Software de control atencional: incrementar la productivitat davant el PC

Nom Estudiant: Ramon Padilla Mensa
Grau en Enginyeria Informàtica
Àrea d'intel·ligència artificial

Nom Consultor/a: Dr. David Isern Alarcón
Nom Professor/a responsable de l'assignatura: Dr. Carles Ventura Royo

Data Lliurament: 02/01/2019

FITXA DEL TREBALL FINAL

Títol del treball:	*Software de control atencional: incrementar la productivitat davant el PC*
Nom de l'autor:	*Ramon Padilla Mensa*
Nom del consultor/a:	*Dr. David Isern Alarcón*
Nom del PRA:	*Dr. Carles Ventura Royo*
Data de lliurament (mm/aaaa):	*01/2019*
Titulació o programa:	*Grau en Enginyeria Informàtica*
Àrea del Treball Final:	*Intel·ligència artificial*
Idioma del treball:	*Català*
Paraules clau	*evitar distraccions, productivitat, monitoratge.*

Resum del Treball (màxim 250 paraules): *Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball*

L'objectiu del projecte es centra en augmentar la productivitat d'un usuari que està treballant al PC.

Amb aquest propòsit, s'ha desenvolupat un software amb la capacitat d'obtenir unes mètriques (cercar cares a través de la webcam, activitat del ratolí...) i extraure'n conclusions que permetin avisar a l'usuari quan el seu nivell de productivitat ha caigut o està en alt risc de caure.

El software està enfocat en dues funcionalitats:

Funcionalitat 1: Quan hi ha evidències de falta de productivitat, es fa sonar una alarma.

Funcionalitat 2: Un algoritme predictiu jutja a l'usuari a partir de les mètriques obtingudes amb la finalitat de predir si la productivitat de l'usuari es troba en risc. En cas afirmatiu, l'usuari rep una notificació d'alerta.

D'altra banda, també s'ha inventat un nou subtipus de xarxa neuronal i s'ha estudiat la seva viabilitat.

En definitiva, es pot dir que el projecte ha estat un èxit.

Abstract (in English, 250 words or less):

The goal of the project is focused on increasing the productivity of a user which is working on the PC.

For this purpose, a software has been developed with the ability of getting metrics (look for faces though the webcam, mouse activity...) and extracting conclusions which allow the user to be warned when his/her producitivity has fallen or is in high risk to fall.

The software is about two main features:

Feature 1: When there is evidence of lack of productivity, an alarm will sound.

Feature 2: A predictive algorithm judges the user according to the obtained metrics in order to predict the possibility that the user's productivity is at risk. If so, the user receives a warning notification.

On the other hand, a new neural network subtype has been invented and its viability has been studied.

In short, it can be said that the project has been a success.

Índex

Llista de figures

índex d'il·lustracions

1. Introducció

1.1 Context i justificació del Treball

Per decidir quin projecte presentar com a TFG m'he basat en tres requisits:

- Un projecte que englobi, en certa mesura, el contingut acadèmic del grau d'enginyeria informàtica.
- Un projecte orientat a la intel·ligència artificial.
- Un projecte sobre una temàtica que conegui bé.

La meva motivació per proposar aquest projecte és que em costa molt concentrar-me i, quan estic fent feina al PC, em distrec contínuament davant del més mínim pensament o estímul (tinc TDAH i sinestèsia).
D'altra banda, però, crec que aquesta dificultat d'evitar les distraccions afecta a tothom o pràcticament a tothom. Per tant, crec que és un projecte interessant, ja que soluciona o intenta solucionar un problema que afecta molta gent i que cada cop n'està afectant a més; a causa, en gran part, de la tecnologia i l'estil de vida que tenim actualment.
A més a més, malgrat que la productivitat és un aspecte molt important, no hi ha gaires eines per millorar-la. Sí que n'hi ha algunes com *Booster*, *Focusatwill* o *RescueTime*. Però aquest tipus d'eines no estan gaire popularitzades i, en general, solen estar més enfocades a gestionar i organitzar millor el temps i no tant a detectar distraccions. A més a més, no n'he trobat cap que faci el que vull fer jo. Per tant, es podria considerar una necessitat a cobrir.

Així doncs, he considerat interessant el fet de desenvolupar un *software* que ajudi a l'usuari a ser més productiu.

1.2 Objectius del Treball

L'objectiu és desenvolupar un *software* que sigui capaç de monitorar a l'usuari (a través de la *webcam* es controlarà que l'usuari estigui davant el PC, també es dedicarà a controlar que no entri a llocs web poc productius com *Facebook* o *Twitter*, que vagi movent el ratolí de tant en tant...) per detectar caigudes de productivitat i ajudar així als usuaris a estar més concentrats i ser més productius quan treballen al PC.

El resultat d'aquest monitoratge desencadenarà en dues funcionalitats principals:

Objectius imprescindibles

Funcionalitat 1: Si hi ha evidències clares de que l'usuari no està sent prou productiu s'activarà una alarma.

Dins de la funcionalitat 1 hi ha els següents objectius:

1- Detecció que l'usuari està inactiu o fora del PC durant un temps determinat.

2- Detecció que l'usuari s'ha connectat a un lloc web el qual es troba dins la llista de webs prohibides o està utilitzant el navegador web (tot això podrà ser configurat al gust de l'usuari).

3- Detecció que l'usuari no mou el ratolí durant un temps determinat (el temps màxim d'inactivitat abans de fer sonar l'alarma podrà ser configurat).

4- Implementació d'alguna funcionalitat extra com fer sonar alguna alarma a una hora concreta o fer sonar música relaxant de fons.

Objectius prescindibles

Funcionalitat 2: El *software* es dedicarà a cercar patrons que puguin estar relacionats amb una falta de productivitat.

Dins de la funcionalitat 2 hi ha el següent objectiu:

5- Implementar un sistema d'intel·ligència artificial al núvol en el qual s'enviaran dades dels diferents usuaris indicant un patró que va succeir i, si va desencadenar en una falta de productivitat evident (funcionalitat 1) o no. D'aquesta manera, es podrà aplicar algun algoritme supervisat (ja sigui una xarxa neuronal, arbre de decisió...) que permeti predir si un usuari acabarà sent improductiu (primera funcionalitat) o no durant els segons vinents. En cas que es consideri que té una probabilitat alta d'acabar sent improductiu, li apareixerà una petita notificació per pantalla recordant-li que té feina per fer.

1.3 Enfocament i mètode seguit

S'haurà de desenvolupar un *software* des de zero.

Des d'un punt de vista conceptual, no tinc cap altre *software* com a referència, ja que no he trobat cap eina existent que faci el mateix que el que tinc previst fer en aquest treball.

Des d'un punt de vista tècnic, disposo de molta informació a la xarxa:

Per obtenir i processar les imatges de la webcam es podrà utilitzar *OpenCV* [I]; és una llibreria de visió per ordinador amb llicència *BSD*.

Per fer la predicció (funcionalitat 2) s'haurà d'investigar una mica més sobre algoritmes predictius, atès que segurament serà la part més complicada del projecte. En tot cas, també disposo de molta informació. Com ja vaig comentar en el seu moment, de la mateixa manera que pel processament d'imatges existeixen llibreries com *OpenCV* que et permeten implementar les funcionalitats desitjades amb menys dificultat i menys temps, per la predicció també existeixen *altres llibreries com TensorFlow [II], Deeplearning4j [IV] o Weka [III]* que poden ser de gran utilitat a l'hora d'implementar el sistema predictiu.
Tanmateix, vaig deixar pendent d'estudiar la possibilitat d'utilitzar alguna d'aquestes llibreries de tercers per desenvolupar el sistema predictiu o, fins i tot, utilitzar alguna implementació *Java* feta. Finalment, he decidit no utilitzar cap llibreria de tercers per desenvolupar el sistema predictiu i, enlloc, desenvolupar-lo jo des de zero. El motiu d'aquesta decisió és que malgrat que amb una d'aquestes llibreries o implementacions el desenvolupament del sistema predictiu seria més senzill i, probablement, s'obtindrien unes prediccions més bones, he tingut en compte el fet que jo mai havia desenvolupat cap xarxa neuronal i que, per tant, seria interessant desenvolupar-la des de zero per entendre millor el funcionament d'aquest tipus d'algoritme. Sobretot, a nivell matemàtic.
En cap cas puc assegurar que la part predictiva del *software* acabi funcionant correctament, però si que tinc raons prou sòlides com per considerar la viabilitat d'estudiar-ho i, finalment, intentar implementar-ho.

1.4 Planificació del Treball

Llenguatge de programació
He considerat que *Java* és el llenguatge de programació més adequat per desenvolupar aquest projecte per diferents raons:

- És el llenguatge de programació que més domino.
- És un dels llenguatges més eficients (des del punt de vista de la velocitat de processament) que hi ha.
- Existeix una gran comunitat que hi dona suport.
- Té un molt bon control de les excepcions, permetent que el programa pugui seguir funcionant normalment després d'haver-se produït un error inesperat.
- És multiplataforma.

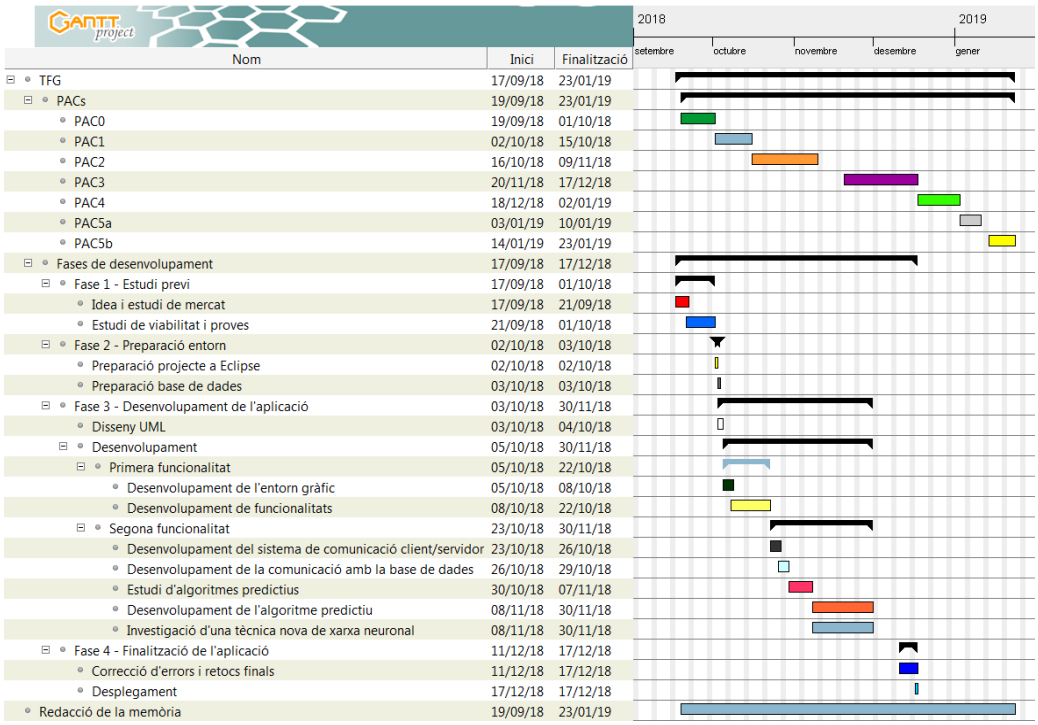
Eines

- Com a IDE vaig decidir utilitzar Eclipse, ja que és l'entorn de desenvolupament que conec millor.
- Com a gestor de base de dades vaig decidir utilitzar PostgreSQL perquè és gratuït, té una bona comunitat de suport, és molt estable i escalable.
- Com a llibreries de tercers vaig decidir utilitzar OpenCV per ser, segons el meu parer, la millor llibreria gratuïta de visió per ordinador. Té una

comunitat de suport molt extensa, permet ajustar molt bé els paràmetres que t'interessen, és molt eficient, disposa de funcionalitats molt variades, molts projectes OpenSource fan ús de OpenCV i és multiplataforma.
Com ja s'ha comentat anteriorment, es va descartar la possibilitat d'utilitzar altres llibreries de tercers per la implementació del sistema predictiu.

Fonts d'informació i documentació

- Internet.
- Llibres.



Nom	Inici	Finalització
TFG	17/09/18	23/01/19
PACs	19/09/18	23/01/19
PAC0	19/09/18	01/10/18
PAC1	02/10/18	15/10/18
PAC2	16/10/18	09/11/18
PAC3	20/11/18	17/12/18
PAC4	18/12/18	02/01/19
PAC5a	03/01/19	10/01/19
PAC5b	14/01/19	23/01/19
Fases de desenvolupament	17/09/18	17/12/18
Fase 1 - Estudi previ	17/09/18	01/10/18
Idea i estudi de mercat	17/09/18	21/09/18
Estudi de viabilitat i proves	21/09/18	01/10/18
Fase 2 - Preparació entorn	02/10/18	03/10/18
Preparació projecte a Eclipse	02/10/18	02/10/18
Preparació base de dades	03/10/18	03/10/18
Fase 3 - Desenvolupament de l'aplicació	03/10/18	30/11/18
Disseny UML	03/10/18	04/10/18
Desenvolupament	05/10/18	30/11/18
Primera funcionalitat	05/10/18	22/10/18
Desenvolupament de l'entorn gràfic	05/10/18	08/10/18
Desenvolupament de funcionalitats	08/10/18	22/10/18
Segona funcionalitat	23/10/18	30/11/18
Desenvolupament del sistema de comunicació client/servidor	23/10/18	26/10/18
Desenvolupament de la comunicació amb la base de dades	26/10/18	29/10/18
Estudi d'algoritmes predictius	30/10/18	07/11/18
Desenvolupament de l'algoritme predictiu	08/11/18	30/11/18
Investigació d'una tècnica nova de xarxa neuronal	08/11/18	30/11/18
Fase 4 - Finalització de l'aplicació	11/12/18	17/12/18
Correcció d'errors i retocs finals	11/12/18	17/12/18
Desplegament	17/12/18	17/12/18
Redacció de la memòria	19/09/18	23/01/19

Il·lustració 1: Diagrama de Gantt.

El diagrama de Gantt mostra la planificació proposada.

1.5 Breu sumari de productes obtinguts

- La memòria.
- Projecte Eclipse i base de dades en una màquina virtual.

1.6 Breu descripció dels altres capítols de la memòria

Capítol 2: Explicació detallada del disseny i el desenvolupament del *software*. Indicació de les decisions preses i com s'han enfrontat.
Explicació dels resultats obtinguts.
Capítol 3: Introducció a les xarxes neuronals i explicació de la implementació d'una xarxa neuronal. Investigació d'un subtipus de xarxa neuronal nova. També inclou una breu descripció d'un senzill algoritme genètic.
Capítol 4: Explicació del que s'ha aprés durant aquest treball i si s'han assolit els objectius funcionals i temporals.
Capítol 5: Glossari dels termes utilitzats.
Capítol 6: Referències bibliogràfiques utilitzades.
Capítol 7: Explicació de la implementació amb diagrames UML i anàlisi de la xarxa neuronal. Manual d'instal·lació de la part servidor.

2. Desenvolupament del treball

2.1 Fonaments arquitectònics del projecte

El *software* està orientat a una arquitectura *client-servidor*. Existint, per tant, un programa client (amb interfície gràfica) situat al PC dels usuaris i un programa servidor situat al PC servidor, conjuntament o no, amb la base de dades (segons convingui pot interessar més un model o l'altre).

-Programa client: Conté tota la funcionalitat 1.

1- Detecció que l'usuari no està davant del PC.
2- Detecció que s'ha accedit a una pàgina web no permesa segons la configuració escollida.
3- Detecció que l'usuari fa una estona determinada que no mou el ratolí.
4- Configuració d'una alarma a una hora determinada.
5- Fer sonar ones binaurals *[V] i [VI]*.
6- Activar les notificacions comunicant-se amb el servidor (funcionalitat 2).

-Programa servidor: Conté tota la funcionalitat 2.

1- Comunicació amb el client i amb la base de dades.
2- Afegir noves mètriques (patrons) rebudes pel programa client a la base de dades.
3- Entrenar el sistema predictiu amb els patrons de la base de dades.
4- Predir si l'usuari està reduint la seva productivitat a partir d'un patró concret enviat pel programa client.

2.2 Funcionalitat del *software*

Com ja s'ha comentat anteriorment, *Attentional control* és un *software* que permet fer un monitoratge de l'activitat de l'usuari al PC amb la finalitat de detectar conductes pròpies de la falta de productivitat davant del PC. D'aquesta manera, si es detecta alguna d'aquestes conductes, s'avisarà a l'usuari.
Aquest *software*, al avisar a l'usuari quan aquest deixa de ser productiu no només està ajudant a que aquest estigui concentrat mentre el *software* està en marxa, sinó que també està influenciant a que aquest usuari adquireixi uns hàbits/conductes inconscients que li permetin ser més productiu quan no estigui usant el *software*.

Attentional control disposa d'una interfície gràfica que permet a l'usuari usar el *software* de forma personalitzada per activar només aquelles característiques que li interessin i configurar-les d'acord a les seves necessitats.

La interfície gràfica està dividida en 5 apartats:

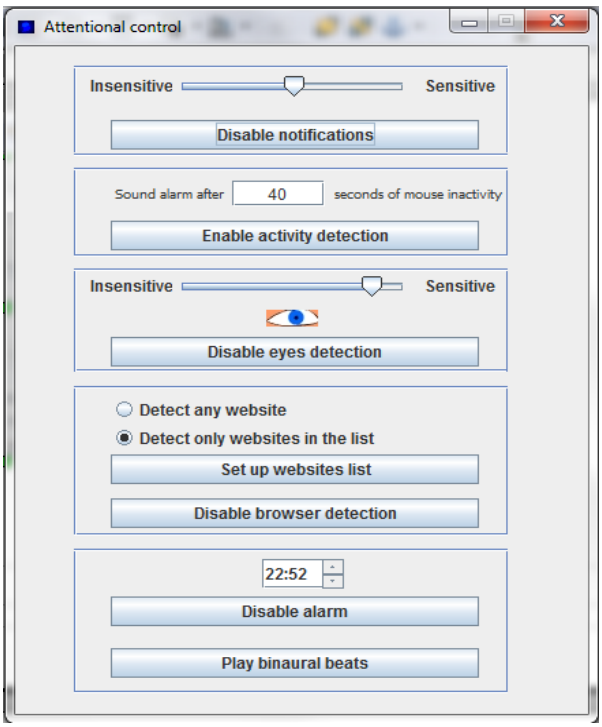
Apartat 1 (notificacions): Aquesta característica permet que a l'usuari li apareguin notificacions quan el *software* considera que hi ha un alt risc de que deixi de ser productiu.

Apartat 2 (monitoratge del ratolí): Es farà sonar una alarma quan s'hagi detectat que el ratolí no s'ha mogut durant un temps determinat.

Apartat 3 (monitoratge dels ulls): Aquesta característica farà sonar una alarma quan faci aproximadament un minut que el *software*, a través de la *webcam*, no pugui detectar els ulls de l'usuari.

Apartat 4 (monitoratge del navegador): Aquesta característica permet controlar l'activitat del navegador fent sonar una alarma en cas que detecti algun accés no permès.

Apartat 5 (altres característiques): Permet que l'usuari faci sonar una alarma a una hora determinada i fer sonar ones binaurals.



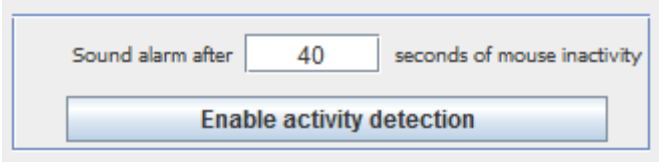
Il·lustració 2: Aspecte de l'entorn gràfic.

En la Il·lustració superior es pot veure l'aspecte del *software*. Veiem que hi ha 5 apartats: el primer apartat correspon a la funcionalitat 2, la resta a la funcionalitat 1.

2.2.1 Funcionalitat 1

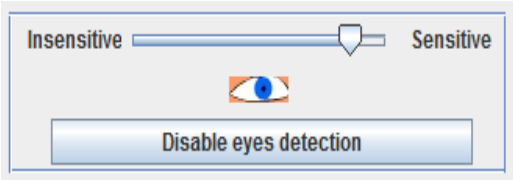
La funcionalitat 1 es va poder implementar en la seva totalitat; dins del termini de la planificació proposada.

Explicació del funcionament:

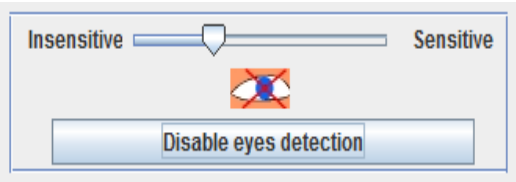


Il·lustració 3: Activació i configuració del monitor d'activitat.

En aquest apartat podem configurar un temps màxim d'inactivitat del ratolí. Si el ratolí es troba totalment inactiu durant el temps especificat, sonarà l'alarma.



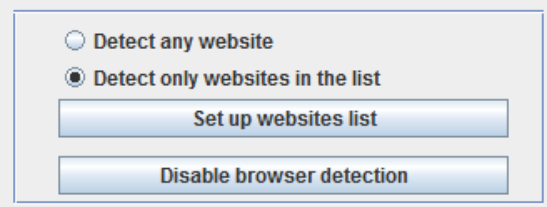
Il·lustració 4: Activació i configuració de la detecció d'ulls.



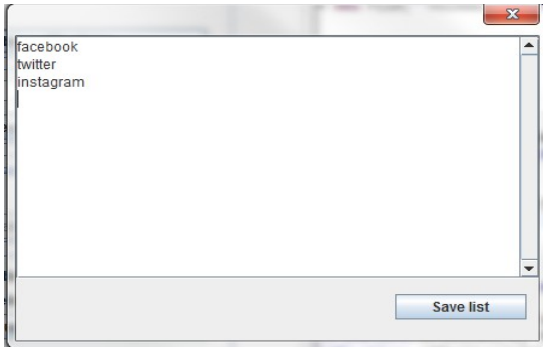
Il·lustració 5: Activació i configuració de la detecció d'ulls.

En aquest altre apartat podem activar el monitoratge de detecció facial a través de la *webcam* del nostre PC. A l'apartat es pot veure l'icona d'un ull (amb una creu o sense creu): Mentre no aparegui la creu, no hi ha perill que soni l'alarma atès que el monitor ens està detectant la cara (el fet que l'icona sigui un ull i no pas una cara es deu a que el monitor, en realitat, està cercant una cara amb dos ulls dins. Això és important perquè, per exemple, tu pots estar davant del PC però mirant el mòbil...).
Si apareix la creu vol dir que no ens està detectant els ulls i, a partir d'aquest moment, ens donarà aproximadament un minut en el que intentarà tornar a detectar-nos els ulls. Si no ho aconsegueix sonarà l'alarma.

Podem ajustar la sensibilitat de la detecció, ja que segons algunes variables (il·luminació, distància entre la nostra cara i la webcam...) haurem d'ajustar el monitor assignant-li més o menys sensibilitat.

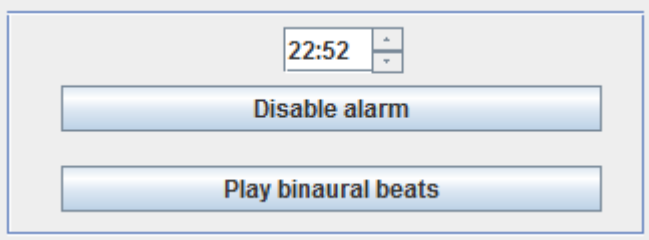


Il·lustració 7: Activació de la detecció del navegador.



Il·lustració 6: Configuració de la detecció del navegador.

En aquest apartat podem decidir si volem permetre l'accés a qualsevol lloc web menys als llocs web especificats dins de la llista de webs prohibides o que directament no es permeti utilitzar el navegador web.
Si premem el botó “Set up websites list” se'ns obrirà un diàleg on podrem determinar la llista de webs prohibides.



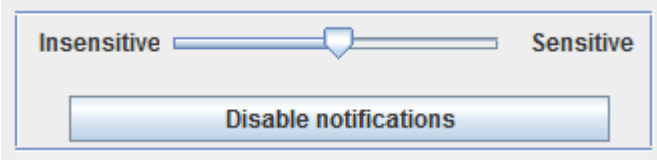
Il·lustració 8: Funcionalitats extra.

Finalment, tenim un apartat de funcionalitats extra on podem establir una alarma a una hora determinada (hem de marxar, ja hem treballat prou per avui...) i fer sonar *ones binaurals* (és un tipus de "música" que ajuda a relaxar-se i concentrar-se millor, especialment a persones amb dèficit atencional).

2.2.2 Funcionalitat 2

La funcionalitat 2 s'ha implementat dins dels terminis de la planificació proposada.
Prèviament a la implementació, es van estudiar els principals algoritmes predictius per decidir quin utilitzar com a *kernel* de la funcionalitat 2.

Explicació del funcionament des del punt de vista de l'usuari:



Il·lustració 9: Activació i configuració de les notificacions.

Quan s'activen les notificacions, es comença a monitorar a l'usuari enviant unes dades (anònimes) al servidor que serveixen tan per predir si l'usuari està a punt de despistar-se/deixar de ser productiu, com per millorar el mateix sistema predictiu.
Cada cop que el sistema predictiu (*server-side*) arribi a la conclusió que hi ha una probabilitat considerable de que l'usuari deixi de ser productiu, apareixerà

una petita notificació per pantalla avisant a l'usuari amb frases d'alerta o motivació.
Com en altres apartats del *software*, també podem configurar la sensibilitat de les notificacions perquè apareguin amb més o menys freqüència.

2.2.2.1 Estudi d'algoritmes predictius

Es va estudiar la implementació d'alguns dels principals algoritmes predictius, amb la finalitat de decidir quin implementar en la funcionalitat 2, tant des del punt de vista de la dificultat i carrega de temps de la implementació de l'algoritme, com de la conveniència i adequació tenint en compte el tipus de problema a resoldre.
En concret, es van tenir en consideració els següents algoritmes:

Regressió lineal *[VII] i [VIII]***:** És una tècnica molt antiga que consta en generar una línia entre un conjunt de punts amb la finalitat de trobar una relació entre les variables de *input(X)* i les de *output(Y)*. És fàcil i ràpid d'implementar. A més a més, és un algoritme amb un aprenentatge ràpid però no funciona massa bé quan hi ha atributs irrellevants (no relacionats amb el *output* que s'intenta predir) o correlacionats. A més a més, com és obvi, no està enfocat a problemes no lineals.

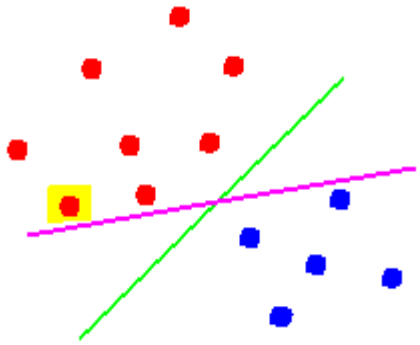
Regressió logística *[VII] i [VIII]***:** Conceptualment és podria dir que és força semblant a la regressió lineal però més sofisticada, atès que genera una línia corba (en forma de 'S') que emmarca la frontera entre les dues classes. Com la regressió lineal, és ràpid i fàcil d'implementar i té un aprenentatge ràpid però és poc tolerant als atributs irrellevants i correlacionats.

KNN (K-Nearest Neighbors) *[VII] i [VIII]***:** Com el seu nom indica, intenta classificar un punt relacionant-lo amb altres punts basant-se en la distància. La idea és que els atributs es trobin escalats o normalitzats, d'aquesta manera es pot calcular fàcilment la distància amb els altres punts. A més a més, cada atribut pot estar ponderat segons la seva rellevància.

Xarxa neuronal *[VII] i [VIII]***:** Una xarxa neuronal és un algoritme que intenta simular un cervell mitjançant unes neurones lògiques que es comporten d'una forma molt similar a les neurones biològiques. Concretament, aquestes neurones tenen una funció d'activació (per decidir amb quina força disparen o si disparen un 1 o un 0, això dependrà de la funció d'activació) i estan connectades unes amb les altres mitjançant unes connexions (sinapsis) que incorporen un pes/ponderació.
L'objectiu és que que els pesos d'aquestes connexions es vagin modificant segons les dades d'entrenament (punts) rebudes amb la finalitat de configurar

aquesta xarxa, de manera que el *output* final que disparen les neurones de l'ultima capa predigui a quina classe pertany un punt determinat.
La implementació d'una xarxa neuronal acostuma a ser complexa i per obtenir bones prediccions es necessita una gran quantitat de *training data*. A més a més, acostumen a requerir un gran processament computacional per entrenar-se i a tardar molt temps (comparat amb altres algoritmes) en fer aquest entrenament.
Malgrat això, una xarxa neuronal ben entrenada pot arribar a ser molt precisa.

SVM (Support Vector Machines) *[VII] i [VIII]*: L'objectiu és aconseguir un hiperplà que separi els punts de les dues classes guardant el màxim de marge possible amb els punts més propers de cada classe.



Il·lustració 10: Exemple de SVM.

En la il·lustració veiem que l'hiperplà verd és millor que el lila, ja que malgrat els dos hiperplans aconsegueixen classificar bé tots els punts, el lila no deixa gaire marge amb els punts i això vol dir que, en cas que es vulgui predir a quina classe pertany un punt X (imaginem que pertany a la classe vermella), si aquest punt es trobés una mica més avall que el punt vermell ressaltat de color groc, seria classificat com a classe blava quan en realitat pertany a la classe vermella. D'altra banda, com que l'hiperplà verd deixa més marge, encara que el punt X quedés una mica per sota del punt vermell ressaltat, seria igualment classificat com a vermell, ja que l'hiperplà verd deixa prou marge.

A l'hora de prendre una decisió sobre quin algoritme implementar, es va arribar a considerar, en un primer moment, en la possibilitat d'implementar diversos algoritmes i fer-los predir conjuntament mitjançant unes votacions, com seria el cas de *Random Forest*. Però es va descartar per falta de temps.
Finalment, es va concloure que el millor seria implementar una xarxa neuronal, no pas per cap raó tècnica en concret, sinó per la mera curiositat que sentia cap a les xarxes neuronals.
Potser això pot semblar una decisió quelcom emocional, però crec que l'objectiu d'un treball d'aquest tipus recau més en aspectes com demostrar els

coneixements adquirits, les ganes d'aprendre, la creativitat, la curiositat... que no pas en la qualitat final del producte desenvolupat.
A més a més, dels 5 algoritmes estudiats, la xarxa neuronal és segurament el més complicat d'implementar i, en problemes no lineals, permet obtenir molt bons resultats quan el volum de dades d'entrenament és prou gran. Per tant, segons el meu parer, aquesta decisió es pot considerar més aviat positiva que negativa.

Per una altra banda, també es va considerar convenient la implementació d'un *algoritme genètic [VIII]* que permetés l'entrenament continu de xarxes neuronals amb configuracions diverses amb la finalitat d'obtenir-ne una que fos la més precisa possible.

En conclusió, es va decidir que s'implementaria una xarxa neuronal i un petit algoritme genètic amb la finalitat de fer les prediccions de la funcionalitat 2 (*server-side*) i, per tant, fer arribar notificacions a l'usuari quan hi hagi un alt risc que aquest deixi de ser productiu.

Un cop presa aquesta decisió, vaig començar a documentar-me sobre com implementar una xarxa neuronal des de zero.
Com ja he comentat a l'apartat 1.3, vaig decidir no utilitzar cap llibreria ni implementació de tercers perquè considero que fer una implementació des de zero és la millor manera d'aprendre.

3. Xarxa neuronal

3.1 Introducció a les xarxes neuronals

Les xarxes neuronals són models predictius, inspirats en el camp de la neurologia, que intenten desenvolupar una intel·ligència a través d'unes neurones i connexions artificials.
Evidentment, aquestes xarxes neuronals artificials no poden ser comparades amb les xarxes neuronals biològiques; Si bé és cert que són comparables des d'un punt de vista conceptual/filosòfic, no ho són des d'un punt de vista més pràctic/funcional.

Com ja s'ha expressat anteriorment, el *software* desenvolupat utilitza una xarxa neuronal per predir si l'usuari acabarà despistant-se o no (funcionalitat 2) i, en cas afirmatiu, mostrar-li una notificació per pantalla.

3.1.1 Breu introducció a les xarxes neuronals biològiques

Malgrat la intensa recerca que la neurologia ha dut a terme durant les últimes dècades, encara no se sap com funciona una xarxa biològica més enllà d'haver aconseguit traçar algunes línies generals i específiques, a més de múltiples hipòtesis. Aquest fet, doncs, suposa un obstacle no només per les xarxes neuronals artificials sinó que també per la intel·ligència artificial en general.

Una de les hipòtesis més ben valorades i, alhora, altament aplicables dins del camp de la intel·ligència artificial, és la teoria de la intel·ligència de Jeff Hawkins *[IX]* (ell mateix fa anys que la està intentant aplicar a *Numenta*). Per aquest motiu, pot ser interessant basar-nos en aquesta teoria (encara que no estigui provada) per explicar de manera pràctica i amena com funciona (o sembla que funciona) una xarxa neuronal biològica:

El cervell humà conté unes 80-90 mil milions de neurones, això sense contar la medul·la espinal ni el sistema nerviós perifèric. Aquestes neurones estan connectades entre sí a través de dendrites i axons que, des d'una perspectiva descriptiva, es podrien considerar “branques” o “cables” per on s'envien i reben senyals. Les dendrites reben senyals enviades per altres neurones i els axons envien senyals a altres neurones. Tanmateix, això no sempre és així, ja que també hi poden haver connexions tipus dendrita-dendrita, per exemple. Com a curiositat, els axons poden arribar a ser molt llargs i això implica que hi pot haver una comunicació directa entre dues neurones que es troben separades per diversos centímetres i que, per tant, hi pot haver una comunicació directa entre dues àrees diferents del cervell.

Cada una d'aquestes neurones té un mecanisme elemental basat en un llindar (threshold) que en cas de ser superat per la suma dels impulsos rebuts a través de les dendrites, provoca que la neurona *dispari* un senyal (de major o menor intensitat) a través del seu axó que arribarà a les dendrites d'altres neurones. I així successivament.
A partir de la comprensió d'aquest mecanisme, també s'ha d'entendre que existeix un altre mecanisme de configuració que permet crear i destruir connexions i, també, modificar la connectivitat amb les altres neurones segons l'estat dels punts de connexió entre dues neurones (sinapsis). No és necessari entrar en detalls, però sí que cal entendre que aquest mecanisme de configuració, en el fons, el que està fent és que la xarxa neuronal s'adapti a allò que necessita i això implica que la forma d'intel·ligència va canviant amb el temps. D'aquesta forma, doncs, si jo intento aprendre una llengua, aquesta adaptació neuronal no només permetrà que aprengui aquesta llengua sinó que també crearà uns camins neuronals que faran que, més tard, quan intenti aprendre una altra llengua, hem resulti raonablement més fàcil que la primera vegada. Aquest fet es deu a que els camins que aquesta configuració neuronal està creant són, en realitat, patrons generals amb la finalitat de fer associacions. Aquestes associacions permeten que a través de les dades/experiències del passat combinades amb les del present, el cervell intenti predir el futur. Per això, el cervell no es caracteritza per ser especialment bo processant dades. En el que és realment bo és memoritzant i recordant de forma associativa.

Els següents són exemples de la utilitat pràctica d'aquesta configuració:

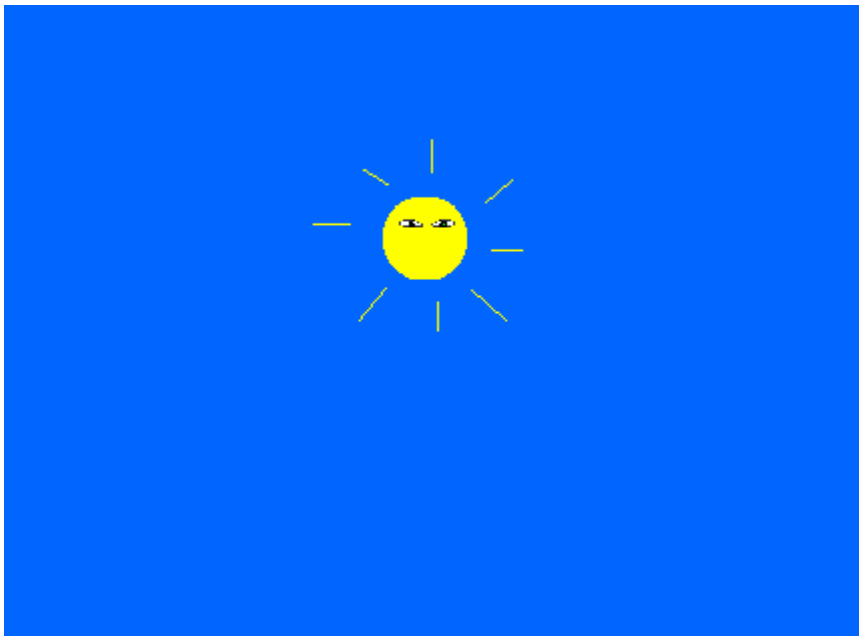
1) Si es vol expressar que es té la necessitat de fer quelcom, en català es poden utilitzar expressions com “he de” o “hauria de”. Aquestes dues expressions s'associen amb la necessitat de fer quelcom perquè durant la vida de l'individu s'han experimentat diferents situacions en les que una persona que sentia la necessitat de fer alguna cosa ha utilitzat aquests mots. D'altra banda, aquestes expressions s'associaran entre sí, no només pel simple fet de compartir el mateix significat (obtingut dels diferents contextos en els que s'han emprat aquestes expressions), sinó també per les similituds literals reconegudes a través del so, escriptura... ja que la segona paraula de l'expressió és compartida en ambdós casos i, la primera, apunta en els dos casos al verb “haver” que, alhora, té un altre significat.

2) La repetició incrementa el nivell d'aquesta configuració. Per exemple, quan una persona viu envoltada de gent que fa un ús inadequat de la llengua utilitzant expressions errònies com “tinc que”, és molt probable que aquesta persona, involuntàriament, acabi adoptant aquesta expressió i deixi d'utilitzar “he de” o “hauria de”. A més a més, el fet d'acabar adoptant-la quan parla, escriu o pensa reforçarà la configuració neuronal de manera que, si algun dia decideix deixar d'utilitzar aquesta expressió, pot trobar-hi certa dificultat, ja que la configuració ha creat un patró fort i, inconscientment, seguirà dient “tinc que” encara que s'esforci en no fer-ho.

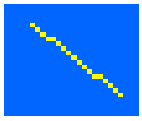
3) Com que el cervell intenta predir el futur basant-se en el passat i el present, si, per exemple, una persona toca un objecte i aquest té el tacte esperat

segons les experiències passades i la informació del present, la persona no donarà gaire importància a la sensació tàctil que aquest objecte li genera. Ara bé, si el tacte no és l'esperat, la persona es sorprendrà intentant obtenir un raonament (basant-se en altres experiències del passat i la informació del present) que expliqui aquesta irregularitat. Com a curiositat, el cervell augmentarà el seu nivell d'activitat i això provocarà que el ritme cardíac i la respiració de la persona també augmentin per proveir al cervell d'oxigen, a més de possibles expressions facials/corporals com arrugar el seny que, en aquest cas, l'ajudarà a incrementar el seu nivell de concentració.

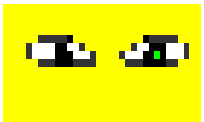
4) Les imatges que s'envien al cervell no són, en realitat, imatges completes sinó troços de les imatges captades pels ulls; mostrant petites imatges que el cervell considera significatives i ignorant tot allò que considera irrellevant.



Il·lustració 11: Imatge mostrada al ull.



Il·lustració 14: Imatge que li arriba al cervell.



Il·lustració 12: Imatge que li arriba al cervell.

Il·lustració 13: Imatge que li arriba al cervell.

La il·lustració 11 podria representar el dibuix fet per un infant. Quan una persona veu aquest dibuix, la imatge no entra “de cop” al cervell sinó que entra en petites imatges que mostren les diferents “característiques” de la imatge

global. L'ull, segurament, capturarà primer la il·lustració 12 perquè és estranya, genera emocions, és complexa, ens recorda a una persona i, en general, és d'especial interès pel cervell. Després capturarà les il·lustracions 14 (per la seva forma lineal) i 13 (per la seva forma ovalada). El fons de color blau, al ser tot igual, inexpressiu i ser un color típic del cel, segurament serà el punt que el cervell li donarà menys importància. Com a curiositat, si quan veiem la imatge ens sembla que la veiem tota de cop és perquè el cervell reconstrueix la imatge i, com que durant unes mil·lèsimes de segon, els humans tenim memòria eidètica, ens fa la impressió que la veiem "de cop" o "sencera". Però això és, en realitat, una il·lusió.

5) El cervell recorda molt millor les experiències associades a emocions. Per exemple, un alumne tindrà més facilitat per recordar el que li explica un professor que li cau bé, que el fa riure i amb qui empatitza, abans que allò que li explica un professor avorrit i distant.

3.1.2 Breu introducció a les xarxes neuronals artificials

Les xarxes neuronals artificials segueixen el mateix principi que les biològiques; hi ha uns nodes i unes connexions entre aquests nodes. Els valors d'aquestes connexions es modifiquen amb l'experiència tot generant uns patrons amb la finalitat d'aconseguir una configuració (aprenentatge) el més òptima possible.

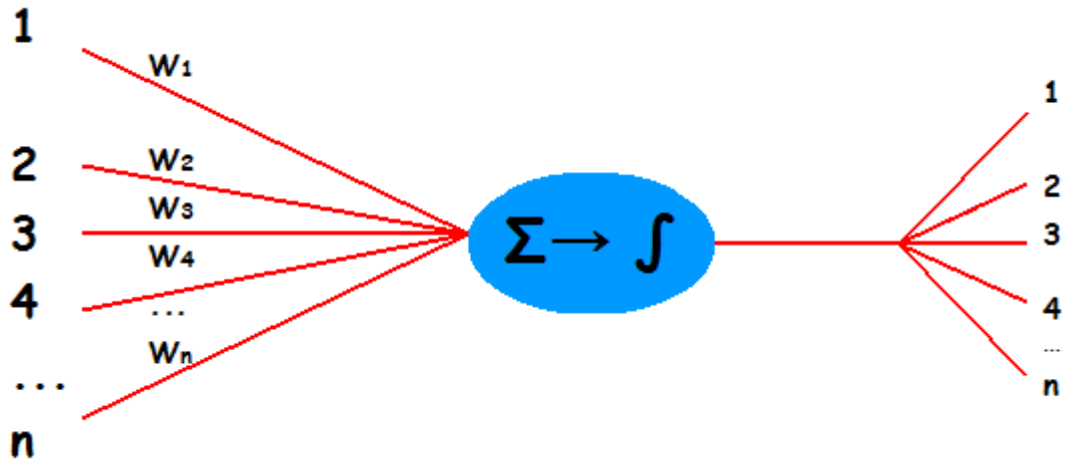
Aquestes xarxes neuronals tenen un conjunt de característiques molt semblants a les biològiques.
Les següents són algunes d'aquestes característiques:

Funció d'activació: Les neurones (nodes), a partir del sumatori de *inputs* rebuts d'altres neurones (a través de les dendrites), apliquen una funció matemàtica per obtenir quin és el valor que han de disparar a les neurones connectades al seu *output* (a través del axó). La funció d'activació equival al "mecanisme elemental basat en un llindar" del punt anterior.

Pesos: Les connexions entre neurones tenen unes ponderacions anomenades pesos que ponderen el *output* que va d'una neurona a una altra amb la finalitat de donar-li més o menys rellevància a aquest *output* i aconseguir, així, millorar la configuració de la xarxa neuronal. Aquests pesos equivalen a la configuració del punt anterior.

Capes (layers): Les xarxes neuronals acostumen a dissenyar-se en forma de capes o columnes paral·leles. D'aquesta manera, la primera columna rep els *inputs* i dispara un *output* a la segona columna, la segona columna dispara a la tercera i així fins a obtenir el *output* de la xarxa neuronal. El cervell no està directament estructurat en capes. Malgrat això, una àrea del cervell, en ser

activada, també activa altres àrees del cervell. Si associem aquestes àrees amb el concepte capa, sí que podríem trobar certa similitud.



Il·lustració 15: Components d'una neurona artificial.

La il·lustració superior mostra els components típics d'una neurona artificial: concretament, veiem que les dendrites reben un valor i li apliquen el pes (ponderació) concret d'aquella connexió. Després, aquest valors ponderats arriben a la neurona que en fa el sumatori i li aplica la funció d'activació. Finalment, la neurona expulsa el *output* a través del seu axó, de manera que les següents neurones reben aquest *output*, li apliquen la ponderació, fan el sumatori...

3.1.3 Comparativa entre les xarxes neuronals bilògiques i les artificials

Comparació a baix nivell *[IX] i [X]*:

Una de les diferències més clares entre els dos tipus de xarxa neuronal es troba en la magnitud: Mentre que les biològiques poden arribar a contenir desenes de milers de milions de neurones, les artificials acostumen a contenir entre desenes i pocs milers. Evidentment, es pot dissenyar una xarxa neuronal artificial raonablement gran sempre que el *hardware* ho permeti tant a nivell de memòria com a nivell del temps d'execució, però no acostumen a donar bons resultats.

A més d'això, malgrat hi ha un mite que diu que les computadores poden processar informació més ràpid que un cervell humà, això no és pas cert. Si bé és cert que un ordinador pot processar una “instrucció” més ràpid que el nostre cervell, no ho pot fer d'una forma simultània massiva. És a dir, en un cervell, milions de neurones poden estar processant (en execució) en paral·lel. Dit d'una altra forma, en cada *cicle de CPU* hi poden haver milions de nuclis processant al mateix temps...

D'altra banda, les xarxes neuronals artificials tenen una estructura més centralitzada i homogènia mentre que les biològiques són descentralitzades i heterogènies.

Comparació a alt nivell *[IX] i [X]*:

Les xarxes neuronals artificials són més “matemàtiques” i, per tant, acostumen a funcionar més bé en entorns rígids, estrets i controlats.
Les xarxes neuronals biològiques, d'altra banda, s'adapten més bé a entorns flexibles, amplis i descontrolats. Son especialment bones en contextualitzar i fer associacions iliterals entre conceptes molt diversos (la creativitat i intuïció que poden assolir ho demostra), aquest fet també implica que tinguin una memòria poc precisa i poc literal.

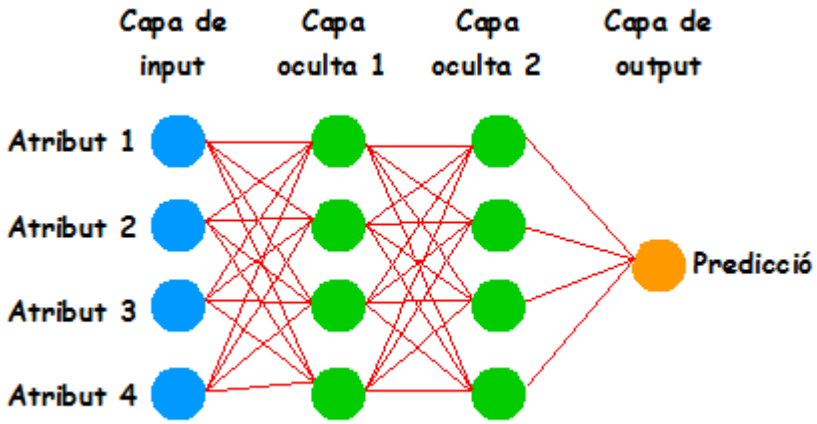
3.2 Funcionament de les xarxes neuronals artificials

Les xarxes neuronals artificials poden arribar a ser molt diverses. Malgrat que conceptualment són el mateix, tant l'estructura com les característiques com la implementació poden arribar a ser molt variades. El fet que formin un conjunt tan heterogeni es deu al fet que cada xarxa neuronal està enfocada a resoldre un problema en concret i això implica basar-se en el model que millor s'adapti al tipus de problema que s'intenta resoldre. Per tant, no hi ha un model millor que un altre de *per ser*, tot dependrà de la situació en particular.

3.2.1 Principals tipus de xarxes neuronals artificials

Tipus de xarxes neuronals artificials:

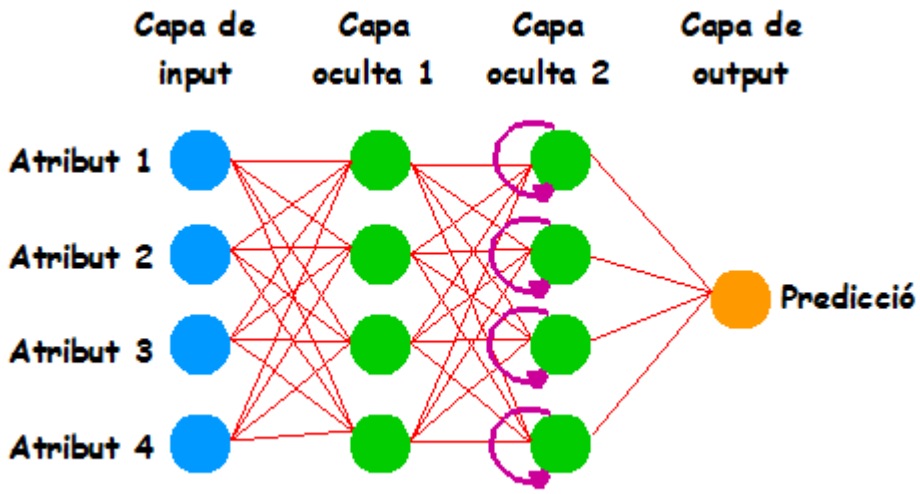
Feedforward Neural Network: Aquesta és, probablement, la forma més simple de xarxa neuronal. En aquest cas, les neurones sempre disparen endavant. Per exemple, si el *input* entra per l'esquerra, les neurones dispararan sempre a la capa que tenen immediatament a la seva dreta, sense produir-se cap tipus de recursivitat o salt. Les FNN és la forma més genèrica de xarxa neuronal i acostuma a funcionar bé en la majoria d'aplicacions.



Il·lustració 16: Exemple de Feedforward Neural Network.

La il·lustració superior mostra una possible estructura de FNN; la capa de input seria la primera en disparar, després dispararia la capa oculta 1, després la capa oculta 2 i, finalment, la capa de output.

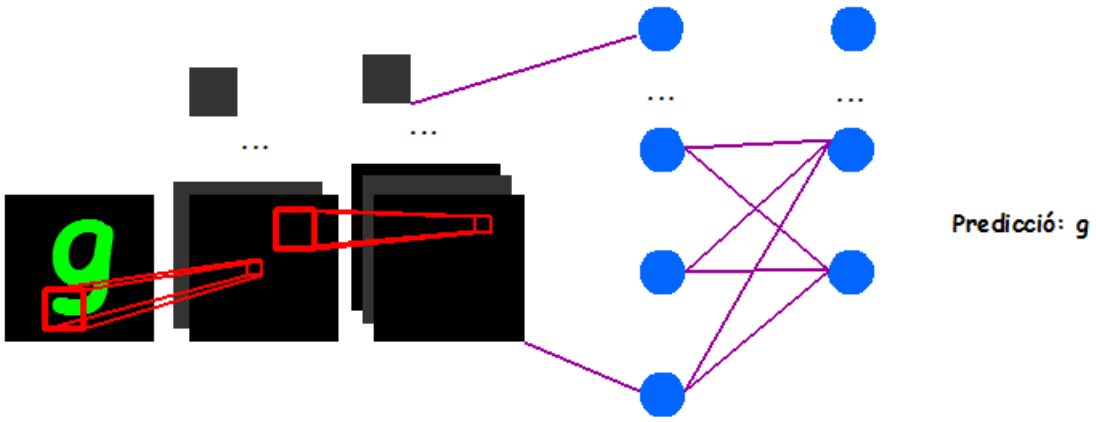
Recurrent Neural Network: Les Xarxes Neuronals Recurrents són molt semblants a les anteriors però amb la diferència que les neurones poden memoritzar el *output* produït i, d'aquesta forma, la propera vegada que rebi un *input*, generarà un *output* influenciat pel *output* memoritzat. Això és molt interessant quan necessitem fer una predicció basant-nos en la predicció anterior. Poden ser molt útils en el reconeixement de veu, ja que conèixer la paraula o so anterior, ajuda a predir la paraula o so següent.
Per exemple, si es reconeix el mot "he", és important que la xarxa neuronal pugui recordar aquest *output* perquè la probabilitat que un mot sigui "de" si abans s'ha reconegut "he" és superior a si abans s'ha reconegut "mentre". Per tant, "he" hauria d'influenciar la xarxa neuronal perquè la següent classificació tingui tendència a ser "de" o algun verb del pretèrit perfet.



Il·lustració 17: Exemple de Recurrent Neural Network.

Convolutional Neural Network: Les Xarxes Neuronals Convolucionals són molt interessants de cara al reconeixement de patrons filtrats. Això és especialment útil per la classificació de imatges, ja que permet extreure característiques d'una imatge i obtenir una classificació a partir d'aquestes característiques.

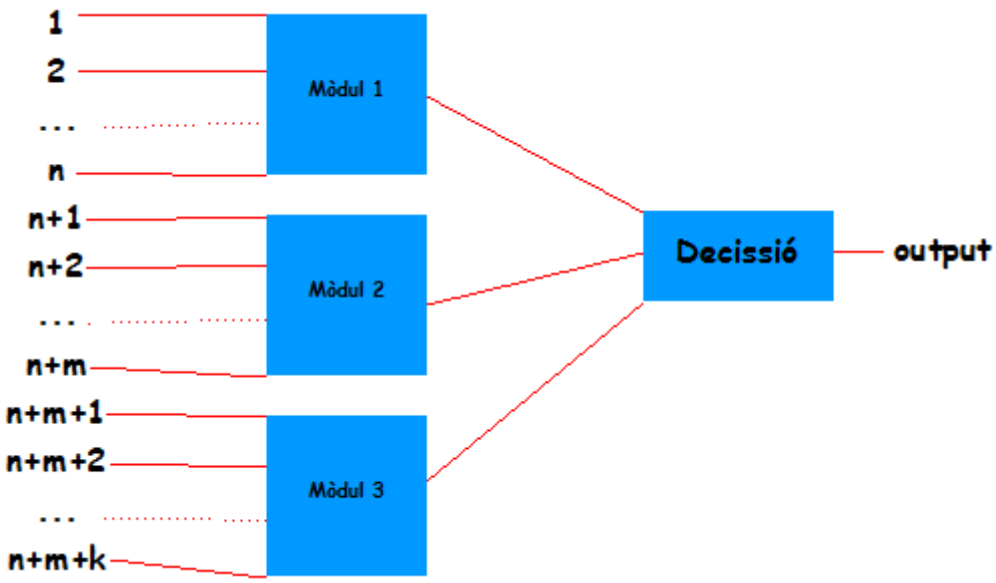
Això és altament interessant tenint en compte el que s'ha explicat a l'apartat 3.1.1; s'ha explicat que el cervell no processa una imatge com un tot, sinó com petites imatges on apareixen trossos de la imatge global.



Il·lustració 18: Exemple de Convolutional Neural Network.

La il·lustració superior explica, d'una manera molt superficial, el funcionament de les CNN. En realitat, la implementació pot arribar a ser molt complexa i es poden arribar a aplicar diversos filtres (filtres d'il·luminació, *histogram equalization*...).

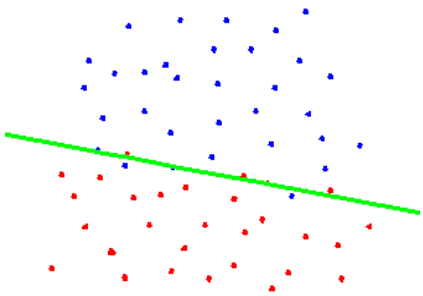
Modular Neural Network: Les Xarxes Neuronals Modulars s'inspiren en la modularitat del cervell (cada àrea del cervell té una funció diferent). El disseny es basa en diverses xarxes neuronals independents que envien els seus *outputs* a un intermediari que acabarà prenent una decisió.
Aquest tipus de xarxa neuronal es pot utilitzar com a substitut d'una xarxa neuronal molt gran i complexa. D'aquesta manera, s'aconsegueix augmentar l'eficiència al requerir-se menys connexions (els nodes d'una xarxa neuronal incrementen el nombre de connexions de forma exponencial), permet entrenar els diferents mòduls de forma independent i això permet implementar i mantenir la xarxa neuronal amb major facilitat.



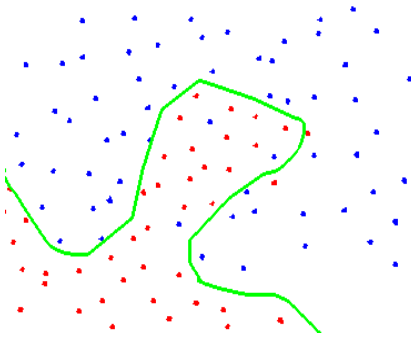
Il·lustració 19: Exemple de Modular Neural Network.

3.2.2 Funcions d'activació

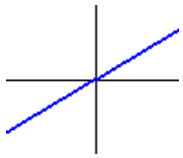
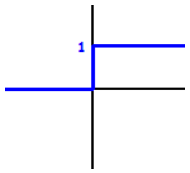
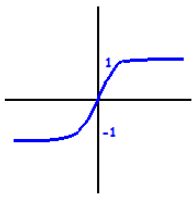
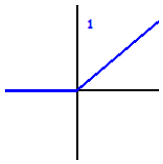
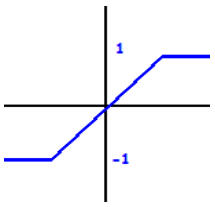
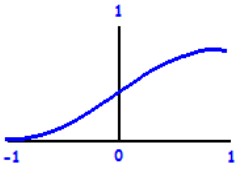
La funció d'activació té un paper molt important de cara a obtenir un bon entrenament de la xarxa neuronal. Naturalment, algunes xarxes neuronals no utilitzen cap funció d'activació, de manera que el *input* és el mateix que el *output*. Quan no hi ha funció d'activació, diem que la xarxa neuronal utilitza una funció d'activació lineal.
La funció d'activació lineal, però, és útil en només una minoria dels casos, perquè la majoria de problemes no tenen una solució lineal en els que es pugui classificar amb una línia recta, obtenint un bon resultat. Per aquest motiu, ens interessa utilitzar una funció que ens permeti traçar una línia amb suficient curvatura com per classificar correctament.



Il·lustració 20: Problema lineal.



Il·lustració 21: Problema no lineal.

Funció d'activació	Equació	Gràfic
Lineal	f(x)=x	
Pas (Step)	$f(x)=\begin{cases}0, & x<0\\1, & x\geq 0\end{cases}$	
Tangent hiperbòlica	$f(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	
Lineal rectificada	$f(x)=\max(0,x)$	
Trams	$\begin{cases}-1 & si \quad x<-1\\ x & si \quad -1\leq x\leq 1\\ 1 & si \quad x>1\end{cases}$	
Logística	$f(x)=\frac{1}{1+e^{-x}}$	

Lineal: És la funció d'activació més simple. Com ja s'ha dit, no hi ha funció d'activació. Per tant, el *output* és igual al *input*.

Pas: El *output* pot ser un 1 o un 0 (dispara o no), no hi ha punts entremitjos. Aquest funció d'activació té dos grans inconvenients:

El primer és a l'hora d'actualitzar els pesos, ja que d'aquesta funció binària s'obté sempre una derivada de 0. Per tant, actualitzar els pesos de forma adequada és complicat...
El segon és que al ser una funció binària no té punts entremitjos i, per tant, una petita modificació a un pes o *bias* pot provocar que la funció passi de 0 a 1 o viceversa i, per tant, generar un fort canvi en tota la xarxa neuronal.

Tangent hiperbòlica: Aquesta funció d'activació (també anomenada tanh), acostuma a funcionar força bé. Té el gran avantatge que permet qualsevol nombre real d'entre -1 i 1. Desgraciadament, a causa de la seva forma, quan la funció d'activació d'una neurona es troba prop del 1 o el -1, la inclinació és molt suau i això provoca que, si el *gradient* calculat és petit, la neurona es quedi "encallada" a un punt i no pugui actualitzar-se correctament, fent que la xarxa neuronal no aprengui correctament. Aquest fet es coneix com a esvaniment del *gradient* (*gradient vanishing*) i és un dels majors problemes que presenten les xarxes neuronals.

Lineal rectificada: La funció d'activació lineal rectificada (més coneguda com a ReLU) s'ha popularitzat molt durant els últims anys a raó dels bons resultats que acostuma a donar. Aquesta funció té l'avantatge que no genera *gradient vanishing*. Així doncs, amb aquesta funció d'activació es soluciona un dels principals inconvenients que presenten moltes funcions d'activació. Per contra, el fet que quan $x \leq 0$ el *output* sigui sempre 0, pot provocar que una neurona quedi atrapada amb un *output* de 0 i cap *gradient* pugui canviar el seu *output*. En aquest cas, la neurona es quedarà desactivada de forma indeterminada i es podria dir que es "mor". Per solucionari aquest problema, es poden utilitzar variants de ReLU, com és el cas de Leaky ReLU que és igual a ReLU però amb la diferència que la meitat esquerra de la funció no és completament horitzontal, sinó que fa una mica de pendent. D'aquesta forma, quan $x \leq 0$, el *output* és lleugerament inferior a 0. A mesura que *x* s'aproximi a 0, el *output* també ho farà i, per tant, al haver-hi certa pendent, evitem que una neurona es "mori".

Trams: Aquesta funció d'activació és semblant a ReLU. El problema és que quan arribem a 1 o -1, també ens trobem amb el problema anterior de "mort neuronal".

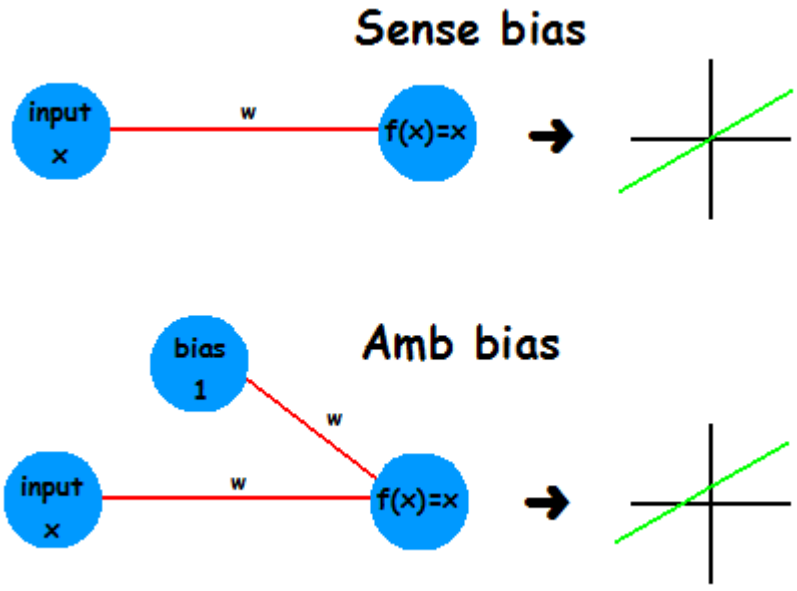
Logística: És molt semblant a la Tangent hiperbòlica però amb algunes diferències com que té un rang [0, 1] enlloc de [-1, 1]. Malauradament, també presenta *gradient vanishing*.

En resum, segons el context ens pot interessar més implementar una funció d'activació o una altra. De fet, hi ha xarxes neuronals que utilitzen diferents funcions d'activació en les diferents capes.
Hem vist que les neurones de les primeres capes tenen problemes a l'hora d'actualitzar-se, ja que la seva actualització té un efecte molt gran sobre la resta de la xarxa neuronal perquè el *output* de les neurones de totes les capes posteriors depenen d'aquesta actualització.

D'altra banda, malgrat que la tangent hiperbòlica i la logística presentin el problema de *gradient vanishing*, també tenen l'avantatge de tenir una forma molt suavitzada i això fa que les actualitzacions de pesos no acostumin a ser massa grans i, alhora, prou grans com perquè en cada actualització els pesos es modifiquin mínimament. En el cas de la tangent hiperbòlica, s'aprecia una corba més descarada que permet obtenir unes derivades més grans que amb la logística i, per tant, una convergència més ràpida.
El fet que la logística tingui un rang de [0, 1] pot ser un inconvenient en alguns casos encara que també pot ser avantatjós en algunes aplicacions.
També cal tenir en compte que aquestes dues funcions d'activació tenen major complexitat computacional que les altres funcions d'activació esmentades en aquest apartat i el temps d'entrenament pot arribar a variar molt segons la funció d'activació seleccionada.

3.2.3 Utilització del *bias* (biaix)

A la il·lustració 15 s'ha mostrat l'exemple del que podria ser una neurona artificial bàsica. Tanmateix, hi ha un component opcional (anomenat *bias*) que ajuda a centrar el *output* de la xarxa neuronal, fent de *threshold*.
El *bias* es comporta simplement com una neurona extra en cada una de les capes. Aquesta neurona no rep cap *input*, ja que el seu *output* és sempre un valor constant (normalment 1). Malgrat això, sí que té connexions amb les neurones "normals" que s'actualitzen i afecten a la xarxa neuronal, ajudant a estabilitzar-la.

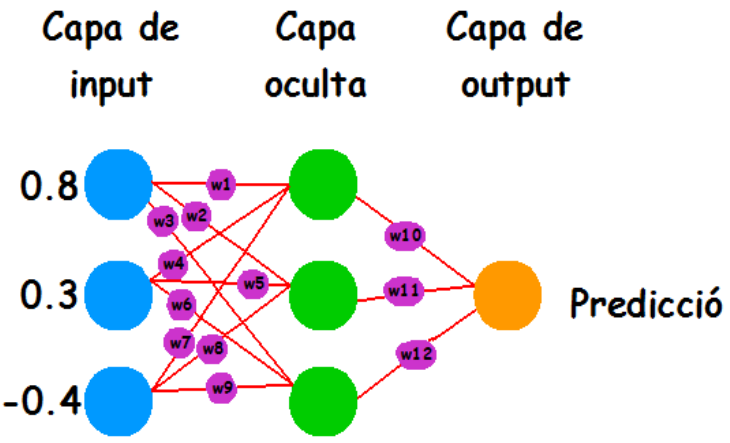


Il·lustració 22: Funció del bias.

A la il·lustració superior veiem un exemple molt senzill amb una funció d'activació lineal. Veiem que sense *bias* la funció, com és lògic, té un *output* de 0 quan el *input(x)* és 0. En canvi, amb *bias* la línia queda més o menys elevada segons les dades d'entrenament.

3.2.4 Forward propagation

Forward propagation és la propagació del *input* (dades obtingudes del monitoratge de l'usuari) a través de les diferents capes de la xarxa neuronal per acabar convertint-se en un *output* (en el nostre cas, si l'usuari s'acabarà despistant o no).



Il·lustració 23: Exemple pràctic d'una xarxa neuronal de tipus perceptró mutlicapa amb una capa input de 3 neurones, una capa oculta de 3 neurones i una capa output amb una neurona.

Per entendre aquest concepte ens basarem en la xarxa neuronal de la il·lustració superior i simularem el seu funcionament.
Com es pot veure, es tracta d'una xarxa neuronal de tipus perceptró multicapa força senzilla. Les neurones blaves són les neurones de *input*, les verdes les neurones de la capa oculta i la taronja la neurona de *output*. Per qüestions de simplificació, s'han obviat els *bias*.
Imaginem que utilitzem la funció d'activació logística i volem fer una predicció tenint els valors d'entrada normalitzats a 0.8, 0.3 i -0.4.
A més a més, tenim els següents pesos seleccionats a l'atzar:

w1: -0.6,	w2: 0.4,	w3: 0.5,	w4: 0.55,	w5: -0.65,	w6: 0.67,
w7: 0.2,	w8: 0.42,	w9: -0.4,	w10: 0.6,	w11: 0.53	w12: 0.48

$$f(x) = \frac{1}{1+e^{-x}}$$

Il·lustració 24: Funció logística.

Capa input:

La capa de input és la més simple de totes, simplement hem d'aplicar la funció d'activació amb el valor de *input*.

$$\text{neurona1} = \frac{1}{1+e^{-0.8}} = 0.68997$$

$$\text{neurona2} = \frac{1}{1+e^{-0.3}} = 0.57444$$

$$\text{neurona3} = \frac{1}{1+e^{-(-0.4)}} = 0.40131$$

Il·lustració 25: output de les neurones de la capa de input.

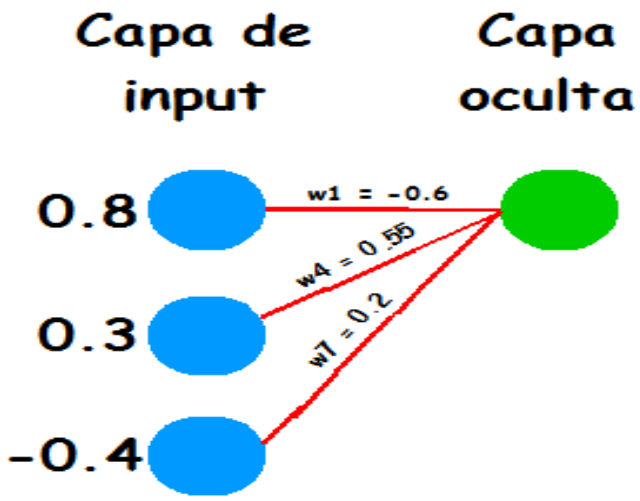
Capa oculta:

Ara que la capa de input (capa anterior) ja ha "disparat" un *output*, hem de multiplicar el pes de cada connexió amb el *output* de la neurona de la capa anterior d'on surti aquesta connexió i fer-ne el sumatori. A aquesta operació se l'anomena *net input* i se l'anomena així perquè és precisament l'entrada neta que acabarà arribant a la neurona.

La formula del *net input* és la següent:

$$net\ input = b + \sum_{i=1}^{n} Xi \cdot Wi$$

En el nostre cas, al no utilitzar *bias*, podem ignorar la “b”.



Il·lustració 26: Connexions que intervenen en el sumatori de la primera neurona de la capa oculta.

A la il·lustració superior veiem les connexions que afecten a la primera neurona de la capa oculta. A partir d'aquí ja podem fer el sumatori:

$$net\ input = 0.8\cdot(-0.6) + 0.3\cdot 0.55 + (-0.4)\cdot 0.2 = -0.395$$

Ara es fa el mateix amb les altres dues neurones de la capa oculta:

$$net\ input\ n2 = 0.8\cdot(0.4) + 0.3\cdot(-0.65) + (-0.4)\cdot 0.42 = -0.043$$

$$net\ input\ n3 = 0.8\cdot 0.5 + 0.3\cdot 0.67 + (-0.4)\cdot(-0.4) = 0.761$$

Finalment, per obtenir el *output* de cada una de les neurones de la capa oculta, s'aplica la funció d'activació:

$$\text{neurona1} = \frac{1}{1+e^{0.395}} = 0.40251$$

$$\text{neurona2} = \frac{1}{1+e^{0.043}} = 0.48925$$

$$\text{neurona3} = \frac{1}{1+e^{-0.761}} = 0.68157$$

Capa output:

Per acabar, fem el mateix entre la capa oculta i la capa de output:

$$\textit{net input} = 0.40251 \cdot 0.6 + 0.48925 \cdot 0.53 + 0.68157 \cdot 0.48 = 0.82796$$

$$\frac{1}{1+e^{-0.82796}} = 0.69592$$

3.2.5 Backpropagation

La *backpropagation* (retropropagació) serveix per anar ajustant els pesos de les connexions segons l'error total actual de la xarxa neuronal amb la finalitat d'aconseguir que quan la xarxa neuronal convergeixi, aquesta es trobi en el *mínim global* o, almenys, en algun *mínim local* acceptable (raonablement proper al *mínim global*).
Utilitzarem una tècnica d'optimització anomenada *gradient descent,* que s'aplica a partir de l'error de les dades d'entrenament.
Existeixen 3 mètodes principals a l'hora d'aplicar el *gradient descent*:

Stochastic Gradient Descent: Per cada element del *training set* amb el que fem un *forward propagation* calculem l'error i actualitzem els pesos a partir d'aquest error.

Batch Gradient Descent: En aquest cas, no s'actualitzen els pesos després de cada element del *training set*, sinó que s'actualitzen després d'haver fet el *forward propagation* amb tots els elements del *training set*.

Mini-Batch Gradient Descent: És el mateix que l'anterior però enlloc de fer un *forward propagation* amb el *training set* sencer, ho fem dividint el *training set* en diferents conjunts.

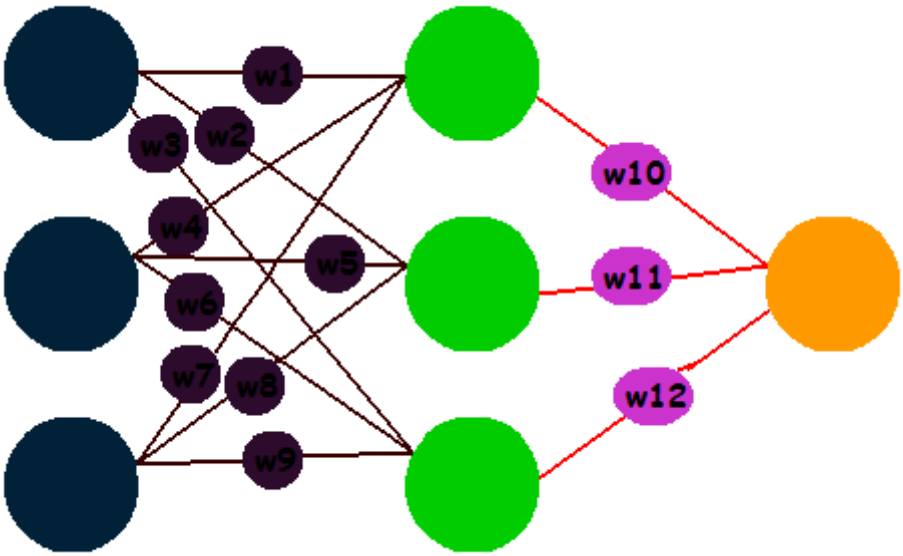
En concret, *backpropagation* consta de 3 passos que es van repetint de forma iterativa:

1- *Forward propagation* amb les dades d'entrenament i calcul d'error del *output*.
2- Obtenir el *gradient* del pes de totes les connexions.
3- Actualització dels pesos de les connexions (pesos sinàptics) a partir del *gradient*, el *learning rate* i, opcionalment, altres elements com *momentum*.

A mode d'exemple, utilitzarem la xarxa neuronal descrita en l'apartat anterior (3.2.4) per demostrar el funcionament de la retropropagació:

Pas 1:
Per simplicitat, utilitzarem el mètode *Stochastic Gradient Descent.* Utilitzarem el *input* de l'apartat anterior (0.8, 0.3 i -0.4) com a exemple d'un element del *training set*. Aquest, genera un *output* de 0.69592.

Pas 2:
S'ha d'obtenir el *gradient* de tots els pesos de la xarxa neuronal. El *gradient* del pes és, en realitat, una derivada parcial. És a dir, és la “inclinació” que hi ha entre aquest pes i l'error total i, això vol dir, “com impactarà un canvi d'aquest pes sobre l'error total?”.
Aquests *gradients* es calcularan utilitzant la regla de la cadena i això implica que es calcularan de dreta a esquerra (de l'última capa fins a la primera).



Il·lustració 27: Primer s'han de calcular els gradients dels pesos que es troben entre la capa oculta i la capa de output.

Per tant, el que interessa saber ara és el següent:

$$\frac{\partial E}{\partial W10}, \frac{\partial E}{\partial W11} \text{ i } \frac{\partial E}{\partial W12}$$

En altres paraules, “Com afecten els pesos w10, w11 i w12 sobre l'error total?”. Per a obtenir aquests *gradients* s'han de tenir en compte diferents paràmetres i, per aquest motiu, s'utilitzarà la regla de la cadena.
Primer s'ha de saber com afecta el *output* (Z) generat pel *item* del *training set* sobre l'error total (E), després com afecta el *net input* de la neurona de la capa de *output* (netO) sobre Z i, finalment, com afecta el pes en particular sobre netO.

Això és:

$$\frac{\partial E}{\partial Wi} = \frac{\partial E}{\partial Z}\frac{\partial Z}{\partial netO}\frac{\partial netO}{\partial Wi}$$

(Per expressar un pes sinàptic s'utilitza, normalment, la notació Wij, on “i” fa referència a la neurona d'una capa i “j” fa referència a la neurona de la capa posterior. Per simplificació s'utilitzarà “i” com a referència al número de pes dins del rang [1-12]).

Si es calculen les derivades parcials:

$$\frac{\partial E}{\partial Z} = -(t - z) \qquad \frac{\partial Z}{\partial netO} = z(1 - z) \qquad \frac{\partial netC}{\partial Wi} = outCO$$

(outCO fa referència a la capa oculta).

Per tant:

$$\frac{\partial E}{\partial Wi} = (z - t)\, z(1 - z)\, outCO$$

Aleshores, si suposem que el *output* esperat és, per exemple, 1 (t=1):

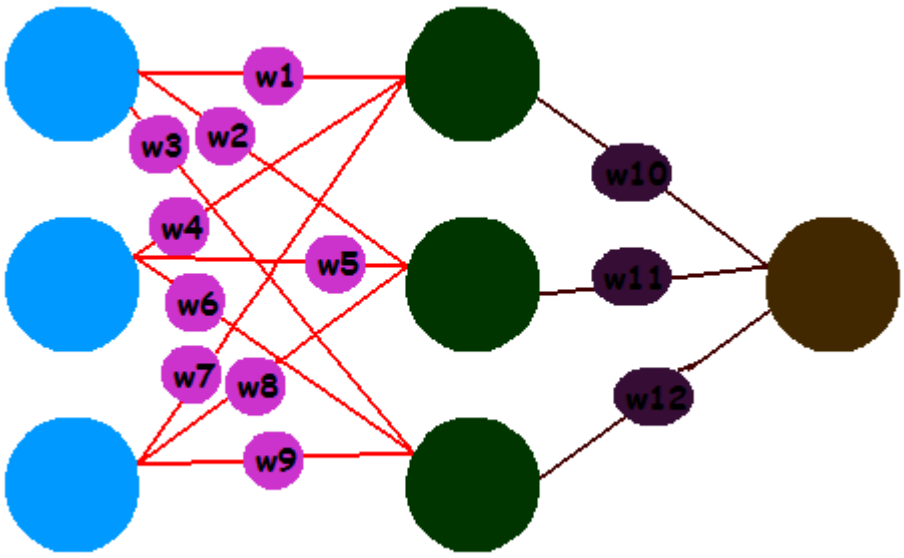
$$\frac{\partial E}{\partial W10} = (0.69592 - 1) \cdot 0.69592(1 - 0.69592) \cdot 0.40251 = -0.025901$$

$$\frac{\partial E}{\partial W11} = (0.69592 - 1) \cdot 0.69592(1 - 0.69592) \cdot 0.48925 = -0.031482$$

$$\frac{\partial E}{\partial W12} = (0.69592 - 1) \cdot 0.69592(1 - 0.69592) \cdot 0.68157 = -0.043858$$

Aquests són els *gradients* dels pesos w10, w11 i w12. Tot i això, per calcular la resta de *gradients* es podrà utilitzar una part de l'equació anterior que és comuna. En concret, $delta_z$, que és la derivada de la funció d'activació (logística) de la neurona de la capa de output multiplicat per la diferència entre el *output* d'aquesta neurona(z) i el *output* esperat (t):

$$\delta z = (z - t)\, z(1 - z)$$



Il·lustració 28: Ara es calcularan els pesos que es troben entre la capa de input i la capa oculta.

Per calcular els pesos w1-w9 s'haurà de seguir el mateix patró però allargant la regla de la cadena fins arribar al pes del qual es vol obtenir el *gradient*.

$$\frac{\partial E}{\partial Wi} = \frac{\partial E}{\partial netO}\frac{\partial netO}{\partial outCO}\frac{\partial outCO}{\partial netCO}\frac{\partial netCO}{\partial Wi}$$

$$\frac{\partial E}{\partial Wi} = outCO(1-outCO)outCI \cdot Wj \cdot \delta z$$

(outCI fa referència a la capa de input, mentre que W_j fa referència al pes que hi ha entre outCO i netO).

I això és:

$$\frac{\partial E}{\partial W1} = 0.40251 \cdot (1-0.40251) \cdot 0.68997 \cdot 0.6 \cdot (0.69592-1) \cdot 0.69592 \cdot (1-0.69592) = -0.0064065$$

$$\frac{\partial E}{\partial W4} = 0.40251 \cdot (1-0.40251) \cdot 0.57444 \cdot 0.6 \cdot (0.69592-1) \cdot 0.69592 \cdot (1-0.69592) = -0.0053338$$

$$\frac{\partial E}{\partial W7} = 0.40251 \cdot (1-0.40251) \cdot 0.40131 \cdot 0.6 \cdot (0.69592-1) \cdot 0.69592 \cdot (1-0.69592) = -0.0037263$$

I la resta de *gradients*:

$$\frac{\partial E}{\partial W2} = -0.00588 \quad \frac{\partial E}{\partial W5} = -0.0048955 \quad \frac{\partial E}{\partial W8} = -0.00342 \quad \frac{\partial E}{\partial W3} = -0.0046252 \quad \frac{\partial E}{\partial W6} = -0.0038508$$
$$\frac{\partial E}{\partial W9} = -0.0026902$$

3.2.6 Weights updating

Per actualitzar els pesos, restarem, al valor actual del pes, el producte del *gradient* obtingut pel pes en qüestió i el *learning rate*. En cas d'utilitzar *momentum*, es sumarà el producte del *gradient* calculat en la iteració anterior i el *momentum*.

$$Wi = Wi - \eta \cdot \frac{\partial E}{\partial Wi} + \gamma \cdot \Delta W^{t-1} i$$

(On "η" és el learning rate, "γ" és el momentum i $\Delta W^{t-1} i$ *és el gradient obtingut en la iteració anterior que, en aquest cas, serà zero perquè aquesta és la primera iteració).*

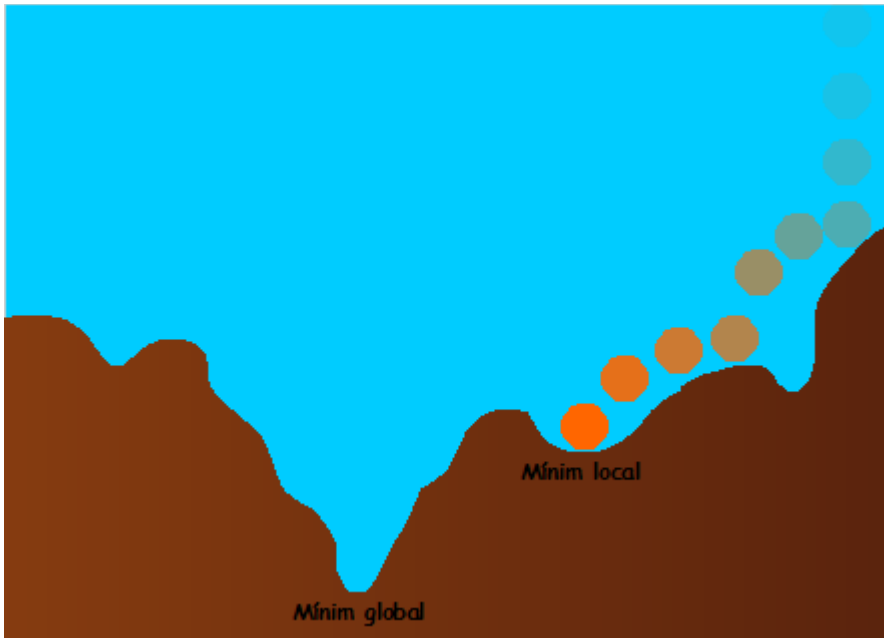
Per tant, suposant que η=0.1:

$$W1 = -0.59936 \quad W2 = 0.40059 \quad W3 = 0.50046 \quad W4 = 0.55053 \quad W5 = -0.64951 \quad W6 = 0.67039 \quad W7 = 0.20037$$
$$W8 = 0.42034 \quad W9 = -0.39973 \quad W10 = 0.60259 \quad W11 = 0.53315 \quad W12 = 0.48439$$

Aquest procés de *forward propagation*, *backpropagation* i *weights updating* s'haurà de fer de forma iterativa per a tots els elements del *training set* i aquesta iteració es repetirà fins que la xarxa neuronal convergeixi.
Si ara es tornés a aplicar *forward propagation* amb el mateix *input* per comprovar com ha impactat l'actualització de pesos sinàptics sobre el *output* de la xarxa neuronal (z) obtindríem z=0.69714. Tenint en compte que abans de la actualització s'havia obtingut z=0.69592 i que el *target output (t)* és t=1. Es pot dir que, basant-se només en aquest element del *training set,* la xarxa neuronal ha millorat, ja que l'error és inferior que abans. Naturalment, és important que el *training set* contingui molts elements per tenir un *training set* representatiu i reduir el *overfitting*.

3.3 Investigació d'una tècnica nova de xarxa neuronal

Durant l'estudi del funcionament de les xarxes neuronals, vaig veure que per explicar el funcionament de l'entrenament de les xarxes neuronals, sovint s'utilitzava l'exemple d'una “pilota” que es deixava caure sobre un “terreny” molt accidentat. Evidentment, això és només un símil per explicar de forma simple una cosa que en realitat és força més complexa.



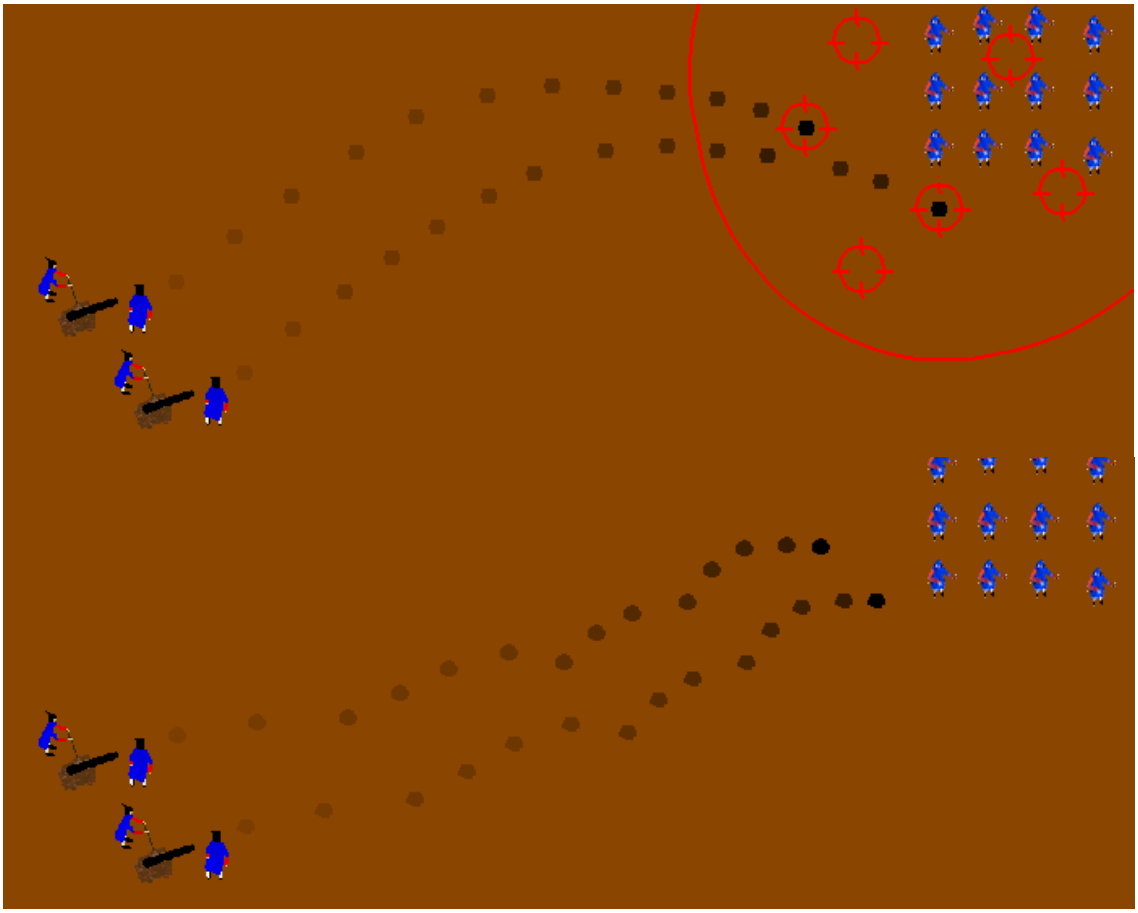
Il·lustració 29: Es deixa caure la pilota que va descenent fins que convergeix en un mínim local.

El gran problema que s'expressava era el fet que es podia saber (i decidir) a quin punt del “espai” es deixava caure la “pilota”, però no es podia pas veure en quin punt del “terreny” es deixava caure amb antelació. És a dir, el fet que convergís en un punt més alt o més baix depenia, en gran part, de l'atzar.
Al imaginar-me aquest símil vaig enrecordar-me (segurament a causa del sistema associatiu que s'ha descrit a l'apartat 3.1.1) d'una tècnica militar dels segles XVII-XVIII i que havia llegit feia anys enrere en diversos llibres i documents històrics. La tècnica havia estat batejada amb el nom francès de *Tir à Ricochet* i s'atribueix a *Sébastien Le Prestre de Vauban*, considerat el millor enginyer militar de l'època.

La tècnica estava enfocada a solucionar un problema comú de l'artilleria de campanya i aquest problema era el fet que, malgrat l'artilleria tenia molt efecte sobre estructures com baluards, torres... tenia molt poc efecte sobre la infanteria, cavalleria i artilleria enemiga, ja que era molt difícil encertar a un blanc tan petit en una època en la que la precisió de les armes de foc era tan baixa... En aquests casos, només tenia eficàcia des d'un punt de vista psicològic, tant pels soldats enemics com pels seus cavalls. Excepte en cas que l'enemic es trobés a prop que, si s'utilitzaven sacs de metralla, sí que podien tenir certa eficàcia.

Per solucionar aquest problema, *Vauban* va proposar el *Tir à Ricochet* que era, bàsicament, disparar els canons amb poca quantitat de pólvora i amb un angle de tir molt reduït, amb la finalitat que la bala (que era rodona) anés rassa i, al impactar contra el terra, enlloc de quedar-se clavada, anés botant com una pilota d'acer pel camp de batalla, emportant-se per davant a tothom que hi hagués en la seva trajectòria.
Vauban va demostrar que aquesta tècnica, ben emprada, incrementava (i molt) la probabilitat de vèncer quan el terreny era l'adequat, fins al punt que al llibre "*De l'attaque et de la défense des places"*, es pot llegir una cita seva que deia:

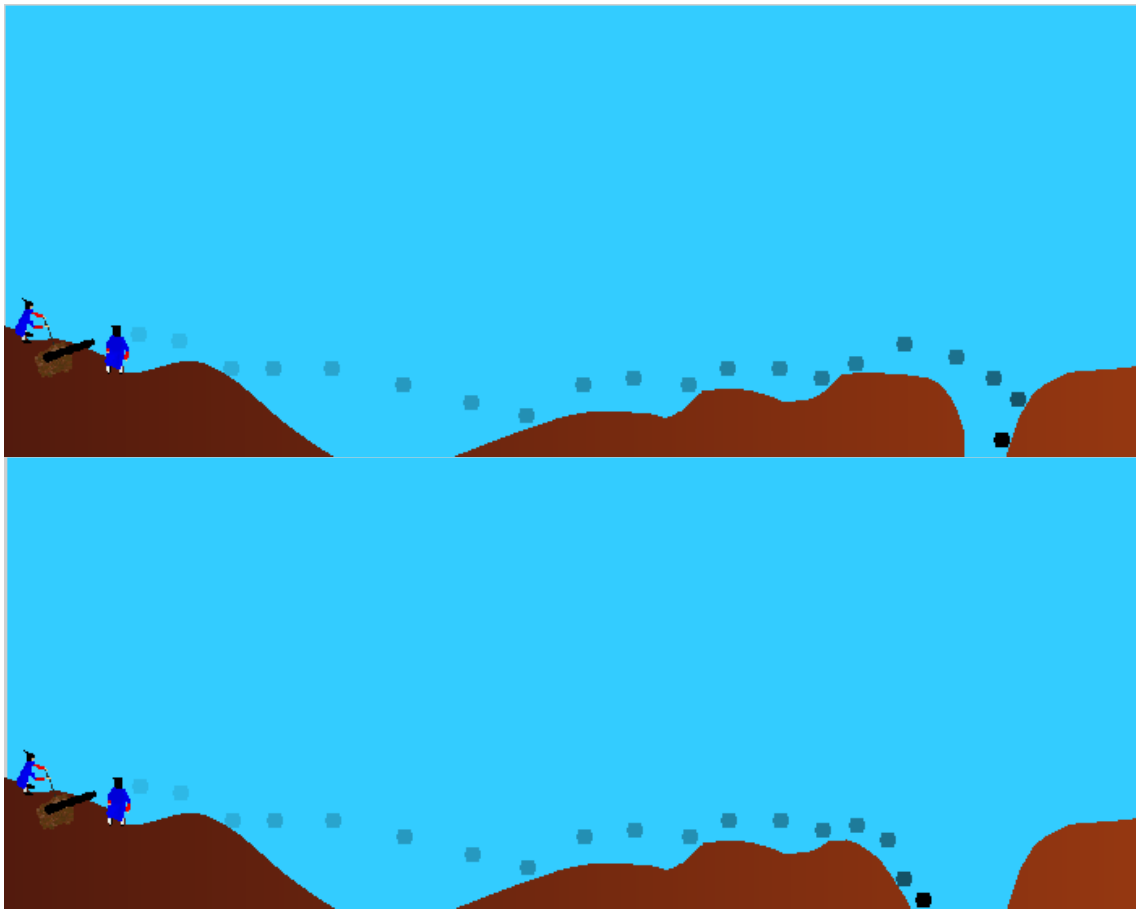
"*...On est bien-tôt accoutumé au Ricochet, qui est la meilleure et la plus excellente manière d'employer utilement le Canon dans les Siéges.*" *[XIII]* (pàg. 75).



Il·lustració 30: Tir à Ricochet (imatge inferior), les bales s'utilitzen com "pilotes" d'acer que boten pel camp de batalla. Amb aquest mètode era molt més fàcil causar baixes que amb el mètode estàndard (imatge superior).

La meva idea era fer exactament el mateix sobre el "terreny" del símil que he exposat. Com que no veiem el "terreny" on deixarem caure la "pilota" (d'alguna manera, equivalent a la falta de precisió de l'artilleria), la idea és que la "pilota" no baixi pel primer pendent que trobi, sinó que sigui una bala de canó disparada "à la Ricochet" i vagi botant pel terreny sense deixar-se caure per cap pendent (només deixant-se caure/influenciar lleugerament per ell), excepte

si es donen certes circumstàncies; per exemple, si l'últim “bot” que ha fet té una caiguda en profunditat que supera un llindar concret i la profunditat actual supera un altre llindar concret. En aquest cas, la “pilota” abandona l'estat de *Ricochet* i fa un *gradient descent* normal i corrent.



Il·lustració 31: Exemple de l'aplicació del “Tir à Ricochet” en xarxes neuronals.

Aquesta extrapolació es basa, doncs, en la hipòtesi que si després d'una actualització de pesos la qualitat de la predicció de la xarxa neuronal ha millorat de forma significativa i, a més a més, la qualitat actual és raonablement bona, això pot implicar que si apliquem *gradient descent* a partir d'aquest punt podem acabar trobant un mínim local força bo.
Com és evident, que es donin aquestes condicions no implica, en absolut, que el mínim local hagi de ser més bo (profund). Per exemple, podem trobar-nos en la situació que la “pilota” es trobi a certa profunditat i l'ultim “bot” hagi suposat un descens important per la “pilota” i que, malgrat això, convergeixi poc després.
Ara bé, l'objectiu d'anàlisi d'aquesta tècnica no és pas que de mitja es puguin obtenir bons resultats, sinó que en els millors casos es puguin obtenir bons resultats, ja que en aquest projecte es van creant instàncies de xarxes neuronals contínuament i la funcionalitat 2 selecciona aquelles xarxes neuronals que han obtingut una major precisió per usar-les per fer la predicció

que farà servei a la mateixa funcionalitat. Per tant, a nosaltres no ens interessa la mitja, sinó només els millors casos.

L'anàlisi d'aquesta nova tècnica es troba a l'annex.

3.4 Creació de xarxes neuronals a partir d'un senzill algoritme genètic

Un algoritme genètic *[XV]* és un algoritme que té l'objectiu de simular el procés de selecció natural fent que aquells algoritmes o agents que s'adapten millor a l'objectiu pel qual han estat programats es "reprodueixin" abans de ser destruïts, mentre que els que no s'han adaptat tan bé són destruïts sense reproduir-se. D'aquesta manera, després d'unes quantes "generacions", s'obtenen uns algoritmes (o agents) més òptims que els del principi.

El sistema predictiu de la funcionalitat 2 incorpora un senzill algoritme genètic que s'encarrega de fer "reproduir" aquelles xarxes neuronals que han obtingut més bons resultats.

Existeixen formes d'algoritmes genètics molt diverses. En aquest cas, el disseny és molt simple:

Inicialització: Es genera una població de xarxes neuronals a l'atzar. Cada un dels espècimens té un conjunt de paràmetres (cromosoma) que es declaren en el moment de generar-los (instanciar-los).
Informació del cromosoma:

- Subtipus de xarxa neuronal (*Estàndard* o *Ricochet*).
- Transformació de dades (*Estandarització* o *Normalització*).
- Nombre de capes de la xarxa neuronal.
- *Learning rate*.
- *Momentum*.
- Llindar d'inclinació (només si el tipus de xarxa neuronal és *Ricochet*).
- Llindar de qualitat (només si el tipus de xarxa neuronal és *Ricochet*).

Evaluació: Tots els espècimens (xarxes neuronals) de la generació s'entrenaran i, al final d'aquest entrenament, cada un d'ells obtindrà una puntuació (la precisió que ha adquirit).

Selecció: Es seleccionen aquells espècimens que tenen una puntuació més alta per ser creuats amb la següent generació.

Encreuament i mutació: En aquesta implementació no es fa un encreuament de gens entre parelles d'espècimens i una posterior mutació d'alguns d'aquests gens, sinó que només hi ha un progenitor i cada un dels gens del descendent és igual al gen respectiu del seu progenitor amb certa variació generada al atzar. Per tant, no hi ha cap encreuament de gens, encara que sí que hi ha mutació (però aquesta té un impacte parcial sobre els diferents gens).

Cal dir, però, que els paràmetres “subtipus de xarxa neuronal” i “transformació de dades” són immutables. Per tant, aquests dos gens seran idèntics entre el progenitor i el seu descendent.
En concret, el mode de funcionament (explicat des d'un punt de vista més proper a la implementació) és el següent:

1) Es crea una generació de xarxes neuronals, generades a l'atzar, que s'entrenen.
2) Un cop acabat l'entrenament es destrueixen les xarxes neuronals d'aquesta generació i es crea una nova generació de xarxes neuronals. Algunes d'aquestes noves xarxes neuronals tindran unes característiques generades completament al atzar (l'algoritme no s'aturarà mai i, per tant, ens interessa que no convergeixi mai) i d'altres seran còpies de les millors xarxes neuronals de la generació anterior amb unes característiques lleugerament diferents (mutacions).
3) Si alguna xarxa neuronal de la generació anterior és prou bona com per considerar-se una de les millors xarxes neuronals de la història (d'entre totes les generacions que han existit dins d'aquest algoritme genètic), s'afegirà a un *ranking* de les millores xarxes neuronals de la història.
4) La nova generació creada en el punt 2 es posarà a entrenar i, un cop s'hagi acabat aquest entrenament, es torna al punt 2 (i així successivament, en forma de bucle).

Seguint els quatre punts anteriors s'obté un *ranking* de les millors xarxes neuronals creades fins al moment, aquest *ranking* s'anirà actualitzant (millorant) amb el temps; a mesura que vagin apareixen noves xarxes neuronals que siguin més òptimes que les que es troben al *ranking*.
S'utilitzarà aquest *ranking* per fer les prediccions de la funcionalitat 2, mitjançant votacions ponderades. En altres paraules, cada una de les xarxes neuronals del *ranking* indicarà el seu *output* (si considera que l'usuari deixarà de ser productiu o no) i aquest *output* (vot) valdrà més o menys segons la qualitat de la xarxa neuronal en qüestió. Es farà un sumatori de tots els vots i, a partir d'aquí, es decidirà la resolució final.

4. Conclusions

4.1 Assoliment dels objectius plantejats inicialment

Hi havia 4 objectius imprescindibles i 1 de prescindible:
1- Detecció que l'usuari està inactiu o fora del PC durant un temps determinat (imprescindible).
2- Detecció que l'usuari s'ha connectat a un lloc web no permès (imprescindible).
3- Detecció que l'usuari no mou el ratolí durant un temps determinat (imprescindible).
4- Implementació d'alguna funcionalitat extra com fer sonar alguna alarma a una hora concreta o fer sonar música relaxant de fons (imprescindible).
5- Implementar un sistema d'intel·ligència artificial al núvol (prescindible).

Tots aquests 5 objectius s'han pogut assolir correctament.

4.2 Seguiment de la planificació

La planificació proposada s'ha assolit correctament. Tot i això, la tasca "Estudi d'algoritmes predictius" es va assolir més ràpidament del previst arribant a la conclusió que el millor era utilitzar una xarxa neuronal. Conseqüentment, es va decidir afegir una nova tasca "Investigació d'una tècnica nova de xarxa neuronal" que es va fer en paral·lel amb la tasca “Desenvolupament de l'algoritme predictiu” en la que s'intentava dissenyar un nou tipus de xarxa neuronal i comprovar la seva viabilitat.
Aquesta nova tasca es va poder assolir correctament i es van poder extreure conclusions. Hauria estat interessant poder fer moltes més proves de les que es van fer però, naturalment, aquesta era només una tasca del projecte i, per tant, tampoc se li podien dedicar moltes més hores de les que ja se li van dedicar.

Totes les tasques programades en la proposta inicial eren necessàries per obtenir el resultat desitjat i tampoc va caldre afegir-ne cap (a part de la tasca esmentada en el paràgraf anterior; però tampoc era estrictament necessària, sinó un extra).

4.3 Experiència i lliçons apreses

En aquest punt crec que és convenient parlar en primera persona.

Abans de tancar el TFG, m'agradaria relatar alguns aspectes rellevants (de forma autocrítica) de la meva experiència fent-lo.
En concret, crec que vaig ser quelcom temerari a l'hora de planejar totes les funcionalitats que tindria el software:
Per començar, considero que els requisits implicaven una programació i una recerca massa voluminosos pel temps disponible. Això va suposar haver de

dedicar més hores diàries de les previstes i que alguns punts del treball acabessin sent funcionals i acceptables però sense assolir el punt de robustesa i solidificació que, personalment, hauria desitjat.

D'altra banda, decidir implementar la xarxa neuronal des de zero també va ser temerari, ja que mai n'havia implementat cap i no era prou conscient de la dificultat que em podia trobar (especialment, a nivell matemàtic) a l'hora d'implementar-la, tenint també en compte que no disposava de massa temps. Si bé és cert que, afortunadament, no van aparèixer massa complicacions, podrien haver aparegut, provocant el no assoliment de la planificació i la funcionalitat 2.

Des d'una perspectiva més concreta, considero que va ser un error el fet de considerar que la funció logística era la millor funció d'activació, en el context de la funcionalitat 2, sense haver-ne provat d'altres i haver fet una comparativa. És cert que ho vaig justificar, però al no haver implementat les diferents funcions d'activació i haver fet el *testing* necessari amb cada una d'elles, aquesta justificació no deixa de ser una hipòtesi no provada.

També vull aclarir que malgrat no vaig poder demostrar que la tècnica nova de xarxa neuronal que em vaig inventar inspirada en el *Tir à Ricochet* funcionés (en els millors casos) millor que la estàndard, sí que considero que va ser interessant intentar redissenyar un algoritme existent i analitzar els resultats obtinguts. A més a més, també crec que aquesta nova tècnica (conjuntament amb el software desenvolupat) dóna una tonalitat blanca i creativa a aquest treball.

Finalment, vull expressar que ha estat molt interessant el fet d'estudiar i desenvolupar una xarxa neuronal des de zero i, va ser precisament durant el seu desenvolupament, que vaig adonar-me del potencial que tenen aquests petits “cervells artificials” que permeten descobrir patrons d'alta complexitat (poden arribar a expressar qualsevol funció matemàtica imaginable) sobre un conjunt de dades.
També ha estat molt gratificant el fet de desenvolupar una eina informàtica que pot tenir certa utilitat per a persones que treballen al PC i es distreuen fàcilment (a mi, personalment, m'ha ajudat molt).

4.4 Línies de treball futur

Si s'hagués de donar continuïtat a aquest projecte, seria en les següents línies:

- Obtenir opinions dels usuaris i mantenir el *software* en conseqüència.
- Posar en pràctica tècniques de *Data Mining* per millorar el software (i això inclou, verificar que la xarxa neuronal funcioni bé a l'hora de fer prediccions amb els usuaris reals).
- Fer un estudi més exhaustiu (tant en amplitud com en profunditat) de la xarxa neuronal de subtipus *Ricochet*.

5. Glossari

Agents: Entitat capaç d'actuar de manera intel·ligent segons el context i l'estat en el que es trobi.

Algoritme: Conjunt de pasos que, a partir d'uns valors d'entrada, és capaç de resoldre un problema generant la sortida adequada.

Algoritme supervisat: Algoritme d'aprenentatge computacional que utilitza una conjunt de dades i els seus resultats per entrenar-se, de manera que quan un cop entrenat se li passa un conjunt de dades del qual no se'n sap el resultat, l'algoritme és capaç de predir-lo.

Arquitectura client-servidor: Model de disseny de sotware en el que el conjunt de tasques que es realitzen es reparteixen entre el una part client i una altra part servidor.

Artilleria de campanya: Artilleria mòbil i relativament poc pesada, utilitzada en els camps de batalla.

Booster: Software per millorar la productivitat.

Cicle de CPU: És cada senyal que rep el processador per a realitzar una instrucció o part d'una instrucció.

Dataset: Conjunt de dades que s'utilitzen tant per entrenar un algoritme com per comprovar/validar la precisió d'aquest.

Deeplearning4j: Llibreria de deep learning per *Java*.

Diagrama de Gantt: Diagrama per planificar el temps previst per fer certes tasques.

Diagrames UML: Llenguatge de modelat de sistemes de software.

Diàleg: Finestra filla que s'obre dins del context d'una altra finestra que queda inusable fins que la finestra filla es tanca.

DNS: Sistema que tradueix noms a direccions IP.

Eclipse: És un Entorn de Desenvolupament Integrat (IDE).

Epoch: Cada una de les iteracions de tots els vectors d'entrenament d'un training set.

Facebook: Xarxa social.

Focusatwill: Software per millorar la productivitat.

Gradient vanishing: És un problema que apareix en una xarxa neuronal quan el càlcul d'un gradient és tan petit que no permet modificar el seu valor.

Hardware: Components físics d'un ordinador.

Hiperplà: Una recta que divideix un espai en dues meitats.

IDE: Entorn de desenvolupament integrat.

Item: Cada un dels elements d'un dataset.

Input: Valors d'entrada.

IP: Nombre identificatori d'un node dins d'una xarxa.

Java: Llenguatge de programació.

Kernel: Nucli.

Learning rate: Ponderació en la que s'actualitzen els gradients.

Memòria eidètica: Habilitat de recordar imatges projectant-les en la imaginació de forma molt precisa.

Mínim global: Punt mínim absolut (màxima precisió) de la xarxa neuronal.

Mínim local: Punt mínim no absolut (convergència no òptima) de la xarxa neuronal.

Momentum: Fa que el gradient descent agafi certa embranzida segons l'última actualització de pesos per ajuda a evitar una convergència massa ràpida a un mínim local.

Nivell de significació: Indica com de improbable és que la precisió obtinguda depengui del atzar.

Nucli de CPU: Cada CPU que conté el microprocessador.

Ones binaurals: Un tipus de so/música que ajuda a incrementar la concentració.

OpenCV: Llibreria de visió per ordinador.

OpenSource: Un tipus de llicència en el que el codi font es fa públic.

Output: Sortida generada.

Overfitting: Es produeix aquest problema quan un algoritme es sobre-ajusta al conjunt d'entrenament.

PC: Ordinador personal.

Perceptró multicapa: Una xarxa neuronal formada per múltiples capes.

Ports: Interfície per on es poden enviar i rebre dades.

PostgreSQL: Gestor de base de dades.

Random forest: Mètode de classificació, regressió o semblant que genera un conjunt d'arbres de decisió que, entre tots, decideixen (mitjançant votacions) quin serà el output final.

Regla de la cadena: Formula matemàtica.

RescueTime: Software per millorar la productivitat.

Sel·lecció natural: Procés evolutiu d'una espècie basat en la descendència dels gens que millor s'adapten al medi.

Sinestèsia: Condició neurològica en la qual es barregen els sentits.

Software: Programa informàtic.

Tallafocs: Sistema de seguretat informàtica.

TDAH: Transtorn per Dèficit d'Atenció i Hiperactivitat.

TensorFlow: Llibreria especialitzada en deep learning.

Test data/set: Conjunt de dades de prova.

Threshold/llindar: Frontera lògica.

Training data/set: Conjunt de dades d'entrenament.

Twitter: Xarxa social.

Webcam: Càmera preparada per usar conjuntament amb aplicacions informàtiques.

Weka: Llibreria de machine learning.

6. Bibliografia

I) Bryan WC Chung, Hong Kong, 2017 , *Pro processing for images and computer vision with OpenCV*
https://discovery.uoc.edu/iii/encore/record/C__Rb1065803__Sopencv__Orightresult__U__X6?lang=spi&suite=def

II) TensorFlow (Wikipedia) https://en.wikipedia.org/wiki/TensorFlow (del 8 al 14 d'octubre del 2018)

III) Weka (Wikipedia) https://en.wikipedia.org/wiki/Weka_(machine_learning) (del 8 al 14 d'octubre del 2018)

IV) Deeplearning4j (Wikipedia) https://en.wikipedia.org/wiki/Deeplearning4j (del 8 al 14 d'octubre del 2018)

V) LANE, J. D., S. J. KASIAN, J. E. OWENS AND G. R. MARSH, 1998, *Binaural Auditory Beats Affect Vigilance Performance and Mood*
https://www.sciencedirect.com/science/article/abs/pii/S0031938497004368
(12 de novemebre del 2018)

VI) Richard Cauley Kennerly, *An Empirical Investigation Into the Effect of Beta Frequency Binaural-beat Audio Signals on Four Measures of Human Memory, ADD / ADHD*
http://www.astarapoint.com/Binaural.pdf
(12 de novemebre del 2018)

VII) Shai Shalev-Shwartz i Shai Ben-David, Nova York, 2014, *Understanding Machine Learning: From Theory to Algorithms*

VIII) Stuart J. Russell i Peter Norvig, 2015, Artificial Intelligence: A modern approach (Third edition)

IX) Jeff Hawkins amb Sandra Blakeslee, 2005, On intelligence: How a New Understanding of the Brain will Lead to the Creation of Truly Intelligent Machines

X) O.S. Eluyode i Dipo Theophilus Akomolafe, Comparative study of biological and artificial neural networks
https://pdfs.semanticscholar.org/e6b7/2382550aa56b9403b524308e3303137f6e8f.pdf
(24 de novembre del 2018)

XI) Michael Taylor, 2017, Neural networks: A visual introduction for beginners

XII) Yann LeCun, Leon Bottou, Genevieve B. Orr i Klaus-Robert Müller, Efficient BackProp
http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf

XIII) Sébastien Le Prestre de Vauban, 1737, De l'attaque et de la defense des places / par Mr. de Vauban.

XIV) Bottou, L, 1991, Stochastic gradient learning in neural networks: Proceedings of Neuro-Nımes

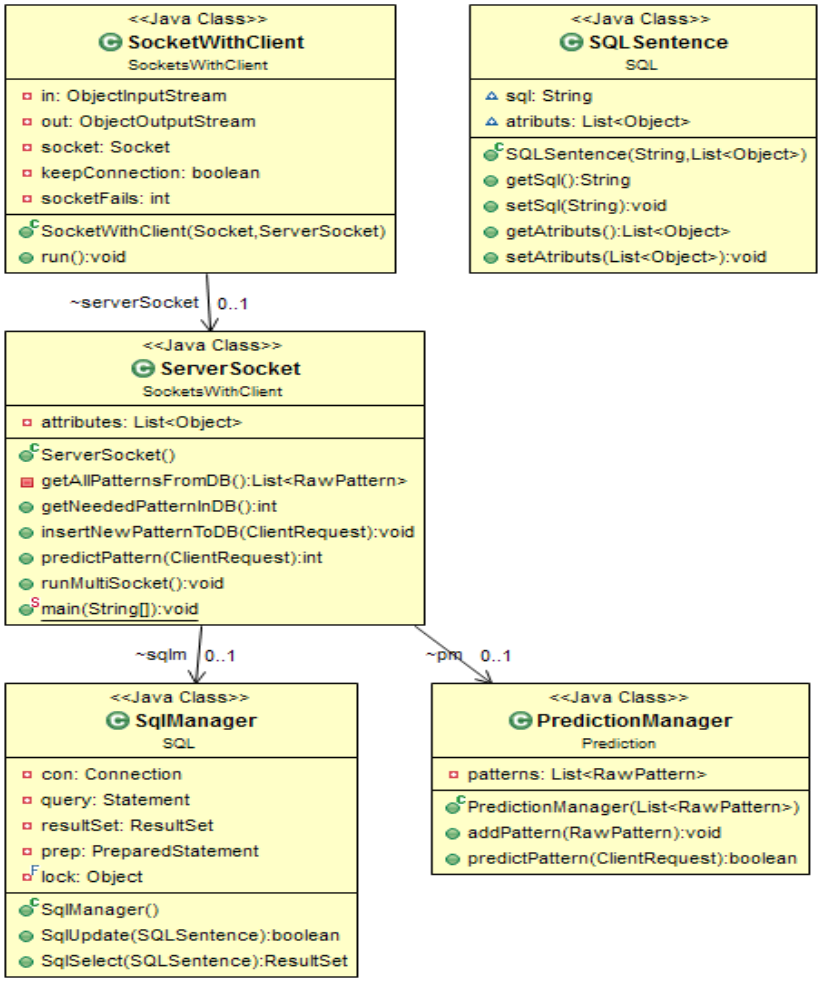
XV) Algoritme genètic (Wikipedia) https://es.wikipedia.org/wiki/Algoritmo_gen%C3%A9tico (del 20 al 28 de desembre del 2018)

7. Annexos

7.1 Implementació

7.1.1 Diagrames de classes

Classes del server-side:



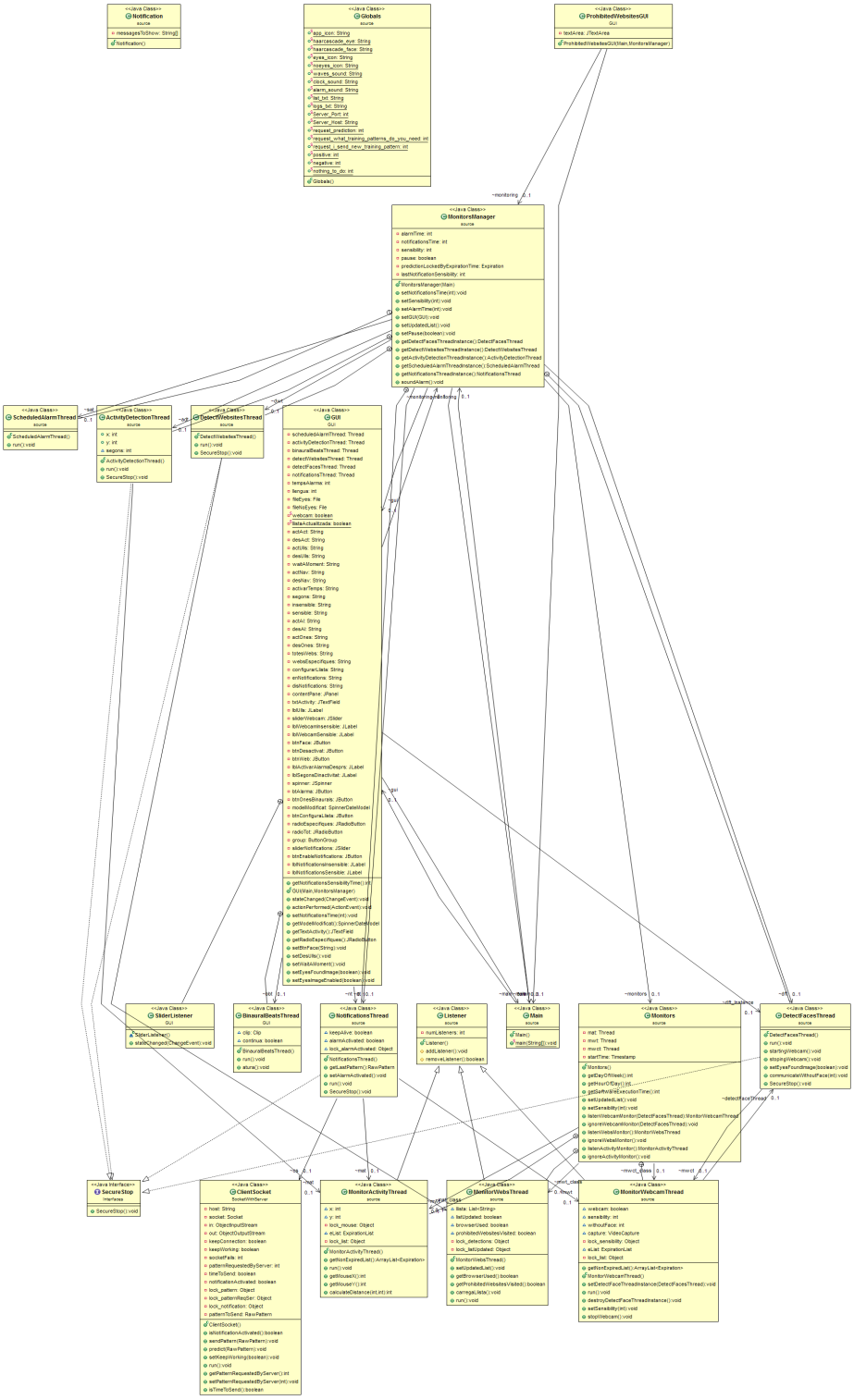
Il·lustració 32: Server-side

ServerSocket: S'encarrega de rebre les connexions entrants i assignar-lis un *SocketWithClient*. També conté els mètodes genèrics relacionats amb el sistema predictiu.

SocketWithClient: S'encarrega de la comunicació amb el client.

PredictionManager: S'encarrega de fer les prediccions (encara no implementat del tot).
SqlManager i SQLSentence: S'encarreguen d'interactuar amb la base de dades.

Classes del client-side:



Il·lustració 33: Client-side

Notification: Mostra una notificació per pantalla.

Globals: Variables globals (paths, configuració del servidor...).

ProhibitedWebsitesGUI: Mostra un *Dialog* per configurar la llista de llocs web prohibits.

MonitorsManager: Gestiona els diferents monitors (detecció de l'usuari a través de Webcam, activitat del ratolí...).

Monitors: Incorpora els diferents monitors que seran controlats des de *MonitorsManager*.

GUI: Interfície gràfica del *software*.

DetectWebsitesThread, ActivityDetectionThread, DetectFacesThread: Fan consultes als monitors per decidir si cal fer sonar l'alarma o no.

BinauralBeatsThread: Fa sonar un àudio que incrementa la concentració de l'usuari.

ScheduledAlarmThread: Fa sonar una alarma a una hora concreta.

ClientSocket: S'encarrega de la comunicació amb el servidor.

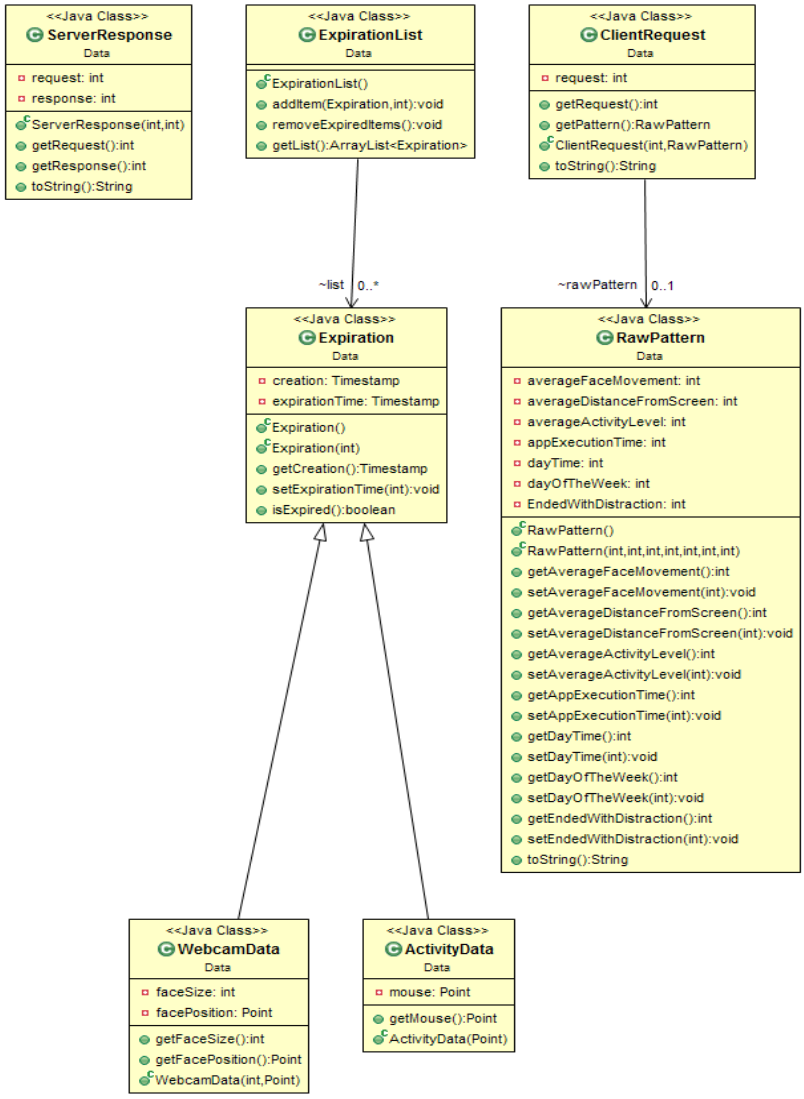
Main: Inici del programa.

Listener: Permet tenir controlat el nombre de de *listeners* d'un monitor.

NotificationsThread: Utilitza el *ClientSocket* per comunicar-se amb el servidor.

MonitorActivityThread, MonitorWebsThread, MonitorWebcamThread: Són les classes que recullen dades per si algun *listener* les necessita.

Classes compartides:



Il·lustració 34: Classes compartides

ServerResponse, ClientRequest, RawPattern: Classes utilitzades per encapsular la informació que s'envia per *socket*.

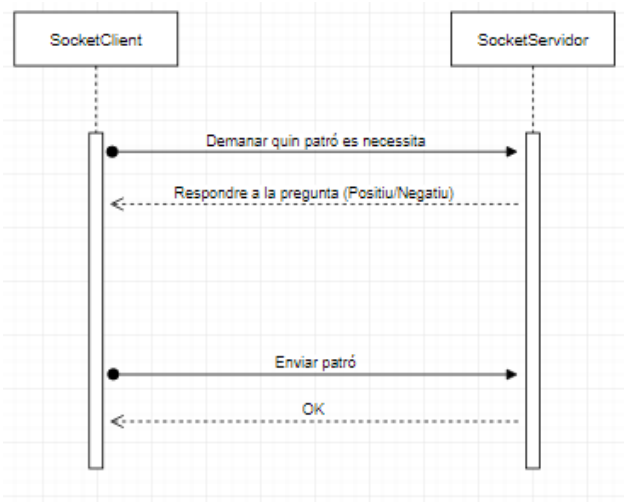
ExpirationList: Permet desar objectes amb un temps de vida concret.

Expiration: S'utilitza com a classe *pare* per les classes que interessa afegir-les a una *ExpirationList*.

WebcamData, ActivityData: Son classes *filles* de *Expiration* per poder afegir-les dins d'una *ExpirationList*.

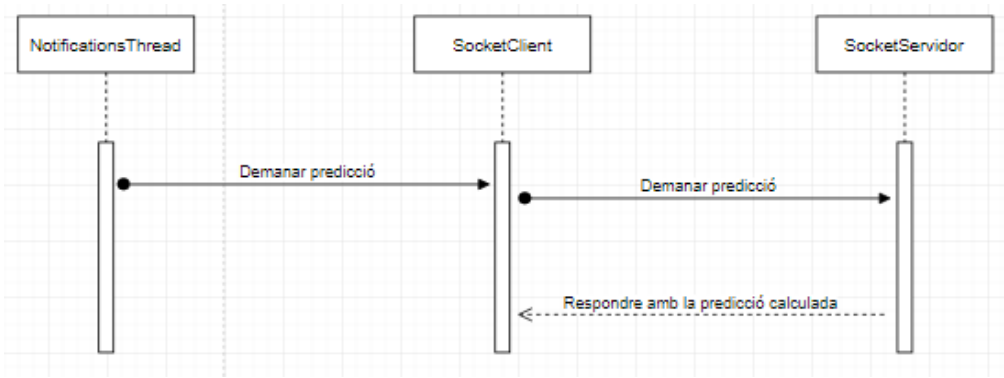
7.1.2 Diagrames de flux

La comunicació entre el client i el servidor és simple i només es basa en 3 tipus de request:
-Demanar quin tipus de patró es necessita.
-Enviar el patró que es necessita.
-Demanar que es faci una predicció.



Il·lustració 35: Demanar quin patró es necessita

La il·lustració 5 mostra com s'envien els patrons d'entrenament; el client li demana al servidor si necessita un patró positiu o negatiu i, un cop té el patró requerit, li envia al servidor.



Il·lustració 36: Demanar que es faci una predicció

La il·lustració 6 mostra com es demana que es faci una predicció; s'envia al servidor un patró i el servidor respon si el patró ha donat positiu o negatiu.

7.2 Anàlisi de la xarxa neuronal

En aquest apartat s'explicarà com s'ha implementat la xarxa neuronal de la funcionalitat 2 i s'analitzarà la nova tècnica de xarxa neuronal que s'ha proposat en aquest projecte (la xarxa neuronal inspirada en el *Tir à Ricochet,* apartat 3.3).

7.2.1 Disseny de la xarxa neuronal

La xarxa neuronal té les següents característiques:

Funció d'activació: logística.
Gradient descent: *Stochastic Gradient Descent*.
Bias: Sí.
Tipus de xarxa neuronal: *Feedforward Neural Network*.
Subtipus de xarxa neuronal: Configurable (*Estàndard* o *Ricochet*).
Transformació de dades: Configurable (*Estandardització* o *Normalització*).
Nombre de capes de la xarxa neuronal: Configurable.
Learning rate: Configurable.
Momentum: Configurable.
Altres: Llindar d'inclinació i de qualitat (només si el tipus de xarxa neuronal és *Ricochet*).

Es va decidir utilitzar una funció d'activació logística perquè té una forma suavitzada que permet actualitzar els pesos d'una forma molt gradual. Presenta alguns inconvenients com el *gradient vanishing*, però tenint en compte que la funcionalitat 2 estarà tot el dia creant xarxes neuronals noves (de forma indefinida) i que, per fer les prediccions, només s'utilitzaran les millors xarxes neuronals que s'hagin creat, es pot assumir el risc que el *gradient vanishing* comporta, ja que no es depèn només d'una sola instància de xarxa neuronal.

També és va decidir utilitzar *Stochastic Gradient Descent* amb per questions de rendiment (sobretot quan el *training set* es fa molt gran) i perquè, com que després de cada element del *training set*, es fa una actualització de pesos, això permet generar cert “soroll” i “atzar” en la línia de convergència. D'aquesta forma, impedim que la xarxa neuronal convergeixi massa ràpidament i forcem a que “busqui” una mica més abans de convergir.

L'ús de *biaix* es va considerar important per ajudar a moure les funcions d'activació a la dreta o a l'esquerra per adaptar-se millor al llindar.

D'altra banda, pel tipus de problema que es vol resoldre, *Feedforward Neural Network* és el tipus de xarxa neuronal més adequat, ja que no necessitem fer cap tipus de recursivitat, modularitat...

Finalment, com que la resta de característiques són configurables, no s'ha pres cap decisió predeterminada.

7.2.2 Comprovació del funcionament de la xarxa neuronal

Per comprovar el correcte funcionament de la xarxa neuronal s'han fet entrenaments amb diversos conjunts de dades. Tots els *Datasets* són de classificació binària i contenen el mateix nombre d'elements per cada una de les dues classes. Això és així, car els patrons de la funcionalitat 2 poden pertànyer a dues classes (deixarà de ser productiu o no) i el programa servidor sol·licita als programes clients patrons de la classe menys nombrosa dins de la base de dades de patrons:

```
0 Total error 0.2542977
Training set -> Correct:407 Incorrect:405   50.123154%
Test set -> Correct:204 Incorrect:204   50%

1 Total error 0.22476548
Training set -> Correct:593 Incorrect:219   73.02956%
Test set -> Correct:185 Incorrect:223   45%

2 Total error 0.22170466
Training set -> Correct:727 Incorrect:85   89.53202%
Test set -> Correct:185 Incorrect:223   45%

3 Total error 0.21985811
Training set -> Correct:774 Incorrect:38   95.3202%
Test set -> Correct:191 Incorrect:217   46%

4 Total error 0.21808115
Training set -> Correct:786 Incorrect:26   96.798035%
Test set -> Correct:195 Incorrect:213   47%

5 Total error 0.21631883
Training set -> Correct:802 Incorrect:10   98.76847%
Test set -> Correct:203 Incorrect:205   49%

6 Total error 0.21455583
Training set -> Correct:810 Incorrect:2   99.75369%
Test set -> Correct:204 Incorrect:204   50%

7 Total error 0.21277839
Training set -> Correct:812 Incorrect:0   100.0%
Test set -> Correct:204 Incorrect:204   50%

18 Total error 0.18953897
Training set -> Correct:808 Incorrect:4   99.50739%
Test set -> Correct:210 Incorrect:198   51%

19 Total error 0.1871529
Training set -> Correct:808 Incorrect:4   99.50739%
Test set -> Correct:225 Incorrect:183   55%

20 Total error 0.18462485
Training set -> Correct:808 Incorrect:4   99.50739%
Test set -> Correct:246 Incorrect:162   60%

21 Total error 0.18193834
Training set -> Correct:808 Incorrect:4   99.50739%
Test set -> Correct:269 Incorrect:139   65%

22 Total error 0.17908584
Training set -> Correct:808 Incorrect:4   99.50739%
Test set -> Correct:286 Incorrect:122   70%

23 Total error 0.17606893
Training set -> Correct:808 Incorrect:4   99.50739%
Test set -> Correct:295 Incorrect:113   72%

24 Total error 0.17289741
Training set -> Correct:809 Incorrect:3   99.63054%
Test set -> Correct:308 Incorrect:100   75%

25 Total error 0.16959031
Training set -> Correct:809 Incorrect:3   99.63054%
Test set -> Correct:315 Incorrect:93   77%

176 Total error 0.029883184
Training set -> Correct:790 Incorrect:22   97.29064%
Test set -> Correct:380 Incorrect:28   93%
```

Il·lustració 37: Data set:Banknote

Es pot veure com durant els primers *epochs* es genera un fort *overfitting* sense que el *test set* es vegi afectat. Entre els *epochs* 20-30 la precisió del *test set* comença a incrementar-se i, finalment, acaba convergint amb una precisió del *test set* de 93% i un 97% el *training set*. Per tant, la xarxa neuronal ha aprés obtenint una bona precisió malgrat hi hagi cert *overfitting*.

```
0 Total error 0.1741177
Training set -> Correct:674 Incorrect:45   93.74131%
Test set -> Correct:198 Incorrect:163   54%

1 Total error 0.17055438
Training set -> Correct:708 Incorrect:11   98.4701%
Test set -> Correct:201 Incorrect:160   55%

2 Total error 0.16985118
Training set -> Correct:702 Incorrect:17   97.635605%
Test set -> Correct:204 Incorrect:157   56%

3 Total error 0.16918428
Training set -> Correct:698 Incorrect:21   97.07928%
Test set -> Correct:204 Incorrect:157   56%

4 Total error 0.16855018
Training set -> Correct:694 Incorrect:25   96.52295%
Test set -> Correct:207 Incorrect:154   57%

5 Total error 0.16794567
Training set -> Correct:690 Incorrect:29   95.96662%
Test set -> Correct:208 Incorrect:153   57%

6 Total error 0.16736807
Training set -> Correct:685 Incorrect:34   95.27121%
Test set -> Correct:209 Incorrect:152   57%

7 Total error 0.16681445
Training set -> Correct:683 Incorrect:36   94.99304%
Test set -> Correct:211 Incorrect:150   58%
```

```
888 Total error 0.093980685
Training set -> Correct:623 Incorrect:96   86.648125%
Test set -> Correct:227 Incorrect:134   62%

889 Total error 0.093979135
Training set -> Correct:623 Incorrect:96   86.648125%
Test set -> Correct:228 Incorrect:133   63%
```

Il·lustració 38: Data set: Messidor features

En aquest cas la precisió obtinguda és només del 63% amb un *training set* del 86%. Per tant, la precisió no és massa bona i el *overfitting* excessiu.

```
0 Total error 0.21759245
Training set -> Correct:216 Incorrect:105   67.28972%
Test set -> Correct:86 Incorrect:75   53%

1 Total error 0.1767627
Training set -> Correct:309 Incorrect:12   96.26168%
Test set -> Correct:123 Incorrect:38   76%

2 Total error 0.1648002
Training set -> Correct:311 Incorrect:10   96.884735%
Test set -> Correct:134 Incorrect:27   83%

3 Total error 0.15315597
Training set -> Correct:311 Incorrect:10   96.884735%
Test set -> Correct:141 Incorrect:20   87%

4 Total error 0.14183804
Training set -> Correct:312 Incorrect:9   97.196266%
Test set -> Correct:147 Incorrect:14   91%

5 Total error 0.13105
Training set -> Correct:311 Incorrect:10   96.884735%
Test set -> Correct:150 Incorrect:11   93%

6 Total error 0.12096579
Training set -> Correct:310 Incorrect:11   96.57321%
Test set -> Correct:153 Incorrect:8   95%

7 Total error 0.111704975
Training set -> Correct:310 Incorrect:11   96.57321%
Test set -> Correct:153 Incorrect:8   95%
```

```
80 Total error 0.032082785
Training set -> Correct:313 Incorrect:8   97.50779%
Test set -> Correct:155 Incorrect:6   96%
```

Il·lustració 39: Data set: Breast cancer

En el cas de *breast cancer* s'obté un aprenentatge molt ràpid i una precisió del *test set* del 96% i un 97% el *training set*. Ara bé, tenint en compte que el *dataset* és poc nombrós, és normal obtenir una bona precisió i que l'entrenament sigui ràpid. La precisió real, probablement, seria inferior a aquest 96%.

```
0 Total error 0.24530537
Training set -> Correct:90 Incorrect:91   49.723755%
Test set -> Correct:46 Incorrect:45   50%

1 Total error 0.23818345
Training set -> Correct:90 Incorrect:91   49.723755%
Test set -> Correct:46 Incorrect:45   50%

2 Total error 0.23321974
Training set -> Correct:90 Incorrect:91   49.723755%
Test set -> Correct:46 Incorrect:45   50%

3 Total error 0.2299944
Training set -> Correct:113 Incorrect:68   62.430943%
Test set -> Correct:46 Incorrect:45   50%

4 Total error 0.22794652
Training set -> Correct:114 Incorrect:67   62.983425%
Test set -> Correct:46 Incorrect:45   50%

5 Total error 0.22666365
Training set -> Correct:116 Incorrect:65   64.0884%
Test set -> Correct:46 Incorrect:45   50%

6 Total error 0.225865
Training set -> Correct:140 Incorrect:41   77.34807%
Test set -> Correct:46 Incorrect:45   50%

7 Total error 0.22536775
Training set -> Correct:150 Incorrect:31   82.872925%
Test set -> Correct:49 Incorrect:42   53%

8 Total error 0.2250562
Training set -> Correct:146 Incorrect:35   80.66299%
Test set -> Correct:52 Incorrect:39   57%
```

Il·lustració 40: Data set: Random data

Aquest *Dataset* està format per valors creats de forma aleatòria. Malgrat això, dins l'aleatorietat també es poden trobar certs patrons i el *test set* arriba fins a una precisió del 57%. Tot i això, tenint en compte el fort *overfitting* i que el *dataset* no disposa de molts elements, aquests resultats no són gens fiables. Però sent valors triats a l'atzar no s'esperava més.

```
0 Total error 0.2534085
Training set -> Correct:75 Incorrect:32   70.09346%
Test set -> Correct:28 Incorrect:27   50%

1 Total error 0.24615872
Training set -> Correct:96 Incorrect:11   89.71962%
Test set -> Correct:28 Incorrect:27   50%

2 Total error 0.24503063
Training set -> Correct:96 Incorrect:11   89.71962%
Test set -> Correct:28 Incorrect:27   50%

3 Total error 0.24385479
Training set -> Correct:96 Incorrect:11   89.71962%
Test set -> Correct:28 Incorrect:27   50%

4 Total error 0.242514
Training set -> Correct:96 Incorrect:11   89.71962%
Test set -> Correct:28 Incorrect:27   50%

5 Total error 0.24088512
Training set -> Correct:95 Incorrect:12   88.78505%
Test set -> Correct:32 Incorrect:23   58%

6 Total error 0.23883232
Training set -> Correct:93 Incorrect:14   86.915886%
Test set -> Correct:41 Incorrect:14   74%

7 Total error 0.23619942
Training set -> Correct:94 Incorrect:13   87.85047%
Test set -> Correct:44 Incorrect:11   80%

8 Total error 0.2328084
Training set -> Correct:94 Incorrect:13   87.85047%
Test set -> Correct:50 Incorrect:5   90%

9 Total error 0.22846474
Training set -> Correct:95 Incorrect:12   88.78505%
Test set -> Correct:50 Incorrect:5   90%

10 Total error 0.22298178
Training set -> Correct:95 Incorrect:12   88.78505%
Test set -> Correct:50 Incorrect:5   90%

11 Total error 0.21621725
Training set -> Correct:97 Incorrect:10   90.654205%
Test set -> Correct:52 Incorrect:3   94%

22 Total error 0.1052985
Training set -> Correct:102 Incorrect:5   95.3271%
Test set -> Correct:54 Incorrect:1   98%

23 Total error 0.09846663
Training set -> Correct:102 Incorrect:5   95.3271%
Test set -> Correct:55 Incorrect:0   100%

24 Total error 0.09251404
Training set -> Correct:102 Incorrect:5   95.3271%
Test set -> Correct:55 Incorrect:0   100%

25 Total error 0.08734693
Training set -> Correct:102 Incorrect:5   95.3271%
Test set -> Correct:55 Incorrect:0   100%

26 Total error 0.08292376
Training set -> Correct:102 Incorrect:5   95.3271%
Test set -> Correct:55 Incorrect:0   100%

27 Total error 0.07922492
Training set -> Correct:102 Incorrect:5   95.3271%
Test set -> Correct:55 Incorrect:0   100%

28 Total error 0.07615966
Training set -> Correct:102 Incorrect:5   95.3271%
Test set -> Correct:55 Incorrect:0   100%
```

Il·lustració 41: Data set: Haberman

L'entrenament és molt ràpid i de seguida s'arriba a una precisió del 100%. Però no és tan estrany tenint en compte que el *dataset* és molt petit. És cert, però, que al haver-hi només 3 atributs no es necessiten tants elements, però malgrat això, 162 elements segueix sent massa poc.

```
0 Total error 0.1022483
Training set -> Correct:1934 Incorrect:99   95.13035%
Test set -> Correct:455 Incorrect:566   44%

1 Total error 0.061574362
Training set -> Correct:1964 Incorrect:69   96.605995%
Test set -> Correct:424 Incorrect:597   41%

2 Total error 0.05102378
Training set -> Correct:1966 Incorrect:67   96.70438%
Test set -> Correct:420 Incorrect:601   41%

3 Total error 0.04562869
Training set -> Correct:1968 Incorrect:65   96.80275%
Test set -> Correct:418 Incorrect:603   40%

4 Total error 0.04224391
Training set -> Correct:1968 Incorrect:65   96.80275%
Test set -> Correct:420 Incorrect:601   41%

5 Total error 0.039924603
Training set -> Correct:1968 Incorrect:65   96.80275%
Test set -> Correct:421 Incorrect:600   41%

6 Total error 0.038212936
Training set -> Correct:1973 Incorrect:60   97.04869%
Test set -> Correct:423 Incorrect:598   41%

7 Total error 0.036848456
Training set -> Correct:1973 Incorrect:60   97.04869%
Test set -> Correct:424 Incorrect:597   41%

8 Total error 0.035694722
Training set -> Correct:1972 Incorrect:61   96.99951%
Test set -> Correct:429 Incorrect:592   42%

9 Total error 0.034690212
Training set -> Correct:1972 Incorrect:61   96.99951%
Test set -> Correct:435 Incorrect:586   42%

10 Total error 0.03380277
Training set -> Correct:1973 Incorrect:60   97.04869%
Test set -> Correct:438 Incorrect:583   42%

512 Total error 7.583787E-4
Training set -> Correct:2033 Incorrect:0   100.0%
Test set -> Correct:939 Incorrect:82   91%

513 Total error 7.558463E-4
Training set -> Correct:2033 Incorrect:0   100.0%
Test set -> Correct:940 Incorrect:81   92%

514 Total error 7.533315E-4
Training set -> Correct:2033 Incorrect:0   100.0%
Test set -> Correct:940 Incorrect:81   92%
```

Il·lustració 42: Data set: Chess

L'aprenentatge és lent, ja que és un *dataset* extens tant a nivell de nombre d'elements com de nombre d'atributs. Tot i això, s'arriba a una precisió del 92% en el *test set* i un 100% en el *training set*. Hi ha *overfitting* però tenint en compte el nombre d'elements del *dataset*, aquest 92% és força fiable amb un nivell de significació (α) reduït.

```
0 Total error 0.21233802
Training set -> Correct:1748 Incorrect:555   75.90099%
Test set -> Correct:578 Incorrect:577   50%

1 Total error 0.20516524
Training set -> Correct:2302 Incorrect:1   99.95658%
Test set -> Correct:578 Incorrect:577   50%

2 Total error 0.20136885
Training set -> Correct:2302 Incorrect:1   99.95658%
Test set -> Correct:578 Incorrect:577   50%

3 Total error 0.19770604
Training set -> Correct:2302 Incorrect:1   99.95658%
Test set -> Correct:578 Incorrect:577   50%

4 Total error 0.1939373
Training set -> Correct:2301 Incorrect:2   99.913155%
Test set -> Correct:578 Incorrect:577   50%

5 Total error 0.19002013
Training set -> Correct:2287 Incorrect:16   99.30525%
Test set -> Correct:578 Incorrect:577   50%

6 Total error 0.18593325
Training set -> Correct:2260 Incorrect:43   98.13287%
Test set -> Correct:573 Incorrect:582   49%

7 Total error 0.18166547
Training set -> Correct:2243 Incorrect:60   97.39471%
Test set -> Correct:569 Incorrect:586   49%

8 Total error 0.17721324
Training set -> Correct:2242 Incorrect:61   97.35128%
Test set -> Correct:669 Incorrect:486   57%

9 Total error 0.17258185
Training set -> Correct:2238 Incorrect:65   97.1776%
Test set -> Correct:855 Incorrect:300   74%

10 Total error 0.167783
Training set -> Correct:2238 Incorrect:65   97.1776%
Test set -> Correct:914 Incorrect:241   79%

11 Total error 0.16283536
Training set -> Correct:2239 Incorrect:64   97.221016%
Test set -> Correct:962 Incorrect:193   83%

12 Total error 0.15776432
Training set -> Correct:2239 Incorrect:64   97.221016%
Test set -> Correct:1058 Incorrect:97   91%

13 Total error 0.15259902
Training set -> Correct:2239 Incorrect:64   97.221016%
Test set -> Correct:1134 Incorrect:21   98%

14 Total error 0.14737304
Training set -> Correct:2239 Incorrect:64   97.221016%
Test set -> Correct:1138 Incorrect:17   98%

15 Total error 0.14212193
Training set -> Correct:2239 Incorrect:64   97.221016%
Test set -> Correct:1140 Incorrect:15   98%
```

Il·lustració 43: Data set: Occupancy

Com en el cas anterior, s'arriba a una precisió molt bona i sense *overfitting*. Tenint en compte que el *dataset* és molt nombrós, l'interval de confiança és molt bo.

En conclusió, crec que malgrat els resultats de *messidor features* són preocupants, els resultats dels altres 6 *datasets* són acceptables. Per tant, encara que, naturalment, tot és millorable, dono per vàlida la implementació de la xarxa neuronal.

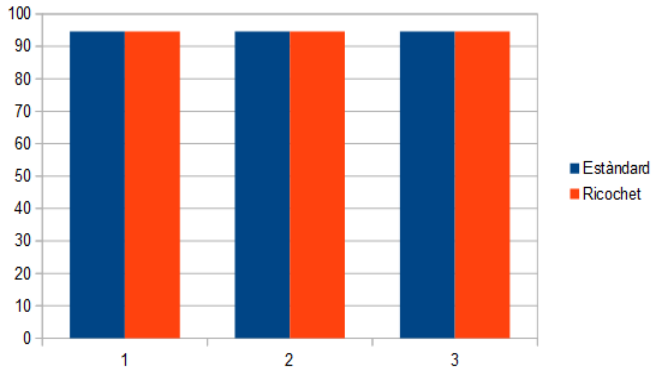
7.2.3 Anàlisi de la nova tècnica de xarxa neuronal proposada (inspirada en el *Tir à Ricochet*)

Com ja s'ha comentat al punt 3.3, aquesta tècnica està enfocada en la suposició de que si en l'última actualització de pesos la xarxa neuronal ha millorat considerablement i, a més a més, l'entrenament fet fins al moment permet superar una precisió determinada, començar a fer *gradient descent* des d'aquest punt pot permetre assolir el mínim global o un mínim local força òptim.
En concret, si això fos cert, *Ricochet* hauria d'incrementar, lleugerament, la desviació estàndard de les precisions del conjunt de les seves instàncies. D'aquesta manera, aquestes instàncies no tindrien tanta tendència a trobar-se dins de la mitja i això permetria que les millors instàncies d'aquest mètode fossin superiors a les millors instàncies del mètode estàndard. Naturalment, les pitjors també podrien ser inferiors a les del mètode estàndard.
Si aquesta hipòtesi fos certa, com que el sistema predictiu només utilitza les millors instàncies que s'han arribat a generar durant tota l'execució del programa servidor (és a dir, els casos excepcionalment bons), s'aconseguiria incrementar la precisió del sistema predictiu.

Per comprovar-ho, per cada *dataset* s'han entrenat 200 instàncies de cada un dels dos subtipus de xarxa neuronal. Evidentment, les dues xarxes neuronals tenien els mateixos paràmetres (learning rate, momentum...).

A continuació es mostren les gràfiques de les tres millors instàncies per cada *dataset*:

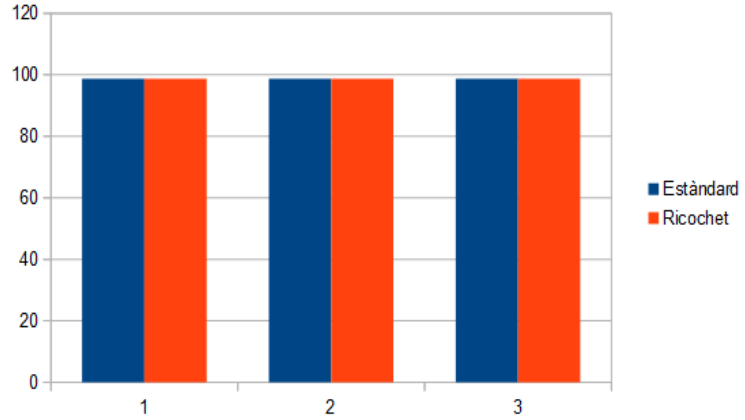
Banknote:



Il·lustració 44: Estàndard vs Ricochet (Banknote)

En aquest cas, les millors instàncies tenien la mateixa precisió (94.60784%). Per tant, no hi podem trobar cap diferència evident.

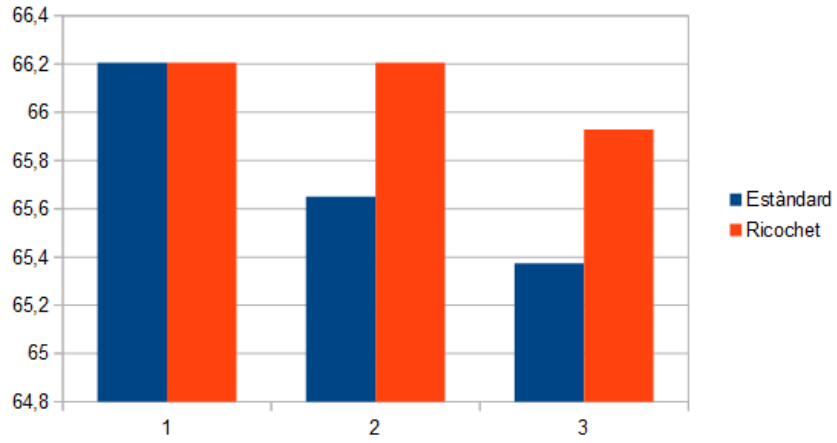
Breast cancer:



Il·lustració 45: Estàndard vs Ricochet (Breast cancer)

En aquest cas, totes les instàncies també obtenen la mateixa precisió (98.75776%).

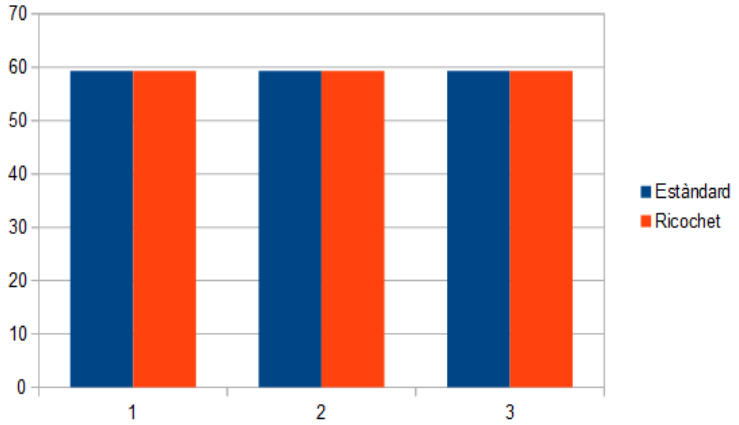
Messidor features:



Il·lustració 46: Estàndard vs Ricochet (Messidor features)

Les primeres instàncies de cada subtipus de xarxa neuronal obtenen la mateixa precisió(66.20499%). En les segones instàncies *Ricochet* és superior (66.20499% vs 65.65097%). En les terceres instàncies *Ricochet* torna a ser superior (65.92798% vs 65.37396%).

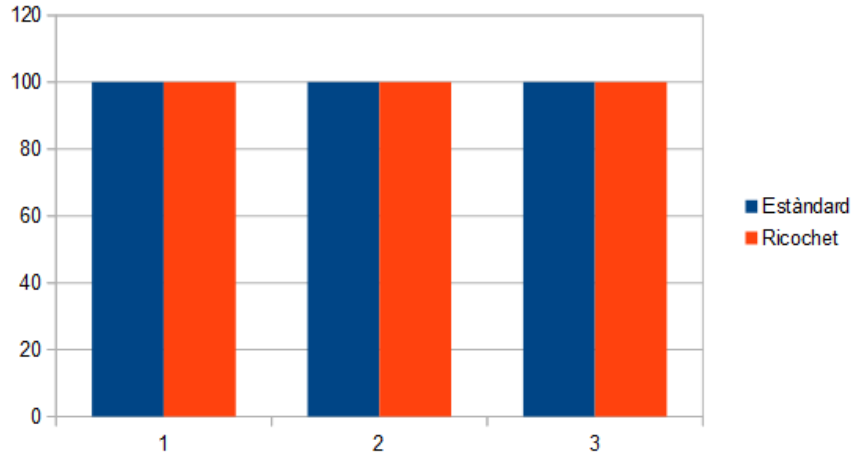
Random data:



Il·lustració 47: Estàndard vs Ricochet (Random data)

A *Random data* totes les millors instàncies han obtingut la mateixa precisió (59.34066%).

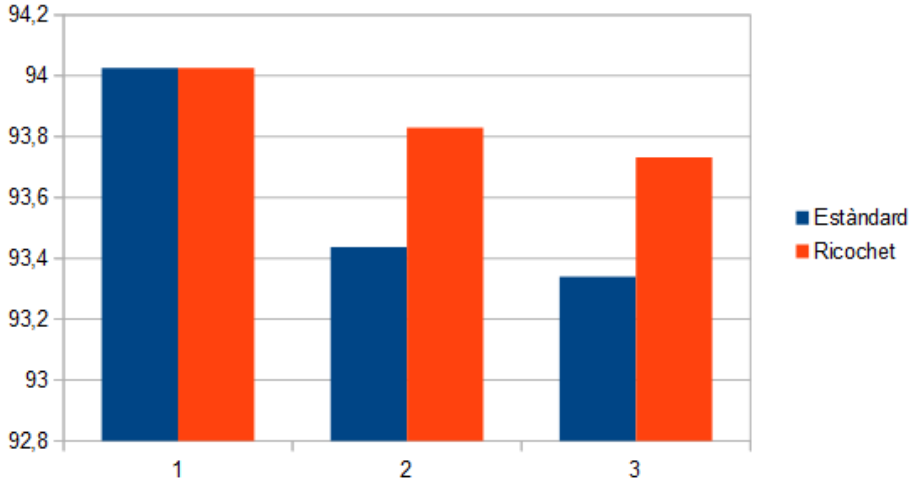
Haberman:



Il·lustració 48: Estàndard vs Ricochet (Haberman)

En el cas de Haberman, totes les instàncies (no només les tres millors) han obtingut una precisió del 100%.

Chess:

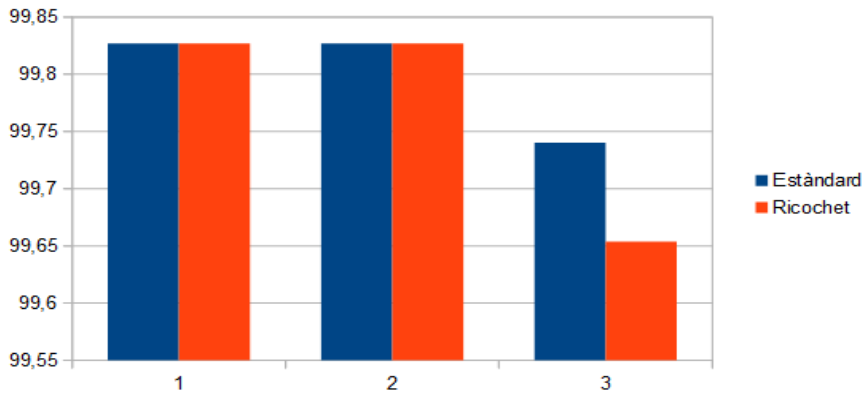


Il·lustració 49: Estàndard vs Ricochet (Chess)

En aquest cas, les primeres instàncies han estat iguals (94.02546%). La segona millor instància ha estat lleugerament millor amb *Ricochet* (93.829575%

vs 93.437805%). En la tercera ha tornat a estar lleugerament superior amb *Ricochet* (93.731636% vs 93.33987%).

Occupancy:



Il·lustració 50: Estàndard vs Ricochet (Occupancy)

En les primeres i segones posicions s'ha obtingut la mateixa precisió (99.82684%). En tercera posició, *Estàndard* ha estat lleugerament superior (99.74026% vs 99.65368%).

En conclusió, no hi ha cap evidència que demostri que *Ricochet* sigui superior al mètode estàndard en les millors instàncies. És cert, que en alguns *datasets* sembla tenir una lleugera superioritat però tenint en compte que s'han fet les proves en només set *datasets* diferents i que no s'han provat de crear milers d'instàncies per cada *dataset* (per falta de temps), aquesta diferència també podria haver estat producte del factor atzar.

D'altra banda, aquests *tests* s'han executat donant-li tant a *Ricochet* com a *Estàndard* els mateixos paràmetres (*learning rate*, *momentum*, nombre de capes...) i això no implica que els resultats dels dos subtipus de xarxa neuronal no pugessin tenir diferències més significatives amb uns altres paràmetres o, fins i tot, amb altres tipus de *datasets*.

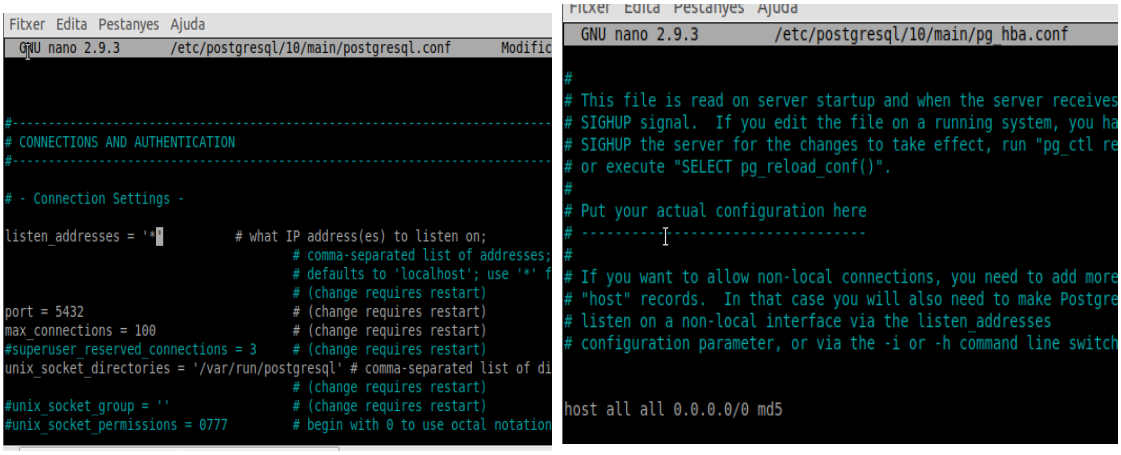
Així doncs, considero que no es pot donar cap resposta clara però, de moment, si hagués de dir quelcom, diria que si existeix alguna diferència, aquesta és poc significativa.

7.3 Instal·lació del servidor

- Com a SO del servidor, el més adequat és instal·lar alguna versió lleugera de Linux. En tot cas, com que els programes en *Java* són multiplataforma, també es pot instal·lar un Windows.

- El software ha estat desenvolupat en la versió 1.7 de *Java*. Per tant, el servidor ha de tenir instal·lat *Java* 1.7 o una versió superior.

- És necessari instal·lar la gestor de bases de dades PostgreSQL.



Il·lustració 51: Fitxers postgresql.conf i pg_hba.conf

Com es pot veure en la il·lustració superior, s'hauria de revisar la configuració predeterminada dels fitxers p*ostgresql.conf* i *pg_hba.conf.* Cal ajustar la configuració a les necessitats específiques que es tinguin.

- Per crear la base de dades i la taula s'haurà de fer el següent:
  1) *create database AttentionalControl* **(crear la base de dades)**
  2) *\c AttentionalControl* **(connectar-s'hi)**
  3) *create table Patterns(id serial primary key, aFaceMovement int, aDistanceFromScreen int, aActivityLevel int, appExecutionTime int, dayTime int, dayOfWeek int, isDistraction int);* **(crear la taula)**

- Configurar els permisos i accessos que es considerin necessaris.

- Executar el programa amb la comanda *java -jar attentionalControl_Server.jar*.

- Si els clients no aconsegueixen establir una connexió amb el servidor s'haurien de revisar els DNS/IPs, ports, tallafocs...

7.4 Vídeo sobre el funcionament del software

Accedir a https://www.youtube.com/watch?v=XbFIHHW0m0M per veure un exemple del funcionament del software.