agregació* i es fonamenta en fer la mitjana els resultats de diferents crides a kernels. Tenint en compte les propietats dels kernels, veurem que la nova funció de kernel és, efectivament, un kernel. En els propers apartats estudiarem si hi ha alguna solució existent al problema que hem plantejat i justificarem l'elecció dels següents passos a seguir per implementar el nou kernel.

1.3 Justificació

Arribats a aquest punt hem identificat quin és el problema al qual pretenem donar solució. El següent pas lògic passaria per, tenint en compte aquesta informació, definir els objectius i requeriments del nostre projecte —en definitiva, l'abast del projecte—. Abans, però, és convenient que duguem a terme un estudi de les solucions existents per al nostre problema, si és que n'hi ha alguna. Seguidament, justificarem l'elecció dels passos a seguir basant-nos en l'estat de l'art que tot just haurem definit.

1.3.1 Solucions existents

En aquest apartat estudiarem les possibles solucions (parcials o totals) que existeixen en l'actualitat per al problema que hem definit anteriorment.

Tal com hem especificat en la definició del problema, durant aquest projecte pretenem treballar dissenyant i implementant funcions de kernel que tractin amb vectors de variables no reals. També hem comentat que limitarem els mètodes kernel amb els quals tractarem a les *Support Vector Machines* (SVM). Així doncs, per tal d'analitzar l'estat de l'art del problema, estudiarem les diferents llibreries i paquets de software d'aprenentatge automàtic que proporcionen els llenguatges de programació R, Python, Matlab i C++ per tractar amb mètodes kernel. D'aquesta manera, veurem quines rutines treballen amb les SVM per a cada llenguatge i si permeten abordar el problema d'emprar múltiples kernels on cadascun tracti amb diferents variables d'un mateix conjunt de dades.

R

R és un llenguatge de programació i un entorn per a la computació estadística i els gràfics [15]. El llenguatge proveeix una gran varietat de tècniques estadístiques i gràfiques com ara el modelatge lineal, la classificació o el clustering, entre d'altres. A més a més, disposa d'un sistema de paquets (*packages*) que permet a l'usuari emprar col·leccions de funcions i *datasets* creats per la comunitat. Amb relació a les SVM hem identificat dos paquets diferents:

e1071 El paquet `e1071` comprèn un conjunt de funcions creades pel Departament d'Estadística de la Universitat Tècnica de Viena [16]. Entre els seus àmbits d'acció hi ha les SVM. Si bé és cert que disposa d'una gran quantitat de mètodes, el que ens interessa s'anomena `svm`. Aquest s'encarrega d'entrenar models d'SVM. Amb relació als kernels, permet determinar quin kernel emprar d'entre els de tipus lineal, polinòmic, *radial basis* i *sigmoid*. Permet tractar problemes de classificació *multiclasse*, però no dóna suport als problemes amb múltiples kernels. Val a remarcar que el codi font està programat en C++ i fa servir la llibreria LIBSVM.

kernlab El paquet `kernlab` pren el nom del títol «*Kernel-Based Machine Learning Lab*». Es tracta d'un conjunt de funcions d'aprenentatge automàtic basades en kernels [17]. Com també passava amb l'anterior paquet, disposa d'un mètode (`ksvm`) que s'encarrega d'entrenar models d'SVM. Amb relació als kernels, disposa d'una varietat més àmplia de mètodes que permeten crear-los i tractar-los. Per exemple, la funció `dots` facilita la generació de kernels de tipus gaussians, polinòmics o laplacians, entre d'altres. També permet emprar com a kernel qualsevol altra funció que implementi un producte escalar de la classe kernel. Aquest fet facilita que els usuaris puguin fer servir kernels especials per a determinats *datasets*. Tot i així, novament ens trobem que no permet tractar problemes de múltiples kernels de manera nativa i, altre cop, el paquet està basat en la llibreria LIBSVM.

Python

Python és un llenguatge de programació de propòsit general i alt nivell [18]. Durant els últims anys ha agafat força en l'àmbit de l'aprenentatge automàtic, malgrat que moltes de les tasques que requereixen eficiència computacional i velocitat estan implementades en C o C++.

De la mateixa manera que R, Python disposa d'un sistema de paquets (*packages*). En l'àmbit de l'aprenentatge automàtic destaca `scikit-learn`. Aquest paquet integra eines simples i eficients per l'anàlisi de dades predictiva [19]. Com és d'esperar, també dóna suport a les SVM. Permet entrenar models d'SVM i dur a terme prediccions fent servir el model generat. També dóna llibertat per triar quin kernel emprar i, a més a més, permet usar kernels definits per l'usuari. De tota manera, el paquet fa servir les llibreries LIBSVM i LIBLINEAR per dur a terme els còmputs.

Matlab

Matlab és un llenguatge de programació i una plataforma de còmput numèric emprada majoritàriament per l'anàlisi de dades, el desenvolupament d'algorismes i la creació de models [20].

A diferència dels anteriors llenguatges, Matlab ofereix un aplicatiu anomenat *classification learner* per treballar amb problemes d'aprenentatge automàtic. En l'àmbit de les SVM, permet entrenar models fent servir kernels RBF, lineals, polinòmics i *sigmoid*. Si bé és cert que l'aplicació és simple i intuïtiva, no disposa de massa capacitat de personalització.

A banda d'aquest aplicatiu, Matlab també disposa d'un sistema de paquets. Els paquets s'anomenen *toolbox* i, d'igual manera que amb R o Python, permeten agrupar funcions i classes. Hem pogut identificar alguns paquets que tracten amb mètodes kernel (per exemple, la *Kernel Methods Toolbox*), però no són massa populars ni estan especialitzats.

C++

C++ és un llenguatge de programació multiparadigma d'alt nivell creat per Bjarne Stroustrup l'any 1983 als laboratoris Bell. El llenguatge hereta gran part de la sintaxi del seu predecessor, el llenguatge C. Una de les seves principals característiques és el seu alt rendiment. Així doncs, no és difícil adonar-se que és un gran aliat a l'hora d'entrenar models d'aprenentatge automàtic. En aquest àmbit destaquen dues llibreries:

LIBSVM LIBSVM és un software integrat de codi obert per a la classificació i regressió de vectors suport [21]. La llibreria està implementada en C++, però disposa d'una gran quantitat d'interfícies i extensions a altres llenguatges com ara Java, R, Python o Julia. A més a més, consta de múltiples funcionalitats com ara la classificació *multiclasse*, l'ús de *cross validation* per a la selecció de models o una interfície gràfica que mostra els resultats de la classificació o regressió. Una altra característica important és que permet emprar diferents kernels i implementar-ne de nous.

shark Shark és una llibreria modular programada en C++. Aquesta implementa múltiples mètodes per resoldre problemes d'optimització lineal i no lineal, així com algorismes d'aprenentatge automàtic basats en kernels, xarxes neuronals i altres tècniques d'aprenentatge automàtic [22].

Com era d'esperar, la llibreria permet entrenar models d'SVM i fer prediccions a partir dels resultats obtinguts. El que és interessant d'aquesta llibreria és la seva capacitat per treballar amb les funcions de kernel. Un dels punts a destacar és el suport a les combinacions lineals de kernels (LKC, de l'anglès *Linear Kernel Combinations*) [23].

La idea de les LKC neix del fet que per resoldre alguns problemes mitjançant mètodes kernel pot ser beneficiós emprar una combinació lineal de kernels que actuï com a un "kernel compost"en lloc de fer sevir un únic kernel. En altres paraules, essent k_1 i k_2 kernels, aleshores —d'acord amb les propietats dels kernels— una combinació lineal de k_1 i k_2 tal que $k = \theta_1 k_1 + \theta_2 k_2$ també resulta ser un kernel (sempre que $\theta_1, \theta_2 \geq 0$). Tenint això en compte, podem generar un kernel compost que sigui combinació lineal d'un conjunt de kernels:

$$k = \sum_i \theta_i k_i$$

Si bé és cert que aquesta metodologia no soluciona el problema que hem plantejat, és una aproximació amb la qual podríem treballar i estudiar-ne la viabilitat per adaptar-la al nostre cas d'estudi.

1.3.2 Justificació de l'elecció

Arribats a aquest punt, hem estudiat les diferents possibilitats que ofereixen els llenguatges de l'apartat anterior i les llibreries o paquets dels quals disposen. Tenint això en compte, hem determinat que l'opció més adient serà emprar el llenguatge C++ per implementar les funcions de kernel necessàries, així com el kernel d'agregació. Seguidament, integrarem la nostra solució en la llibreria LIBSVM.

Per bé que la llibreria *shark* sembla robusta i està dotatda d'algunes característiques interessants, LIBSVM combina la velocitat del C++ amb la seva fàcil usabilitat i escalabilitat. A més a més, està interficiada amb una gran quantitat de llenguatges[1]. Per últim, sabem que LIBSVM es pot executar per la línia de comandes, fet que facilita la creació d'*scripts*.

No hem contemplat el fet de programar amb un altre llenguatge que no sigui C++. És així perquè la llibreria LIBSVM (programada en C++) ja ens facilita l'escalabilitat als llenguatges més comuns de l'àmbit de l'aprenentatge automàtic (com ara Python, R, Julia, Java, etc.) en cas que ens sigui necessari.

1.4 Abast

1.4.1 Objectius del projecte

L'*objectiu principal* d'aquest projecte és, en definitiva, dissenyar, implementar i estudiar el kernel d'agregació per a vectors de variables no reals. Com hem comentat anteriorment, el kernel d'agregació haurà de fer la mitjana dels valors resultants a múltiples crides a funcions de kernel. Per determinar amb quines variables tractarem, requerirem d'un procés de recerca en l'àmbit. Finalment, un cop hàgim implementat el kernel, experimentarem amb ell tot intentant resoldre alguns problemes de classificació.

A banda de l'objectiu esmentat anteriorment, hem establert els següents *subobjectius*:

- Generar una llista dels tipus de variables que sembla raonable tractar amb mètodes d'Aprenentatge Automàtic.
- Fer un estat de l'art sobre els tipus de variables per a les quals ja hi ha dissenyada i no necessàriament implementada alguna funció de kernel.
- Dissenyar i implementar funcions de kernel per a variables amb domini no real.
- Dissenyar i implementar el kernel d'agregació que faci servir les funcions de kernel anteriorment esmentades per a un mateix conjunt de dades.
- Integrar el kernel d'agregació en el software LIBSVM.
- Resoldre un o més problemes de classificació fent servir la nova versió de LIBSVM amb el kernel d'agregació implementat.

1.4.2 Requisits funcionals

Els *requisits funcionals* són les declaracions dels serveis que proveirà el sistema [24]. En el nostre cas, hem establert els següents requisits:

- **Funcionalitat del kernel**: Un dels principals requisits que haurà de complir el kernel proposat com a solució serà ésser funcional. És a dir, mitjançant l'entrenament d'una SVM que integri el nostre kernel haurà de ser possible resoldre diferents problemes del domini establert.
- **Integració del kernel solució**: En aquest projecte únicament es dissenyaran i implementaran funcions de kernel i kernels. Així doncs, per poder resoldre problemes concrets és imperatiu que el kernel final s'integri correctament en l'eina de classificació i regressió de vectors suport *LIBSVM*.

[1] Vegeu l'apartat «*Interfaces and extensions to LIBSVM*» de la pàgina `https://bit.ly/3cUPwxv`.

- **Integració de les funcions de kernel**: Tal com hem explicat en apartats anteriors, per assolir els objectius del nostre projecte serà necessari que agrupem múltiples funcions de kernel en un únic kernel. Així doncs, és un requisit indispensable que aquestes funcions siguin funcionals i s'integrin correctament en el kernel final.

1.4.3 Requisits no funcionals

Els *requisits no funcionals* d'un sistema són aquells que no es refereixen a funcions específiques del sistema, sinó a les propietats emergents d'aquest (Per exemple: fiabilitat, emmagatzematge...). En el cas que ens concerneix, hem establert els requisits següents:

- **Eficiència dels algorismes**: Un factor que cal tenir en compte és l'eficiència dels algorismes programats durant el desenvolupament del projecte. Entrenar models SVM no és —en molts casos— una tasca veloç i, per tant, ens hem d'assegurar que les funcions de kernel que dissenyem siguin eficients. D'aquesta manera contribuirem a minimitzar el temps de còmput global.

- **Usabilitat del sistema**: Tal com hem comentat en altres apartats, el kernel solució proposat s'integrarà en la llibreria *LIBSVM*. Serà necessari tenir cura del procés d'integració per tal que la usabilitat del software no es vegi afectada.

- **Reusabilitat del sistema**: El kernel que pretenem dissenyar integrarà múltiples funcions de kernel. Tot i així, no és difícil adonar-se que poden haver-hi moltes altres funcions de kernel que tractin amb variables el domini de les quals no tindrem en consideració durant el desenvolupament d'aquest projecte. Així doncs, intentarem dissenyar el kernel de manera que el procés d'addició de noves funcions sigui el més fàcil i intuïtiu possible.

1.4.4 Actors implicats

En aquest apartat s'esmentaran i es descriuran dels actors que directament o indirecta estaran implicats en el projecte.

- **Autor principal (desenvolupador)**: Aquest projecte estarà desenvolupat íntegrament per l'Arnau Arqué Martínez, alumne de quart curs del Grau en Enginyeria Informàtica de la Facultat d'Informàtica de Barcelona (UPC) en motiu de la realització del Treball de Final de Grau (TFG).

- **Director del projecte**: El director d'aquest projecte serà el professor Lluís A. Belanche del departament de Ciències de la Computació de la Unitat Transversal de Gestió de l'Àmbit TIC Campus Nord (UTG CNTIC) (UPC).

- **Col·laboradors indirectes**: Aquest projecte pretén —entre altres fites— agrupar múltiples implementacions de funcions de kernel en un únic kernel. Algunes de les implementacions han estat dissenyades i programades per altres investigadors en l'àmbit dels mètodes kernel. Malgrat que no col·laboraran de manera activa en el projecte, considero que cal fer-los menció com a actors implicats.

- **Usuaris**: El codi resultant del projecte serà obert. Així doncs, tot estudiant, professor, expert o aficionat en l'àmbit de l'aprenentatge automàtic i, més concretament, dels mètodes kernel tindrà accés i podrà emprar la solució derivada d'aquest treball.

- **Beneficiaris**: Els beneficiaris d'aquest projecte seran totes aquelles persones que obtinguin algun tipus de profit gràcies als projectes derivats de la solució proposada en aquest treball.

1.4.5 Obstacles

Durant el desenvolupament d'un projecte és molt important prendre consciència dels possibles obstacles que caldrà superar. He considerat com a obstacle tota situació existent que pugui provocar un entrebanc en el correcte transcurs del treball.

- **Inexperiència en l'àmbit d'estudi**: No és cap secret que en el Grau d'Enginyeria Informàtica de la FIB es tracten una gran quantitat de temàtiques diferents. Si bé és cert que aquest fet proporciona a l'estudiantat una àmplia visió en el camp de la informàtica, també cal tenir en compte que no facilita aprofundir en tots els conceptes per separat. És per això que em considero poc experimentat en l'àmbit dels mètodes kernel i hauré de dedicar temps a fer recerca per poder ampliar els meus coneixements.
- **Potència de còmput**: Crear i entrenar models basats en aprenentatge automàtic sovint sol significar haver de treballar amb *datasets* de grans dimensions. En funció de la grandària d'aquests, el temps d'execució dels algorismes pot ser molt elevat en computadors amb potència de càlcul estàndard. Serà necessari tenir en compte aquest fet i evitar que suposi un impacte en la planificació del projecte.
- **Calendari tancat**: El fet de tenir un calendari d'entregues del projecte tancat pot suposar un impediment de cara a assolir els objectius establerts. És per això que caldrà tenir un control acurat del temps previst per a cada tasca.
- **Complexitat del projecte**: Tant l'autor com el director del projecte hem considerat que la tasca a realitzar no és trivial. Per evitar que aquest fet suposi un problema caldrà dedicar un elevat nombre d'hores setmanals al projecte. A més a més, serà imperatiu mantenir una comunicació fluida entre l'autor i el director per tal de minimitzar els possibles errors de desenvolupament.

1.4.6 Riscs

He considerat com a risc potencial tota situació desfavorable que es pugui donar durant el transcurs del projecte i que impliqui un perill per al correcte desenvolupament d'aquest.

- **Planificació poc acurada**: Donada la poca experiència en la gestió de projectes d'aquesta envergadura, hi ha la possibilitat que la planificació no sigui del tot acurada. Serà necessari tenir en compte aquest fet perquè, si es dóna el cas, la reacció sigui pràcticament immediata i no suposi un impediment de cara a assolir els objectius establerts.
- **Restriccions derivades de la Covid-19**: En el moment de la redacció d'aquest document la situació epidemiològica derivada de la Covid-19 és poc estable. Aquest fet pot significar que durant el desenvolupament del projecte s'estableixin restriccions per pal·liar amb els efectes de la pandèmia. Malgrat la indiscutible necessitat de les restriccions, no es pot obviar que podrien implicar dificultats afegides en certes etapes del projecte. Per exemple, el tancament de les biblioteques de la UPC podria impedir l'adquisició de nova documentació.
- **Avaries del hardware**: En un projecte d'enginyeria el mal funcionament dels dispositius de treball pot suposar greus entrebancs en algunes etapes del desenvolupament. Per aquest motiu, fóra bo disposar d'eines alternatives per poder reprendre el projecte en cas que es produeixi alguna avaria.

2 Planificació del projecte

2.1 Metodologia

2.1.1 Metodologia de treball

Un factor de gran utilitat en la gestió de projectes és determinar una metodologia de treball. D'aquesta manera s'estableixen les bases de com es treballarà durant el període de desenvolupament del projecte. En aquest apartat s'estudiarà i concretarà quines metodologies són aplicables al nostre projecte.

Valoració inicial

Per tal de determinar el tipus de metodologia a seguir s'han estudiat dues variants: les metodologies tradicionals i les metodologies àgils.

Les *metodologies tradicionals* es fonamenten en dur a terme una documentació exhaustiva del projecte i se centren a complir la planificació establerta en la fase inicial del treball. En definitiva, els pilars fonamentals són la documentació, la planificació i els processos [25].

D'altra banda, les *metodologies àgils* responen a la necessitat de pal·liar amb certs problemes que sorgeixen en l'aplicació de les metodologies tradicionals. Aquest tipus de metodologies es caracteritzen per retardar la presa de decisions i dur a terme una planificació adaptativa del projecte. Són de gran utilitat en projectes de desenvolupament complex [25].

Elecció final: *SCRUM*

Tenint en compte que el nostre projecte no és trivial i que l'autor no està abastament familiaritzat amb la temàtica, sembla raonable pensar que una metodologia àgil serà més beneficiosa que no pas una de tradicional. Més concretament, treballarem seguint el model de la metodologia *SCRUM*.

La metodologia SCRUM és adequada per a projectes amb un desenvolupament complex i amb la necessitat d'un cert grau de flexibilitat en la planificació [26]. Una de les principals característiques és la divisió del desenvolupament en diferents etapes anomenades *sprints*. Cada *sprint* consta d'un període de desenvolupament, seguit d'una revisió dels resultats i, finalment, un reajustament del projecte d'acord amb la informació recaptada a les revisions.

En definitiva, aquesta metodologia s'adequa raonablement als menesters del nostre projecte.

2.1.2 Eines de desenvolupament

Les eines de desenvolupament engloben els aplicatius, programaris i les llibreries necessaries per al correcte desenvolupament del projecte. En el nostre cas farem servir el següent:

- **Git**: Git és un sistema de control de versions gratuït i de codi obert [27]. El farem servir per mantenir un control sobre els diferents canvis que duguem a terme al llarg del projecte. A més a més, ens pot ser molt útil per mantenir el projecte actualitzat en cas que emprem múltiples dispositius durant l'etapa de desenvolupament.

- **Fork**: L'aplicatiu Fork és un client de Git parcialment gratuït [28]. Malgrat que disposa d'una versió de pagament, les funcionalitats de la versió gratuïta ens són suficients per al projecte que ens concerneix.
- **LIBSVM**: LIBSVM és un software integrat per a la classificació i regressió de *support vectors* [21]. El nostre projecte consisteix en dissenyar, implementar i estudiar mètodes kernel. LIBSVM ens permetrà entrenar models basats en *Support Vector Machines* (SVM) mitjançant els kernels que hàgim creat.
- **Texmaker**: Texmaker és un editor gratuït i multiplataforma de LaTeX [29]. Farem servir l'aplicatiu per redactar la documentació del projecte.
- **Jabref**: Jabref és un aplicatiu gratuït que proporciona una interfície simple i intuïtiva per facilitar l'edició d'arxius BibTeX [30]. Farem servir Jabref per gestionar les cites i referències establertes en la memòria del proejcte.

2.1.3 Eines de seguiment i valoració

L'eina bàsica de seguiment i valoració dels resultats obtinguts serà les reunions amb el tutor del projecte. D'acord amb la metodologia descrita anteriorment (*SCRUM*), es planificarà una reunió per setmana en la qual es revisaran els avenços assolits durant el transcurs d'aquesta. A més a més, es reajustaran els continguts dels següents *sprints* en cas que sigui necessari.

A banda de les reunions amb el director del projecte farem servir un *indicador d'eficàcia* per mesurar el percentatge d'assoliment dels objectius. El càlcul es durà a terme en finalitzar cada *sprint* i s'emprarà la fórmula següent per quantificar-lo:

$$\textit{Indicador Eficàcia } (\%) = \frac{\textit{Nº Objectius Assolits}}{\textit{Nº Objectius Establerts}} \cdot 100$$

2.2 Descripció i estimació temporal de les tasques

2.2.1 Grups de tasques i *sprints*

Un cop hem definit i descrit els objectius principals del projecte, ha arribat el moment plantejar-nos quines són les activitats que haurem de dur a terme per tal d'assolir aquests objectius.

Tot i així, abans de definir les tasques concretes, considero que serà clarificador dividir el nostre projecte en grups d'activitats. Cada grup tindrà una temàtica diferenciada respecte a la resta de grups. A partir d'aquí, ens serà molt més fàcil definir les tasques individuals i estimar la seva extensió temporal. Els grups que hem definit són els següents:

- **Recerca**: Dins aquest grup s'inclouran les tasques associades a la recerca i l'estudi de diferents àmbits del treball. Aquestes tasques ens seran necessàries per, posteriorment, poder començar les tasques d'implementació. Totes les tasques d'aquest grup tindran associat un codi amb el prefix REC.
- **Implementació**: Aquest grup englobarà les tasques de disseny i implementació dels mètodes kernel i kernels així com les tasques d'integració d'aquests en el software LIBSVM. El codi de les tasques d'aquest grup tindrà el prefix IMP.
- **Experimentació**: Aquest bloc inclourà les tasques d'experimentació fent ús dels mètodes kernel i kernels implementats anteriorment. Les tasques d'aquest grup tindran associat un codi amb prefix EXP.

- **Gestió**: Dins aquest grup hi haurà les tasques associades a la gestió del projecte com ara la contextualització o la planificació temporal d'aquest. El codi de les tasques d'aquest grup tindrà prefix GES.
- **Documentació**: Contindrà totes les tasques de documentació que es duran a terme al llarg del projecte. Les tasques d'aquest grup tindran un codi amb prefix DOC.
- **Control**: No hem d'oblidar que aquest treball segueix la metodologia àgil SCRUM. Tal com hem explicat en apartats anteriors, aquesta metodologia es basa a dividir el projecte en diferents *sprints*. Cada *sprint* consta d'una etapa de desenvolupament seguida d'una etapa de revisió i una d'adaptació. Així doncs, hem creat aquest grup per tal d'incloure les tasques de revisió i adaptació de cada *sprint*. Les tasques d'aquest grup tindran associat un codi amb prefix CRL.

Havent definit els grups de tasques, ja podem fer una planificació de quins seran els *sprints* que es duran a terme al llarg del projecte. En total, es dividirà el treball en quatre *sprints* (enumerats per ordre d'execució): L'*sprint* associat a la gestió (GES), el de recerca (REC), el d'implementació (IMP) i el d'experimentació (EXP). Més endavant detallarem l'extensió temporal de cadascun dels *sprints*.

2.2.2 Descripció de les tasques

Arribats a aquest punt, hem definit quins seran els àmbits en què es repartiran les tasques del projecte. Seguidament, determinarem les tasques que conformen cada grup i l'estimació temporal d'aquestes. Abans, però, farem un breu comentari de l'extensió temporal global del projecte.

Tal com s'indica a la normativa del Treball de Final de Grau (TFG) de la FIB [31], el TFG té assignat un total de 18 crèdits i la càrrega de treball és de 30 h per crèdit. Tot plegat equival a una càrrega horària total de 540 h. Tenint en compte que el treball s'inicia el 22 de febrer de 2021 i finalitza el 2 de juliol de 2021 (sense incloure el període de presentacions), hi ha assignades un total de 30 h per setmana. Aquest nombre d'hores és el que es repartirà entre les diferents tasques del projecte. Tot i així, a les tasques de control (CRL) se'ls assignarà un extra de 30 h. Per tant, la durada total estimada del projecte és de 570 h.

Tasques de recerca (REC)

- **REC1**: Estudiar quins tipus de variables sembla raonable tractar amb mètode d'aprenentatge automàtic. L'estimació temporal d'aquesta tasca és de 25.5 h.
- **REC2**: Estudiar per a quins tipus de variables ja hi ha dissenyada alguna funció de kernel. L'estimació temporal d'aquesta tasca és de 25.5 h.
- **REC3**: Analitzar el software LIBSVM i estudiar com es pot estendre. L'estimació temporal d'aquesta tasca és de 51 h.

Tasques d'implementació (IMP)

- **IMP1**: Dissenyar mètodes kernel per a variables amb domini no real. L'estimació temporal d'aquesta tasca és de 38.25 h.
- **IMP2**: Implementar en C++ mètodes kernel per a variables amb domini no real. L'estimació temporal d'aquesta tasca és de 38.25 h.
- **IMP3**: Dissenyar un kernel que agrupi múltiples funcions de kernel. L'estimació temporal d'aquesta tasca és de 25.5 h.

- **IMP4**: Implementar en C++ el kernel dissenyat anteriorment. L'estimació temporal d'aquesta tasca és de 25.5 h.
- **IMP5**: Integrar el kernel implementat anteriorment en el software LIBSVM. L'estimació temporal d'aquesta tasca és de 25.5 h.

Tasques d'experimentació (EXP)

- **EXP1**: Plantejar i definir els experiments a realitzar. L'estimació temporal d'aquesta tasca és de 12.75 h.
- **EXP2**: Recopilar i analitzar les dades necessàries. L'estimació temporal d'aquesta tasca és de 12.75 h.
- **EXP3**: Executar el software fent servir el kernel prèviament generat sobre les dades recopilades. L'estimació temporal d'aquesta tasca és de 25.5 h.
- **EXP4**: Analitzar els resultats obtinguts en l'execució. L'estimació temporal d'aquesta tasca és de 25.5 h.

Tasques de gestió (GES)

- **GES1**: Contextualitzar i definir l'abast del projecte. L'estimació temporal d'aquesta tasca és de 30 h.
- **GES2**: Planificar el projecte. L'estimació temporal d'aquesta tasca és de 30 h.
- **GES3**: Analitzar el pressupost i la sostenibilitat del projecte. L'estimació temporal d'aquesta tasca és de 30 h.

Tasques de control (CRL)

- **CRL1**: Etapes de revisió (SCRUM). Reunions amb el director del projecte. L'estimació temporal d'aquesta tasca és de 18 h.
- **CRL2**: Etapes de reajustament (SCRUM). L'estimació temporal d'aquesta tasca és de 12 h.

Tasques de documentació (DOC)

- **DOC1-4**: Tasques associades a la documentació dels grups GES, REC, IMP i EXP, respectivament. L'estimació temporal de cadascuna de les tasques és de 30, 18, 27 i 13.5 h, respectivament.
- **DOC5**: Documentació final. L'estimació temporal d'aquesta tasca és de 30 h.

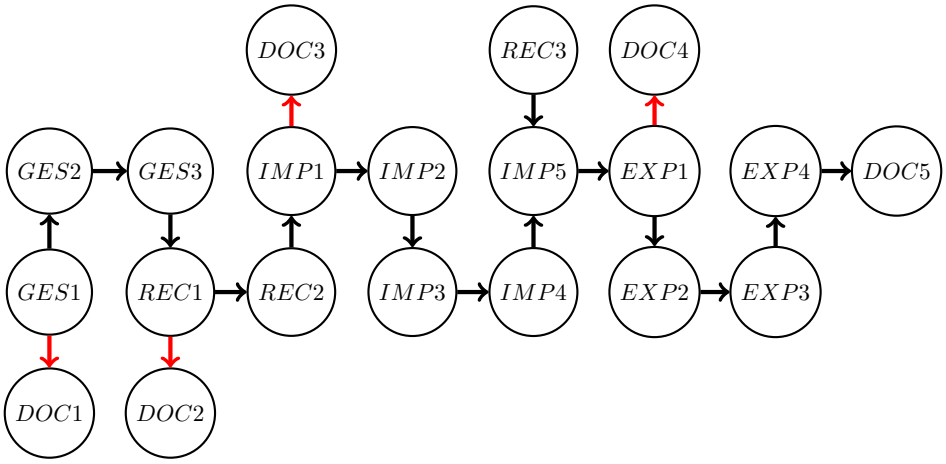
2.2.3 Dependències entre tasques

En general, la planificació d'aquest projecte ha estat dissenyada de tal manera que les tasques s'executin en cascada. Tot i així, hi ha algunes excepcions. És per això que hem decidit plasmar les dependències entre les tasques mitjançant el graf dirigit de la figura 2.1.

Tot node del graf representa una tasca, mentre que les arestes tenen un significat o un altre en funció del seu color. Sigui (i, j)[1] una aresta dirigida del graf, si aquesta és de color **vermell**, significa que la tasca j només es podrà iniciar un cop la tasca i hagi començat. En canvi, si l'aresta és de color **negre**, significa que la tasca j només podrà començar un cop la tasca i hagi finalitzat.

[1] i i j són nodes del graf que, alhora, representen tasques.

Figura 2.1: Graf de dependències entre tasques.



2.2.4 Recursos necessaris

En aquest apartat definirem quins són els recursos necessaris per dur a terme les tasques que hem especificat anteriorment. Distingirem entre dos tipus de recursos principals: els recursos *humans* i els recursos *materials*.

Recursos humans

En l'àmbit dels recursos humans destaquen un total de quatre figures que s'encarregaran de desenvolupar les diferents tasques del projecte. Noti's que, en el cas que ens concerneix, aquestes figures s'han d'interpretar com a rols que «interpretarà» la mateixa persona; a saber, l'autor del projecte.

- **Cap de projecte (CP)**: El cap de projecte serà l'encarregat de dur a terme els tràmits associats a les tasques de gestió (GES, DOC1). A més a més, serà qui dirigeixi les reunions (CRL1) i, en cas necessari, reajusti el projecte (CRL2). Finalment, un cop acabat el desenvolupament del projecte, serà qui faci el redactat final (DOC5).
- **Investigador d'aprenentatge automàtic (I)**: Serà l'encarregat de dur a terme les tasques associades a la recerca (REC) i documentar-les (DOC2). A més a més, també participarà a les reunions periòdiques (CRL1).
- **Enginyer d'aprenentatge automàtic (E)**: L'enginyer tindrà l'encàrrec d'executar les tasques d'implementació (IMP) i experimentació del projecte (EXP), així com la seva redacció (DOC{3,4}). Donades les seves capacitats, també col·laborarà en la tasca d'anàlisi del software LIBSVM (REC3, DOC2), juntament amb l'investigador. Finalment, també participarà a les reunions del projecte (CRL1).
- **Analista de dades (A)**: L'analista de dades ajudarà en les tasques d'experimentació. Més concretament, a l'hora de recopilar i analitzar les dades necessàries per als experiments proposats (EXP{1,2}, DOC4). Igual que la resta d'integrants, també participarà a les reunions (CRL1).

Recursos materials

En l'àmbit dels recursos materials diferenciarem dos tipus de recursos: els recursos *software* i els recursos *hardware*.

Taula 2.1: Taula–resum de les tasques.

Codi	Resum de la tasca	Nº *sprint*	Temps (h)	Dependències	Rols	Recursos
REC1	Estudi de variables	2	25.5	GES3	I	H1
REC2	Estudi de mètodes kernel existents	2	25.5	REC1	I	H1
REC3	Anàlisi de LIBSVM	2	51	–	I, E	S3, H1
IMP1	Disseny de mètodes kernel	3	38.25	REC2	E	H1
IMP2	Implementació de mètodes kernel	3	38.25	IMP1	E	S{1,2}, H1
IMP3	Disseny de kernels	3	25.5	IMP2	E	H1
IMP4	Implementació de kernels	3	25.5	IMP3	E	S{1,2}, H1
IMP5	Integració de kernels a LIBSVM	3	25.5	REC3, IMP4	E	S{1,2,3}, H1
EXP1	Plantejament dels experiments	4	12.75	IMP5	E, A	H1
EXP2	Recopilació de dades	4	12.75	EXP1	E, A	S2, H1
EXP3	Execució dels experiments	4	25.5	EXP2	E	S{1,2,3}, H1
EXP4	Anàlisi dels resultats	4	25.5	EXP3	E	H1
GES1	Contextualització i abast	1	30	–	CP	H1
GES2	Planificació temporal	1	30	GES1	CP	H1
GES3	Pressupost i sostenibilitat	1	30	GES2	CP	H1
DOC1	Documentació de l'*sprint* 1	1	30	GES1	CP	S4, H1
DOC2	Documentació de l'*sprint* 2	2	18	REC1	I, E	S4, H1
DOC3	Documentació de l'*sprint* 3	3	27	IMP1	E	S4, H1
DOC4	Documentació de l'*sprint* 4	4	13.5	EXP1	E, A	S4, H1
DOC5	Documentació final	–	30	EXP4	CP	S4, H1
CRL1	Revisió del projecte	1-4	18	–	Tots	H1
CRL2	Reajustament del projecte	1-4	12	–	CP	H1
Total	–	–	**570**	–	–	–

Font: Elaboració pròpia.

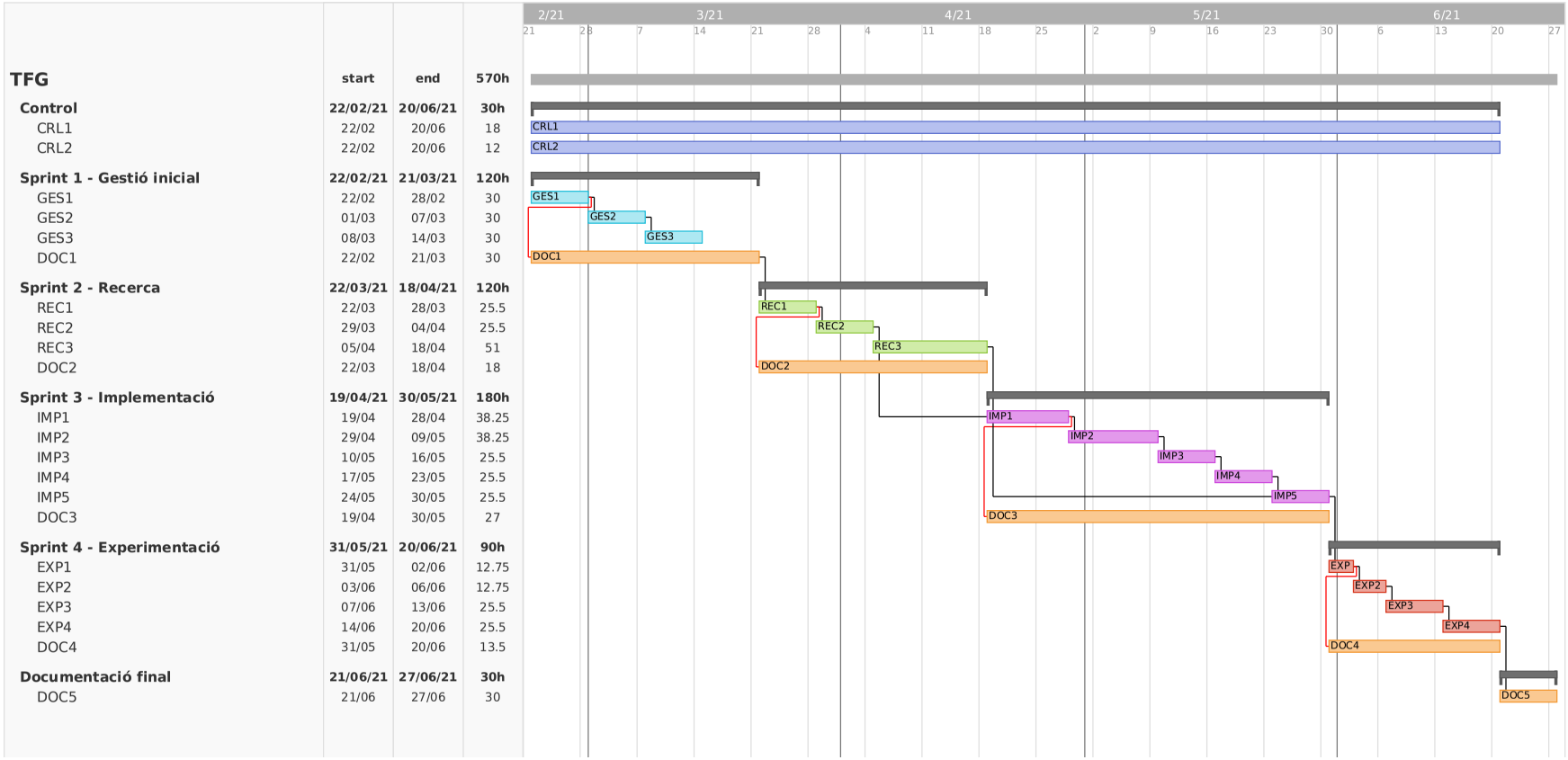
- **Recursos software**: Com a recursos software ens serà necessari un editor de codi (S1), un client de Git (S2) i, sens dubte, l'aplicatiu LIBSVM (S3). A més a més, necessitarem un editor de documents en LaTeX (S4). Finalment, per treballar amb l'ordinador, també requerirem un sistema operatiu estable. En el nostre cas, treballarem amb MacOS BigSur.
- **Recursos hardware**: En l'àmbit dels recursos hardware, farem servir un ordinador principal (H1) iMac de 27" de l'any 2015 amb un processador 4 GHz Intel Core i7 de quatre nuclis i memòria RAM de 16 GB.

2.2.5 Taula–resum i diagrama de Gantt

Arribats a aquest punt, ja tenim la capacitat de generar una taula–resum amb tota la informació establerta per a cada tasca. A la taula 2.1 podem veure, per a cada tasca, el seu codi, una breu descripció, les dependències i els rols i recursos necessaris per dur-la a terme. A l'última fila hi ha el nombre total d'hores estimades del projecte.

En la figura 2.2 podem veure el diagrama de Gantt associat al nostre projecte. Les tasques que s'hi aprecien s'han agrupat en funció de l'*sprint* al qual pertanyen.

Figura 2.2: Diagrama de Gantt del projecte. Elaboració pròpia fent ús de l'eina *Teamgantt* [32].



2.3 Gestió del risc

No és difícil adonar-se que, tot suposant tenir una planificació molt acurada del projecte, poden sorgir imprevistos que afectin el correcte desenvolupament d'aquest. En apartats anteirors hem comentat quins són els riscs que contemplem. En aquest apartat revisarem tots els riscs i intentarem establir plans de xoc per evitar que suposin un impacte en els objectius del projecte.

- **Gestió de la planificació poc acurada**: Si bé és cert que, com que no som experts en l'àmbit, podem flaquejar amb relació a la precisió de la planificació del projecte, val a dir que tenim molts recursos i indicadors per evitar que aquest fet suposi un inconvenient. Precisament, un dels avantatges d'emprar una metodologia àgil és l'elevada capacitat de reacció que ofereix. En cada *sprint* disposem d'etapes de revisió i reajustament. Així doncs, sembla raonable pensar que disposem dels recursos necessaris per pal·liar amb els efectes d'aquest risc. Tot i així, una mala planificació pot suposar un impacte en totes les tasques del projecte. Establirem que per a cada replanificació necessitarem 5 h extra. Assumint que les replanificacions es duran a terme al final de cada *sprint* i tenint en compte que hi ha un total de quatre *sprints*, requerirem com a màxim de 20 h extra.

- **Gestió de les restriccions associades a la Covid-19**: En aquest sentit tenim poc marge de maniobra. En l'hipotètic cas que tanquin o modifiquin l'horari de les biblioteques a causa de la Covid-19, haurem de contactar amb els serveis bibliotecaris de la UPC per intentar —si fos de menester— buscar una alternativa a l'ús de llibres físics. Les tasques que es podrien veure afectades són, sobretot, les de gestió i recerca (GES1-3 i REC1-3). És així perquè són les que requeriran més consultes a fonts d'informació. És difícil estimar el nombre d'hores que pot repercutir en la durada del projecte. Tot dependrà del temps de resposta del personal de la UPC i de la duresa de les restriccions imposades. Estimarem, però, que necessitarem un total de 5 h extra per a cada grup de tasques (10 h en total).

- **Gestió de les avaries de hardware**: En cas que el correcte funcionament de l'ordinador principal es vegi compromès, disposarem d'un dispositiu de treball de reforç. En l'àmbit dels recursos hardware s'afegirà un ordinador portàtil MacBook Pro de 13" de l'any 2016 amb processador Intel Core i5 de doble nucli i 8GB de RAM. Per tant, no sembla que aquest risc pugui afectar greument al desenvolupament del projecte. Tot i així, les tasques que es veurien més compromeses són les d'implementació i experimentació (IMP1-5 i EXP1-4). Ens serà necessari instal·lar novament el programari i descarregar les dades i la documentació necessària que, previsiblement, haurem desat al *Git* amb anterioritat. Estimarem que, en cas d'avaria, necessitarem un total de 5 h extra.

Havent estudiat tots els possibles riscs, hem pogut trobar alternatives i solucions per tal d'evitar que suposin un impediment per assolir els objectius del projecte. Així doncs, sembla raonable pensar que el projecte acabarà a temps.

2.4 Pressupost

En aquest capítol durem a terme un estudi econòmic del projecte. Començarem estimant i identificant els costos associats a tots els seus àmbits. Finalment, proposarem i descriurem els mecanismes que s'empraran per controlar les desviacions que puguin aparèixer durant el transcurs del projecte.

2.4.1 Identificació i estimació dels costos

En aquesta primera secció identificarem i estimarem els costos de manera incremental. Començarem calculant les despeses associades al personal. Seguidament, analitzarem els costos genèrics i

les contingències. A continuació estudiarem les despeses dels imprevistos i acabarem determinant el pressupost total del projecte.

Costos de personal

En apartats anteriors hem determinat quins rols estaran involucrats en aquest projecte i quines seran les tasques que duran a terme. Ara, però, ens és necessari determinar les despeses associades a la seva contractació. Per fer-ho, hem buscat el sou mitjà anual de cadascun dels rols a la pàgina *Glassdoor*[2]. Un cop recopilats els sous anuals, hem emprat la següent fórmula per calcular el sou per hora:

$$Sou_{hora}\ (€) = \frac{Sou_{any}\ (€/any)}{250\ (dies\ laborables/any)\ \cdot\ 8\ (hores/dia)}$$

Per obtenir el cost real per hora associat a cada rol haurem de multiplicar el Sou_{hora} per 1.3. D'aquesta manera, estarem tenint en compte les despeses de la seguretat social. A la taula 2.2 hi ha una recopilació dels sous per hora i els costos per hora de cada rol.

Un cop determinats els costos per hora de cada rol, haurem de calcular el cost associat al personal de cada tasca i, finalment, de tot el projecte (suma dels costos de totes les tasques). Ho podem veure a la taula 2.3.

Taula 2.2: Sous i costos per hora del personal.

Rol	**Sou anual (€)**	**Sou/hora brut (€/h)**	**Cost/hora (€/h)**
Cap de projecte (CP)	46000	29.49	38.33
Investigador d'AA (I)	18200	11.67	15.17
Enginyer d'AA (E)	38000	24.36	31.67
Analista de dades (A)	28600	18.33	23.83

Font: Glassdoor (`https://www.glassdoor.es/`).

Per saber el nombre d'hores que duu a terme cada rol en cada tasca només cal dividir el nombre d'hores associat a cada tasca (que podeu veure a la taula 2.1) entre els rols que hi participen. Dit d'una altra manera, s'ha repartit el temps estimat per a cada tasca equitativament entre els rols que hi participen.

Costos genèrics

En l'àmbit dels costos genèrics destaquen les despeses associades al *hardware* i les associades a l'*espai de treball*. Noti's que no hem inclòs cap cost de software perquè tot el programari que emprarem durant el projecte és lliure.

[2] *Glassdoor*: `https://www.glassdoor.es/`

Taula 2.3: Costos per tasca associats al personal.

Tasca	**Cost CP (€)**	**Cost I (€)**	**Cost E (€)**	**Cost A (€)**	**Total (€)**
REC1		386.75			386.75
REC2		386.75			386.75
REC3		386.75	807.50		1194.25
IMP1			1211.25		1211.25
IMP2			1211.25		1211.25
IMP3			807.50		807.50
IMP4			807.50		807.50
IMP5			807.50		807.50
EXP1			201.88	151.94	353.81
EXP2			201.88	151.94	353.81
EXP3			807.50		807.50
EXP4			807.50		807.50
GES1	1150.00				1150.00
GES2	1150.00				1150.00
GES3	1150.00				1150.00
DOC1	1150.00				1150.00
DOC2			285.00		285.00
DOC3			855.00		855.00
DOC4			213.75	160.88	374.63
DOC5	1150.00				1150.00
CRL1	172.50	68.25	142.50	107.25	490.50
CRL2	460.00				460.00
Total (€)	6382.50	1228,50	9167.50	572.00	**17350.50**

Font: Elaboració pròpia.

Tal com hem comentat a l'apartat dels recursos hardware, farem servir un ordinador principal iMac de 27". En primer lloc, calcularem el cost per hora que suposa l'adquisició del producte mitjançant la següent fórmula:

$$Cost_{HW}\ (€/h) = \frac{PVP\ (€) - ValorResidual\ (€)}{3\ (anys\ d'amortització)\ \cdot\ 250\ (dies\ laborables/any)\ \cdot\ 8\ (hores/dia)}$$

Estimarem que el valor residual dels productes hardware serà el 10% del preu d'adquisició. Així doncs: $ValorResidual\ (€) = 0.1 \cdot PVP\ (€)$.

Seguidament, calcularem el cost dels recursos hardware imputat al nostre projecte. Donat que l'ordinador es farà servir en totes les tasques, multiplicarem el $Cost_{HW}$ pel nombre total d'hores del projecte. Podeu veure les despeses resultants a la taula 2.4.

Taula 2.4: Costos genèrics associats al hardware.

Recurs	PVP (€)	Cost/hora (€/h)	Cost imputable (€)
iMac 27" 2015	2629	0.39	224.78

Font: Applesfera (`https://bit.ly/3qKh5yb`).

Taula 2.5: Costos genèrics associats a l'espai de treball.

Espai	Cost (€/mes)	Cost + IVA (€/mes)	Nº mesos	Cost total (€)
Utopicus Pl. Cat.	300	363	4	1452

Font: Utopicus (`https://bit.ly/30Fs55s`).

Amb relació a l'espai de treball, contractarem un servei d'oficina de *coworking*. D'aquesta manera tindrem una única despesa associada al lloc de treball, la sala de reunions, l'electricitat i l'internet. L'espai s'anomena *Utipicus Plaça Catalunya*. L'hem escollit per les seves elevades prestacions i l'alta accessibilitat (ubicació propera a les línies L1, L2, L3 i L4 del Metro de Barcelona i accés les 24 h del dia). El resum dels costos es troba a la taula 2.5.

Contingència

Les despeses de contingència representen el marge de seguretat del pressupost. En el nostre cas, les hem calculat prenent un 15% dels costos genèrics i del personal. Podeu veure els resultats a la taula 2.6.

Taula 2.6: Costos de contingència.

Tipus cost	Cost (€)	% de Contingència	Cost contingència (€)
Personal	17350.50	15	2602.58
Genèric	1676.78	15	251.52
Total			**2854.09**

Font: Elaboració pròpia.

Imprevistos

Tenint en compte el que hem establert en l'apartat de la gestió del risc, l'únic imprevist que implicaria una partida extra de pressupost seria una avaria de hardware. En aquest cas, farem servir un ordinador portàtil MacBook Pro de 13" de l'any 2016. Podeu veure el resum dels costos a la taula 2.7

Taula 2.7: Costos d'imprevistos.

Recurs	**PVP (€)**	**Preu imputable (€)**	**% de Risc**	**Cost final (€)**
MacBook Pro 13" 2016	1699	145.26	20	29.05

Font: Applesfera (`https://bit.ly/3cL4ia7`).

Pressupost final

Per acabar, farem el càlcul total del pressupost del projecte. Únicament serà necessari fer la suma de les despeses declarades en els apartats anteriors. Podeu veure els resultats a la taula 2.8.

Taula 2.8: Pressupost final del projecte.

Concepte del cost	**Cost (€)**
Personal	17350.50
Genèric	1676.78
Contingències	2854.09
Imprevistos	29.05
Total	**21910.42**

Font: Elaboració pròpia.

2.4.2 Control de la gestió

Un cop definit el pressupost del projecte ens és necessari establir mecanismes de control per evitar desviacions en aquest àmbit. En el cas que ens concerneix, només poden haver-hi desviacions de dos tipus: *desviacions de cost* i *desviacions temporals*. Una desviació de cost representa una variació de la despesa estimada en algun àmbit del projecte. D'altra banda, les desviacions temporals representen alteracions en la dedicació horària estimada del projecte. Per calcular l'abast d'aquestes desviacions farem servir els següents indicadors numèrics:

- **Desviacions de cost**: $Desviació_{cost} = (Cost_{estimat} - Cost_{real}) \cdot Hores_{reals}$
- **Desviacions temporals**: $Desviació_{temps} = (Hores_{estimades} - Hores_{reals}) \cdot Cost_{estimat}$

Per tal de valorar si s'han produït desviacions, es durà a terme un control de la gestió en cada reunió de seguiment del projecte. Aquest control vindrà acompanyat del càlcul de les desviacions mitjançant els indicadors definits anteriorment. En cas que $Desviació_{cost} < 0$ o $Desviació_{temps} < 0$, significarà que els costos o terminis estimats han estat insuficients i, per tant, serà necessari reajustar el pressupost i les tasques. En canvi, si $Desviació_{cost} \geq 0$ o $Desviació_{temps} \geq 0$, voldrà dir que les estimacions eren correctes i que disposem d'un cert marge de pressupost o temps.

2.5 Sostenibilitat

2.5.1 Autoavaluació

Abans de respondre l'enquesta d'autoavaluació que se'ns proposa tenia una opinió formada i meditada del paper que juguem els informàtics en l'àmbit de la sostenibilitat. Havent analitzat les preguntes i valorat les respostes, em refermo en la meva convicció que desenvolupem un rol clau.

No és cap secret que l'abast de la informàtica ha crescut —i està creixent— de manera exponencial des de fa anys. La creació dels primers computadors va marcar l'inici d'una nova era amb límits encara inimaginables. Com a partícips en actiu i influents directes en aquest àmbit som en part responsables d'un dels temes en l'ordre del dia mundial: el medi ambient. Si bé és cert que en l'actualitat la meva petjada ecològica en l'àmbit professional és mínima, considero que, d'ara endavant, haig de ser molt conscient de l'impacte dels meus projectes en l'entorn. És imperatiu que, tenint en compte l'extremadament delicada situació climàtica actual, s'intentin minimitzar al màxim les despeses ecològiques.

D'altra banda, considero que els efectes que pot tenir la meva implicació en l'àmbit econòmic dels meus futurs projectes són relativament baixos. És així perquè durant la nostra formació no se'ns ha entrenat per a aquests afers. Comprenc la necessitat d'entendre els conceptes bàsics de l'economia, però benveig que siguin els economistes especialitzats els que s'encarreguin de controlar aquest àmbit.

Finalment, també sóc conscient de la necessitat d'implicació social que requereixen els projectes informàtics. Avui en dia, amb un món cada cop més connectat per les xarxes socials i internet és imperatiu que, com a desenvolupadors, ens impliquem en incloure els principis de justícia social, equitat, diversitat i transparència en els nostres projectes. No hem d'oblidar l'impacte que poden tenir les tecnologies de la informació i de la comunicació en la societat. És la meva responsabilitat com a futur enginyer vetllar per maximitzar l'impacte positiu de les meves creacions en la societat.

2.5.2 Dimensió econòmica

Tal com hem comentat a l'apartat del pressupost, el cost estimat del projecte és de 21910.42 €. Donada l'alta capacitat de reutilització i actualització que tindrà el producte final, considero que no es tracta d'una xifra elevada. Amb relació a una possible reducció de les despeses, penso que es tracta d'una eventualitat poc probable. És així, perquè únicament s'han comptabilitzat els recursos estrictament necessaris per a la compleció dels objectius establerts.

Actualment, si algun projecte requereix un kernel per a vectors de diferents tipus de variables no reals que estigui implementat en el software LIBSVM, primer haurà de dissenyar i implementar el kernel específic. Posteriorment, es podrà iniciar la resolució del problema en concret. Aquesta etapa prèvia, però, suposarà un impacte negatiu en el cost total del projecte.

La nostra solució ofereix un kernel funcional per a la resolució de problemes que requereixin l'ús de mètodes kernel. Aquest kernel integrat en el software LIBSVM agruparà múltiples funcions de kernel i permetrà a l'usuari triar quina o quines fer servir per a cada variable del problema en qüestió. El fet que tot el codi generat i el software sigui lliure permetrà als investigadors estalviar-se el cost de confecció d'aquest kernel. Així doncs, podran dedicar aquests diners a altres afers com, per exemple, una ampliació de l'abast de l'estudi.

2.5.3 Dimensió mediambiental

Amb relació a l'impacte mediambiental d'aquest projecte, considero que no serà gens elevat. Malgrat haver establert que hi haurà diferents rols participant en el treball, és sabut que tots ells seran representats per l'únic autor del projecte. Aquest fet implica que només serà necessari emprar un únic recurs hardware durant el desenvolupament del projecte; a saber, un iMac de 27" de l'any 2015.

Segons la pàgina oficial d'*Apple*[3], l'ordinador en qüestió consumeix uns 63 W mentre està inactiu i 240 W en funcionament. Assumint que es treballarà durant 6 h al dia i estimant que un 20% del temps l'ordinador restarà en mode repòs, podem estimar que:

- El 20% del temps (1.2 h) el consum serà de $Consum_{repòs} = 6 \cdot 0.2 \cdot 63 = 0.076$ kWh.
- El 80% del temps (4.8 h) el consum serà de $Consum_{actiu} = 6 \cdot 0.8 \cdot 240 = 1.152$ kWh.

Tenint en compte que únicament estarem emprant un ordinador, veig difícil una reducció del consum associat al projecte posat en producció.

2.5.4 Dimensió social

Personalment, em prenc aquest projecte com una oportunitat de seguir ampliant els meus coneixements en relació amb l'aprenentatge automàtic. El cert és que durant els anys que he estat cursant la carrera a la FIB no he tingut gaires oportunitats per treballar en aquest àmbit. Aquest projecte em permetrà dedicar quatre mesos a un camp que trobo extraordinàriament interessant i que em motiva a seguir avançant i assumir nous reptes en el camí de la meva vida acadèmica i professional.

Si bé és cert que un kernel de per si no té la capacitat de millorar la nostra societat, crec que les seves aplicacions poden marcar una gran diferència. Sempre m'ha agradat plantejar els meus projectes des del punt de vista de com poden suposar un impacte positiu en la gent. En aquest cas, els mètodes kernel tenen un ampli ventall d'aplicacions; des de la detecció de diferents tipus

[3] `https://support.apple.com/es-es/HT201918`

de càncer [33] fins a l'anàlisi de les xarxes socials per evitar casos d'assetjament [34]. Considero que la solució que proposarem pot ser d'ajut en el desenvolupament de projectes que representin un benefici real per a la nostra societat.

Tenint en compte que no hi ha cap altre kernel que agrupi múltiples funcions de kernel i s'integri en el software LIBSVM, considero que hi ha una necessitat real del projecte que ens concerneix. Gràcies a la solució que proposarem, diferents equips d'investigació podran desenvolupar els seus projectes basats en mètodes kernel. A més a més, podran emprar el software a cost zero.

3 Recerca de variables

En aquest capítol durem a terme un estat de l'art sobre diferents tipus de variables sense estructura. A més a més, investigarem quines són les aplicacions dels tipus de dada en problemes moderns i si existeixen funcions de kernel dissenyades per a la variable en qüestió.

3.1 Variables numèriques contínues

Les variables numèriques contínues són aquelles que prenen com a valors nombres de la recta real $\mathbb{R}$. Aquest tipus de variable ha estat bastament estudiat. En definitiva, estem parlant de nombres reals. Així doncs, considerem que no té massa sentit enumerar aplicacions d'aquesta tipologia. Tot i així, sí que ens interessa saber quines funcions de kernel podem emprar. Per tant, i malgrat que han estat àmpliament estudiades, n'enumerarem algunes.

Kernel polinòmic

El kernel polinòmic és una funció de kernel emprada per tractar vectors de variables numèriques. Aquest mètode representa la similitud entre dues mostres en un *feature space* generat a partir de polinomis sobre les mostres originals. D'aquesta manera, permet l'aprenentatge de models no lineals [35].

Freqüentment podem trobar-lo expressat de la forma:

$$k_{poly}(x, z) = (\langle x, z \rangle + R)^d \tag{3.1}$$

On k_{poly} és el kernel definit sobre un espai vectorial X de dimensió n i R i d són paràmetres. L'expressió $\langle \cdot, \cdot \rangle$ representa el producte escalar entre els dos arguments.

Tot i així, també podem derivar el kernel polinòmic a partir d'un altre kernel k_1. Ens queda, doncs, una expressió de la forma:

$$k_{poly}(x, z) = p(k_1(x, z)) \tag{3.2}$$

On $p(\cdot)$ és qualsevol polinomi amb coeficients positius.

Radial Basis Function (RBF)

Un altre kernel per a variables numèriques és el *Radial Basis Function*, també conegut com RBF o *Gaussian Kernel*. Els kernels gaussians són els més coneguts i emprats i han estat bastament analitzats en múltiples àmbits d'estudi [35]. Una possible formulació del kernel RBF és la següent:

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

On $\sigma > 0$ i $||x - z||^2$ representa la distància euclídea quadràtica entre ambdós *feature vectors*. El paràmetre σ controla la flexibilitat del kernel de manera similar a com ho fa d en el cas del kernel polinòmic 3.1.

Sovint trobem el kernel expressat de manera que $\gamma = \frac{1}{2\sigma^2}$. Així doncs, ens queda:

$$k_{rbf}(x, z) = exp(-\gamma||x - z||^2) \tag{3.3}$$

Una curiositat amb relació al kernel RBF és que el seu *feature space* té un nombre infinit de dimensions [36].

3.2 Variables binàries

Una variable binària és aquella que únicament pot prendre els valors diferenciats de 0 i 1 [37]. És a dir, per a tota variable binària b, s'ha de donar que $b \in \{0, 1\}$. Aquest tipus de variables no ens són pas desconegudes en l'àmbit de la informàtica i les matemàtiques. És així perquè els autòmats programables en l'actualitat treballen amb el *bit* com a unitat d'informació bàsica, que és, en definitiva, una variable binària.

En l'àmbit de l'aprenentatge automàtic aquest tipus de variable és molt freqüent. A l'hora d'associar atributs als individus d'un conjunt de dades, el que pretenem és donar informació envers aquests. Les variables o característiques binàries ens permeten augmentar el nivell d'expressivitat i introduir conceptes com sí–no, tot–res o obert–tancat, entre d'altres. Generalitzant aquests conceptes, podem determinar que el valor 1 indica *presència* d'alguna cosa, mentre que el valor 0 n'indica l'*absència*.

En aquest cas particular no té massa sentit parlar d'*aplicacions* de les variables binàries. És així perquè poden emprar-se en qualsevol àmbit que requereixi aquest tipus d'expressivitat. Així doncs, passarem directament a descriure alguns tipus de funcions de kernel que podem emprar per tractar amb aquestes dades.

Funció de kernel d'Aitchison i Aitken

L'any 1976, J. Aitchison i CGG. Aitken varen proposar una funció de kernel definida en espais d'entrada discrets [38]. L'anomenaren *Aitchison and Aitken Kernel* (AA-kernel). Més concretament, els autors van definir la funció de kernel per estimar funcions de densitat en espais d'entrada binaris.

Curiosament, els autors, si bé és cert que ho intuïren, no varen arribar a demostrar que l'AA-kernel era semidefinit positiu. Més endavant, però, H. Y. Mussa ho va fer al seu article *The Aitchison and Aitken Kernel Function Revisited* [39]. A l'article, Mussa també comenta que l'AA-kernel s'ha emprat en múltiples experiments en l'àmbit de la quimioinformàtica, tot oferint bons resultats. Els problemes tenien com a objectiu classificar molècules representades en forma de variables binàries.

L'expressió de l'AA-kernel és la següent:

$$k(x_i, x_j; \lambda) = \lambda^{n-d(x_i,x_j)} \left(\frac{1-\lambda}{c-1}\right)^{d(x_i,x_j)}$$

On $x_i,\ x_j$ són vectors de dades binàries tals que $x_i, x_j \in \mathcal{B}^n$ amb $\mathcal{B} = \{0, 1, \ldots, c-1\}$, n és el nombre d'entrades discretes que té cada variable i $c \geq 2$ és el nombre de categories que pot tenir cada entrada. En el nostre cas, donat que estem tractant amb variables binàries, s'estableix $c = 2$. D'aquesta manera es dóna que $\mathcal{B} = \{0, 1\}$. A més a més, λ és un hiperparàmetre que queda definit en el rang $0.5 < \lambda < 1$.

A la definició de la funció, també podem trobar l'expressió $d(x_i, x_j)$. Es tracta d'una funció que denota el nombre de discordances entre els elements de x_i i x_j. Queda definida com:

$$d(x_i, x_j) = (x_i - x_j)^T (x_i - x_j)$$

Simple Matching Coefficient

El *Simple Matching Coefficient* (SMC) és un estadístic que es fa servir per mesurar la similitud entre mostres de dades binàries [40]. Més concretament, l'SMC mesura la ràtio del nombre total d'atributs coincidents respecte al nombre total d'atributs de les dades. És a dir:

$$SMC = \frac{\text{Núm. atributs coincidents}}{\text{Núm. total d'atributs}}$$

Malgrat que no hi ha massa marge d'error en la comprensió de l'expressió anterior, encara podem trobar una definició més acurada del càlcul de l'SMC.

Siguin $x_i,\ x_j$ dues dades binàries tals que $x_i,\ x_j \in \{0,1\}^n$ i:

N_{00} = Núm. d'atributs amb valor 0 tant en x_i com en x_j.
N_{01} = Núm. d'atributs amb valor 0 en x_i i valor 1 en x_j.
N_{10} = Núm. d'atributs amb valor 1 en x_i i valor 0 en x_j.
N_{11} = Núm. d'atributs amb valor 1 tant en x_i com en x_j.

L'SMC defineix l'índex de simililtud entre x_i i x_j com:

$$SMC(x_i, x_j) = \frac{N_{00} + N_{11}}{N_{00} + N_{01} + N_{10} + N_{11}} \tag{3.4}$$

Índex de Jaccard

L'índex de Jaccard és un altre estadístic emprat per mesurar la similitud entre dos conjunts de dades. Fou definit per l'ecòleg Paul Jaccard a l'article *The Distribution of the Flora in the Alpine Zone* [41]. A diferència de l'SMC, l'índex de Jaccard determina el grau de similitud entre dues mostres com la mida de la seva intersecció dividida per la mida del conjunt resultant de la seva unió. És a dir, essent A i B dues mostres:

$$Jac(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{3.5}$$

De la mateixa manera que hem fet amb l'SMC a l'equació 3.4, també podem definir el càlcul de l'índex de Jaccard fent ús dels termes $N_{00},\ N_{01},\ N_{10},\ N_{11}$. Novament, essent $x_i,\ x_j$ dues dades binàries tals que $x_i,\ x_j \in \{0,1\}^n$, ens queda que:

$$Jac(x_i, x_j) = \frac{N_{11}}{N_{01} + N_{10} + N_{11}} \tag{3.6}$$

En aquest cas, noti's que l'índex de Jaccard queda indefinit quan $N_{01} + N_{10} + N_{11} = 0$, ja que $Jac(x_i, x_j) = \frac{0}{0}$. Dit amb altres paraules, hi haurà una indefinició de l'índex de Jaccard quan totes les components de x_i i x_j coincideixin i prenguin el valor 0. Així doncs, en cas de programar aquesta funció de kernel haurem de definir el valor resultant de la comparació entre dades que compleixin aquestes característiques.

L'índex de Jaccard té un homòleg en l'àmbit de l'estimació de la similitud entre dades no binàries i s'anomena l'índex de Tanimoto [42]. El càlcul es duu a terme de la mateixa manera que hem definit a 3.5.

3.3 Variables categòriques

Les variables categòriques són aquelles que emmagatzemen dades amb un domini format per un nombre finit de valors o categories diferents [43]. Podem agrupar les variables categòriques en dues tipologies: variables categòriques *nominals* i *ordinals* [44].

Les variables categòriques *nominals* són aquelles els valors de les quals no presenten cap ordre intrínsec. Un possible exemple en seria una variable associada a la religió que segueix un individu. Aquesta podria prendre valors d'entre, per exemple, {Ateisme, Cristianisme, Islam, Budisme}. No és difícil adonar-se que no s'intueix cap possible ordenació objectiva dels valors expressats anteriorment i, per tant, la variable *religió* seria categòrica nominal.

D'altra banda, les variables categòriques *ordinals* són aquelles que prenen valors categòrics amb un ordre diferenciat. Per exemple, una variable que emmagatzema la informació associada a la qualitat d'un producte. Els possibles valors en ordre ascendent de satisfacció serien: {Molt dolent, Dolent, Normal, Bo, Molt Bo}.

En l'actualitat, l'aparició de variables categòriques en problemes sol significar haver de fer una conversió de variable categòrica a vector de variables binàries. És així a causa de la dificultat per trobar kernels que tractin directament amb variables categòriques en els programaris convencionals. El procediment seguit habitualment és el següent: essent C una variable categòrica amb domini $\{c_1,\ c_2, \ldots,\ c_n\}$. Per tractar aquesta variable amb una funció de kernel, generaríem un vector binari $B = (b_1,\ b_2, \ldots,\ b_n)$ amb $B \in \{0,1\}^n$. Tot individu tindria un vector B com a atribut en representació del valor que pren la variable C en cada cas. Si l'individu tingués associat el valor $C = c_j$ amb $1 \leq j \leq n$, el representaríem mitjançant el vector B de manera que $b_i = 1$ i $\forall i \in [1, n],\ i \neq j,\ b_i = 0$. Per exemple, si un haguéssim de representar el valor *blau* d'entre la llista de categories $\{$*blau*, *verd*, *vermell*$\}$, definiríem el vector $B = (1,\ 0,\ 0)$.

Aplicacions

Podem trobar aplicacions de les variables categòriques en múltiples àmbits d'estudi. N'és un exemple l'ús d'aquesta tipologia de variables en la medicina.

Al dataset *Breast Cancer Dataset*[1] del repositori UCI d'aprenentatge automàtic [45], podem trobar dades associades a pacients amb càncer de mama proporcionades pel *University Medical Center* de Ljubljana, Yugoslavia. Aquestes contenen atributs categòrics com el pit afectat (dret, esquerre), la mida del tumor (0–4, 5–9,..., 90–99) o l'estat menopàusic del pacient (<40, >40, premeno).

Un altre exemple en seria l'ús de variables categòriques per determinar el tipus de sang d'un individu. Per resoldre problemes mèdics de classificació ens pot interessar saber la tipologia de la sang de les persones en qüestió per tal de determinar possibles patologies. Els valors que podria prendre aquesta variable serien {O-, O+, B-, B+, A-, A+, AB-, AB+}.

El kernel *Univariate*

A l'article *Kernel Functions for Categorical Variables with Application to Problems in the Life Sciences* [46] es proposen una família de kernels positius definits per tractar problemes descrits fent ús d'informació de tipologia categòrica. El kernel que es proposa s'anomena *Univariate* i s'expressa com:

[1] *Breast Cancer Dataset*: `https://bit.ly/3iS9e1y`

$$k_{uni}(z_i, z_j; \alpha) = \begin{cases} h_\alpha(P_Z(z_i)) & \text{if } z_i = z_j \\ 0 & \text{if } z_i \neq z_j \end{cases} \tag{3.7}$$

On $z_i,\ z_j$ són dades categòriques i Z és la variable categòrica amb funció de probabilitat P_Z (*i.e.* $P_Z(z_l)$ indica la probabilitat que Z prengui el valor z_l). Aquesta funció de probabilitat està definida com $P_Z : Z \to [0, 1]$.

Noti's que α és un hiperparàmetre que serà necessari estimar. La funció h_α és una *funció d'inversió* definida com $h_\alpha : [0, 1] \to (0, 1)$ i s'expressa com:

$$h_\alpha(z) = (1 - z^\alpha)^{1/\alpha} \tag{3.8}$$

La funció d'inversió té com a objectiu introduir un comportament no lineal en el càlcul de l'índex de similitud entre les dades.

Valorant la funció de kernel des del punt de vista d'una funció de similitud, el kernel *univariate* retornarà un 0 (*i.e.* no semblança) sempre que els valors z_i i z_j no coincideixin. Podríem arribar a pensar que, en cas contrari, el kernel hauria de retornar un 1 (*i.e.* màxima semblança). Tot i que els kernels amb aquest comportament existeixen (i també s'exposen a l'article amb el nom d'*overlap kernels*), l'*univariate* fa un raonament més acurat del concepte de *semblança* i estableix que no totes les coincidències tenen la mateixa importància.

No és difícil adonar-se que si la probabilitat $P_Z(z_i)$ és molt baixa, el fet que hi hagi una coincidència entre dos individus amb valor z_i és molt significatiu. D'altra banda, si la probabilitat $P_Z(z_i)$ és molt elevada, significa que z_i és un valor molt comú. En conseqüència, el fet que dos individus prenguin el mateix valor no és tan rellevant. Aquest és, doncs, el paper de la funció h_α. En funció del paràmetre α, la valoració de la *importància* d'una coincidència en el valor z_i serà una o una altra.

3.4 Variables difuses

Les variables difuses són aquelles que ens permeten emmagatzemar informació amb un grau d'incertesa o inexactitud associat. Podem distingir fins a dos tipus de variables difuses: les variables difuses qualitatives i els nombres difusos. Abans, però, d'explicar en què consisteixen aquestes variables, és convenient introduir els conjunts difusos.

Tal com explica L. Zadeh al seu article *Fuzzy sets, fuzzy logic, and fuzzy systems* [47], essent $\mathcal{X}$ un espai d'objectes, un conjunt difús $\mathcal{A}$ sobre $\mathcal{X}$ es defineix com:

$$\mathcal{A} = \{x \in \mathcal{X} \ \mid \ \mu_{\mathcal{A}}(x) > 0\}$$

Diem que la funció $\mu_{\mathcal{A}}$ *caracteritza completament* el conjunt difús $\mathcal{A}$. En definitiva, s'encarrega de determinar si un element de l'univers de discurs $\mathcal{X}$ pertany o no al conjunt $\mathcal{A}$. És per això que $\mu_{\mathcal{A}}$ s'anomena *funció de pertinença*. Podem definir-la de la següent manera:

$$\mu_{\mathcal{A}} \ : \ \mathcal{X} \to [0, 1]$$

Així doncs, $\mu_{\mathcal{A}}(x) = 0$ implicaria que l'element $x \in \mathcal{X}$ no pertany a $\mathcal{A}$, mentre que $\mu_{\mathcal{A}}(x) = 1$ indicaria una pertinença total de x a $\mathcal{A}$.

Ara sí, passem a definir les diferents tipologies de variables difuses. Les *variables difuses qualitatives* (VDQ) o variables lingüístiques, són variables que prenen conjunts difusos com a valors.

És a dir, el domini d'una VDQ es pot entendre com un conjunt de conjunts difusos. Donat que un conjunt difús està caracteritzat per la seva funció de pertinença, el domini d'una VDQ també es pot interpretar com un conjunt de funcions de pertinença $\{\mu_{\mathcal{A}_1},\ \mu_{\mathcal{A}_2}, \ldots,\ \mu_{\mathcal{A}_n}\}$. Aquest conjunt s'anomena partició difusa [48].

Els *nombres difusos* (ND) són un altre tipus de variable difusa. A diferència de les VDQ, els ND estan formats per tres components: un valor nominal Ω, una desviació superior ds i una desviació inferior di. La notació més emprada per expressar els ND és $\Omega\ ^{ds}_{di}$. Noti's que $\Omega,\ di,\ ds \in \mathbb{R}$.

Els nombres difusos han estat bastament emprats en múltiples àmbits d'estudi. Un d'ells és l'enginyeria industrial, on els ND serveixen per expressar, entre d'altres, les toleràncies dels diferents materials de construcció. Per exemple, podem expressar la tolerància del diàmetre d'un tub d'acer com $40^{+0.2}_{-0.1}\ [mm]$.

Aplicacions

Un cop descrit el rerefons teòric en el qual s'emmarquen les variables difuses, creiem convenient comentar algunes de les seves aplicacions en diferents àmbits. D'aquesta manera pretenem justificar per què és necessari tenir una funció de kernel que tracti amb aquesta tipologia de dades.

Astronomia

Un dels camps d'aplicació de les variables difuses és l'astronomia. L'observació espacial es troba en l'ordre del dia dels avenços científics. És així perquè representa una font constant d'informació amb relació a l'origen de la Terra i l'univers. Alguns àmbits d'estudi són la documentació d'exoplanetes (planetes similars a la Terra), de supernoves, l'observació d'eclipsis en estrelles fora del Sistema Solar i fins i tot l'estudi i control d'asteroides que poden suposar una amenaça per la vida al nostre planeta.

Un dels principals problemes en l'observació espacial és discernir entre la llum provinent de fonts astronòmiques i la d'impactes al telescopi dels rajos còsmics (rajos de partícules d'alta energia que impacten en l'atmosfera terrestre a altes velocitats) [49].

Una manera simplificada d'explicar com emprar variables difuses per a la classificació de rajos lumínics és la següent: Definim una funció de pertinença fent servir les estrelles que ja tenim localitzades. Seguidament, generem, entre d'altres, una variable difusa que emmagatzemi la direcció d'on provenen els rajos ($direction = \{north,\ south,\ east,\ west\}$). A partir d'aquí s'intenta classificar si el raig prové de les estrelles o si es tracta d'un raig còsmic.

Medicina

Un altre àmbit d'ús de les variables difuses és la medicina. N'és un exemple el que comenta S. Alayón al seu article *A Fuzzy System for Helping Medical Diagnosis of Malformations of Cortical Development* [50]. En aquest, exposa les dificultats que hi ha a l'hora de diagnosticar malformacions del còrtex. Entre d'altres, hi ha una manca de consens en la nomenclatura o molta variabilitat entre les característiques de les imatges dels diferents tipus de malformacions. És per això que la diagnosi acaba basant-se en interpretacions subjectives de les troballes en les neuroimatges.

Alayón exposa una possible solució que implica generar un sistema que faci servir variables difuses per emmagatzemar la informació que recopila l'expert i, seguidament, emprar un model de classificació que ajudi a la diagnosi.

Kernel *cross product* sobre conjunts difusos

En aquest apartat comentarem alguns dels kernels que proposa J. Guevara en el seu article *Kernels on fuzzy sets* [51]. Més concretament, començarem enunciant el kernel *cross product* per a conjunts difusos.

Abans, però, d'exposar la funció de kernel, establirem algunes convencions. Definim $\mathcal{F}(\Omega)$ com el conjunt de tots els conjunts difusos sobre l'univers de discurs Ω i les funcions de pertinença com $\mu : \Omega \to [0,1]$. Donat que els conjunts difusos estan caracteritzats, tal com hem comentat en apartats anteriors, per la seva funció de pertinença, farem servir indistintament la notació $\mu(x)$ tant per denotar el *grau de pertinença* de l'element $x \in \Omega$ com per referir-nos al *conjunt difús* $\mu \in \mathcal{F}(\Omega)$.

Arribats a aquest punt, ja estem en disposició de donar la definició del kernel *cross product*. El kernel *cross product* és, en definitiva, una funció de kernel $k_\times : \mathcal{F}(\Omega) \times \mathcal{F}(\Omega) \to \mathbb{R}$ definida com:

$$k_\times(\mu_X,\ \mu_Y) = \sum_{\substack{x \in \text{supp}(\mu_X),\\ y \in \text{supp}(\mu_Y)}} k_1 \otimes k_2((x, \mu_X(x)),\ (y, \mu_Y(y)) \tag{3.9}$$

On, $\mu_X(x)$ i $\mu_Y(y)$ són els graus de pertinença dels elements $x, y \in \Omega$ als conjunts difusos μ_X i μ_Y, respectivament i k_1 i k_2 són dues funcions de kernels definides sobre $\Omega \times \Omega$ i $[0,1] \times [0,1]$ respectivament. El *support* (supp) d'un conjunt difús es defineix com:

$$\text{supp} = \{x \in \Omega \ \mid \ \mu_X(x) > 0\}$$

A més a més, el *producte tensorial* $k_1 \otimes k_2 : (\Omega \times [0,1]) \times (\Omega \times [0,1]) \to \mathbb{R}$ es defineix com:

$$k_1 \otimes k_2\ (x,\ \mu_X(x),\ y,\ \mu_Y(y)) = k_1(x,y) \cdot k_2(\mu_X(x), \mu_Y(y))$$

En altres articles [52], Guevara ha demostrat també que el kernel *cross product* $k_\times$ serà definit positiu sempre que els kernels k_1 i k_2 també ho siguin.

Noti's que en l'expressió general de la funció de kernel 3.9 estem assumint que el conjunt supp és finit. Ara bé, en cas que no sigui així i tinguem cardinalitat infinita dels conjunts $\text{supp}(\mu_X)$ i $\text{supp}(\mu_Y)$, també podem definir una variant del kernel *cross product* de manera que:

$$k_\times(\mu_X,\ \mu_Y) = \int \int_{\substack{x \in \text{supp}(\mu_X),\\ y \in \text{supp}(\mu_Y)}} k_1 \otimes k_2((x, \mu_X(x)),\ (y, \mu_Y(y))$$

Ara bé, en aquest cas serà necessari verificar que k_1 i k_2 són funcions de kernel contínues.

A banda de la funció de kernel que acabem d'exposar, J. Guevara també en proposa d'altres com ara el kernel *non-singleton* per a conjunts difusos [51] o el kernel basat en distàncies sobre conjunts difusos [53]. Tot i així, per a la tasca que ens ocupa no ens és necessari proposar més funcions de kernel per a variables difuses i, per tant, només n'enunciarem els noms.

4 Estudi i implementació del kernel d'agregació

En aquest capítol començarem enunciant i discutint el kernel d'agregació. Seguidament, durem a terme un ànalisi del software LIBSVM original en la seva última versió (3.25). Finalment, enunciarem i justificarem les modificacions que cal aplicar per integrar el kernel d'agregació a la llibreria.

4.1 El kernel d'agregació

Tal com hem comentat en capítols anteriors, l'objectiu principal d'aquest projecte és enunciar i dissenyar una funció de kernel que permeti tractar amb vectors formats per components de diferents tipus. Per exemple, el kernel en qüestió haurà de poder comparar dues mostres amb components de tipus [continu, continu, categòric, binari, categòric]. Aquest fet per si sol ens avança moltes coses amb relació a la implementació d'un kernel d'aquestes característiques. Tot i així, en aquesta secció ens limitarem a enunciar la nova funció de kernel.

La idea principal del que anomenarem **kernel d'agregació** (o *aggregation kernel*, en anglès) és la següent: siguin x i z els vectors d'entrada de cardinalitat $d \in \mathbb{R}^+$. Per a cada component $x_i,\ z_i$ durem a terme la crida a un kernel compatible amb la seva tipologia. Si tractem totes les components dels vectors amb un kernel per separat —ja veurem que, en alguns casos, no serà així—, obtindrem d valors resultants de les comparacions $k(x_i, z_i)$ amb $1 \leq i \leq d$. Per calcular el valor associat a la similitud entre x i z, farem el còmput de la mitjana aritmètica de les crides als subkernels $k(x_i, z_i)$. Vegem-ho de manera més formal:

Siguin x i z dues mostres de cardinalitat d, les components de les quals són de diferents tipologies. Definirem el kernel *inner* (de l'anglès «intern») com:

$$k_{inner}(x, z) = \frac{1}{d} \sum_{i=1}^{d} k_i(x_i, z_i) \tag{4.1}$$

On $\forall i \in [1, d]$, k_i és un kernel de domini compatible a les variables $x_i,\ z_i$. Farem referència a aquests kernels k_i com a *subkernels*.

Ara bé, no és difícil que sorgeixi la pregunta de per què el kernel no s'anomena d'«agregació», tal com hem comentat a l'inici de l'apartat. Doncs bé, és així perquè hem decidit separar la definició del kernel d'agregació en dues parts. El kernel *inner* de l'eqüació 4.1 s'encarrega del còmput de la mitjana aritmètica dels valors resultants de les crides als subkernels. Ara, però, volem introduir un hiperparàmetre que analitzarem més endavant. Vegem-ho:

$$k_{aggr}(x, z) = \frac{\exp(\gamma\ k_{inner}(x, z)) - 1}{\exp(\gamma) - 1} \tag{4.2}$$

Ara sí, hem arribat a l'equació final 4.2 del kernel d'agregació. Noti's l'aparició de l'hiperparàmetre γ. El paràmetre γ definirà fins a quin punt arriba la influència d'una mostra de *training* en la generació de l'hiperplà que separa les dades. Si al valor de γ és molt alt, acabarem disposant d'un hiperplà que separi les dades tenint en gran consideració les mostres properes al *decision*

boundary i pràcticament obviant les mostres llunyanes. En canvi, si γ pren un valor molt baix, les dades llunyanes tindran un pes considerablement elevat en la definició de l'hiperplà, mentre que les dades properes no marcaran la diferència. Tal com podem veure, cap de les dues situacions sembla òptima. És per això que a l'hora d'implementar el kernel, serà necessari estimar l'hiperparàmetre γ per tal que prengui un valor adequat a cada situació. L'única restricció amb relació a aquest hiperparàmetre és que $\gamma \in \mathbb{R}^+$.

4.1.1 Subkernels

Tal com hem explicat a l'apartat anterior, necessitarem disposar d'alguns kernels de diferents tipologies per fer-los servir com a *subkernels* del kernel d'agregació 4.2, 4.1. Arribats a aquest punt, entren en joc els kernels que hem estudiat al capítol associat a la recerca de les variables. Tot i haver recopilat una quantitat raonable de kernels, per qüestions de limitació temporal únicament implementarem com a subkernels les funcions associades als kernels RBF, *univariate*, Jaccard i *Simple Matching Coefficient* (SMC). A més a més, reformularem les funcions per evitar tenir massa hiperparàmetres a estimar, fet que provocaria una caiguda considerable de l'eficiència de l'algorisme. Així doncs, procedim a enunciar els nous subkernels:

RBF

Podeu trobar l'equació original del kernel RBF a 3.3. Si ens fixem, la funció de kernel en qüestió disposa d'un hiperparàmetre γ. En definitiva, ja estimarem el paràmetre γ a l'hora d'executar el kernel d'agregacio. És per aixó que, en aquest cas, establirem $\gamma = 1$, de manera que:

$$k_{rbf}(x, z) = \exp(-||x - z||^2) \tag{4.3}$$

Univariate

De la mateixa manera que amb el kernel d'agregació, el kernel *univariate* 3.7 disposa d'un paràmetre α que establirem com a $\alpha = 1$ per evitar fer-ne l'estimació. Així doncs, el nou subkernel quedarà com:

$$k_{uni}(z_i, z_j; \alpha) = \begin{cases} h_\alpha(P_Z(z_i)) & \text{if } z_i = z_j \\ 0 & \text{if } z_i \neq z_j \end{cases} \tag{4.4}$$

On la funció h_α original 3.8 estarà redefinida com:

$$h_\alpha(z) = 1 - z$$

Jaccard i SMC

Amb relació als kernels de Jaccard 3.6 i SMC 3.4, no hi ha cap hiperparàmetre que hàgim d'estimar. Així doncs, implementarem els kernels tal com els vàrem formular en una primera instància.

4.2 Anàlisi del software LIBSVM original

Aquesta secció té com a objectiu analitzar software original de LIBSVM. D'aquesta manera podrem identificar en quins punts del codi haurem d'aplicar modificacions per tal que la llibreria assimili i sigui capaç de treballar amb el nou kernel d'agregació.

```
1  struct svm_node {
2      int index;
3      double value;
4  };
```

Índexs de codis 4.1: Estructura de l'`svm_node` original.

Tal com hem explicat en apartats anteriors, LIBSVM [21] és un software integrat per a la classificació (C–SVM, ν–SVC) i regressió (ϵ–SVR) de vectors suport i per a l'estimació de distribució (*one class*–SVM). A més a més, permet tractar amb problemes de classificació multiclasse.

La versió a partir de la qual integrarem el nou kernel d'agregació serà la 3.25, publicada l'abril del 2021. A la pàgina oficial de LIBSVM [54] podeu trobar el fitxer *zip* per obtenir el codi de la versió esmentada.

Un cop som al directori principal, podem distingir diferents subdirectoris com ara `~/libsvm-3.25/{java, matlab, python}/` que inclouen els codis de la llibreria en els llenguatges de programació Java, MATLAB i Python. Ara bé, al directori base també podem identificar els codis que executen el software en C++: `svm.h`, `svm.cpp`, `svm-train.c`, `svm-predict.c` i `svm-scale.c`. Aqeuests programes contenen el codi font de la llibreria. A continuació, procedirem a identificar-ne les estructures i parts més importants.

4.2.1 Capçaleres i estructures bàsiques

El fitxer `svm.h` és el fitxer de capçalera de la llibreria. És a dir, conté la declaració de les estructures bàsiques sobre les quals es fonamenta el software. A més a més, també hi ha la definició (capçalera) de les funcions que a `svm.cpp` s'encarreguen d'entrenar l'SVM i cridar a les funcions de kernel, entre altres utilitats.

Si bé és cert que el fitxer inclou capçaleres de funcions, el que ens interessa estudiar i valorar són les estructures bàsiques. Aquestes estructures representen, en definitiva, els fonaments de la llibreria. Per tant, sembla raonable que en un futur pròxim haurem de modificar-les per tal d'implementar el nou kernel.

El node bàsic

L'estructura més bàsica és l'`svm_node`. Un `svm_node` és un `struct` que conté un `int` i un *double*. El seu codi és molt simple i el podem veure al fragment de codi 4.1.

L'`svm_node` té la funcionalitat d'emmagatzemar *una* característica (atribut) d'*un* individu determinat. És a dir, suposem que tenim un dataset de n entrades i m atributs. Podríem dir, doncs, que idealment tot individu $i \in [1, n]$ tindrà associades m característiques, de manera que: $i = (c_1,\ c_2, \ldots,\ c_m)$. Per poder representar aquest individu mitjançant `svm_node`'s, haurem de tenir m structs `svm_node`, que representaran els parells ($index$, $value$) de cada *feature* de l'individu en qüestió. Més endavant veurem com s'enllacen internament els `svm_node` per arribar a formar l'estructura de matriu que representa el dataset.

Emmagatzematge del problema

Una altra estructura important és l'`svm_problem`. Amb els `svm_node` hem vist com emmagatzemar la informació associada a un atribut d'un individu en concret. Ara bé, com s'*enllacen* els

```
1  struct svm_problem {
2      int l;
3      double *y;
4      struct svm_node **x;
5  };
```

Índexs de codis 4.2: Estructura de l'`svm_problem` original.

diferents nodes d'un individu? On s'emmagatzemen els *atributs de classe* d'aquests? Doncs bé, aquest és el paper que juga l'`svm_problem`. Podeu veure el codi al fragment 4.2.

L'`svm_problem` és un `struct` que conté tres atributs: `l` és un enter que serveix per emmagatzemar el nombre total d'individus (entrades) del dataset. En canvi, `*y` és un vector de `double` que emmagatzema els atributs de classe de cada individu. Noti's, doncs, que la mida de `y` és `l`. Finalment, `**x` és un vector de vectors d'`svm_node`. Per simplificar, ens podem imaginar `x` com una matriu $n \times m$ on cada fila representa un vector d'`svm_node`'s on cadascun d'aquests representa un atribut de l'individu associat a la fila en qüestió.

Gestió dels paràmetres i el model final

Arribats a aquest punt, ja sabem quines són les estructures que emmagatzemen les dades (atributs) i els problemes (conjunt de dades i atributs de classe). Ara bé, per executar una SVM ens seran necessaris més paràmetres. En són alguns exemples, el grau d del polinòmi per al kernel polinomic, el paràmetre γ per al kernel RBF o la C per a les C-SVM, entre d'altres.

Precisament l'`struct svm_parameter` s'encarrega d'emmagatzemar la informació esmentada al paràgraf anterior, entre d'altres. Sembla raonable que en diferents punts de l'execució de l'SVM —sigui durant l'entrenament o a l'hora de fer prediccions— requereixi alguns paràmetres per ser emprats en els algorismes. En resum, l'`svm_parameter` recopila la informació i els paràmetres necessaris per poder executar amb èxit els algorismes que conformen l'SVM.

Un cop hem entrenat l'SVM, tot emprant les estructures esmentades amb anterioritat, ens serà necessari emmagatzemar els resultats obtinguts. Posteriorment, els podrem fer servir per dur a terme prediccions sobre les dades de testatge. Aquesta és la funció que desenvolupa l'`struct svm_model`. Aquesta estructura de dades emmagatzema, entre d'altres, el nombre de vectors suport obtinguts en l'entrenament de l'SVM, els vectors suport com a tal i el nombre de classes. A més a més, també guarda una còpia de l'`svm_parameter`. És així perquè malgrat haver finalitzat l'etapa d'entrenament, els paràmetres ens seran necessaris per dur a terme les prediccions.

4.2.2 Funcions i mètodes principals

Un cop exposades les estructures bàsiques que LIBSVM fa servir per entrenar i fer prediccions mitjançant l'SVM, ha arribat el moment de comentar on es duen a terme les tasques importants i quins mètodes són els encarregats de fer-ho.

El fitxer `svm.cpp` conté totes les definicions, classes, algorismes i mètodes necessaris per entrenar l'SVM mitjançant les dades d'entrenament i fer, posteriorment, les prediccions sobre les dades de testatge. En aquesta secció descriurem algunes de les classes i mètodes d'aquest programa.

El primer amb què ens topem si revisem l'arxiu és amb la classe **Cache**. Aquesta classe s'encarrega de simular un sistema de cache per tractar eficientment amb les dades de la matriu de kernel. Cal tenir en compte que, en funció de la mida de les dades, una matriu de kernel pot ser extremadament gran. Tant, que podria donar-se el cas en què no hi hagi prou espai a la memòria per emmagatzemar-la. LIBSVM contempla aquest fet i ofereix la classe `Cache` i els seus mètodes com a solució.

El següent que ens agradaria comentar és la classe **Kernel**. Aquesta classe s'encarrega de tractar amb tot el que estigui relacionat amb les funcions de kernel. El que més destaca és que conté el codi dels algorismes de les diferents funcions de kernel.

Una altra de les classes interessants a comentar és **Solver**. Aquesta s'encarrega d'implementar l'algorisme de programació quadràtica per entrenar l'SVM. Més concretament, LIBSVM implementa l'algorisme de *Sequential Minimal Optimization* (SMO) proposat per Fan, Chen, Lin et al. al seu article *Working set selection using second order information for training support vector machines* [55].

Per acabar, comentarem breument que a `svm.cpp` també hi trobem altres mètodes independents com `svm_save_model` o `svm_load_model`, entre d'altres. Aquests en concret ens seran útils per desar el model generat posterior a l'etapa d'entrenament de l'SVM i carregar-lo a l'inici de l'etapa de testatge.

4.2.3 Etapa de generació del model

El fitxer `svm-train.c` conté el codi encarregat de dur a terme tots els passos necessaris previs a la crida al `svm_train(...)` de `svm.cpp`.

El programa té una estructura molt simple: està format per un `main` que s'encarrega de dur a terme les crides als diferents mètodes de `svm-train.c` i `svm.cpp`. A banda del mètode `main`, també destaquen els mètodes `parse_command_line`, que fa el *parse* dels paràmetres rebuts per la línia d'ordres i *read_ problem*, que llegeix el dataset en format LIBSVM.

4.2.4 Etapa de predicció

Un cop hem dut a terme l'entrenament de l'SVM mitjançant l'execució del fitxer `svm-train.c`, ha arribat el moment de dur a terme prediccions sobre les dades de testatge fent ús del model generat. Aquest és el paper del codi de `svm-predict.c`. Aquest requereix el model i les dades de testatge i, a partir d'aquí, duu a terme les prediccions. Finalment, retorna dos possibles resultats: *accuracy* en cas que es tracti d'un problema de classificació i *mean squared error* i *squared correlation coefficient* en cas que sigui un problema de regressió.

4.2.5 Escalat de les dades

El programa `svm-scale.c` s'encarrega de fer un escalat de les dades previ a l'etapa d'entrenament. A partir d'un dataset amb dades reals en format LIBSVM, el programa escala les dades en el rang per defecte $[-1, 1]$. Si l'usuari ho requereix, pot establir un rang d'escalat personalitzat mitjançant les opcions de la línia d'ordres.

Amb relació al nostre projecte, no farem massa incidència en aquest fitxer ni la funcionalitat que ofereix. És així, perquè considerem que hi ha una manera millor per fer l'escalat de les dades. En apartats posteriors comentarem quin serà el nou programa encarregat de fer l'escalat i la manera en que ho fa.

```
1  struct svm_node {
2      int index;
3      svm_node *next_node = NULL;
4
5      virtual ~svm_node() { next_node = NULL; }
6
7      virtual feature_type get_type() const = 0;
8      virtual bool set_value(char* value) = 0;
9      virtual svm_node* next_address() = 0;
10     virtual void print_info() const = 0;
11 };
```

Índexs de codis 4.3: Estructura del nou `svm_node`.

4.3 Implementació del kernel d'agregació

Arribats a aquest punt, hem enunciat el kernel d'agregació i hem analitzat el software original de LIBSVM. Ara, doncs, haurem de determinar les modificacions que ens permetin implementar i, posteriorment, emprar el kernel d'agregació dins la llibreria.

Començarem analitzant les estructures bàsiques de LIBSVM i aplicant els canvis necessaris per permetre l'ús del nou kernel.

4.3.1 Estructures bàsiques

Tal com hem vist a seccions anteriors, LIBSVM té diferents estructures bàsiques. A saber, els `svm_node`, l'`svm_problem`, l'`svm_parameter` i l'`svm_model`. En aquesta secció, revisitarem cadascuna d'aquestes estructures i estudiarem els canvis que cal aplicar-los per esdevenir compatibles amb la nova versió de la llibreria.

`svm_node`

En primer lloc modificarem l'estructura de l'`svm_node`. Tal com podem veure en la seva generació inicial 4.1, l'`svm_node` únicament permet emmagatzemar valors de tipus `double`, *i.e.* reals. Ara, però, el kernel d'agregació requereix tractar amb múltiples tipus de dades.

La manera en què inclourem aquest nou coneixement a LIBSVM és modificant l'`svm_node` i establint-lo com a un `struct` abstracte (virtual, en C++). Més concretament, serà un `struct` virtual pur. Podeu trobar la nova definició de l'`svm_node` al fragment de codi 4.3.

Un cop el node esdevé virtual, també hem de generar les subestructures que definiran el tipus concret de la dada que conté l'`svm_node`. Tal com podem veure, hem modificat l'atribut *value* del node original per un punter al següent `svm_node`. D'aquesta manera, podrem enllaçar nodes de diferents tipologies i generar tots els individus del dataset.

A més a més, hem afegit diferents mètodes virtuals per tal de facilitar el tractament amb els nodes. Noti's que mantenint l'estructura de l'`svm_node` i fent-la virtual aconseguim que tots els fragments de codi on es feien servir `svm_node`'s, puguin seguir funcionant de manera pràcticament normal. Ara bé, cal tenir cura dels punts on s'accedeix als antics atributs *value* dels `svm_node`. Tot i així, la llibreria LIBSVM està programada de manera que l'accés als nodes es duu a terme únicament a les funcions de kernel. Per tant, només hem de modificar els accessos als `svm_node` en un punt localitzat del codi.

```
1  struct categorical_node: svm_node {
2      char value[CATLEN];
3
4      // Destructora
5      ~categorical_node() {}
6
7      // Metodes
8      feature_type get_type() const override { return CATEGORICAL; }
9
10     // S'espera un string amb la categoria correcta ('value')
11     bool set_value(char* value) override {
12         strncpy(this->value, value, CATLEN);
13         if (strlen(value) > CATLEN-1) this->value[CATLEN-1] = '\0';
14         return true;
15     }
16
17     svm_node* next_address() override {
18         #warning "'svm_node.next_address' does not ensure that a 'svm_node' is
19         stored in the resulting address"
20         return this + 1;
21     }
22
23     void print_info() const override {
24         printf("[INFO] CATEGORICAL node with value %s.\n", value);
25     }
26 };
```

Índexs de codis 4.4: Estructura del `categorical_node`.

Donat que els codis de les estructures dels subnodes són relativament llargues, només mostrarem a manera d'exemple el subnode que contindrà les variables categòriques: el nou `categorical_node`. Podeu trobar el seu codi al fragment 4.4.

Tal com podem veure, un node categòric no és res més que un *string*. Hem emprat un *c–string* i no pas una instància de la classe `string` perquè LIBSVM no fa ús de cap estructura C++ preestablerta (com ara `vector` o `string`). Els creadors ho argumenten per qüestions d'eficiència i de maniobrabilitat del codi. Per la nostra part, tot i que no compartim la filosofia, hem decidit seguir aquests principis per evitar un estil de codi fragmentat.

La mida de les categories s'ha establert mitjançant una constant `CATLEN` definida inicialment a 12 bytes.

Ja hem vist tant en el codi de l'`svm_node` com del `categorical_node` l'ús d'un tipus d'atribut anomenat `feature_type`. Aquest tipus de variable és una enumeració que hem creat per facilitar la identificació dels tipus de node. A la versió original de LIBSVM ja hi havia enumeracions per identificar els tipus de kernel i de SVM.

`svm_problem`

Amb relació a l'`svm_problem` 4.2 no hem hagut d'aplicar cap modificació. En definitiva, un problema segueix estant format pel nombre d'instàncies, els atributs de classe de cadascuna d'aquestes i les mateixes instàncies (que segueixen essent, en definitiva, `svm_node`'s).

```
1  struct aggregation_kernel_params {
2      // Per indicar el num. de feature a la qual ens referim
3      int index;
4
5      virtual kernel_type get_type() const = 0;
6      virtual bool free_params() { return false; }
7      virtual void print_info() const = 0;
8  };
```

Índexs de codis 4.5: Estructura de l'`aggregation_kernel_params`.

```
1  struct univariate_params: aggregation_kernel_params {
2      double *probs;      // Probabilitats de cada categoria
3      double alpha;       // Hiperparametre alpha per a la funcio 'h'
4      int numCategories;  // Num total de categories
5      char **categories;  // Categories
6
7      // Metodes ...
8  };
```

Índexs de codis 4.6: Resum de l'estructura de l'`univariate_params`.

`svm_parameter`

En l'àmbit dels paràmetres de la SVM, ara hem de tenir en compte que cada atribut o *feature* de les dades pot tenir associat un kernel diferent. És així perquè, en definitiva, d'això tracta el kernel d'agregació. Així doncs, ja no ens és útil tenir un únic `svm_parameter`, sinó que en necessitem tants com nombre d'atributs tingui el dataset.

Per implementar aquesta novetat hem generat una nova estructura anomenada `aggregation_kernel_params`. Podeu veure el seu codi al fragment 4.5.

Podem veure que, de la mateixa manera que l'`svm_node`, l'estructura dels paràmetres del kernel d'agregació és virtual. És així perquè, en definitiva, podem tenir múltiples tipus de variable i múltiples funcions de kernel per tractar-les. És possible que necessitem diferents paràmetres per a cadascun dels kernels. Un exemple de subparàmetre necessari és el del kernel *univariate*. Podem veure el seu codi (resumit) al fragment 4.6.

Per tractar amb el kernel *univariate* necessitarem disposar de totes les categories possibles i la probabilitat associada a cadascuna. A més a més, noti's que hem afegit l'atribut per emmagatzemar l'hiperparàmetre `alpha`. Si bé és cert que nosaltres, per simplificar, l'hem establert a $\alpha = 1$, deixem tot i així, l'atribut inclòs per si algú altre vol establir-lo d'una altra manera.

Al paràgraf anterior hem parlat d'«establir» el paràmetre `alpha`. És inevitable, doncs, que sorgeixi la pregunta: com establirem `alpha` i la resta d'`aggregation_kernel_params`? Respondrem a aquesta pregunta en apartats posteriors, en els quals introduirem una nova manera d'interactuar amb la llibreria a través d'un *fitxer de configuració*.

Per incloure els `aggregation_kernel_params` a l'estructura original d'`svm_parameter`, hem modificat aquesta última estructura, tot afegint un array d'`aggregation_kernel_params` i un enter que comptabilitzi el nombre de paràmetres (*i.e.* *features* o atributs) que emmagatzemarà aquest array.

`svm_model`

Amb relació a l'`svm_model`, no hem hagut d'aplicar-li cap modificació. És així perquè, recordem, l'`svm_model` ja conté un atribut que emmagatzema l'`svm_parameter`. Així doncs, donat que l'`svm_parameter` ara conté els `aggregation_kernel_params`, no cal modificar en cap cas l'estructura que emmagatzema el model generat.

4.3.2 Lectura de les dades: el fitxer de configuració

El format de lectura de les dades és exactament el mateix que hi havia amb la versió original de LIBSVM. Cal introduir un fitxer en el qual cada línia representi un individu del dataset. El format en el qual ha d'estar enumerat tot individu és el següent:

$$\texttt{C} \;\; \texttt{1:value}_1 \;\; \texttt{2:value}_2 \;\; \texttt{3:value}_3 \;\; \ldots \;\; \texttt{d:value}_d$$

On `C` és l'atribut de classe i d és el nombre d'atributs total. Noti's que en la nova versió de LIBSVM amb el kernel d'agregació no es permet l'aparició de *missing values* a les dades. És així perquè tractar amb els valors perduts s'escapa de l'àmbit d'estudi d'aquest TFG. La tasca s'establirà com a treball futur.

Ara, però, a diferència del que passava amb la versió original de LIBSVM, els atributs $\texttt{value}_i$ poden prendre valors no numèrics.

A banda del dataset, el nou software —sempre que s'utilitzi fent servir el kernel d'agregació— requerirà l'addició d'un *fitxer de configuració*. El fitxer de configuració té com a objectiu comptar i especificar les *features* o atributs que apareixen al dataset i els seus respectius tipus. El format d'aquest fitxer s'especifica a continuació:

```
D
index_1    kernel_type_1    [options_1]
index_2    kernel_type_2    [options_2]
...
index_m    kernel_type_m    [options_m]
```

On `D` és el nombre total d'atributs del conjunt de dades. Algú podria preguntar-se perquè cal especificar els $\texttt{index}_i$ i fins i tot perquè l'últim índex no té subíndex d. Tot plegat és així perquè el fitxer de configuració permet deixar índexs d'atributs sense definir. En aquests casos, el software interpretarà que totes les *features* amb índexs no introduïts al fitxer de configuració s'hauran de tractar mitjançant el kernel RBF. Ara bé, sempre s'haurà de complir que $m <= d$ i, a més a més, la numeració de les features ha de començar per l'índex 1.

No cal patir per possibles errors al fitxer de configuració. El nou software compta amb una funció de *parse* del fitxer i notificarà a l'usuari qualsevol errata que s'hagi pogut produir.

Tal com hem comentat en apartats anteriors, aquesta primera versió de LIBSVM amb el kernel d'agregació suportarà dades numèriques, categòriques i binàries. Així doncs, tindrà implementats els subkernels RBF 4.3, *univariate* 4.4 i els de jaccard 3.6 i SMC 3.4. En conseqüència, tots aquests kernels es poden especificar al fitxer de configuració. Les claus `kernel_type` que s'accepten al fitxer de configuració per a cadascun són `rbf`, `uni`, `jac` i `smc`, respectivament. Les `[options]` que permeten estan llistades a continuació:

```
rbf   -g [gamma]

uni   -n [nºcategories] -c [categories] -p [probabilitats] -a [alpha]
```

`jac` $\emptyset$

`smc` $\emptyset$

4.3.3 Estimació dels paràmetres γ i C

A l'hora d'executar el programa `svm-train`, que duu a terme la generació del model, podem especificar algunes opcions per indicar el comportament desitjat de l'SVM. No les comentarem totes —són moltes—, però sí que creiem convenient fer menció a aquelles opcions que hem implementat arran de la introducció del kernel d'agregació.

En primer lloc, l'opció `-t 5` és l'encarregada d'indicar que s'ha d'emprar el kernel d'agregació. A conseqüència d'aquesta opció, també es requereix l'opció `-cfg config_file` per tal d'indicar quin és i on es troba el fitxer de configuració.

A banda de les opcions esmentades anteriorment, també s'ha introduït la possibilitat d'estimar els paràmetres γ del kernel d'agregació 4.2 i C de l'SVM. Les opcions establertes per aquest afer són `-g -1` i `-c -1` respectivament. Si no s'especifica el contrari, es durà a terme l'estimació de C mitjançant un procés de 10–*fold cross validation*. Per establir un nombre diferent de *folds*, cal introduir l'opció `-v num_folds`.

Per estimar el valor de γ calcularem la distància quadràtica de totes les dades al *feature space*, de la següent manera:

Siguin x, x' dues dades qualsevols tals que $x \neq x'$,

$$||\Phi(x) - \Phi(x')||^2 = k_{inner}(x, x') + k_{inner}(x, x') - 2k_{inner}(x, x')$$

Tenint en compte que $k_{inner}(x, x') = k_{inner}(x, x')$, podrem fer el còmput de la distància com:

$$||\Phi(x) - \Phi(x')||^2 = 2(1 - k_{inner}(x, x'))$$

Seguidament, ordenarem els resultats obtinguts de menor a major. Per acabar prendrem el primer Q_1 i el tercer quartil Q_3 i establirem γ com el valor mitjà entre ambdós. Així doncs:

$$\gamma = \frac{Q_1 + Q_3}{2}$$

D'altra banda, un cop hàgim estimat el valor de γ, estimarem el valor de C. El procediment serà el següent: per a cada valor de C del conjunt $C = \{10^l \mid l = -2...2\}$, durem a terme un 10–*fold cross validation* (o un CV amb el nombre de *folds* que hagi establert l'usuari) amb les dades de *training* i la γ establerta anteriorment. Per avançar d'un valor de l al següent, hi haurà un *step*= +0.5. Per als problemes de regressió ens quedarem amb aquell valor de C que proporcioni un *CV Squared Correlation Coefficient* més alt. D'altra banda, quan tractem amb problemes de classificació, ens quedarem amb la C que proporcioni una precisió de CV més alta.

4.3.4 Implementació dels kernels

En aquesta secció comentarem alguns aspectes amb relació a la implementació dels kernels *inner*, d'agregació i la resta de subkernels.

En general el fet d'implementar nous kernels a la llibreria LIBSVM original no ha estat costós. Únicament hem hagut de modificar el mètode `k_function` de la classe `Kernel` tal com es mostra al fragment de codi 4.7.

```
1  double Kernel::k_function(const svm_node *x, const svm_node *y, const
       svm_parameter& param) {
2      switch(param.kernel_type) {
3          case LINEAR:
4              return dot(x,y);
5          case POLY:
6              return powi(param.gamma*dot(x,y)+param.coef0,param.degree);
7
8          [...]
9
10         case AGGREGATION: {
11             double kavg = inner_loop_aggr(x, y, param.numFeatures, param.
       aggr_params);
12             return (exp(param.gamma * kavg)-1) / (exp(param.gamma)-1);
13         }
14
15         default:
16             return 0;  // Unreachable
17     }
18 }
```

Índexs de codis 4.7: Fragment del mètode `k_function` de la classe `Kernel`.

El codi contingut al `case AGGREGATION` és l'associat al kernel d'agregació 4.2, mentre que la crida a `inner_loop_aggr` conté el codi del kernel *inner* 4.1. El mètode `inner_loop_aggr` únicament conté un for-loop que itera sobre els diferents atributs i fa les crides adients als subkernels.

Cal tenir en compte que LIBSVM és un software que ja té implementat un sistema per a l'execució dels diferents kernels. Mitjançant l'ús de la classe Cache —que hem comentat a apartats anteriors—, aconsegueix emmagatzemar les parts de la matriu de kernel que més convé en tot moment. D'aquesta manera, s'assoleix un elevat nivell d'eficiència i eficàcia.

Per acabar, únicament resta dir que tot el codi associat als nous kernels d'agregació, *inner* i subkernels es troben al fitxer `svm.cpp`.

5 Experimentació

5.1 Eines d'experimentació

Abans de començar a enunciar els experiments que hem dut a terme, creiem convenient comentar quines han estat les eines que hem generat per poder realitzar-los de manera satisfactòria.

En primer lloc, hem generat un fitxer Python `tools_aggr.py` que conté els següents mètodes: `parse_config_file`, que duu a terme el parse d'un fitxer de configuració i notifica els errors adients i `check_data_format`, que comprova que el dataset que rep per paràmetre estigui en format LIBSVM i, en cas que no sigui així, notifica l'error adientment.

El segon programa que hem proporcionat és `scale_aggr.py`. Si bé és cert que LIBSVM ja oferia un fitxer per escalar les dades (`svm-scale.c`), ara cal tenir en compte que podem estar tractant amb variables de tipus no numèric. Així doncs, `scale_aggr.py` té en consideració aquest fet rebent com a paràmetre el fitxer de configuració del conjunt de dades. A més a més, a diferència de l'escalat que oferia LIBSVM, hem establert que les dades s'escalin seguint amb la següent fórmula:

$$\frac{x - \hat{\mu}}{\hat{\sigma}}$$

On x seran les dades numèriques, $\hat{\mu}$ la mitjana mostral i $\hat{\sigma}$ la desviació estàndard (*standard deviation*).

Un altre programa interessant és `get_categories.py`. Ens hem trobat amb alguns datasets que tenien variables categòriques, però no enunciaven quines eren. Amb aquest programa, proporcionarem un dataset en format LIBSVM i un conjunt d'índexs dels atributs i ens retornarà les diferents categories dels *features* en qüestió.

Finalment, també proporcionem el programa `create_dataset.py`, que genera un conjunt de dades sintètic d'acord amb els principis que establirem en apartats posteriors durant l'etapa d'experimentació amb dades sintètiques.

Per acabar, únicament comentar que totes les eines i programes les podeu trobar al directori `~/libsvm-aggr/tools/` de la nova versió de la llibreria LIBSVM. A més a més, tots els programes disposen explícitament de missatges d'informació d'ús (*usage*) per qualsevol que necessiti emprar-los.

5.2 Experimentació amb dades sintètiques

Els primers experiments que durem a terme es fonamentaran en el fet d'emprar un conjunt de dades generat de manera sintètica. El prinipal atractiu d'aquest tipus de proves serà permetre'ns crear les dades en funció de les nostres necessitats. És a dir, entre d'altres, podrem establir a plaer el nombre de dades, de caractarístiques i, fins i tot, de classes que tindran els nostres datasets.

En els propers apartats començarem definint els fonaments dels conjunts de dades que emprarem. Seguidament, analitzarem els conceptes més importants de la implementació del generador de datasets. Per acabar, executarem l'SVM emprant el kernel d'agregació sobre les dades generades i n'analitzarem els resultats obtinguts.

5.2.1 Definició dels experiments

Abans d'iniciar el procés d'implementació del generador de dades i posterior execució dels experiments, haurem de determinar un conjunt de nocions bàsiques. Entre aquestes destaquen el nombre d'experiments que realitzarem amb dades sintètiques, el nombre d'entrades o individus que contindrà cada conjunt de dades i el nombre i tipus de les característiques de cada individu del dataset.

Durem a terme un total de tres experiments. Cada experiment emprarà un dataset sintètic amb diferent nombre d'entrades. El nombre de dades serà de 100, 1000 i 10000, respectivament. L'objectiu de la presa d'aquesta decisió és poder executar l'SVM i el kernel d'agregació amb un conjunt de dades petit, mitjà i gran i estudiar-ne el seu rendiment per a les diferents mides dels conjunts.

Els conjunts de dades tindran un total de vint característiques (*features*) per entrada sense valors perduts (*missing values*). Deu d'aquestes característiques seran contínues i la resta (deu) seran binàries. Els individus tindran assignat un atribut de classe d'entre dos de possibles. En apartats posteriors especificarem com es determinarà el valor de cada *feature* i l'atribut de classe en cada cas.

5.2.2 Implementació del generador de dades

Per tal de generar les dades seguint la definició establerta a l'apartat anterior hem creat el fitxer Python `create_dataset.py` que es troba al directori `~/libsvm-aggr/tools/` de la llibreria. Podeu veure les instruccions d'ús introduïnt la comanda `python3 create_dataset.py -h`.

Pel que fa a l'explicació que durem a terme en aquest apartat, només ens interessa saber que el programa requereix com a paràmetres el nombre d'entrades n i el nombre de *features* de cada tipus m que haurà de contenir el dataset sintètic generat. Per exemple, la crida `python3 create_dataset.py -n 100 -m 10` crearà un dataset de 100 entrades i 10 *features* contínues i binàries (20 en total). En el nostre cas, el n variarà en funció de si estem generant el dataset petit (100), mitjà (1000) o gran (10000) i m serà constant a 10 per a tots els conjunts de dades.

La primera qüestió que abordarem serà el procediment a seguir per generar els valors de les 10 característiques binàries. La idea bàsica serà generar 10 dades binàries independents. En primer lloc, prendrem 10 probabilitats seguint una distribució uniforme en l'interval $(0, 1)$, de manera que $\forall i \in [1, 10],\ p_i \in (0, 1)$, on p_i és la probabilitat associada a la *feature* f_i. Seguidament, emprarem una distribució de bernoulli per determinar el valor binari de cada característica, de manera que $f_i = Ber(p_i)$.

Un cop determinats els valors de les característiques binàries, haurem d'establir un procediment per determinar-ne el de les contínues. Per generar les dades emprarem una distribució normal multivariada $\mathcal{N}(\mu, \Sigma)$, on μ serà un vector de 10 mitjanes i Σ una matriu de covariàncies. Noti's que Σ haurà de ser semidefinida positiva i simètrica. Per generar el vector de mitjanes μ prendrem 10 valors aleatoris en el rang $[-5, 5)$. Per generar, en canvi, la matriu de covariàncies Σ, crearem una matriu quadrada aleatòria A i definirem $\Sigma = AA^T$.

Arribats a aquest punt, només resta definir la manera en què atorgarem l'atribut de classe a cada entrada generada. Hem establert les següents normes:

if paritat(x_{ib}) parell $\rightarrow c_i = 0$.

else if $D_M(x_{ic}, \mu) < \mathrm{T} \rightarrow c_i = 1$.

else $c_i = 0$.

On $x_i,\ i \in [1, n]$ és una entrada del dataset, c_i és el seu atribut de classe, x_{ib} i x_{ic} és el subvector de característiques binàries i contínues, respectivament i D_M és la distància de Mahalanobis. Inicialment establirem $T = \text{average}(\mu)$. Tot i així, modificarem el valor de T i reassignarem els atributs de classe fins que obtinguem el $50\% \pm 5\%$ de les dades amb l'atribut de classe 0 i la resta amb l'atribut de classe 1.

5.2.3 Execució i anàlisi dels resultats

Seguidament procedirem a executar la nova versió del software LIBSVM amb els datasets sintètics. Per a l'etapa de *training* establirem que s'estimin els paràmetres γ i C amb un 10 *fold cross validation* (opcions `-g -1` i `-c -1`, respectivament). Determinarem que s'usi una C-SVC (opció `-s 0`) i, evidentment, el kernel d'agregació (opció `-t 5`). A més a més, per a cada conjunt de dades (petit, mitjà i gran), emprarem un 80% de les dades per conformar els respectius conjunts de *training*, mentre que el 20% restant de les dades es faran servir per al *testing*. Les crides seran, doncs, de la forma:

```
./svm-train -s 0 -t 5 -cfg syn.cfg -g -1 -c -1 data.train model
./svm-predict data.test model output
```

Podeu trobar els resultats associats al temps d'execució de l'SVM amb els datasets petit (100 entrades), mitjà (1000 entrades) i gran (10000 entrades) a les taules 5.1, 5.2 i 5.3, respectivament.

Com era d'esperar, els temps d'execució han anat en augment a mesura que el nombre de dades era major. Podem veure que l'estimació del paràmetre C és la que més temps requereix d'entre les tasques realitzades en l'etapa de generació del model. Tot i així, aquest fet no suposa cap sorpresa ni inconvenient, tenint en compte que estem aplicant un 10 *fold cross validation* per estimar-la en cada cas.

Amb relació als errors obtinguts, el *dataset petit* ha resultat amb un error de *testing* del **40%** (8 mostres mal classificades respecte a un total de 20 mostres). L'execució amb el *conjunt mitjà* ha obtingut un error del **3.5%** (7 mostres mal classificades d'un total de 200). Finalment, l'execució amb el *conjunt gran* ha obtingut un error del **2.45%** (49 mostres mal classificades d'un total de 2000).

Podem veure que, com era d'esperar, a mesura que augmentem el nombre de dades, l'error disminueix. Tot i així, per poder afirmar que els resultats són positius, restaria comprovar l'interval de confiança en el qual s'emmarca l'error.

Essent n el nombre de dades del conjunt de *testing* en cada cas i $\hat{p}$ l'error obtingut, sabem que $\hat{p}$ segueix una distribució binomial $Bin(n, p)$, on p és l'error real. A més a més, pel teorema de De Moivre–Laplace [56], sabem que una distribució binomial es pot aproximar adequadament per una distribució gaussiana sempre que es compleixi que (1) n sigui suficientment gran, que (2) $\hat{p} > 0.05$ i que (3) $\hat{p}(1 - \hat{p}) > 0.05$. Podem veure com en tots tres casos es compleixen les

Taula 5.1: Temps d'execució resultant de l'experimentació amb el dataset *sintètic petit*.

Concepte	**Temps d'execució (s)**	**% respecte al total**
Estimació γ	0.0001	0.08
Estimació C	0.12	98.28
Generació del model	0.001	1.20
Predicció	0.0005	0.44
Total	**0.12**	**≈100**

Font: Elaboració pròpia.

Taula 5.2: Temps d'execució resultant de l'experimentació amb el dataset *sintètic mitjà*.

Concepte	**Temps d'execució (s)**	**% respecte al total**
Estimació γ	0.006	0.04
Estimació C	15.85	98.90
Generació del model	0.14	0.87
Predicció	0.03	0.19
Total	**16.02**	**≈100**

Font: Elaboració pròpia.

condicions requerides.

Podem obtenir un interval de confiança per a p al 95% de significança fent ús de l'equació 5.1 [57], on n i $\hat{p}$ mantenen el significat especificat en paràgrafs anteriors.

$$\hat{p} \pm 1.967\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \tag{5.1}$$

L'interval de confiança obtingut per a tots tres resultats és, doncs:

- Dataset petit ($n = 20,\ \hat{p} = 0.4$): 0.4 ± 0.215
- Dataset mitjà ($n = 200,\ \hat{p} = 0.035$): 0.035 ± 0.026
- Dataset gran ($n = 2000,\ \hat{p} = 0.0245$): 0.0245 ± 0.007

D'aquests resultats en podem extreure que quan augmentem el nombre de dades, reduïm l'interval de confiança. Dit d'una altra manera, el marge d'error de $\hat{p}$ es redueix. Així doncs, podem concloure que a mesura que augmentem el nombre de dades, no només rebaixem l'error de predicció $\hat{p}$, sinó que també reduïm el seu marge d'error.

5.3 Experimentació amb dades reals

Un cop realitzada l'experimentació amb les dades sintètiques, creiem que pot ser interessant testar el nou software amb dades obtingudes del món real. El procediment que seguirem serà dur

Taula 5.3: Temps d'execució resultant de l'experimentació amb el dataset *sintètic gran*.

Concepte	**Temps d'execució (s)**	**% respecte al total**
Estimació γ	0.22	0.01
Estimació C	3404.76	98.82
Generació del model	37.21	1.08
Predicció	3.09	1.09
Total	**3445.28**	**≈100**

Font: Elaboració pròpia.

a terme una recerca per recopilar diferents conjunts de dades que compleixin amb les condicions que requereix la nova llibreria. A saber, que no hi hagi valors perduts o que les *features* no siguin únicament contínues, entre d'altres.

Hem pogut trobar dos datasets amb un nombre de dades raonable que satisfan els nostres requisits: el *Contraceptive Method Choice Dataset* i el *Flags Data Set*. Ambdós conjunts es poden trobar a l'*UCI Machine Learning Repository* [45].

5.3.1 *Contraceptive Method Choice Data Set*

El dataset *Contraceptive Method Choice Data Set* [58] es troba disponible al repositori de dades per a Aprenentatge Automàtic UCI [45]. Aquest conté un subconjunt de les dades de la *National Indonesia Contraceptive Prevalence Survey* que recopila informació envers dones casades d'Indonèsia que o bé no estan embarassades o bé no ho saben.

El conjunt de dades està pensat per ésser tractat com a un problema de classificació i no conté *missing values*. A més a més, disposa de 1473 entrades i 9 atributs de tipus Categòric, Enter i Binari. Més concretament, les *features* que es recopilen de cada individu (dona casada) són les següents:

1. Edat (Numèrica).
2. Nivell educatiu (Categòrica: $\{1 = \text{baix},\ 2,\ 3,\ 4 = \text{alt}\}$).
3. Nivell educatiu del marit (Categòrica: $\{1 \equiv \text{baix},\ 2,\ 3,\ 4 \equiv \text{alt}\}$).
4. Nombre de fills que té (Numèrica).
5. Religió (Binària: $0 \equiv$ no islam, $1 \equiv$ islam).
6. Treballa? (Binària: $0 \equiv$ sí, $1 \equiv$ no)
7. Professió del marit (Categòrica: $\{1,\ 2,\ 3,\ 4\}$).
8. Nivell de vida (Categòrica: $\{1 \equiv \text{baix},\ 2,\ 3,\ 4 \equiv \text{alt}\}$).
9. Com considera l'exposició als mitjans? (Binària: $0 \equiv$ bona, $1 \equiv$ dolenta)
10. Mètode anticonceptiu emprat (Atribut de classe: $\{1 \equiv$ No en fa servir, $2 \equiv$ A llarg termini, $3 \equiv$ A curt termini$\}$).

Taula 5.4: Temps d'execució resultant de l'experimentació amb el dataset *contraceptive*.

Concepte	**Temps d'execució (s)**	**% respecte al total**
Estimació γ	0.008	0.04
Estimació C	22.27	98.65
Generació del model	0.25	1.1
Predicció	0.05	0.21
Total	**22.57**	**≈100**

Font: Elaboració pròpia.

Tal com podem veure, l'atribut que determina la classe de cada instància és el número 10. Així doncs, en l'etapa de predicció, haurem de determinar per a cada individu del conjunt de testatge si no fa ús dels mètodes anticonceptius (classe 1), si n'utilitza a llarg termini (classe 2) o si n'usa a curt termini (classe 3).

Tenint en compte la tipologia de les variables anteriorment exposades, haurem de definir quin kernel emprarem per a cadascuna a l'hora d'executar l'SVM amb el kernel d'agregació. Per a totes les característiques numèriques emprarem el kernel RBF. Per a les categòriques farem servir el kernel Univariat. Finalment, per a les *features* binàries, farem servir el kernel de Jaccard. D'acord amb això, generarem el fitxer de configuració.

Execució i anàlisi dels resultats

A continuació durem a terme l'execució de l'SVM amb el dataset *Contraceptive Method Choice Data Set*. Per a l'etapa de *training* establirem que s'estimin els paràmetres γ (opció `-g -1`) i C (opció `-c -1`). Determinarem l'ús d'una C-SVC (opció `-s 0`) amb el nou kernel d'agregació (opció `-t 5`). Farem servir el 80% de les dades per generar el model (fitxer `cmc.train`) i el 20% restant per a l'etapa de *testing* (fitxer `cmc.test`). Noti's que el fitxer de configuració s'estableix amb l'opció `-cfg cmc.cfg`. A més a més, el model generat es desarà a `cmc.model` i l'output de l'etapa de *testing* a `cmc.output`. Així doncs, les crides són les següents:

```
./svm-train -s 0 -t 5 -cfg cmc.cfg -g -1 -c -1 cmc.train cmc.model
./svm-predict cmc.test cmc.model cmc.output
```

Podeu trobar els resultats associats al temps d'execució a la taula 5.4.

Tal com podem veure, els resultats amb relació al temps d'execució són molt favorables. Tant la generació del model com les prediccions es duen a terme de manera pràcticament immediata. De la mateixa manera que amb els experiments sintètics, l'estimació del paràmetre C és la tasca que més temps requereix (més del 98% del temps). El motiu de tot plegat és haver establert un 10 *fold cross validation* per estimar el paràmetre.

Per acabar, comentarem l'error de predicció obtingut. En aquest cas, hem obtingut un error de predicció del **48.81%**. En una primera instància podríem arribar a pensar que és un mal resultat. Tot i així, hem pogut comparar els nostres resultats amb els obtinguts a l'article «*A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty-three Old and New Classification Algorithms*» [59]. A l'estudi fan servir el dataset *contraceptive* amb múltiples

algorismes d'aprenentatge automàtic com ara *linear discriminant analysis* (LDA), *quadratic discriminant analysis* (QDA) o logistic discriminant analysis (LOG) entre molts d'altres. Malgrat que no podem comparar els resultats en termes del temps d'execució (perquè entre els algorismes emprats a l'estudi no s'inclou l'SVM), ens podem fer una idea del rang d'errors de predicció obtinguts per tal d'estimar si el nostre resultat és acceptable o no.

El rang d'errors obtinguts a l'estudi mencionat oscil·la entre un mínim del 43% i un màxim del 60%. Així doncs, sembla raonable pensar que l'elevat error obtingut en la nostra execució (48.81%) és degut a una mala correlació entre les dades i no pas a un mal comportament del nou kernel d'agregació.

5.3.2 *Flags Data Set*

El dataset *Flags Data Set* [60] es troba disponible a l'*UCI Machine Learning Repository* [45]. Aquest conté informació associada a diferents països del món i les seves banderes. Les dades han estat recopilades del «*Collins Gem Guide to Flags*» [61].

El conjunt consta d'un total de 194 instàncies i 30 atributs. La tipologia dels atributs és numèrica, categòrica i binària. El dataset està pensat per resoldre, tal com ens interessa, problemes de classificació. A més a més, no hi ha valors perduts.

Per a cada país, es recopila la següent informació (entre parèntesis especifiquem el tipus de dada):

1. Nom (String).
2. Continent (Categòrica: {1=N.Amèrica, 2=S.Amèrica, 3=Europa, 4=Àfrica, 5=Àsia, 6=Oceania}).
3. Quadrant geogràfic tenint en compte el *centre* del món com el punt intersecció entre el meridià de Greenwich i el paral·lel d'Equador (Categòrica: {1=NE, 2=SE, 3=SW, 4=NW}).
4. Àrea (Numèrica).
5. Nombre d'habitants (Numèrica).
6. Idioma (Categòrica: {1=Anglès, 2=Espanyol, 3=Francès, 4=Alemany, 5=Eslau, 6=Altres Indoeuropees, 7=Xinès, 8=Àrab, 9=Japonès/Turc/Finès/Hongarès, 10=Altres}).
7. Religió (Atribut de classe: {0=Catòlica, 1=Altres cristianes, 2=Musulmana, 3=Budista, 4=Hindú, 5=Ètnica, 6=Marxista, 7=Altres}).
8. Nombre de barres verticals a la bandera (Numèrica).
9. Nombre de barres horitzontals a la bandera (Numèrica).
10. Nombre de colors diferents a la bandera (Numèrica).

11-17. Presència de color {Vermell, Verd, Blau, Groc/Daurat, Blanc, Negre, Taronja} a la bandera (Binària: 0=Absència, 1=Presència).

18. Color predominant a la bandera (Categòrica).

19-23. Nombre de {Cercles, Creus verticals, Creus diagonals, Seccions en quarts, Símbols de sol o estrella} a la bandera (Numèrica).

24-28. Presència de {Símbol de lluna creixent, Triangles, Imatge inanimada, Imatge d'un ésser animat, Text} a la bandera (Binària: 0=Absència, 1=Presència).

29. Color de la cantonada superior esquerra (Categòrica).

30. Color de la cantonada inferior dreta (Categòrica).

A diferència de l'ocorregut en l'anàlisi del dataset *Contraceptive Method Choice* de l'apartat anterior, ara hi ha algunes qüestions amb relació als atributs que hem d'abordar abans de poder iniciar l'entrenament de l'SVM amb el kernel d'agregació. Ho veurem a continuació.

En primer lloc, podem veure que l'atribut 1 és de tipus *string*. Cal tenir en compte que no es tracta d'una variable de tipus categòric. Cada país tindrà el seu nom i no se'n repetirà cap. Així doncs, el kernel que hauríem d'emprar per comparar dos noms de països diferents ha de ser necessàriament un kernel orientat a strings. Donat que en l'actualitat la versió de LIBSVM amb el kernel d'agregació no inclou kernels per al tractament d'strings, desestimarem aquest atribut a l'hora d'entrenar l'SVM i fer les posteriors prediccions.

També podem veure que, malgrat que els atributs 18, 29 i 30 són de tipus categòric, no se'ns faciliten les categories corresponents. Mitjançant l'eina `get_categories.py` (exposada a l'inici de l'experimentació) hem pogut constatar que les categories són les següents:

18. {black, blue, brown, gold, green, orange, red, white}

29. {black, blue, gold, green, orange, red, white}

30. {black, blue, brown, gold, green, orange, red, white}

Finalment, només resta comentar que l'etiqueta de classe de cada instància ve donada per l'atribut 7 de la llista. Així doncs, el problema a resoldre es basarà a classificar els països de les dades de testatge entre les religions catòlica, altres cristianes, musulmana, budista, hindú, ètnica, marxista o altres.

Execució i anàlisi dels resultats

Novament, durem a terme l'execució de l'SVM. Ara, però, emprarem el dataset de *flags*. Per a l'etapa de training establirem que s'estimin els paràmetres γ (opció `-g -1`) i C (opció `-c -1`). A més a més, emprarem una C-SVC (opció `-s 0`), juntament amb el nou kernel d'agregació (opció `-t 5`). De la mateixa manera que amb els anteriors experiments, emprarem un 80% de les dades per generar el model (fitxer `flag.train`) i el 20% restant per dur a terme l'etapa de *testing* (fitxer `flag.test`). Noti's que el fitxer de configuració s'introdueix amb l'opció `-cfg flag.cfg`. Finalment, el model generat es desarà a `flag.model` i l'output de l'etapa de *testing* a `flag.output`. Les crides són les següents:

```
./svm-train -s 0 -t 5 -cfg flag.cfg -g -1 -c -1 flag.train flag.model
./svm-predict flag.test flag.model flag.output
```

Tal com podem veure als resultats de la taula 5.5, els temps d'execució semblen favorables. Novament, la generació del model i l'etapa de predicció és pràcticament immediata. De la mateixa manera que amb la resta d'experiments, l'estimació del paràmetre C és la que requereix més temps (més del 80% del total). És així a conseqüència de l'elevat nombre de *folds* que hem establert a l'hora de dur a terme *cross validation*.

Taula 5.5: Temps d'execució resultant de l'experimentació amb el dataset *flags*.

Concepte	**Temps d'execució (s)**	**% respecte al total**
Estimació γ	0.0003	0.02
Estimació C	1.49	98.5
Generació del model	0.02	1.32
Predicció	0.002	0.15
Total	**1.51**	**≈100**

Font: Elaboració pròpia.

Amb relació a l'error de predicció, aquest cop ha estat d'un **23.08%**. Els resultats, doncs, semblen favorables. Cal destacar que és molt curiós com, tenint en compte els nostres resultats, hi ha una clara correlació entre la bandera d'un país (els colors que la formen, el nombre de ratlles, el nombre de creus verticals i inclinades, etc.) i la religió majoritària en aquest.

En aquest cas, seguirem un procediment alternatiu per valorar la qualitat dels nostres resultats. En els experiments anteriors o bé hem calculat l'interval de confiança o bé hem comparat els resultats amb els d'altres estudis. Ara, però, hem pensat a fer-ho diferent.

En moltes ocasions, quan els conjunts de dades contenen atributs de diferents tipologies (numèrica, categòrica, binària, etc.), s'opta per tractar totes les dades com si fossin numèriques. Per exemple, si tenim un atribut categòric que pot prendre els valors {vermell, verd, blau}, transformarem tots els atributs d'aquest tipus d'acord amb les associacions {vermell=1, verd=2, blau=3}. Un cop convertides les dades, s'executa l'SVM fent ús d'un kernel per a variables numèriques, com pot ser, per exemple, l'RBF. Aquesta metodologia estalvia temps als investigadors, ja que eviten la creació d'un kernel fet a mida com pot ser el kernel d'agregació. Tot i així, és possible que sacrificar l'expressivitat que proporcionen les diferents tipologies de variables tingui un impacte negatiu en els resultats obtinguts.

A continuació, seguirem la metodologia enunciada al paràgraf anterior per resoldre el problema de *flags*. El primer que farem serà, en cas que sigui necessari, convertir totes les dades del dataset a numèriques. Seguidament, executarem la versió original de LIBSVM 3.25 i compararem l'error de predicció obtingut amb el que ha proporcionat l'execució del mateix problema emprant el kernel d'agregació. Per convertir les dades categòriques a numèriques seguirem el següent procediment:

Sigui c una variable categòrica i $[c_1,\ c_2, \ldots,\ c_n]$ la llista de possibles categories, substituirem totes les aparicions de c_i pel seu índex i a la llista. Així doncs, si tenim una variable categòrica amb possibles valors [*petit*, *mitjà*, *gran*], substituirem totes les aparicions de *petit* pel valor 1, les de *mitjà* pel valor 2 i les de *gran* pel valor 3.

Un cop generat el nou conjunt de dades, farem servir la versió 3.25 de LIBSVM per resoldre el problema. Establirem l'ús d'una C-SVC (opció `-s 0`) i l'ús del kernel RBF (opció `-t 2`). Destinarem el 80% de les dades per generar el model (fitxer `flag.num.train`) i el 20% restant a l'etapa de *testing* (fitxer `flag.num.test`). Noti's que no cal emprar cap fitxer de configuració perquè estem executant la versió original de la llibreria. Finalment, el model generat es desarà a `flag.num.model` i l'output de l'etapa de *testing* a `flag.num.output`. Les crides que hem dut

a terme són les següents:

```
./svm-train -s 0 -t 2 flag.num.train flag.num.model
./svm-predict flag.num.test flag.num.model flag.num.output
```

L'error de *testing* obtingut és del **64.1%**. Així doncs, sembla raonable pensar que els nostres arguments eren acceptables. Hem pogut veure com l'ús del kernel d'agregació tot tractant les variables en funció de la seva tipologia ens ha proporcionat un error molt més baix —recordem, del 23.8%— que no pas tractant les dades com a numèriques i emprant únicament el kernel RBF.

Considerem que aquest resultat és molt significatiu. Si bé és cert que cal dur a terme més proves, és possible que estiguem davant d'una eina veritablement útil per resoldre problemes mitjançant LIBSVM. Cal tenir en compte que per emprar la nova versió de la llibreria amb el kernel d'agregació inclòs, únicament cal pagar el cost de generar un simple fitxer de configuració. Per contra, el guany en expressivitat, eficiència i precisió en les prediccions pot marcar la diferència amb relació a altres softwares emprats en la literatura.

6 Conclusions

Arribats a aquest punt, hem conclòs amb les tasques associades a la recerca envers els tipus de variables, a l'estudi i la implementació del kernel d'agregació i a l'experimentació amb el nou software de LIBSVM. Ara, doncs, creiem convenient analitzar la feina feta i extreure'n les conclusions oportunes.

En aquest apartat durem a terme una valoració dels objectius proposats durant l'etapa de planificació i estimarem si han estat assolits o no. Seguidament, valorarem la planificació establerta a l'inici del projecte. A continuació, analitzarem els obstacles que hem detectat al llarg del treball i com hem sabut neutralitzar-los. Per acabar, durem a terme una valoració personal del projecte i establirem quins són els possibles treballs futurs que poden sorgir a partir dels fonaments establerts per aquest projecte.

6.1 Anàlisi del compliment dels objectius

En capítols anteriors d'aquest projecte hem introduït els objectius que era necessari assolir per donar el treball per enllestit. En aquesta secció els comentarem un a un i valorarem si hem complert el que proposàrem.

El primer objectiu (O1) era «*Generar una llista dels tipus de variables que sembla raonable tractar amb mètodes d'Aprenentatge Automàtic*». Considerem aquest objectiu com a complert. És així perquè al llarg de l'etapa de recerca envers els tipus de variables, hem enunciat els tipus de variable que estudiaríem i hem justificat la seva elecció.

El segon objectiu (O2) fou «*Fer un estat de l'art sobre els tipus de variables per a les quals ja hi ha dissenyada i no necessàriament implementada alguna funció de kernel*». Al llarg del capítol de recerca de variables hem enunciat els tipus de variable que volíem estudiar i hem dut a terme un estat de l'art. Hem definit la tipologia de les diferents variables, hem justificat la seva tria i hem trobat aplicacions d'aquestes en diferents àmbits d'estudi. Finalment, hem cercat i trobat un o més funcions de kernel que tractin amb la variable. Així doncs, considerem que hem assolit l'objectiu satisfactòriament.

El tercer objectiu (O3) era «*Dissenyar i implementar funcions de kernel per a variables amb domini no real*». Tal com hem pogut veure al capítol d'estudi i implementació del kernel d'agregació, hem implementat funcions de kernel per a variables no contínues. Més concretament, hem introduït funcions de kernel per a variables binàries (kernel SMC i kernel de Jaccard) i per a variables categòriques (kernel *univariate*).

Considerem que els objectius (O4) «*Dissenyar i implementar el kernel d'agregació que faci servir les funcions de kernel anteriorment esmentades per a un mateix conjunt de dades*» i (O5) «*Integrar el kernel d'agregació en el software LIBSVM*» poden ésser comentats en conjunt. De la mateixa manera que amb els kernels esmentats al paràgraf anterior, també hem estat capaços d'implementar el kernel d'agregació. Aquest ens ha permès fer la mitjana dels valors resultants a les crides de tots els subkernels. A més a més, l'hem integrat a la nova versió de LIBSVM. Així doncs, donem els objectius O4 i O5 per assolits.

Finalment, l'últim objectiu (06) requeria «*Resoldre un o més problemes de classificació fent servir la nova versió de LIBSVM amb el kernel d'agregació implementat*». Tal com hem pogut veure al capítol d'experimentació, hem resolt tres problemes (sintètic, *contraceptive* i *flags*) mitjançant el nou software amb el kernel d'agregació integrat. Els resultats han estat satisfactoris i hem pogut comprovar que la nova versió de LIBSVM amb el kernel d'agregació funciona de la manera esperada.

En definitiva, considerem que la feina feta ha permès assolir els objectius que ens vàrem proposar de manera satisfactòria.

6.2 Valoració de la planificació

Durant l'etapa de planificació d'aquest projecte vàrem establir una metodologia a seguir, així com un conjunt de tasques que ens ajudarien a assolir els objectius establerts. En la secció anterior, hem discutit l'assoliment dels objectius. Ara, però, creiem convenient fer una anàlisi de la metodologia establerta i del desenvolupament a escala de tasques del projecte.

La metodologia triada fou l'anomenada SCRUM. Aquesta metodologia es fonamenta en el repartiment de les diferents tasques del projecte en *sprints*. Cada *sprint* consta d'una etapa de desenvolupament, una etapa de revisió dels resultats i una etapa de reajustament del projecte en funció dels resultats obtinguts anteriorment.

Valorem molt positivament l'elecció d'SCRUM com a metodologia. Per definició es tracta d'una metodologia àgil. Aquest fet implica que SCRUM ens ha permès adaptar la planificació del projecte en funció dels nostres avenços i entrebancs. Les etapes de revisió han estat constants i han facilitat en gran part la resolució de dubtes i problemes que han sorgit al llarg del desenvolupament.

D'altra banda, també ens agradaria comentar l'evolució del desenvolupament de les tasques establertes. Tal com hem comentat en capítols anteriors, vàrem dividir el projecte en sis grups de tasques: recerca, implementació, experimentació, gestió, documentació i control. Per a cadascuna (i les seves subtasques), vam establir una estimació temporal. Considerem que majoritàriament hem respectat les nostres estimacions. Tot i així, és cert que vàrem haver de reorganitzar l'etapa de recerca, ja que, finalment, el seu desenvolupament ens requerí més temps del que havíem previst.

En conclusió, considerem que la planificació que vàrem establir ha estat força acurada. No només ens ha permès assolir els objectius, sinó que ens ha ajudat a avançar de manera eficient i efectiva.

6.3 Anàlisi dels obstacles i els riscs

En el desenvolupament d'un treball d'aquesta envergadura és inevitable que sorgeixin imprevistos i entrebancs. En la secció anterior hem comentat com la metodologia SCRUM ens ha ajudat a superar els obstacles i assolir els objectius del projecte. Ara, però, considerem adient valorar alguns d'aquests obstacles i riscs i argumentar què han significat amb relació al correcte desenvolupament del projecte.

Riscs

En les etapes primerenques del projecte vàrem considerar tres grans blocs de riscs: una possible planificació poc acurada, les possibles restriccions derivades de la Covid-19 i les possibles avaries de hardware.

En l'àmbit de les *restriccions per la Covid-19* no hem experimentat cap impediment per poder desenvolupar el projecte. Si bé és cert que en el moment d'iniciar el treball les dades epidemiològiques no eren pas favorables, la situació s'ha relaxat gradualment i ha facilitat un bon entorn de feina. De la mateixa manera, no hem patit cap *avaria de hardware* ni, fins i tot, de software.

Amb relació al risc d'una *planificació poc acurada* considerem que, generalment, no hem patit els seus efectes. L'únic que considerem adient comentar és el fet d'haver fet la recerca de kernels per a alguns tipus de variables que finalment no hem pogut implementar. És el cas de, per exemple, el kernel *cross product* per a variables difuses. El motiu d'aquesta no inclusió del kernel en la nova versió de la llibreria és la manca de temps físic per assolir-ho amb les exigències de qualitat que ens hem imposat al llarg del projecte. Tot i així, considerem que amb una millor planificació a priori podríem haver-ho aconseguit.

Obstacles

Els possibles obstacles que vàrem detectar durant l'etapa de planificació del projecte foren la inexperiència en l'àmbit d'estudi, una possible baixa potència de còmput, el fet de tenir un calendari tancat i l'elevada complexitat del projecte.

Considerem que la *inexperiència en l'àmbit d'estudi* ha suposat un obstacle latent durant el desenvolupament de les tasques. S'ha fet palès —sobretot— en situacions com ara l'estimació dels temps d'execució de les tasques de recerca. Tot i així, gràcies a les etapes de reajustament establertes en la metodologia, hem pogut pal·liar amb els seus efectes.

A diferència del factor de la inexperiència, la *potència de còmput* no ha suposat un gran impediment per poder desenvolupar el projecte. Les eines de desenvolupament han funcionat satisfactòriament i no han esdevingut problemàtiques.

Amb relació al *calendari tancat*, és evident que ha estat un obstacle. Tot i així, no ha implicat grans impediments per poder treballar de manera adient i, finalment, arribar a temps per lliurar la memòria i dur a terme la presentació.

Per acabar, considerem que l'*elevada complexitat* ha estat un obstacle. Tot i així, hem pogut contrarestar-lo amb constància i dedicació a la feina realitzada.

6.4 Treball futur

En aquest apartat comentarem tres possibles ampliacions que, considerem, podrien ser interessants d'ésser implementades en la llibreria.

En primer lloc, creiem que el fet que el software sigui capaç de treballar amb *missing values* és imprescindible. No ho hem estudiat en aquest TFG perquè, en definitiva, s'escapava de les nostres possibilitats en termes de limitació temporal. A banda d'això, la funcionalitat no té cabuda en els objectius que ens havíem proposat de cara al projecte.

Una altra possible ampliació seria afegir el suport de LIBSVM a més tipus de variables. Considerem que hem establert una molt bona base perquè implementar nous kernels i permetre l'ús de noves tipologies de variables sigui una tasca simple i intuïtiva. Alguns exemples de kernels a implementar que apareixen a la literatura podrien ser kernels per a variables difuses [62] (també estudiats en aquest projecte), kernels per tractar text [63] o kernels per tractar amb grafs [64]. És interessant l'opció dels grafs, ja que requeriria plantejar una nova manera d'introduïr les dades. Ja no tindríem únicament un nombre o una paraula com a valor, sinó que seria necessari definir noves pautes per tal que el software sigui capaç d'interpretar els grafs. Les possibilitats més intuïtives serien introduir-los en format *inordre* o *postordre*.

Per acabar, també seria interessant permetre ponderacions dels kernels en el còmput de l'*inner* kernel 4.1. No només això, sinó que sigui el mateix software qui faci l'estimació d'aquests pesos i, tot seguit, generi el model. Una idea preliminar de com variaria la fórmula de l'*inner* kernel seria:

$$k_{inner}(x, z) = \frac{1}{d} \sum_{i=1}^{d} \omega_i k_i(x_i, z_i)$$

On $\omega_i \in [0, 1]$ són les diferents ponderacions i, idealment, $\sum_{i=1}^{d} \omega_i = 1$.

6.5 Valoració personal

En aquest precís instant em trobo redactant les últimes línies d'aquest projecte. Fa quatre mesos, tot just quan estava escollint la temàtica, no em podia imaginar arribar a aquest punt amb un treball que ha significat tant per a mi. Ha estat una etapa dura, però alhora enriquidora. M'he pogut formar en àmbits dels quals no n'era coneixedor i he assolit fites que no m'imaginava. En aquest últim apartat m'agradaria comentar els aspectes bons i no tan bons d'aquest projecte en l'àmbit personal, tot posant punt final a la meva etapa universitària.

Des dels primers contactes amb el director del projecte, en Lluís A. Belanche, preveia que aquest seria un projecte dur, però amb un elevat contingut pedagògic. Sóc una persona ambiciosa i vaig considerar que aquest era un bon repte per acabar el grau. Durant aquests mesos he pogut introduir-me en l'àmbit de la recerca, entre d'altres. Considero que ha estat una experiència que m'ha ensenyat molt. Al llarg de la carrera no solem haver d'enfrontar-nos a problemes com aquest i m'ha agradat poder experimentar i haver de buscar solucions d'entre la literatura a problemes reals associats al projecte.

En l'àmbit de l'aprenentatge automàtic, estic molt satisfet d'haver pogut tractar amb aquest interessantíssim món. De fet, gràcies a aquest projecte he pogut determinar els següents passos de la meva vida acadèmica. Durant l'últim mes he sol·licitat una plaça al Màster en Data Science d'aquesta mateixa institució i he estat acceptat. Espero, doncs, poder seguir formant-me i adquirint coneixements relacionats amb aquesta temàtica. En definitiva, he pres consciència que l'aprenentatge automàtic és el futur. M'és difícil imaginar la infinitat d'aplicacions que en poden sorgir de cara a seguir endavant com a societat i millorar la vida de les persones.

Amb relació a la feina duta a terme amb LIBSVM, em considero afortunat d'haver pogut modificar el software i acabar desenvolupant una versió funcional ampliada. Haig de dir que el codi era una mica confús i he hagut de dedicar moltes hores a desxifrar els diferents processos que es duen a terme per generar el model de l'SVM i, posteriorment, fer les prediccions. Tot i així, valoro molt positivament el resultat final. Crec realment que, si se segueix treballant, l'eina

pot arribar a ser de molta utilitat per als investigadors d'arreu.

Amb relació als resultats obtinguts, els considero molt gratificants. Si bé és cert que m'hagués agradat poder experimentar una mica més, per qüestions de limitació temporal no ha estat possible. Tot i així, els experiments duts a terme em fan pensar de manera optimista i m'alegra que el software al qual he dedicat tant de temps aparentment funcioni de manera satisfactòria. Tant de bo en un futur pugui seguir millorant-lo i incloent noves funcionalitats que el portin a un altre nivell.

Per acabar, m'agradaria fer una reflexió purament personal. Aquests mesos han estat veritablement difícils. I sí, és cert que els coneixements que he adquirit tenen un valor incalculable. Tot i així, em proposo com a *deures* futurs, intentar dedicar-me més temps a mi. El coneixement no pot ser a qualsevol cost. En aquest món en constant efervescència, també hem d'aprendre a parar i escoltar-nos.

Bibliografia

[1] L. López González, "Los orígenes del concepto de inteligencia I: un recorrido epistemológico desde el mundo clásico hasta el Siglo de las luces.," 2013.

[2] A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem," *Proceedings of the London mathematical society*, vol. 2, núm. 1, pàg. 230 - 265, 1937.

[3] ——, *Intelligent machinery*, 1948.

[4] J. McCarthy, M. L. Minsky, N. Rochester i C. E. Shannon, "A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955," *AI magazine*, vol. 27, núm. 4, pàg. 12 - 12, 2006.

[5] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of research and development*, vol. 3, núm. 3, pàg. 210 - 229, 1959.

[6] T. M. Mitchell et al., "Machine learning," 1997.

[7] M. I. Jordan i T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, núm. 6245, pàg. 255 - 260, 2015.

[8] B. E. Boser, I. M. Guyon i V. N. Vapnik, "A training algorithm for optimal margin classifiers," pàg. 144 - 152, 1992.

[9] L. Wang, *Support vector machines: theory and applications*. Springer Science i Business Media, 2005, vol. 177.

[10] E. J. C. Suárez, "Tutorial sobre máquinas de vectores soporte (SVM)," *Tutorial sobre Máquinas de Vectores Soporte (SVM)*, pàg. 1 - 12, 2014.

[11] V. Martínez Crespo et al., "Espacios de Hilbert con núcleo reproductor: el Teorema del Representante. Interpretación del representante como aproximante en aprendizaje," 2020.

[12] T. Gartner, J. W. Lloyd i P. A. Flach, "Kernels for structured data," a *International Conference on Inductive Logic Programming*, Springer, 2002, pàg. 66 - 83.

[13] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini i C. Watkins, "Text classification using string kernels," *Journal of Machine Learning Research*, vol. 2, núm. Feb, pàg. 419 - 444, 2002.

[14] D. Tian, X. Zhao i Z. Shi, "Support vector machine with mixture of kernels for image classification," a *International Conference on Intelligent Information Processing*, Springer, 2012, pàg. 68 - 76.

[15] R. Gentleman i R. Ihaka, *About R Language*, 2021. adr.: `https://www.r-project.org/about.html`.

[16] D. Meyer, E. Dimitriadou, K. Hornik et al., "Package 'e1071'," *The R Journal*, 2019.

[17] A. Karatzoglou, A. Smola, K. Hornik i A. Zeileis, "kernlab – An S4 Package for Kernel Methods in R," *Journal of Statistical Software*, vol. 11, núm. 9, pàg. 1 - 20, 2004. adr.: `http://www.jstatsoft.org/v11/i09/`.

[18] G. V. Rossum i F. D. Jr, *The Python Language Reference*, Python Software Foundation, 2021. adr.: `https://docs.python.org/3/reference/`.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort et al., "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pàg. 2825 - 2830, 2011.

[20] MATLAB, *9.7.0.1190202 (R2019b)*. Natick, Massachusetts: The MathWorks Inc., 2018. adr.: https://uk.mathworks.com/products/matlab.html.

[21] C.-C. Chang i C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, 27:1 - 27:27, 3 2011, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[22] C. Igel, V. Heidrich-Meisner i T. Glasmachers, "Shark," *Journal of Machine Learning Research*, vol. 9, pàg. 993 - 996, 2008.

[23] S. D. Team, *Linear Kernel Combinations (and a bit of MKL)*, 2021. adr.: https://bit.ly/3sdtIUf.

[24] I. Sommerville, "Requerimientos del software," *Ingeniería del software, 7a ed., PEARSON EDUCACIÓN, Madrid, SPA*, pàg. 109 - 110, 2005.

[25] R. G. Figueroa, C. J. Solís i A. A. Cabrera, "Metodologías tradicionales vs. metodologías ágiles," *Universidad Técnica Particular de Loja, Escuela de Ciencias de la Computación*, vol. 9, 2008.

[26] K. Schwaber, "Scrum development process," a *Business object design and implementation*, Springer, 1997, pàg. 117 - 134.

[27] J. L. Scott Chacon, *Git*, 2021. adr.: https://git-scm.com/.

[28] D. Pristupov i T. Pristupova, *Fork*, 2021. adr.: https://git-fork.com/home.

[29] P. Brachet, *Texmaker*, 2021. adr.: https://www.xm1math.net/texmaker/index.html.

[30] O. Kopp, *Jabref*, 2021. adr.: https://www.jabref.org/.

[31] F. I. B. Comissió Permanent, *Normativa del treball final de grau del grau en Enginyeria Informàtica de la Facultat d'Informàtica de Barcelona*, gen. de 2020. adr.: https://bit.ly/38kKjO6.

[32] *Team Gantt*, març de 2020. adr.: https://www.teamgantt.com/.

[33] V. Nahar, S. Al-Maskari, X. Li i C. Pang, "Semi-supervised learning for cyberbullying detection in social networks," a *Australasian Database Conference*, Springer, 2014, pàg. 160 - 171.

[34] M. Hussain, S. K. Wajid, A. Elzaart i M. Berbar, "A comparison of SVM kernel functions for breast cancer detection," a *2011 Eighth International Conference Computer Graphics, Imaging and Visualization*, IEEE, 2011, pàg. 145 - 150.

[35] J. Shawe-Taylor, N. Cristianini et al., *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[36] A. Shashua, "Introduction to machine learning: Class notes 67577," *arXiv preprint arXiv:0904.3664*, 2009.

[37] B. Salazar López, "Variables binarias. El Caso de la Bauxita," *Ingenieria Industrial Online*, juny de 2019. adr.: https://bit.ly/3vzLK3K.

[38] J. Aitchison i C. G. Aitken, "Multivariate binary discrimination by the kernel method," *Biometrika*, vol. 63, núm. 3, pàg. 413 - 420, 1976.

[39] H. Y. Mussa, "The Aitchison and Aitken kernel function revisited," *Journal of Mathematics Research*, vol. 5, núm. 1, pàg. 22, 2013.

[40] H. Jiawei, M. Kamber, J. Pei i D. MINING, "Concepts and techniques," *Beijing: Machinery Industry Press*, vol. 84, pàg. 92 - 99, 2012.

[41] P. Jaccard, "The distribution of the flora in the alpine zone. 1," *New phytologist*, vol. 11, núm. 2, pàg. 37 - 50, 1912.

[42] T. T. Tanimoto, "Elementary mathematical theory of classification and prediction," 1958.

[43] IBM-Corporation, *Variable types*, 2020. adr.: https://www.ibm.com/docs/en/spss-statistics/27.0.0?topic=charts-variable-types.

[44] UCLA, *What is the difference between categorical, ordinal and interval variables?* 2021. adr.: https://bit.ly/3xjzgiI.

[45] D. Dua i C. Graff, *UCI Machine Learning Repository*, 2017. adr.: http://archive.ics.uci.edu/ml.

[46] L. A. Belanche Muñoz i M. Villegas, "Kernel functions for categorical variables with application to problems in the life sciences," a *Artificial intelligence research and development: proceedings of the 16 International Conference of the Catalan Association of Artificial Intelligence*, 2013, pàg. 171 - 180.

[47] L. A. Zadeh, "Fuzzy sets," a *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh*, World Scientific, 1996, pàg. 394 - 432.

[48] S. Alayón Miranda et al., "Diseño de sistemas borrosos recurrentes mediante estrategias evolutivas y su aplicación al análisis de señales y reconocimiento de patrones," 2003.

[49] L. Shamir i R. J. Nemiroff, "Astronomical pipeline processing using fuzzy logic," *Applied Soft Computing*, vol. 8, núm. 1, pàg. 79 - 87, 2008.

[50] S. Alayón, R. Robertson, S. K. Warfield i J. Ruiz-Alzola, "A fuzzy system for helping medical diagnosis of malformations of cortical development," *Journal of biomedical informatics*, vol. 40, núm. 3, pàg. 221 - 235, 2007.

[51] J. Guevara, R. Hirata Jr i S. Canu, "Kernels on fuzzy sets: an overview," *arXiv preprint arXiv:1907.12991*, 2019.

[52] J. Guevara, R. Hirata i S. Canu, "Cross product kernels for fuzzy set similarity," a *2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, IEEE, 2017, pàg. 1 - 6.

[53] ——, "Fuzzy set similarity using a distance-based kernel on fuzzy sets," 2015.

[54] C.-C. Chang i C.-J. Lin, *LIBSVM – A Library for Support Vector Machines*, set. de 2019. adr.: https://www.csie.ntu.edu.tw/~cjlin/libsvm/.

[55] R.-E. Fan, P.-H. Chen, C.-J. Lin i T. Joachims, "Working set selection using second order information for training support vector machines.," *Journal of machine learning research*, vol. 6, núm. 12, 2005.

[56] M. J. García-Ligero, A. Hermoso, J. A. Maldonado, P. Román i F. Torres, *Teorema límite de De Moivre y Laplace*. adr.: https://bit.ly/3qcNG1a.

[57] B. D. Ripley, *Pattern recognition and neural networks*. Cambridge university press, 2007.

[58] T.-S. Lim, *Contraceptive Method Choice Data Set*, https://archive.ics.uci.edu/ml/index.php, From UCI Machine Learning Repository, jul. de 1997. adr.: https://bit.ly/35mkhrO.

[59] T.-S. Lim, W.-Y. Loh i Y.-S. Shih, "A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms," *Machine learning*, vol. 40, núm. 3, pàg. 203 - 228, 2000.

[60] R. S. Forsyth, *Flags Data Set*, https://archive.ics.uci.edu/ml/index.php, From UCI Machine Learning Repository, maig de 1990. adr.: https://bit.ly/2UcIu1v.

[61] E. Moore i D. Ross, *Collins Gem Flags of the World*, HarperCollins, ed. HarperCollins, 1986, ISBN: 978-0004595030. adr.: https://amzn.to/3gAONTR.

[62] Z. Rustam i A. S. Talita, "Fuzzy kernel k-medoids algorithm for multiclass multidimensional data classification," *Journal of Theoretical and Applied Information Technology*, vol. 80, núm. 1, pàg. 147, 2015.

[63] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," a *European conference on machine learning*, Springer, 1998, pàg. 137 - 142.

[64] H. Kashima, K. Tsuda i A. Inokuchi, "Kernels for graphs," *Kernel methods in computational biology*, vol. 39, núm. 1, pàg. 101 - 113, 2004.