



UNIVERSITAT_{DE}
BARCELONA

Treball final de grau

GRAU EN INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

MACHINE LEARNING
APLICAT A LA GENERACIÓ
DE MOODBOARDS

Autor: Ferrer Margarit, Marc

Director: David Buchaca

Realitzat a: Interiorvista

Barcelona, 25 de juny de 2018

Abstract

Moodboards are currently used to represent objects that are formed by the composition of others and seek to obtain a cohesive result among them and generate a new image. Machine learning allows us to train a machine so that it can generate values from the data that we have entered previously, which must be correct.

What we want to obtain is a combination between the two parties to achieve the generation of new images, with totally correct and cohesive objects and different from the images used to train the machine.

So the objective of this project is to create a machine that allows us, based on correct data, moodboards generated correctly, generate correct moodboards without the need for a person to generate them, create an automated system to replace a task slow and mechanical that a series of people realize.

Apart from creating this system, it will also ease your efficiency with respect to the non-automated system or the generation of data in a random way, so we can establish whether the system created is sufficiently effective, fast and accurate when it comes to performing, in this case, new moodboards.

Finally, these tests will be carried out by supervising expert people in order to improve the evaluation.

Resum

Actualment els moodboards s'utilitzen per representar objectes que estan formats per la composició d'altres i pretén obtenir un resultat de cohesió entre ells i generar una imatge nova. El machine learning ens permet entrenar una màquina per tal que aquesta sigui capaç de generar els valors amb les dades que li hem entrat prèviament, les quals han de ser dades correctes.

El que volem obtenir és una combinació entre les dues parts per aconseguir la generació de noves imatges, amb objectes totalment correctes i cohesionats i diferents de les imatges utilitzades per entrenar la màquina.

Així doncs l'objectiu d'aquest projecte és crear una màquina que ens permeti, a partir de dades correctes, moodboards generats correctament, generar moodboards correctes sense la necessitat que una persona els hagi de generar, crear un sistema automatitzat per a reemplaçar una tasca lenta i mecànica que realitzen una de persones.

A part de crear aquest sistema, també s'analitzarà la seva eficiència, respecte al sistema no automatitzat o la generació de dades d'una forma aleatòria, així podrem establir si el sistema creat és prou eficaç, ràpid i precís a l'hora de realitzar, en aquest cas, nous moodboards.

Finalment, aquestes proves es realitzaran mitjançant la supervisió de persones expertes per a tal de millorar l'avaluació.

Agraïments

Vull agrair a ...

Índex

1	Introducció i objectius	1
2	Moodboards	4
2.1	Disseny dels moodboards	4
2.2	Desenvolupament de l'eina	8
2.2.1	Funcionalitats bàsiques	8
2.2.2	Disseny i funcionament de l'aplicació	10
2.2.3	Procés de generació d'un moodboard	16
3	Desenvolupament del Machine Learning	18
3.1	Introducció i antecedents	18
3.2	Tipus d'algoritmes	22
3.2.1	Aprenentatge supervisat	23
3.2.2	Aprenentatge no supervisat	24
3.2.3	Aprenentatge per reforç	25
3.3	Plantejament de l'algoritme	27
4	Màquina de Boltzmann(BM) i Màquina de Boltzmann Restrictiva (RBM)	29
4.1	Màquina de Boltzmann	29
4.1.1	Entrenament Màquina de Boltzmann	30
4.2	Màquina de Boltzmann Restrictiva	32
4.2.1	Entrenament Màquina de Boltzmann Restrictiva	33
4.2.2	Avaluació d'una RBM	36
4.2.3	Dificultats i solucions convencionals	40
4.3	Algorisme seleccionat a implementar en codi	41
5	Desenvolupament de l'algoritme aplicat al problema	44
5.1	Dades obtingudes de l'eina	44
5.1.1	Format de les dades	44
5.1.2	Dades obtingudes del projecte	49
5.2	Dades d'entrenament	49
5.2.1	Format de les dades	49
5.2.2	Creació del codificador i dades obtingudes	51

5.3	Creació i entrenament del RBM	52
5.3.1	Creació del RBM a partir de les dades	53
5.3.2	Entrenament del RBM	54
5.4	Dades de sortida	55
5.4.1	Dades obtingudes del RBM	55
5.4.2	Creació de moodboards a partir de les dades	57
6	Resultats i conclusions i línies de continuació	57

1 Introducció i objectius

Avui en dia, tot i que depenem de les màquines i dels programes per a realitzar tasques que poden ser pesades i lentes encara en queden moltes que es realitzen a mà i que en algun punt hauran de ser automatitzades per tal que es puguin realitzar més ràpidament i d'una forma més precisa.

Així doncs aquest projecte el que es vol aconseguir és que una tasca que actualment costa temps, dedicació i concentració es transformi amb una tasca mecànica, senzilla de realitzar, ràpida i amb bons resultats. Que la persona que l'hagi de realitzar no es trobi amb problemes que no pot solucionar o en dedicar-hi masses hores. Aquesta transformació que volem aconseguir, s'ha de realitzar partir de la tasca original, de la informació que es necessita i en què es basa.

Aquesta tasca consisteix en la generació de moodboards. La pregunta és: **Què és un moodboard?**

Un moodboard és una combinació d'imatges relacionades entre elles que juntes formen una sola imatge. Normalment es creen aquestes combinacions per generar imatges de paisatges, d'aplicacions, decoració i creació d'habitacions.

La tasca que volem substituir, més ben dit, millorar, evolucionar, és la tasca de la generació de moodboards. Actualment per generar moodboards es necessita fer una cerca de totes les imatges que es desitgen col·locar. Un cop triades s'han de col·locar de forma que formin un sol conjunt i anar-les editant per tal de generar una sola imatge.

Aquest procés pot ser molt lent i tediós, ja que buscar totes les imatges necessàries pot necessitar igual que col·locar-les correctament i editar-les per tal que totes s'ajuntin i formin una sola imatge.

Aquest procés se substituirà per un altre de més ràpid conservant alguns aspectes tot i que el procediment canviarà per tal que tot sigui més mecànic i més ràpid.

Com hem dit anteriorment poden haver-hi moodboards de molts tipus, però per aquest projecte ens centrarem amb els moodboards d'interiorisme, és a dir, els que defineixen una estança d'una casa, un dormitori, un bany, una cuina, etc. A part de definir una estança també definiran un estil, cada moodboard pertanyerà a un estil determinat. Així, cada moodboard serà una combinació de mobles i objectes que junts representaran l'estil d'una zona de la casa.

Un cop explicat tota la base de la qual partirem l'objectiu que ens proposem és crear un sistema automatitzat que ens permeti la generació automàtica de moodboards, sense la necessitat d'editar imatges, facilitant la cerca dels mobles i objectes necessaris que es necessiten per a cada zona i generant imatges d'una forma més ràpida i eficient.

Per tal de poder crear aquest sistema utilitzarem tècniques de machine learning per tal de crear una màquina capaç d'aprendre i de generar a partir de les dades

que ha après. D'aquesta forma podem estructurar el projecte amb dues parts essencials, una la generació de moodboards mitjançant eines, ja creades i que explicarem és endavant, per tal d'entrenar la màquina i la segona la creació d'una màquina capaç de llegir els moodboards creats anteriorment, aprendre com estan generats, és a dir, com es distribueixen els diferents elements a cada un dels i generar-ne de nous que siguin correctes.

Com bé he mencionat, farem servir tècniques de machine learning, és a dir algorismes d'aprenentatge automàtic. **Però que entenem per aprenentatge automàtic?**

L'aprenentatge automàtic és un camp de la intel·ligència artificial que està dedicat al disseny, l'anàlisi i el desenvolupament d'algorismes i tècniques que permeten que les màquines evolucionin. Es tracta de crear programes capaços de generalitzar comportaments a partir del reconeixement de patrons o classificació, d'aquesta manera nosaltres el que volem fer és una màquina que sigui capaç de classificar per després poder generar a partir de les dades que ha classificat.

El machine learning ens ha de permetre donada una sèrie d'imatges generades correctament importar-les a la màquina i que aquesta aprengui com estan formades, l'ordre que segueix la distribució de cada imatge dins del moodboard. També ha de capaç de generar noves sèries d'imatges que han de ser correctes.

Aquest tipus de màquines necessiten moltes dades per tal necessitem generar moltes dades per tal que siguem capaç d'entrenar la màquina correctament.

Finalment, tot i que l'objectiu és crear un sistema automatitzat per a millorar la realització d'una tasca, l'objectiu també és analitzar com és de millor aquest sistema, és a dir, com millora respecte de l'altre sistema. Per tal de saber-ho, al final de tot el projecte avaluarem el sistema mitjançant l'ajuda humana, en aquest cas, persones expertes que han generat molts de moodboards i que també han generat els correctes per entrenar la màquina i d'aquesta forma podrem validar les qualitats del sistema mitjançant diverses proves que s'explicaran més endavant.

Estructura de la Memòria

En primer lloc ens centrarem com generem els moodboards, amb quines eines i alguns exemples per tal de crear-los. Aquests moodboards seran els que posteriorment serviran per entrenar la màquina. També explicarem el funcionament de les eines i les diferències amb el mètode "tradicional".

Seguidament explicarem quin tipus de màquina farem, és a dir, el tipus d'algorisme d'aprenentatge automàtic que aplicarem, els motius pels quals hem decidit escollir aquest algorisme i quines són les seves propietats bàsiques per aquest sistema.

Després analitzarem com són les dades d'entrada, com a partir del moodboards generats obtenim, com transformem aquestes imatges per tal d'obtenir dades vàlides per a entrenar la màquina. També definirem com han de ser aquestes dades i com han de ser les dades que volem obtenir de sortida i com les transformarem en imatges de nou.

I finalment analitzarem els resultats obtinguts i els avaluarem mitjançant diferents proves que realitzarem per comprovar la seva eficiència i els resultats envers els moodboards creats prèviament per experts.

2 Moodboards

Abans de començar a explicar les dades, el funcionament de l'algorisme i els resultats ens hem de centrar en la primera part del projecte i una de les més importants, la generació de dades.

Com hem comentat a la introducció les nostres dades són moodboards que defineixen diferents estances d'un habitatge d'un estil concret.

A continuació explicarem com són els moodboards, és a dir, com estan formats els moodboards actuals i com seran els nostres moodboards que generarem amb la nostra eina. També explicarem com està dissenyada la nostra eina, amb quines eines s'ha realitzat i com es creen moodboards seguin l'estàndard que hem decidit.

2.1 Disseny dels moodboards

Com bé ja hem explicat moodboard és una combinació d'imatges relacionades entre elles que juntes formen una sola imatge. Aquesta imatge que formen és una imatge ben generada, és a dir, que hi ha cohesió amb tots els elements de forma que sembli que sigui una fotografia mentre que es tracta una composició de diferents objectes editats manualment mitjançant eines d'edició d'imatge per formar una sola imatge coherent i compacta, la qual mostrarà una habitació d'una casa.

Realitzar moodboards amb aquesta tècnica té avantatges i inconvenients. Un dels problemes més gran que suposa és el temps. Generar un moodboard utilitzant aquest tipus de composició pot suposar una gran quantitat de temps, ja que en primer lloc necessitem buscar aquelles imatges que necessitem, és a dir, dintre d'un rang enorme el qual està format per milers de mobles i objectes escollir els adients segons l'estil i l'estança desitjada.

Un cop seleccionats els elements necessaris cal editar-los un per un per tal de poder combinar-los entre ells i que sembli una fotografia. Tot plegat suposa un esforç i una gran quantitat de temps per a generar un moodboard.

Per altra banda uns dels millors aspectes és la cohesió que s'aconsegueix i la impressió que provoca, ja que sembla una fotografia. Això el que ens permet imaginar-nos molt millor com seria aquella habitació. Ens proporciona una visió molt més real que un conjunt d'imatges ajuntades aleatòriament.

La qüestió que ens plantegem és si la gran quantitat de temps invertida per a generar una sola imatge compensa el resultat obtingut. Per tal de respondre aquest pregunta, hem consultat a experts que una de les seves principals tasques és la generació de moodboards.

Segons ells invertir una gran quantitat de temps per a generar una imatge compensa, ja que el resultat que obtenen és una reflexió de com seria veure en directe, com fer una fotografia de l'habitació per tal de poder-la mostrar. També afirmen

que a l'hora de mostrar habitacions a clients o empreses és una de les formes de captar més la seva atenció perquè en aquestes imatges es reflecteix la realitat del futur, és a dir, es reflecteix el que un dia podrà ser aquesta habitació en una casa qualsevol.

També han apuntat que en realitzar aquestes composicions els hi permeten crear diferents habitacions, diferents formes en ambients totalment diferents i poder afegir una història a darrere de cada composició que permeti a la persona que està veient aquella imatge entendre, endinsar-se dintre d'aquell ambient fictici fins a fer-lo real en la seva ment.

Així doncs, la generació de moodboards utilitzat aquesta tècnica genera molt bons resultats però té com a conseqüència dedicar-hi molt de temps per tal d'obtenir un bon resultat i que captivi aquelles persones a les quals va dirigit.

A partir d'aquest punt el que ens hem de plantejar si utilitzar l'esquema de la tècnica anterior és adient pel nostre projecte, és a dir, si ens podem permetre generar una gran quantitat d'imatges dedicant-hi molt de temps per després utilitzar-les per entrenar una màquina.

La resposta és simple, aquest tipus de tècnica no serveix per a aquest cas, ja que necessitem generar, amb un període curt de temps, una gran quantitat de moodboards. Per tal de solucionar s'ha de plantejar una solució que ens permeti organitzar en un moodboard els elements d'una forma ràpida i que més o menys formin una imatge cohesionada.

Partint d'aquesta premissa es van agafar diferents estudis realitzats sobre moodboards anteriorment generats de cada estança i es van treure una sèrie d'especificacions que cada moodboard hauria de complir, així podríem definir un model per dissenyar una plantilla per a generar moodboards d'una forma més mecànica i més ràpida.

Així doncs una de les especificacions més importants que es va arribar és que cada estança que es volia generar tenia una sèrie d'elements fixes, és a dir, una sèrie de mobles i objectes que sempre hi haurien de ser. D'aquesta forma es va establir un nombre d'objectes per a cada estança i quins havien de ser. Això solucionava una part del problema que era saber quants objectes podíem col·locar a cada moodboard. L'altre problema que va sorgir era com combinar imatges per tal d'aconseguir una imatge cohesionada semblant a les imatges que s'obtenien amb l'altra tècnica. En aquest cas el rang d'objectes disponible era bastant gran, es tractava d'un conjunt d'uns 3000 objectes, i les imatges per a cada objecte eren *thumbs*, imatges amb un fons blanc i l'objecte en petit centrat al mig del fons blanc. La solució de treure el fons blanc editant la imatge no era viable, ja que es tardaria molt i no era l'objectiu que volíem aconseguir.

Partint de tot això, es va arribar a trobar una solució que complia amb les especificacions i més o menys s'obtenia un resultat coherent. Partint de les solucions anteriors es va establir una graella per a cada estança. Aquesta graella contindria a cada casella un tipus d'objecte, la qual més endavant contindria la imatge, el *thumb*, de l'objecte que fos d'aquell tipus. D'aquesta forma s'aconsegueix una certa

cohesió, ja que totes les imatges en tenir el mateix format es formaria una imatge de fons blanc amb petites imatges de cada objecte. Tot i això, sempre hi ha la condició que a una cel·la no hi hagi cap element i quedi de color blanc.

Aquesta solució serviria per a totes les estances disponibles d'un habitatge però si ens centrem amb el projecte, només analitzarem una estança, el dormitori per tant en aquest cas tindrem una sola graella d'imatges que corresponent als objectes que es troben en l'estança. Així doncs es va establir que, en aquest cas, la graella del dormitori tindria 48 elements els quals correspondrien als diferents mobles i objectes que es poden col·locar.

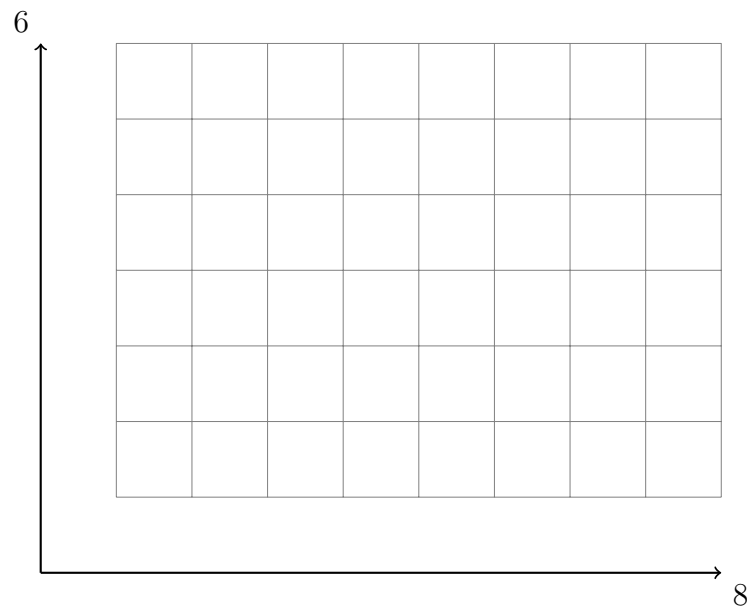


Figura 1: Exemple de la graella de l'estança dormitori.

Cada casella d'aquesta graella tindrà internament, és a dir, d'una forma que no es veurà visualment uns valors que indicaran quins objectes poden anar aquella casella. D'altra banda cada casella també contindrà una o més d'una paraula que descriurà els valors anteriors, d'aquesta forma la persona que hagi d'emplenar la graella amb objectes ha de saber quins poden i quins no. A continuació es mostra la graella final que s'utilitzarà per a la generació de moodboards.

Suelo	Pared A	Pared B	Resto Paredes	Pilar o Columna	Murete	Cornisa	Molduras Frisos
Arrimadero	Zócalos	Cantonera	Escaleras	Techo	Viga o Estructura Compuesta Vigas	Molduras	Rosetón
Puerta	Ventana o Balconera	Otros	Separadores Espaciales	Calefacción	Climatización Ventilador Techo	Chimenea	Interruptores Enchufes
Iluminación Técnica	Cama	Estantería encima cama	Mesita Noche	Lámpara Techo	Armario	Banco/Baúl	Colgadores
Cómoda	Espejode Pie	Galán	Butaca	Lámpara de pie	Mesita Auxiliar	Tocador	Espejo
Alfombra	Cortinas	Textil Cama Principal	Textil Cama Secundario	Textil Cama Secundario 1	Textil Cama Secundario 2	Escritorio	Estantes de Pared

Figura 2: Graella final de l'estança dormitori.

A partir d'aquest punt ja tenim una solució que ens permet generar moodboards de forma ràpida encara que el resultat no sigui el mateix que utilitzant la primera tècnica descrita. Tot i això, en aquest cas preferim velocitat a l'hora de generar que més cohesió, ja que tampoc tenim forma que un cop la màquina generi graelles puguem editar aquestes imatges per tal que s'assemblin al model de la primera tècnica. A partir d'ara necessitem una forma de poder col·locar els objectes disponibles a la graella d'una forma senzilla i eficient.

2.2 Desenvolupament de l'eina

Per tal de poder aplicar el mètode anteriorment explicat necessitem alguna eina que ens permeti accedir a tots els objectes disponibles d'una forma fàcil per tal de poder crear moodboards seguin la graella prèviament definida. Així doncs necessitem crear una eina per realitzar aquesta tasca.

Aquesta eina que volem desenvolupar ha de ser fàcil de fer servir, intuïtiva i ràpida. Partint d'això s'ha de decidir quines tecnologies utilitzar per desenvolupar-la. En aquest cas, com que ja s'havien creat altres eines per altres coses es va decidir aplicar les mateixes tecnologies que en aquest cas són utilitzar Unity i el llenguatge de programació C sharp.

Aquestes dues tecnologies combinades ens permetran desenvolupar una eina capaç de generar moodboards d'una forma ràpida i eficient.

En primer lloc tenim la part de Unity que ens permetrà crear tot el disseny de la interfície i les interaccions necessàries perquè l'usuari pugui col·locar els objectes i així generar moodboards. També ens permetrà visualitzar tots els objectes i tota la informació que necessitem per a la generació.

I en segon lloc la part de C Sharp que controlarà tot a la lògica que hi ha darrere de la interfície. Però el paper més important és la connexió amb la base de dades. Els objectes que tenim disponibles, la seva informació, la graella creada i tots els moodboards generats es guarden en una base de dades que es troba en un servidor. Així doncs per tal de poder accedir a aquesta informació, s'han realitzat crides constants per obtenir-la i això se n'encarrega la part de la lògica de l'aplicació.

Pensant en un esquema model-vista-controlador (MVC), la part de model i vista correspondrien a Unity mentre que tota la part del controlador seria gestionada per fitxers, scripts, amb C Sharp els quals s'han de comunicar amb la base de dades per tal de proporcionar la informació necessària per a mostrar tots els objectes, la graella i altres funcions que ja explicarem.

2.2.1 Funcionalitats bàsiques

Com hem comentat aquesta eina ha de tenir unes funcionalitats bàsiques i essencials per a obtenir un bon resultat. Aquestes funcionalitats són les següents:

- Categoritzar graella predefinida. Com hem dit anteriorment, la graella la qual estableix quin objecte pot anar a cada casella, està organitzada de forma que cada una d'elles conté una sèrie d'identificadors els quals cada un és una subcategoria, un tipus d'objecte de la base de dades. Aquests identificadors no es veuen a la graella, ja que no tenen importància visualment però el que sí que es veu és un resum d'aquestes categories que especifica de forma més general la casella.

Aquests noms i aquests valors s'han de poder modificar per si en algun moment s'ha classificat una casella incorrectament. Així doncs, l'eina ens ha de permetre modificar cada una de les caselles, tant la part visual com els valors que no es mostren que en permeten filtrar objectes.

- Carregar objectes. Com hem mencionat repetidament necessitem objectes per a col·locar. Aquests objectes es troben a una base de dades i el que l'eina ha de fer és obtenir la informació d'aquests i mostrar-la en forma d'imatges per tal que l'usuari pugui col·locar-los correctament a la graella. Així doncs ha d'implementar un sistema de crides a servidor per obtenir les dades i transformar-les per tal que es mostrin correctament i s'emmagatzemi d'alguna forma la informació que posteriorment ens serà útil.
- Filtrar objectes. El rang disponible d'objectes és molt ampli i el que no volem és que l'usuari es passi molt de temps buscant un objecte per a col·locar-lo a una casella. Per tant, el que es necessita és implementar un sistema de filtratge ràpid i fàcil de fer servir.
Per tal d'implementar aquest sistema utilitzarem els valors que amaguen les caselles que són els que ens indiquen aquells objectes que poden anar en ella. Així doncs cada cop que es vulgui col·locar un objecte a una casella només es mostraran aquells objectes que la base de dades retornarà aplicant el filtre amb els valors de la casella. D'aquesta forma aconseguim que l'usuari no s'hagi de trencar al cap buscant objectes i sempre col·loqui objectes vàlids a cada casella.
A part també s'han de poder filtrar aquests objectes, els que ja han estat filtrats, segons cada valor de la casella, és a dir, ha d'haver-hi l'opció de mostrar tots els objectes disponibles de la casella o només els d'un tipus determinat de la casella, només utilitzant un valor de la casella.
- Col·locar objectes. L'aplicació ha de permetre a l'usuari poder col·locar objectes a la graella. Així doncs, la mecànica és que pugui arrossegar la imatge de l'objecte a la graella. També ha de permetre que un cop col·locat l'objecte aquest es pugui treure i/o eliminar.
- Guardar graella. Com que el principal objectiu de l'aplicació és generar moodboards per després entrenar una màquina amb els moodboards generats, l'aplicació ens ha de permetre guardar d'alguna forma el moodboard generat amb tota la distribució d'objectes creada per l'usuari. El format d'aquestes dades que s'exportaran ha de ser l'adient per tal que la màquina pugui llegir-les.
- Crear PDF. Tot i que només necessitem exportar les dades per la màquina també s'ha demanat una forma de visualitzar els moodboards generats i la

informació que contenen. Així doncs, l'aplicació ha de permetre generar d'algun forma un fitxer on figuri tota la informació. La solució és generar un PDF. Aquest fitxer, en primer lloc contindrà una captura de la graella i els seus elements i a continuació a cada pàgina contindrà l'objecte amb tota la seva informació, així obtenim un fitxer fàcil de compartir i el qual conté tota la informació del moodboard generat.

2.2.2 Disseny i funcionament de l'aplicació

A continuació explicarem el disseny de l'aplicació, és a dir, com és la interfície gràfica, per a què serveix cada part que se'ns mostra. Per explicar-ho millor partirem d'una imatge de l'eina, ja que ens servirà per anar assenyalant millor les parts i indicant que fa cada part. Més endavant explicarem quin procés s'ha de seguir per a crear un moodboard. A continuació es mostra la imatge de l'eina.

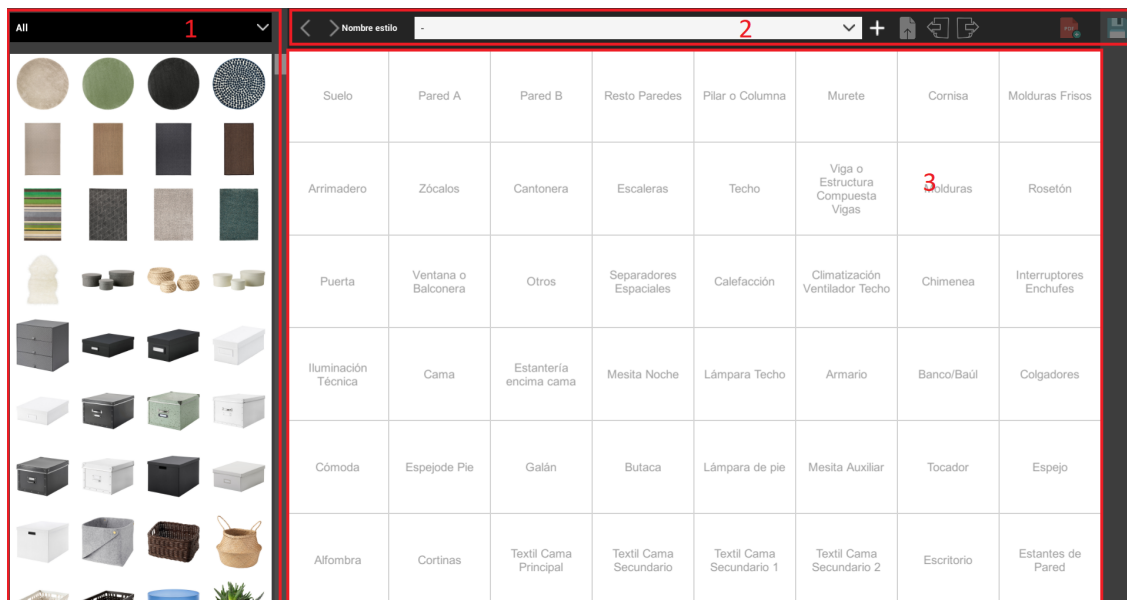


Figura 3: Captura de l'eina.

Podem observar que es tracta d'una interfície senzilla i ben organitzada. A l'hora de dissenyar l'eina es van tenir en compte totes les normes de disseny i d'usabilitat per tal que la distribució de la informació i dels botons fos la més adient per tal que l'usuari pogués treballar d'una forma ràpida i mecànica.

Com podem observar l'eina es divideix en tres parts essencials els quals cadascuna correspon alguna de les necessitats més importants que s'han mencionat anteriorment. A continuació analitzarem les 3 zones per separat explicant el seu objectiu i el funcionament dintre de l'eina.

Primer de tot començarem per la zona 3, la graella. Aquesta graella és la mateixa que s'ha mostrat prèviament i s'ha explicat com estava formada. Podem observar que aquesta graella és prou gran perquè en col·locar un objecte aquest no sigui ni molt petit ni molt gros sinó la mida justa per tal que es pugui veure de quin objecte es tracti. Cada casella de la graella és clicable, és a dir, podem fer clic sobre la casella i aquesta quedarà ressaltada de color verd. El que ens permet és que en seleccionar una casella els objectes es filtrin segons les categories de la casella.

Una altra funcionalitat de les caselles és que si en comptes de fer clic amb el botó esquerre del ratolí el realitzem amb el botó dret, eliminarem l'objecte que es troba a la casella. Aquest funcionament permet a l'usuari una ràpida manipulació de les graelles per tal de poder trobar els objectes adients per a cada una i la facilitat d'eliminar i/o modificar sense la necessitat de realitzar més moviments dels necessaris.

Tal com hem mencionat abans la graella es pot modificar, es poden canviar les categories. Per tal de modificar-la necessitem entrar en el mode edició, en aquest mode, en fer clic amb el botó esquerre ens apareixerà una finestra la qual haurem d'indicar el nom que volem que es mostri i escollir les subcategories que desitgem. Per a escollir una subcategoria primer s'ha de seleccionar el tipus, després la categoria i després la subcategoria.

Per entrar i sortir del mode edició l'usuari només ha de prémer la tecla "F1". Un cop acabada l'edició de les graelles, es guardarà la configuració i es mantindrà fins que no torni a ser modificada de nou per l'usuari. A continuació es mostra una imatge de la finestra que apareix quan ens trobem en el mode edició.



Forma de la finestra del mode edició:

- Name:** Suelo
- Type:** [Menú desplegable]
- Category:** [Menú desplegable]
- Subcategory:** [Menú desplegable] +
- Categories list:**
 - Elementos Estructurales/Pavimento/Compuesto
 - Elementos Estructurales/Pavimento/Continuo
- Buttons:** Eliminar Selección, Cancelar, Guardar

Figura 4: Captura de la finestra del mode edició.

La graella es pot emplenar del tot o deixar caselles buides. No hi ha cap restricció sobre els elements mínims necessaris per a generar un moodboard.

Un cop ja hem explicat l'element principal de l'eina a continuació ens centrarem amb la zona 1, la zona dels objectes. Com hem dit, necessitem visualitzar els diferents objectes que es troben a la base de dades, així doncs necessitem una zona en la qual podem veure i filtrar els diferents objectes.

Només d'iniciar l'aplicació els objectes que es mostraran són tots els objectes disponibles que tenim, ja que no s'ha seleccionat cap cel·la en concret. El rang d'objectes que tenim és molt gran i cada objecte conté una imatge que l'aplicació ha de descarregar. Això ens presenta un problema important que haurem de gestionar d'una forma especial.

El problema que se'ns presenta és que no podem carregar tots els objectes amb les seves corresponents imatges, ja que aquesta càrrega trigaria molt i podria fer lent la tasca de generació de moodboards.

Per tal de solucionar aquest problema crearem un sistema de càrrega semblant als buscadors de les pàgines web, les imatges s'aniran carregant a mesura que es mogui la barra de desplaçament o s'utilitzi la rodeta del ratolí per baixar la barra. Aquest sistema ens permet una càrrega ràpida i fluida de les imatges sense la necessitat de carregar totes les imatges. Així doncs, en iniciar l'aplicació es mostraran un nombre finit d'imatges, aquest número dependrà de la mida de la finestra, ja que com més gran més imatges es mostraran al principi, i a mesura que es vagi desplaçant cap avall s'aniran carregant les següents imatges.

També s'ha implementat que un cop la imatge s'ha descarregat no es torni a descarregar, es vol imitar el sistema de memòria caché però en aquest cas les imatges es guarden internament en l'aplicació el que ens permet que un cop carregada una imatge ja no es tornarà a descarregar, conseqüentment, si s'han carregat tots els objectes amb les seves imatges, ja no es descarregaran més imatges i el filtratge d'objectes serà més ràpid. Aquesta informació desapareixerà un cop l'aplicació es tanqui.

El filtratge d'objectes es realitzarà quan se seleccioni una cel·la. Les diferents opcions de filtratge es mostraran en el desplegable que es troba a la part superior. En tots els casos hi haurà una opció que serà el filtratge de totes les categories, és a dir, de tots els objectes que poden anar a la cel·la i les altres opcions seran per filtrar per categoria, per escurçar el rang d'objectes a col·locar. El sistema de càrrega d'imatges del filtratge és el mateix que s'ha explicat anteriorment, ja que si ja tenim imatges descarregades només les hem de mostrar i en cas contrari, les descarreguem i les guardem per posteriors usos.

Tot el filtratge d'objectes s'obté mitjançant crides a la base de dades aplicant els filtres que té la cel·la. Un altre sistema seria, filtrar a partir de la llista d'objectes descarregats inicialment, però això pot suposar que en algun moment s'afegeixi o es modifiqui un objecte mentre s'està utilitzant l'aplicació i aquest no apareixeria

en el filtratge per això utilitzem les crides a base de dades per obtenir la informació actualitzada. A continuació es mostra un exemple de com funciona el filtre un cop se selecciona una cel·la. Podem veure com la primera opció és la general, la que engloba les dues categories i després podem filtrar per cada una d'elles.

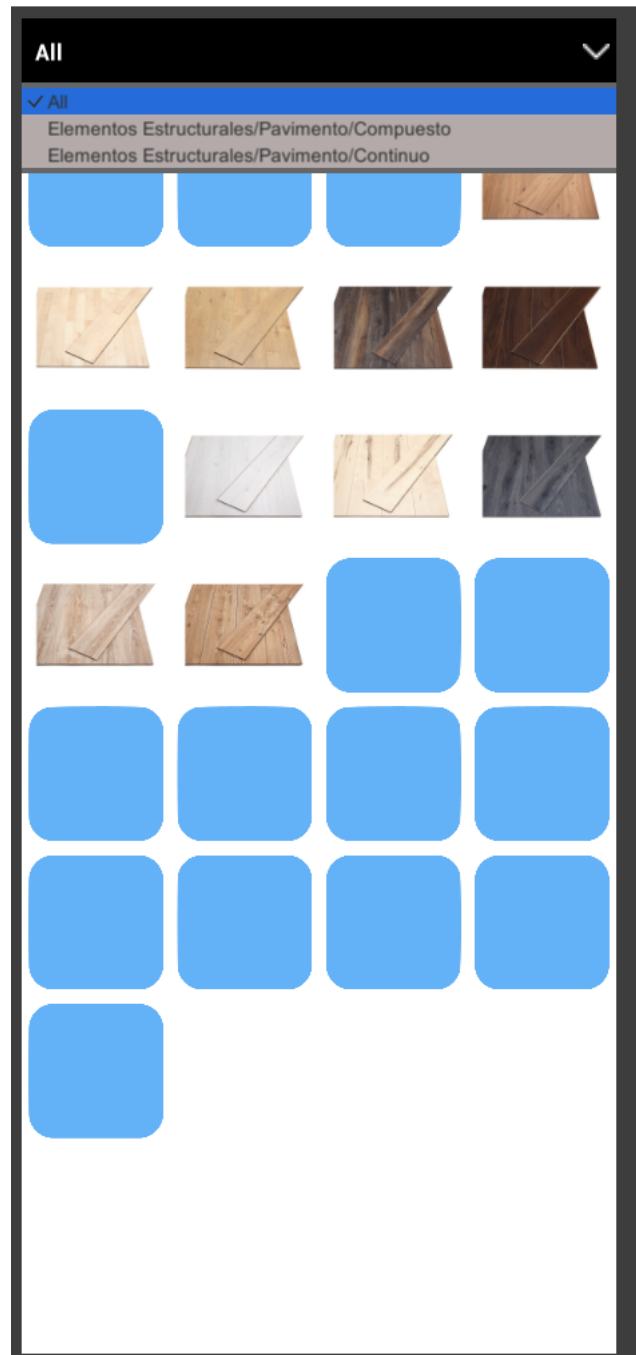


Figura 5: Captura del filtratge després de seleccionar una cel·la.

Un cop ja explicades dues de les parts de l'eina, finalment explicarem la part superior la qual es compon dels diferents botons de l'aplicació per a realitzar diferents

accions. A continuació es mostra una imatge de la barra superior de botons. A partir d'aquestes imatges explicarem la funcionalitat de cada botó a dins l'eina i les diferents parts que la formen.



Figura 6: Barra superior de l'aplicació.

Seguint un ordre d'esquerra a dreta tenim en primer lloc dos botons que són els botons de desfer i refer. L'aplicació té implementat un sistema de gestió d'accions, és a dir, cada cop que es realitza una acció aquesta queda registrada a dins l'aplicació. Això el que ens permet és tenir un control sobre elles i en qualsevol moment saber quines accions s'han realitzat.

A partir d'aquest sistema de gestió hem implementat un sistema de desfer i refer que desfà o torna a fer les accions que s'han realitzat. Això permet que si en un punt ens hem equivocat i volem tornar endarrere no tinguem problema o si s'ha guardat però ens hem equivocat puguem tornar a un punt anterior per seguir endavant.

Aquest sistema guarda totes les accions que es realitza l'usuari amb el ratolí. Aquestes dues icones s'aniran il·luminant a mesura que sigui possible realitzar les accions, és a dir, si no es pot retrocedir o no es pot refer, no es podrà interactuar amb el botó, així s'evita que es realitzin accions no permeses.

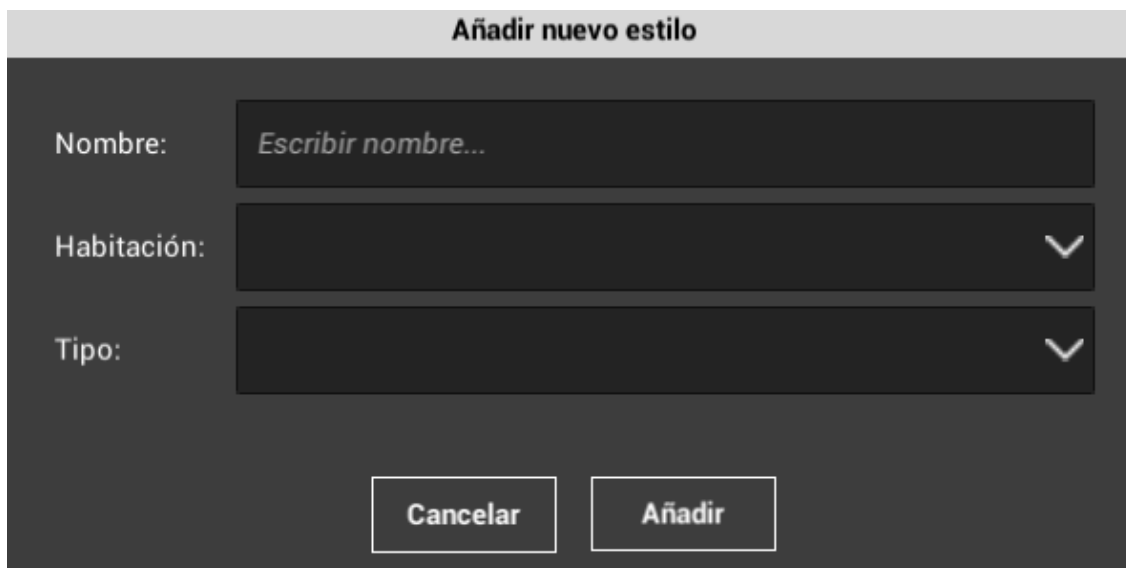
Cada cop que es reinicia l'aplicació és reseteja el sistema i es netegen les accions, quan es guarda un moodboard definitiu també es netegen tot i que es pot seguir realitzant accions que seran afegides al sistema com a noves accions, és a dir, com si es tornés a crear el sistema de gestió. Això permet indicar a l'usuari que ha guardat el que ha realitzat i que a partir d'aquest punt tot el que realitzi serà un nou moodboard.

Després dels botons de desfer i refer tenim el nom del moodboard. Aquí és on ens apareixerà el nom del moodboard que estem generant. En el desplegable també ens apareixerà el nom dels moodboards anteriors que hem creat. Seleccionar un nom d'un moodboard que ja hem creat ens permet crear un nou moodboard amb el mateix o carregar-ne un de nou per a visualitzar-lo.

Per tal de crear un nou moodboard hem de prémer el botó amb el símbol més. En prémer aquest botó ens apareixerà una nova finestra en el qual ens demanarà que entrem unes dades. En primer lloc en demanarà el nom de l'estil, que potser qualsevol nom que l'usuari desitgi. Després utilitzant un desplegable l'usuari ha de seleccionar a quina habitació correspon l'estil i finalment haurà de seleccionar el tipus que correspon.

El tipus es refereix a la distribució d'attrezzo, és a dir, la distribució d'objectes

que es podrien posar a cada moble, a sobre, a sota, els complements per a cada moble que es podrien col·locar per acabar de completar el moodboard. Hi ha sis tipus respecte si estan ordenats o poc i amb quina quantitat. Quan s'afegeix un nou moodboard aquest s'afegirà automàticament al desplegable. A continuació es mostra la pantalla per afegir un nou moodboard.



Añadir nuevo estilo

Nombre:

Habitación:

Tipo:

Cancelar Añadir

Figura 7: Barra superior de l'aplicació.

A continuació segueix el botó per a carregar moodboards ja creats. Això ens permet visualitzar moodboards creats prèviament. Per tal de poder carregar un moodboard correctament s'ha de seleccionar un moodboard. Per fer-ho només cal obrir el desplegable i seleccionar el nom del moodboard desitjat. Si no se selecciona cap nom, la icona no estarà activa, ja que seleccionar un nom és un requisit indispensable per tal de fer la càrrega correctament.

Un cop es premi el botó apareixeran els objectes a la graella segons la distribució del moodboard carregat. L'aparició dels objectes pot ser una mica lenta, ja que per a cada casella ha de buscar l'objecte que hi ha d'anar i si aquest no té imatge descarregar-la. Aquest procés pot ser lent, però si tenim en compte que prèviament s'han carregat totes les imatges de tots els objectes, la col·locació d'aquest ha de ser ràpida.

Com hem dit anteriorment, un moodboard pot contenir diferents distribucions d'objectes, així doncs, en carregar aquest moodboard també hem de poder visualitzar totes les distribucions disponibles. Els dos botons del costat del botó de carregar moodboards ens permetran mostrar les diferents distribucions que té el moodboard. Només de carregar el moodboard es mostrarà la primera distribució de totes. Si només es disposa d'una distribució, els dos botons es trobaran inactius mentre que si hi ha més d'una distribució els botons s'aniran activant per permetre l'avanç

o el retrocés de les distribucions. En qualsevol dels casos, les distribucions si per algun motiu està malament o s'han de modificar és poder canviar els objectes que les formen. En cas de modificació, la distribució es guardarà segons el nom escollit, si es dóna el cas que és el mateix nom, s'afegirà a la distribució corresponent i aquesta podrà ser carregada i visualitzada més endavant.

Seguidament ens trobem amb el botó de generació de PDF. Aquest botó ens permetrà generar un PDF del moodboard actual en el qual apareixerà una captura de la graella amb els elements, amb el nom del moodboard i a continuació a cada una de les pàgines del fitxer, cada objecte amb la seva informació corresponent.

En fer clic sobre el botó ens apareixerà una finestra en la qual ens indicarà a on volem guardar el PDF que es generarà. Un cop seleccionada la ruta només hem de prémer el botó de guardar i a continuació es generarà el PDF. Aquest procés pot tardar una miqueta, ja que generar un fitxer amb molta informació pot ser lent, d'aquesta forma, per indicar que el fitxer s'ha generat correctament, la icona es desactivarà, indicant que s'ha creat el fitxer correctament i que el PDF del moodboard actual ja s'ha generat. En cas de modificació del moodboard, es tornarà a activar la icona per a generar un nou fitxer del moodboard.

Finalment trobem el botó de guardar. Aquest botó ens permet guardar el moodboard generat de forma que s'exporta en un format que després serà tractat per utilitzar-lo amb la màquina. Per tal de poder guardar un moodboard és necessari que aquest tingui un nom si no la icona de guardar no es trobarà activa.

Si en guardar, no existeix cap moodboard amb el nom escollit, es crearà un nou fitxer en el qual s'emmagatzemaran els moodboards amb el mateix nom. En cas contrari, que ja existeixin moodboards amb el mateix nom, s'afegirà la distribució d'objectes al fitxer. D'aquesta forma, per un sol moodboard podem tenir diferents distribucions.

S'ha desenvolupat d'aquesta forma per tal d'assegurar que el format en el qual cada moodboard és exportat és fàcilment manipulable per transformar-lo per un posterior tractament per entrenar la màquina.

2.2.3 Procés de generació d'un moodboard

En aquest punt explicarem els passos a seguir per crear un moodboard correctament, per a generar un moodboard pas a pas utilitzant tota la informació prèviament explicada. A continuació s'enumerarà cada pas i en acabat es mostrarà un diagrama de seqüència per tal d'entendre-ho millor.

1. **Obrir correctament l'aplicació i comprovar que es carreguen tots els objectes i la graella.** En cas contrari pot ser culpa de la connexió a internet o la connexió a base de dades.
2. **Categoritzar la graella.** En cas que la graella estigui buida categoritzar-la per tal de fer el filtratge d'objectes correctament.

3. **Escollir un nom o afegir-ne un de nou.** Escollir un nom pel moodboard per tal de poder guardar i generar el PDF en qualsevol moment.
4. **Emplenar graella.** Emplenar la graella amb els objectes necessaris fins a obtenir un moodboard correcte. Utilitzar les funcions de filtratge per tal de poder col·locar els objectes amb més precisió i més eficaçment.
5. **Guardar moodboard.** Un cop s'hagi acabat d'emplenar el moodboard guardar el moodboard correctament per tal de no perdre els canvis realitzats.
6. **Generar PDF.** Un cop guardat el moodboard guardar el PDF per tenir una referència i una visualització ràpida del moodboard generat.

A continuació es mostra un esquema de seqüència dels passos a realitzar de completar els passos anteriors.

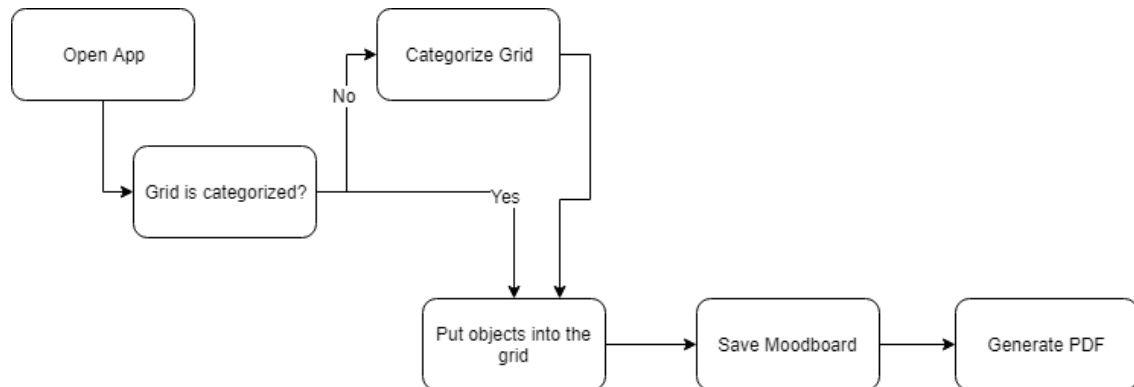


Figura 8: Digrana de seqüència per a generar un moodboard.

3 Desenvolupament del Machine Learning

<https://blog.adext.com/es/machine-learning-guia-completa> resum del machine learning, l'estat actual i les seves utilitats

La part més important de projecte no és el desenvolupament de l'eina i el seu funcionament, sinó que és la part del algorisme. L'algorisme és el que ens permetrà generar automàticament moodboards un cop aquest s'hagi entrenat i és el que ens permetrà, que la generació de moodboards sigui ràpida i automàtica.

Així doncs, a continuació explicarem en què es basa l'algorisme que realitzarem, explicant una mica d'introducció i història per posar en context. També explicarem els diferents tipus i quin d'ells s'ha escollit i com s'ha desenvolupat.

3.1 Introducció i antecedents

Parlar avui d'Intel·ligència Artificial no és una novetat. Ni per a la gent ni per a les empreses o governs.

Si bé sembla una qüestió d'allò més naturalitzada per aquests dies, fins fa no molts anys enrere, parlar d'intel·ligència artificial era un assumpte distant, difícil d'abordar, reservat només per a entesos.

Avui en dia, es parla i s'aplica per qüestions tan bàsiques com l'ús del homebanking, veure pel·lícules en streaming o escoltar música. Fins als cercadors com Google o Yahoo la utilitzen. I a les xarxes socials, els exemples de Facebook i Twitter lideren el segment.

Cada vegada més, es llegeix i es comenta sobre intel·ligència artificial en mitjans massius de comunicació, amb tota facilitat. Fins i tot, comença a guanyar un espai dintre del pressupost propi de les empreses al país: la recerca per la reducció dels costos i l'augment de la productivitat vénen impulsant l'avanç en l'ús de solucions que involucren robots capaços de conversar amb els clients i de sistemes que analitzen milers de dades en pocs segons. En alguns casos, el volum de recursos destinats a aquest segment està previst que augmenti en més de cinc vegades en 2018, en relació a l'aplicat l'any passat.

Dins de l'ampli espectre que abasta la intel·ligència artificial, es troba Machine Learning. **Però que és el Machine Learning?**

El machine learning, conegut en espanyol com aprenentatge automàtic o aprenentatge de màquina, va néixer com una idea ambiciosa de la IA en la dècada dels 60. Per ser més exactes, va ser una subdisciplina de la IA, producte de les ciències de la computació i les neurociències.

El que aquesta branca pretenia estudiar era el reconeixement de patrons (en els processos d'enginyeria, matemàtiques, computació, etc.) i l'aprenentatge per part

de les computadores. En les albors de la IA, els investigadors estaven àvids per trobar una forma en la qual els ordinadors poguessin aprendre únicament des de dades.

Va succeir amb el pas dels anys que el machine learning començar a enfocar-se en diferents assumptes, com ara el raonament probabilístic, investigació basada en l'estadística, recuperació d'informació, i va continuar aprofundint cada vegada més en el reconeixement de patrons (tots aquests assumptes aplicats a processos d'enginyeria, matemàtiques, computació i altres camps relacionats amb objectes físics o abstractes).

Això va ocasionar que en els 90 es separés de la IA per convertir-se en una disciplina per si sola, encara que molts puristes encara la consideren com a part de la IA. Ara, el principal objectiu del machine learning és abordar i resoldre problemes pràctics en on s'apliqui qualsevol de les disciplines numèriques abans esmentades. Com vam establir prèviament, és un camp de les ciències de la computació que, d'acord a Arthur Samuel el 1959, li dona a les computadores l'habilitat d'aprendre sense ser explícitament programades.

Si aquesta definició va resultar molt trivial, posem-d'aquesta manera: és la idea que hi ha algoritmes que poden donar-te troballes o conclusions rellevants obtingudes d'un conjunt de dades, sense que l'ésser humà hagi d'escriure instruccions o codis per això.

D'acord, però ¿què és un algorisme? Doncs no és altra cosa que una seqüència o sèrie d'instruccions, que representen la solució a un determinat problema.

El propòsit del machine learning és que les persones i les màquines treballin de la mà, aquestes màquines han de ser capaços d'aprendre com un humà ho faria. Precisament això és el que fan els algoritmes, permeten que les màquines executin tasques, tant generals com específiques. Si bé al principi les seves funcions eren bàsiques i es limitaven a filtrar emails, avui dia pot fer coses tan complexes com prediccions de trànsit en interseccions molt transitades, detectar càncer, mapejar llocs per generar projectes de construcció en temps real, i fins i tot, definir la compatibilitat entre dues persones.

El principal objectiu de tot aprenent (learner) és desenvolupar la capacitat de generalitzar i associar. Quan traduïm això a una màquina o ordinador, significa que aquestes haurien de poder exercir-se amb precisió i exactitud, tant en tasques familiars, com en activitats noves o imprevistes.

I com és possible això?

Fent que repliquin les facultats cognitives de l'ésser humà, formant models que "generalitzin" la informació que se'ls presenta per realitzar les seves prediccions. I l'ingredient clau en tota aquesta qüestió són les dades.

En realitat, l'origen i el format de les dades no és tan rellevant, ja que el machine learning és capaç d'assimilar una àmplia gamma d'aquests, el que es coneix

com big data, però aquest no els percep com a dades, sinó com una enorme llista d'exemples pràctics.

Podríem dir que les seves algoritmes es divideixen principalment en tres grans categories: supervised learning (aprenentatge supervisat), unsupervised learning (aprenentatge no supervisat) i Reinforcement learning (aprenentatge per reforç). A continuació, detallarem les diferències entre aquestes.

Una vegada que compres el fàcil i pràctic que resulta d'aplicar les tècniques de machine learning a problemes que creies serien impossibles, és quan comences a creure que podria resoldre pràcticament qualsevol problema * -sempre i quan hi hagi suficients dades-.

Per al consumidor modern, el machine learning és un facilitador clau de moltes de les seves tasques quotidianes. Des serveis de traducció, a prediccions climàtiques, fins endevinar el que els usuaris volen amb base a les seves activitats recents; les prestacions que ofereix són incomparables.

Pel que fa als negocis, moltes companyies han començat a incorporar aquesta tecnologia als seus sistemes operatius, amb grans expectatives de millorar i automatitzar els seus processos.

Atès que el machine learning és un sistema basat en el processament i anàlisi de dades que són traduïts a troballes, es pot aplicar a qualsevol camp que compti amb bases de dades prou grans. De moment, alguns dels seus usos més populars i desenvolupats són:

- Classificació de seqüències de DNA
- Prediccions econòmiques i fluctuacions en el mercat borsari
- Mapatges i modelats 3D
- Detecció de frauds
- Diagnòstics mèdics
- Cercadors a Internet
- Sistemes de reconeixement de veu
- Optimització i implementació de campanyes digitals publicitàries

Un exemple de l'últim punt és Adext. Adext és el primer i únic AMaaS (Audience Management as a Service) que aplica Intel·ligència Artificial i Machine Learning a la publicitat digital per trobar la millor audiència o grup demogràfic per a qualsevol anunci. Gestiona de forma automàtica els pressupostos al voltant de 16 públics diferents, en 3 plataformes (AdWords de Google, Facebook i Instagram), optimitzant diverses vegades al dia.

A més, se'ls garanteix sota contracte a les agències que siguin Adext Partners superar el cost per conversió actual (cost per venda o cost per lead més baix) de tots els comptes o campanyes que portin com a agència. Altrament, el servei serà gratis i no es cobra el fee corresponent.

Tot i que ja hem vist el que la IA és capaç d'aportar a les nostres activitats del dia a dia, com podria això beneficiar el món dels negocis?

Bé, ja que les converses i comentaris d'una infinitat de consumidors digitals -el nombre dia amb dia segueix incrementando- li ofereixen a aquest tipus de tecnologies una quantitat d'informació aclaparadora, aquestes contínuament obtenen coneixements nous i detecten tendències més ràpid del que qualsevol humà podria fer-ho.

Si bé és cert que aquesta enorme quantitat de dades la tornarà molt més eficient, requerirà necessàriament de molt talent humà per perfeccionar-se, ja que finalment els ordinadors no tenen un domini tan elevat del llenguatge aplicat al raonament. O el que és, no són precisament hàbils per determinar contextos.

El que significa que perquè el machine learning es desenvolupi en aquestes àrees, els experts en cada camp de treball hauran de trobar el temps per entrenar a les màquines i anar incorporant paulatinament a cada un dels processos que desitgin afinar.

Finalment, com succeeix amb totes les tecnologies, els negocis hauran de començar per entendre els principis bàsics d'aquesta tecnologia, per poder usar-la a favor seu. També s'estima que aquesta -com molts altres derivats de la IA- transformarà per complet el món com el coneixem.

Com podem observar això només reflecteix una petita introducció al intricat món del machine learning. En una època on emergeixen tecnologies innovadores cada vegada que parpellegem, és fàcil perdre en l'allau d'informació i nous conceptes.

3.2 Tipus d'algoritmes

Dintre del conjunt del Machine Learning poder distingir entre diferents tipus que ens definiran diferents tipus d'algoritmes a aplicar durant el desenvolupament d'una màquina. Els principals tipus són els següents:

- Aprenentatge supervisat
- Aprenentatge no supervisat
- Aprenentatge per reforç

A continuació s'analitzarà en profunditat cada un dels tipus anteriors. Seguidament mostra una imatge resum dels diferents tipus i un exemple de cada un.

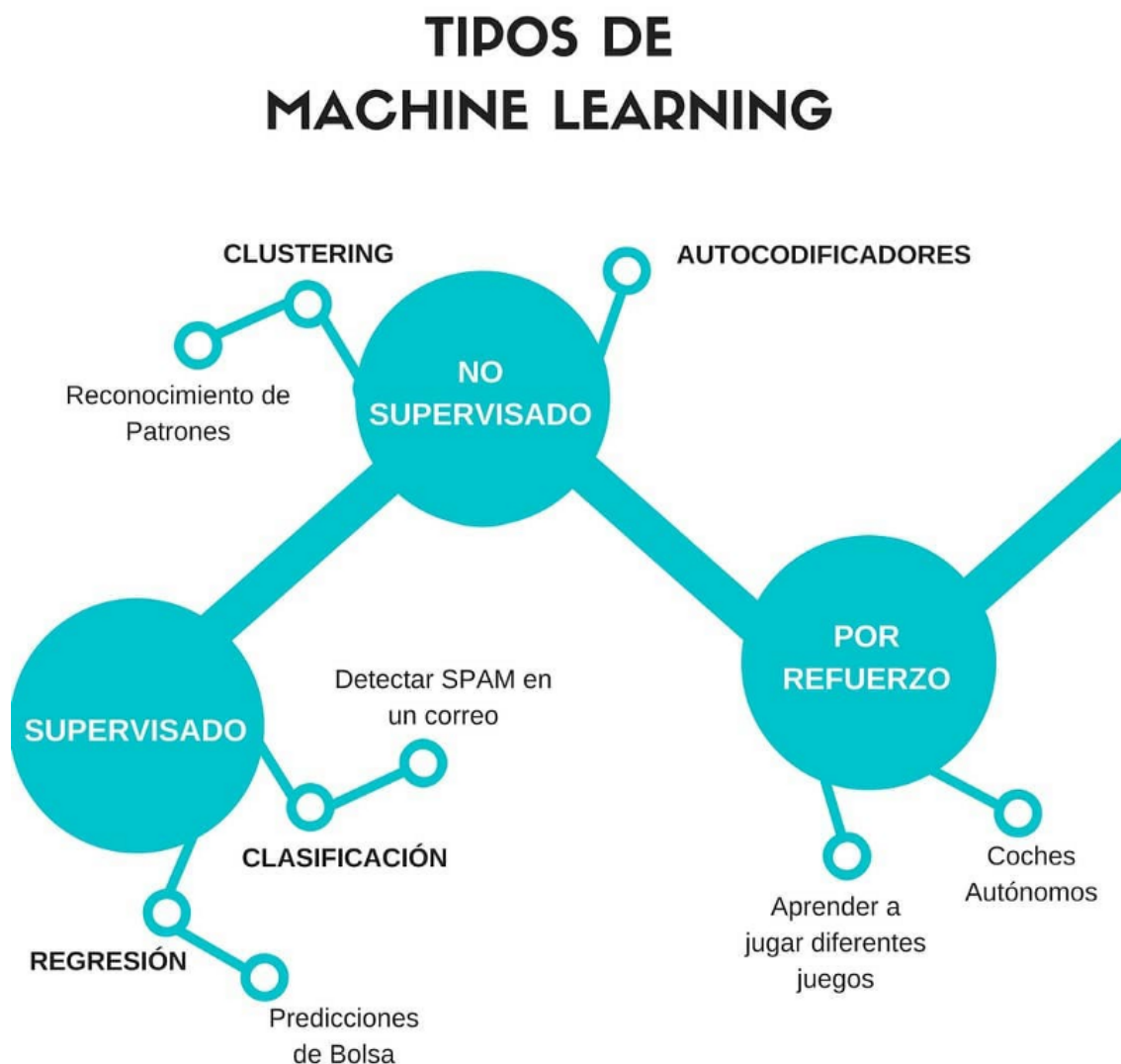


Figura 9: Diagrama dels tipus i exemples.

3.2.1 Aprenentatge supervisat

En l'aprenentatge supervisat, els algorismes treballen amb dades etiquetades (labeled data), intentant trobar una funció que, donades les variables d'entrada (input data), els assigni l'etiqueta de sortida adequada. L'algorisme s'entrena amb un històric de dades i així aprèn a assignar l'etiqueta de sortida adequada a un nou valor, és a dir, prediu el valor de sortida.

Per exemple, un detector de spam, analitza l'històric de missatges, veient que funció pot representar, segons els paràmetres d'entrada que es defineixin (el remitent, si el destinatari és individual o part d'una llista, si l'assumpte conté determinats termes, etc.), l'assignació de l'etiqueta spam o no és spam. Una vegada definida aquesta funció, en introduir un nou missatge no etiquetat, l'algorisme és capaç d'assignar-li l'etiqueta correcta.

L'aprenentatge supervisat se sol usar en problemes de classificació, com a identificació de dígit, diagnòstics, o detecció de frau d'identitat. També s'usa en problemes de regressió, com a prediccions meteorològiques, d'expectativa de vida, de creixement, etc. Aquests dos tipus principals d'aprenentatge supervisat, classificació i regressió, es distingeixen pel tipus de variable objectiu. En els casos de classificació, és de tipus categòric, mentre que, en els casos de regressió, la variable objectiu és de tipus numèric.

Alguns dels més freqüents en aprenentatge supervisat:

1. Arbres de decisió
2. Classificació de Naïve Bayes
3. Regressió per mínims quadrats
4. Regressió Logística
5. Support Vector Machines (SVM)
6. Mètodes "Ensemble" (Conjunts de classificadors)

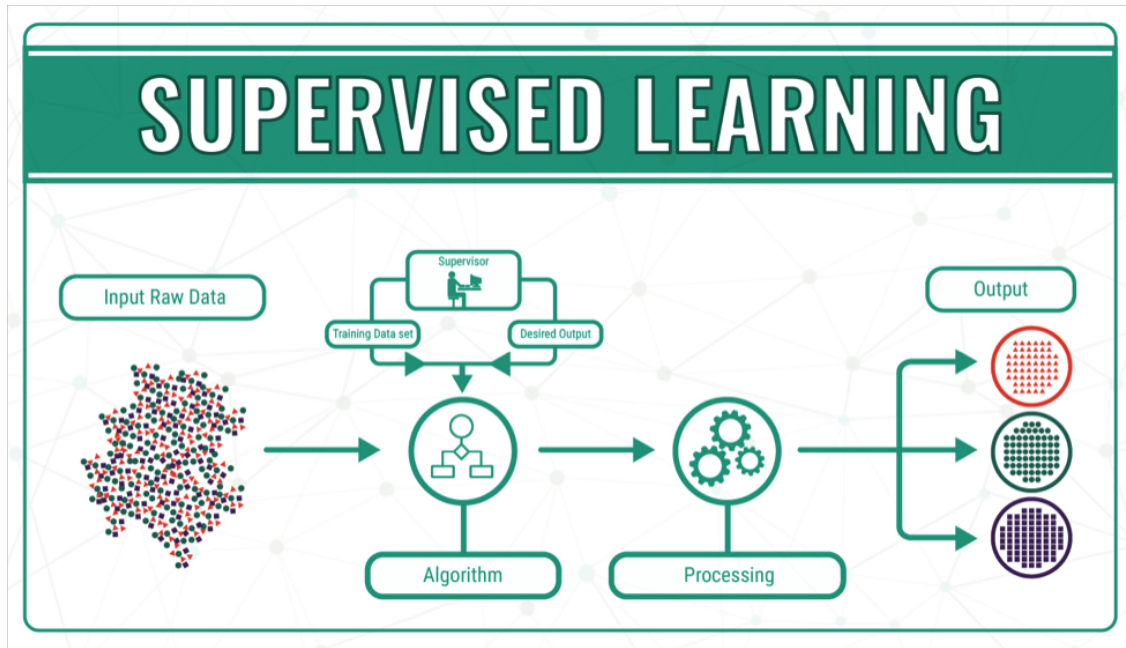


Figura 10: Diagrama d'un algorisme amb aprenentatge supervisat.

3.2.2 Aprenentatge no supervisat

L'aprenentatge no supervisat té lloc quan no es disposa de dades etiquetades per a l'entrenament. Només coneixem les dades d'entrada, però no existeixen dades de sortida que corresponguin a un determinat input. Per tant, només podem descriure l'estructura de les dades, per intentar trobar algun tipus d'organització que simplifiqui l'anàlisi. Per això, tenen un caràcter exploratori.

Per exemple, les tasques de clustering, busquen agrupaments basats en similituds, però gens garanteix que aquestes tinguin algun significat o utilitat. A vegades, en explorar les dades sense un objectiu definit, es poden trobar correlacions esporàdiques curioses, però poc pràctiques. Per exemple, en la gràfica inferior, publicada a la web de Tyler Vigen Spurious Correlations, podem apreciar una forta correlació entre el consum per càpita de pollastre als Estats Units i les seves importacions de petroli.

L'aprenentatge no supervisat se sol usar en problemes de clustering, agrupaments de co-ocurrència i profiling. Si embargament, els problemes que impliquen tasques de trobar similitud, predicció d'enllaços o reducció de dades, poden ser supervisats o no.

Els tipus d'algorisme més habituals en aprenentatge no supervisat són:

1. Algorismes de clustering
2. Anàlisi de components principals
3. Descomposició en valors singulars (singular value decomposition)

4. Anàlisi de components independents (Independent Component Analysis)

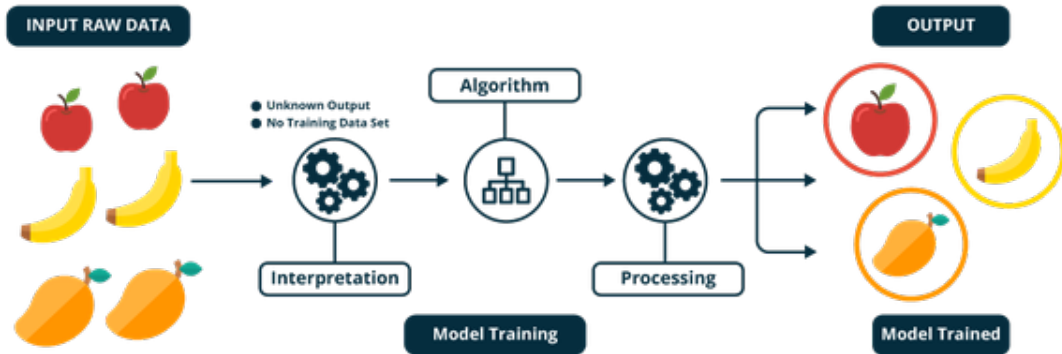


Figura 11: Diagrama d'un algorisme amb aprenentatge no supervisat.

3.2.3 Aprenentatge per reforç

El mètode d'aprenentatge de reforç pretén utilitzar observacions recollides a partir de la interacció amb el medi ambient per prendre accions que maximitzin la recompensa o minimitzin el risc. L'algoritme d'aprenentatge de reforç (anomenat agent) contínuament aprèn de l'entorn d'una manera iterativa. En el procés, l'agent aprèn de les seves experiències sobre el medi ambient fins que explora tot el ventall d'estats possibles.

L'aprenentatge de reforç és un tipus d'aprenentatge automàtic, i també una branca d'Intel·ligència Artificial. Permet que les màquines i els agents de programari determinin automàticament el comportament ideal en un context específic, per maximitzar el seu rendiment. Es requereix feedback simple de recompensa perquè l'agent aprengui el seu comportament; això es coneix com el senyal de reforç.

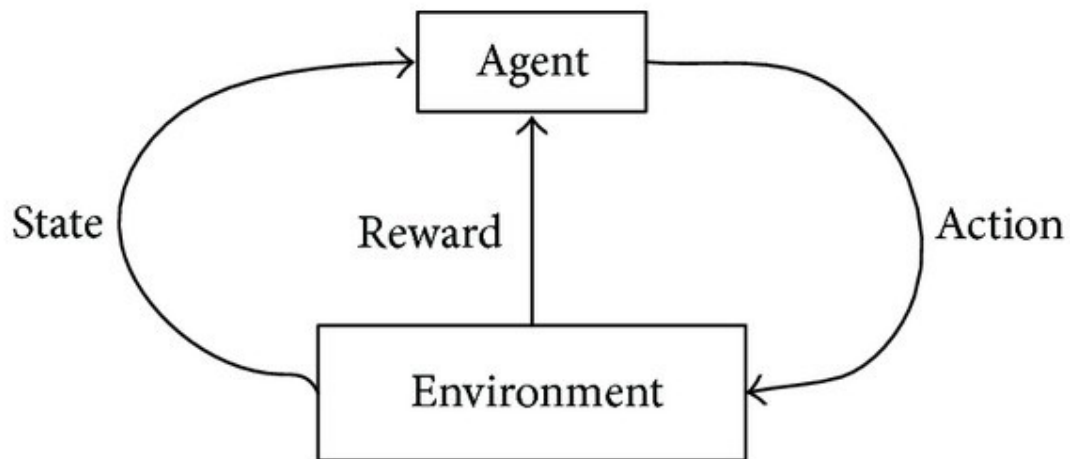


Figura 12: Exemple del funcionament de l'algorisme.

Hi ha molts algorismes diferents que s'ocupen d'aquest problema. De fet, l'educació de reforç es defineix per un tipus específic de problema, i totes les seves solucions es classifiquen com a algorismes d'aprenentatge de reforç. En el problema, se suposa que un agent decideix la millor acció per seleccionar en funció del seu estat actual. Quan es repeteix aquest pas, el problema es coneix com a procés de decisió de Markov.

Per produir programes intel·ligents (també anomenats agents), l'aprenentatge de reforç passa pels següents passos:

1. L'agent l'observa l'estat d'entrada.
2. La funció de presa de decisions s'utilitza per fer que l'agent realitzi una acció.
3. Una vegada realitzada l'acció, l'agent rep recompensa o reforç des del medi ambient.
4. S'emmagatzema la informació del parell d'acció estatal sobre la recompensa.

Els principals algorismes són:

- Q-Learning
- Diferència temporal (TD)
- Xarxes adversàries profundes

3.3 Plantejament de l'algoritme

Un cop ja tenim els coneixements bàsics del machine learning necessitem pensar en quin algoritme necessitem per tal de poder aplicar-lo al nostre problema i desenvolupar-lo correctament.

Pel nostre problema disposarem de dades d'entrada etiquetades, ja que seran les graelles de cada estil, és a dir, les dades d'entrada seran vectors de graelles, els quals cada d'un d'ells serà un moodboard d'un estil concret. Un estil pot tenir diferents graelles, però totes aquestes estaran etiquetades. I el que es vol aconseguir, és obtenir una graella nova donat un estil.

D'aquesta forma el nostre algoritme ha de ser un algoritme de classificació supervisat, ja que el que es vol obtenir és classificar graelles per després poder classificar-ne de noves i com que aquestes estan etiquetades aquest algoritme serà supervisat. En aquests tipus d'algoritmes les dades segueixen un format estàndard compost per dues parts, les etiquetes (labels) i els valors (features).

Les labels podríem dir que és l'output, el valor de sortida, el valor que obtenim després de classificar les dades d'entrada. Aquestes etiquetes poden ser valors únics o classes però són els valors que ens permeten etiquetar les dades correctament.

Les features són variables individuals independents que actuen com a entrada al sistema. Els models de predicció utilitzen funcions per fer prediccions. També es poden obtenir funcions noves a partir de funcions antigues mitjançant un mètode conegut com a enginyeria de característiques. Més simplement, podem considerar que una columna del conjunt de dades sigui una feature. De vegades, aquests també s'anomenen atributs. I la quantitat de funcions s'anomenen dimensions.

Aplicant això al nostre problema obtindríem que les labels serien els estils i les features les diferents graelles que generem a partir de l'eina.

Els algoritmes de classificació esmentats anteriorment classifiquen features d'entrada, és a dir, un cop entrenada la màquina permet classificar qualsevol dada d'entrada i obtenir una label, classificar aquesta dada. Si apliquem aquest comportament al nostre problema, obtindríem que donada una graella la podríem classificar, podríem dir de l'estil que és, de l'estil que pertany, però això, no és el que nosaltres volem aconseguir.

El que volem aconseguir és que donat un estil ens generi una graella nova que segueixi les normes d'aquell estil que ha après la màquina. Volem que les nostres features siguin els estils i les labels les graelles però això no pot funcionar, ja que les labels han de ser valors únics o classes i les nostres graelles són vectors amb vectors, és una llista de dues dimensions. Així doncs necessitem un algoritme que ens permeti generar valors.

Dintre del machine learning hi ha uns algoritmes que ens permeten generar va-

lor a partir dels valors d'entrada, aquests algorismes són els algorismes generatius i discriminatius.

Els algorismes generatius són un model per a generar valors aleatoris d'una dada observable, típicament donats alguns paràmetres ocults. Els algorismes generadors contrasten amb els models discriminadors; un algorisme generador és un gran model probabilista de totes les variables, mentre que un algorisme discriminador proporciona un model sol per les variables etiquetades com a condicionals sobre les variables observades.

Per això un model generador pot ser utilitzat, per exemple, per simular (i.e. generar) valors de qualsevol variable en el model, mentre que un algorisme discriminador permet el mostreig únic de les variables condicionals. A la pràctica les dues classes són vistes com a complementàries o com diferents observacions del mateix procediment.

Així doncs, el que necessitem per al nostre problema és un algorisme generatiu que donats una sèrie de vectors que formen un estil, la màquina sigui capaç de generar una nova graella. Tot i això, aquest model ens provoca una restricció que és que haurem de tenir una màquina entrenada per a cada estil, ja que així aconseguim que els resultats siguin més precisos i correctes, ja que només permetrem que es generin graelles d'un estil per a cada màquina.

Tenim diferents tipus d'algorismes generatius. Els més coneguts són els següents:

- Model de barreja gaussiana i altres tipus de model de mescla
- Model ocult de Markov
- Gramàtica lliure de context probabilística
- Classificador bayesià ingenu
- Averaged one-dependence estimators
- Latent Dirichlet Allocation
- Restricted Boltzman machine

Després d'investigar entre totes les diferents opcions, vam arribar a la conclusió que la millor opció era utilitzar una Màquina de Boltzmann Restrictiva (RBM) que ens permetria generar graelles a partir de les graelles ja generades anteriorment. Aquest tipus de màquina té algunes restriccions a l'hora d'entrar les dades, ja que han de seguir un format específic, a continuació explicarem com funciona aquest tipus d'algorisme i les seves característiques.

4 Màquina de Boltzmann(BM) i Màquina de Boltzmann Restrictiva (RBM)

A continuació explicarem d'una forma més detallada les màquines de Boltzmann i la seva variant, que és la que utilitzarem, la màquina de Boltzmann Restrictiva. Bàsicament ens centrarem en com entrenar i avaluar les màquines utilitzant el gradient estocàstic i analitzar les seves dificultats. També analitzarem els seus avantatges i inconvenients.

4.1 Màquina de Boltzmann

La màquina Boltzmann (BM) és una xarxa neuronal recurrent estocàstica que consisteix en neurones binàries (Haykin, 1998; Ackley et al., 1985). La xarxa està totalment connectada i cada connexió entre dues neurones és simètrica, de manera que l'efecte d'una neurona en l'estat de l'altra és simètrica per a cada parell.

La probabilitat d'un estat particular $X = [x_1, x_2, \dots, x_d]^T$ de la xarxa es defineix per l'energia de BM que es postula com

$$E(x|\theta) = - \sum_i \sum_{j>i} w_{ij} x_i x_j - \sum_i b_i x_i$$

on θ indica paràmetres de la xarxa que consisteixen en una matriu de pes $W = [w_{ij}]$ i un vector de polarització $b = [b_i]$. w_{ij} és el pes de les connexions sinàptiques entre les neurones i i j . Suposem que $w_{ij} = 0$ i $w_{ji} = w_{ij}$. La probabilitat d'un estat x és, doncs,

$$P(x|\theta) = \frac{1}{Z} \exp[-E(x|\theta)] \quad (4.1)$$

on

$$Z(\theta) = \sum_x \exp[-E(x|\theta)]$$

és la constant de normalització. Es dedueix de (3.1) que la probabilitat condicional d'una sola neurona que sigui 0 o 1 donada als estats de les altres neurones es pot escriure de la següent manera:

$$P(x_i = 1|(x_{\setminus i}, W)) = \frac{1}{1 + \exp(-\sum_{j \neq i} w_{ij} x_j - b_i)} \quad (4.2)$$

on $x_{\setminus i}$ denota un vector $[x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_d]^T$.

Les neurones de BM solen estar dividides en visibles i ocultes $x = [v^T, h^T]^T$, on els estats v de les neurones visibles es subjecten a les dades observades, i els estats h de les neurones ocultes poden canviar lliurement. En aquest cas de tenir neurones visibles i ocultes, es pot calcular la probabilitat d'una configuració específica de les neurones visibles marginant les neurones ocultes.

4.1.1 Entrenament Màquina de Boltzmann

Els paràmetres de BM es poden obtenir a partir de les dades utilitzant l'estimació estàndard de màxima versemblança. Donat un conjunt de dades, la probabilitat de registre dels paràmetres de BM és

$$\mathcal{L}(\theta) = \sum_{t=1}^N \log P(v^{(t)}|\theta) = \sum_{t=1}^N \log \sum_h P(v^{(t)}, h|\theta) \quad (4.3)$$

on les mostres $v^{(t)}$ s'assumeixen que són independents entre si, i els estats h de les neurones amagades han de ser marginades.

El gradient de la probabilitat de registre s'obté prenent derivada parcial de $\mathcal{L}(\theta)$ respecte als paràmetres w_{ij}

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \frac{N}{2} [\langle x_i x_j \rangle_d - \langle x_i x_j \rangle_m]$$

on una notació abreviada $\langle \cdot \rangle_{P(\cdot)}$ que denota l'expectativa calculada sobre la distribució de probabilitat $P(\cdot)$. A més, es van utilitzar d i m per designar dues distribucions de probabilitat $P(h|\{v^{(t)}\}, \theta)$ i $P(x|\theta)$, respectivament. Són la probabilitat de les neurones ocultes quan les neurones visibles es subjecten a les mostres, i la probabilitat de totes les neurones sense cap neurona fixa.

Segons el signe de cada terme, els dos termes es poden anomenar la fase positiva i la fase negativa, respectivament. La fórmula d'actualització general d'un paràmetre w_{ij} és

$$w_{ij} \leftarrow w_{ij} + \eta [\langle x_i x_j \rangle_d - \langle x_i x_j \rangle_m] \quad (4.4)$$

on η denota la taxa d'aprenentatge.

Així doncs, a partir d'aquí, deixem que b sigui només un vector de biaixos b_i de les neurones visibles, i c sigui un vector dels biaixos de les neurones ocultes. A continuació, per als biaixos separats de les neurones visibles i ocultes, les regles d'actualització són, en analogia amb la regla d'actualització dels pesos,

$$b_i \leftarrow b_i + \eta [\langle v_i \rangle_d - \langle v_i \rangle_m] \quad (4.5)$$

i

$$c_j \leftarrow c_j + \eta [\langle h_j \rangle_d - \langle h_j \rangle_m] \quad (4.6)$$

on v_i , h_j , b_i i c_j denoten la i i la neurona visible, la j -oculta neurona, el i -biaix visible, i el j biaix amagat.

Tot i que les regles d'activació i aprenentatge de la BM estan clarament formulades, hi ha limitacions pràctiques en l'ús de la BM. Especialment, les fórmules d'actualització basades en gradients (3.4) - (3.6) no són computacionalment factibles, ja que les distribucions requerides tant en les fases positives com negatives només es poden obtenir després de computar la constant de normalització $Z(\theta)$.

Algorithm 1 Passos de mostreig de Gibbs per al BM general

Dibuixa x_0 uniformement des de l'espai de l'estat.

repeat

for $i = 1 \dots d$ **do**

 Mostreig x_i fent l'equació (3.2)

end for

until que es reculli el nombre suficient de mostres, o el mostreig de Gibbs hagi arribat a l'equilibri.

La computació $Z(\theta)$, tanmateix, requereix sumar de forma exponencial moltes possibles configuracions de BM, i és senzillament impossible per a les BM grans.

Un mètode obvi per evitar la computació de la constant de normalització és utilitzar els mètodes de mostreig Markov-Chain Monte-Carlo (MCMC) per calcular el gradient estocàstic. A causa de la simplicitat de la regla d'activació d'una única neurona donada als estats d'altres neurones, un simple mostreig de Gibbs és suficient per obtenir gradients estocàstics.

El mostreig de Gibbs es pot implementar fàcilment perquè la distribució condicional de l'estat d'una sola neurona a BM donada als estats de totes les altres neurones ve donada per (3.2). Una descripció senzilla sobre com realitzar el mostreig de Gibbs amb BM es descriu en Algorisme 1.

Aquest enfocament pot reduir considerablement la càrrega computacional de les regles d'actualització de gradients. Si s'assumeix que la quantitat de mostres necessàries per explicar la distribució de probabilitat de tot l'espai estatal és prou menor que la mida de l'espai estatal, és a dir, el nombre de totes les combinacions possibles dels estats de les neurones, l'aprenentatge de BM ja no és computacional inviable.

Tanmateix, també existeixen altres tipus de limitacions en l'ús del mostreig de Gibbs per a la formació de BM. El major problema es deu a la connectivitat completa de BM. Atès que cada neurona està connectada i influïda per totes les altres neurones, es necessiten tants passos com la quantitat de neurones per obtenir una mostra de l'estat de BM. Fins i tot quan les neurones visibles es mantenen subjectes a les dades de formació, el nombre de passos necessaris per a una sola mostra fresca encara és almenys el nombre de neurones ocultes. Això fa que les mostres

successives de la cadena estiguin altament correlacionades entre elles i aquesta pobre barreja afecta el rendiment de l'aprenentatge.

Una altra limitació d'aquest enfocament és que les distribucions multi modals són problemàtiques per al mostreig de Gibbs (Salakhutdinov, 2009b): a causa de la naturalesa del mostreig de components, les mostres poden perdre alguns modes de distribució.

4.2 Màquina de Boltzmann Restrictiva

Per superar les limitacions pràctiques imposades a la màquina general Boltzmann com el problema del mostreig ineficient, Smolensky (1986) ha proposat una versió estructuralment restringida de la màquina Boltzmann anomenada Restringit Boltzmann Machine (RBM). RBM es construeix eliminant les connexions laterals entre les neurones visibles i les neurones ocultes. Per tant, una neurona visible només tindria vores connectades a les neurones ocultes, i una neurona oculta només tindria vores connectades a les neurones visibles. Ara, l'estructura de RBM es pot dividir en dues capes amb vores interconnectats. La relació entre BM i RBM s'il·lustra a la figura 13.

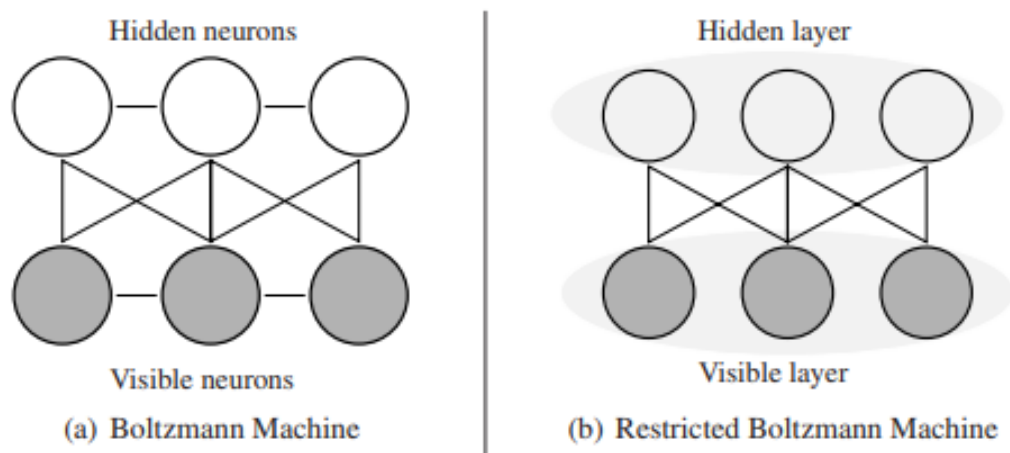


Figura 13: Il·lustració de la relació entre la màquina Boltzmann i la màquina restrictiva de Boltzmann.

Tot i que la restricció imposada podria suggerir que el poder de representació podria haver estat reduït, Le Roux and Bengio (2008) va demostrar que RBM és un aproximador universal que pot modelar qualsevol distribució de probabilitat discreta (Le Roux and Bengio, 2008).

A causa d'aquesta restricció, l'energia i la probabilitat de l'estat han de ser mo-

dificades en conseqüència a la següents fórmules:

$$\begin{aligned} E(v, h|\theta) &= -v^T W h - b^T v - c^T h \\ P(v, h|\theta) &= \frac{1}{Z(\theta)} \exp\{-E(v, h|\theta)\} \end{aligned} \quad (4.7)$$

on ara els paràmetres $\theta = (W, b, c)$ inclouen els biaixos b i c .

Atès que cada neurona oculta és independent una de l'altra donada totes les neurones visibles, és possible extreure explícitament les neurones ocultes i obtenir la probabilitat no normalitzada de les neurones visibles. La probabilitat d'un estat de neurones visibles v és, doncs,

$$P(v|\theta) = \frac{1}{Z(\theta)} \exp(b^T v) \prod_{j=1}^{n_h} (1 + \exp(c_j + \sum_{i=1}^{n_v} w_{ij} v_i)) \quad (4.8)$$

on n_v i n_h són el número de neurones visibles i invisibles respectivament.

4.2.1 Entrenament Màquina de Boltzmann Restrictiva

D'aquesta forma les regles d'aprenentatge d'un RBM passen a ser:

$$w_{ij} \leftarrow w_{ij} + \eta_w [\langle v_i h_j \rangle_d - \langle v_i h_j \rangle_m] \quad (4.9)$$

$$b_i \leftarrow b_i + \eta_b [\langle v_i \rangle_d - \langle v_i \rangle_m] \quad (4.10)$$

$$c_j \leftarrow c_j + \eta_c [\langle v_i \rangle_d - \langle v_i \rangle_m] \quad (4.11)$$

on s'utilitza la mateixa notació $\langle \cdot \rangle_{P(\cdot)}$ que s'ha usat anteriorment.

Encara que no hi ha un fons teòric rigorós sobre l'elecció de les taxes d'aprenentatge, tradicionalment, s'utilitzen taxes d'aprenentatge més petites per aprendre els dos biaixos (Hinton, 2010).

Atès que RBM és un cas especial de BM, és possible utilitzar el mateix mostreig de Gibbs per aprendre. Gràcies a la seva estructura restringida, el mostreig de Gibbs es pot utilitzar de manera més eficient, ja que una capa, ja sigui visible o oculta, les neurones de l'altra capa es converteixen en independents mútuament (vegeu la Figura 14). Aquesta possibilitat de la presa de mostres capaç permet la plena utilització de l'entorn informàtic paral·lel modern.

Tanmateix, a mesura que augmenta el nombre de neurones en RBM, el mostreig de Gibbs hauria de recollir un nombre més gran de mostres per explicar correctament la distribució de probabilitat que representa RBM. A més, a causa de la naturalesa

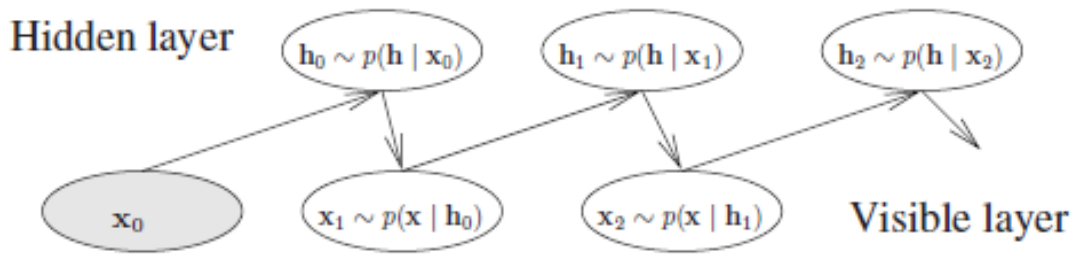


Figura 14: Visualització de la idea de com es realitza el mostreig de Gibbs en capes RBM.

de les mostres de Gibbs, les mostres encara podrien perdre alguns modes de la distribució.

S'han proposat molts enfocaments per superar aquestes dificultats.

Un enfocament popular és l'aprenentatge de divergència contrastiva (CD) proposat per Hinton (2002) com un mètode aproximat per a la formació dels models Product-of-Expert. L'equació (3.8) implica directament que RBM és un cas especial de models PoE, i l'aprenentatge de CD pot ser utilitzat fàcilment per formar RBM.

L'aprenentatge de CD s'aproxima al veritable gradient reemplaçant l'expectativa sobre $P(v, h|\theta)$ amb una expectativa sobre una distribució P_n que s'obté executant n passos del mostreig de Gibbs a partir de la distribució empírica definida per les mostres d'entrenament. La figura 15 il·lustra les distribucions P_0 i P_n .

Per als pesos, la fórmula d'aprenentatge dels CD, llavors, es defineix com

$$w_{ij} \leftarrow w_{ij} + \eta[\langle x_i h_j \rangle_{P_0} - \langle x_i h_j \rangle_{P_n}] \quad (4.12)$$

Cal assenyalar que el cas $n = 0$ produeix la distribució empírica $P(h|\{v^{(t)}\}, \theta)$ utilitzada en la fase positiva, mentre que el cas $n = \infty$ produeix la veritable distribució de la fase $P(x|\theta)$ (Carreira-Perpinayà i Hinton, 2005; Bengio and Delalleau, 2009).

Com es pot preveure del fet que la direcció del gradient no és idèntica al gradient exacte, se sap que l'aprenentatge de CD està parcial (Carreira-Perpinayà i Hinton, 2005; Bengio and Delalleau, 2009). No obstant això, s'ha demostrat que l'aprenentatge en CD funciona bé a la pràctica. Una bona propietat del CD és que, en el cas que la distribució de dades sigui multimodal, les cadenes a partir de cada mostra de dades garanteix que les mostres que s'aproximen a la fase negativa tenen representants de diferents maneres.

Tanmateix, aquest avantatge de l'aprenentatge de CD és el seu desavantatge alhora. Les mostres de P_n no necessàriament expliquen tot l'espai estatal. Per tant, no s'exploren alguns dels modes de la distribució del model, i fins i tot després de l'aprenentatge s'ha convergit, la distribució del model conté els modes que no es

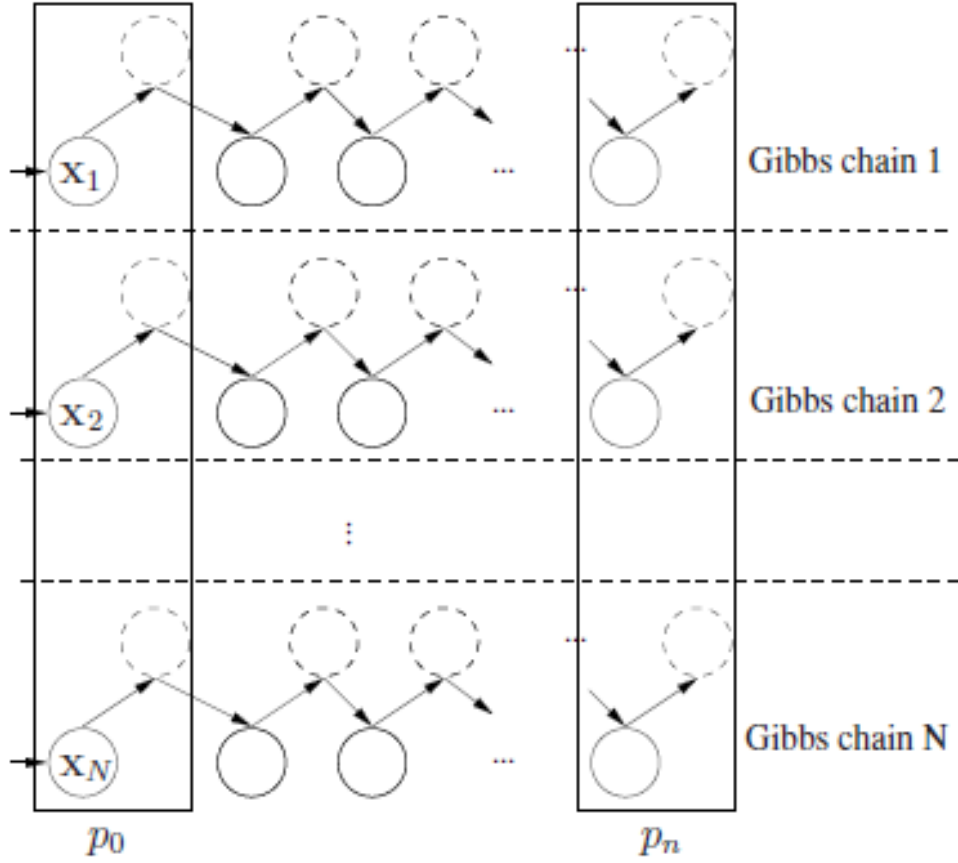


Figura 15: Visualització de com l'aprenentatge del CD obté la distribució empírica utilitzada en la fase positiva i la distribució del model aproximat utilitzada en la fase negativa. A la figura, cada fila representa la cadena de mostreig de Gibbs a partir de cada mostra de dades de formació, i P_0 i P_n indiquen la distribució empírica i la distribució del model aproximat, respectivament.

troben en la distribució de dades definida pel conjunt de dades de formació. Aquest problema es mostra a la figura 16.

Per superar aquest problema, s'han proposat diferents enfocaments basats en l'aprenentatge de CD. Entre ells, l'aprenentatge persistent de divergència contrastiva (PCD) és l'extensió més senzilla de l'aprenentatge de CD (Tieleman, 2008).

A cada pas d'actualització de gradients, l'aprenentatge de CD realitza el mostreig de Gibbs a partir de les mostres de dades de formació, mentre que l'aprenentatge de PCD comença el mostreig a partir de mostres model obtingudes en l'última actualització de degradat. D'aquesta manera, s'espera que les mostres del model exploren els modes de la distribució del model que no són pròxims a les mostres d'entrenament.

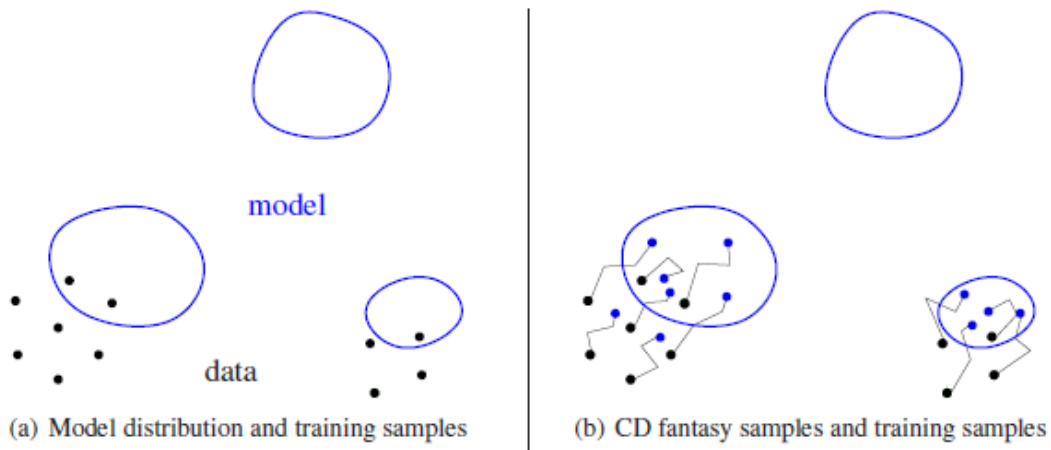


Figura 16: La figura esquerra mostra la distribució del model (blau) i les mostres d'entrenament (punts negres). Els punts blaus en la figura correcta indiquen les partícules de fantasia obtingudes mitjançant l'aprenentatge de CD. És evident que les partícules de fantasia no van explicar tot l'espai en perdre el mode a la part superior.

Tanmateix, l'aprenentatge de PCD encara pateix els modes que falten en la distribució del model a mesura que avança l'aprenentatge. Es deu a la mala barreja del mostreig de Gibbs que produeix mostres altament correlacionades per actualitzacions successives de gradients. Aquest comportament fa que els enfocaments basats en l'aprenentatge en CD pateixin la divergència de la probabilitat (Schulzet, 2010; Fischer and Igel, 2010; Desjardins et al., 2010b, a) si l'aprenentatge es realitza sense escollir manualment heurístiques d'aprenentatge com ara el calendari de tarifes d'aprenentatge, la decadència del pes i l'impuls.

S'han proposat recentment nombrosos enfocaments basats en l'aprenentatge de CDs diferents de l'aprenentatge de la PCD. Per exemple, l'aprenentatge de Fast PCD proposat per Tieleman and Hinton (2009) amplia l'aprenentatge de la PCD mantenint uns pesos ràpids que ajuden a obtenir millors models de mostres.

4.2.2 Avaluació d'una RBM

A continuació explicarem les formes d'avaluar una màquina de boltzamn restrictiva. Tenim tres formes diferents per realitzar l'avaluació. En primer lloc tenim l'avaluació per probabilitat i mostreig rellevant d'importància.

Una forma natural d'avaluar el rendiment d'un RBM entrenat és calcular la probabilitat del model i les probabilitats de realitzar mostres de dades de prova sota el RBM format. A més, tal com es comentarà al capítol 3 i es mostra en el document

de l'autor (Cho et al., 2010), la probabilitat de les mostres de dades aleatòries també es pot utilitzar com a mesura de la bondat de les RBM.

A causa de la restricció estructural, el resum explícit de les neurones amagades és força senzill (vegeu l'equació (4.8)), però, malauradament, la probabilitat d'una observació encara és intractable a causa de la constant de normalització. La constant de normalització només es pot computar exactament sumant exponencialment molts termes, i tret que la dimensionalitat del conjunt de dades sigui molt petita, és senzillament impossible. Per tant, en lloc de computar-ho exactament, cal utilitzar un mètode aproximat.

Per a l'estimació de la constant de normalització, aquesta tesi utilitza mostres d'importància recomenada (Annealed importance sampling, AIS) (Neal, 1998) que s'ha utilitzat amb èxit per calcular la constant de normalització de RBM (Salakhutdinov, 2009b).

L'AIS es basa en un mètode de mostreig d'importància simple (SIS) que podria estimar la proporció de dues constants de normalització. Per a dues densitats de probabilitat $P_A(x) = \frac{P_A^*(x)}{Z_A}$ i $P_B(x) = \frac{P_B^*(x)}{Z_B}$, la relació de dues constants de normalització Z_A i Z_B es pot estimar mitjançant un mètode de mostreig de Monte Carlo sense cap biaix si és possible provar de $P_A(\cdot)$:

$$\frac{Z_B}{Z_A} = E_{P_A}\left[\frac{P_B^*(x)}{P_A^*(x)}\right] \approx \frac{1}{M} \sum_{i=1}^M \frac{P_B^*(x_i)}{P_A^*(x_i)} \quad (4.13)$$

on X_i són mostres de $P_A(X)$. La qualitat de l'aproximació en termes de la variància depèn molt de la proximitat de $P_A(\cdot)$ i $P_B(\cdot)$. Si $P_A(\cdot)$ no és una aproximació gairebé perfecta a P_B , llavors la variància de l'estimació pot ser tan gran com l'infinit. Basat en SIS, AIS calcula la constant de normalització de la distribució del model calculant la ràtio de les constants de normalització de distribucions intermediàries consecutives que van des de l'anomenada distribució bàsica i la distribució de destinació.

La distribució base es tria de tal manera que la seva constant normativa Z_0 es pot calcular exactament i és possible recopilar mostres independents. Una selecció natural de la distribució base per a RBM és RBM amb zero pesos W . Això produeix la constant de normalització

$$Z_0 = \prod_i (1 + \exp\{b_i\}) \prod_j (1 + \exp\{c_j\}) \quad (4.14)$$

on els índexs i i j passen per totes les neurones visibles i ocultes, respectivament.

Mitjançant la computació del producte de les proporcions estimades de les constants de normalització intermèdia i Z_0 , es pot estimar la constant de normalització del RBM objectiu. L'algoritme que implementa AIS es descriu en Algorisme 2.

L'algoritme presentat descriu la construcció de RBM intermedis seguint el que va proposar Salakhutdinov (2009a). La distribució base està representada per RBM amb pesos nuls, però els biaixos són idèntics als del RBM objectiu. No obstant això, cal tenir en compte que hi ha altres possibilitats per construir distribucions intermèdies i triar una distribució bàsica. Per exemple, en els capítols següents, la distribució base és un RBM amb dos pesos zero i biaixos zero, de manera que no cal que cada RBM intermedi mantingui el doble de neurones ocultes que el RBM objectiu.

Algorithm 2 Estimació de la normalització constant per mostreig d'importància recomenada

Crear una seqüència de temperatures T_k tal que $0 = T_0 < T_1 < \dots < T_K = 1$

Crear un RBM base, R_0 amb paràmetres $\theta_0 = (W_0, b, c)$, on $W_0 = 0$

Crear una seqüència de RBMs intermedis R_k tal que

- Té el doble de nodes ocults que el RBM objectiu té
- Els seus paràmetres són $\theta_k = [(1 - T_k)W_0T_kW], [(1 - T_k)b_0T_kb], [(1 - T_k)c_0^T T_k c^T]^T$

for $m = 1 \dots M$ **do**

Obté x_1 de R_0

for $k = 1 \dots K - 1$ **do**

Mostreja x_{k+1} de R_k mitjançant un mostreig de Gibbs d'un pas a partir de x_k .

end for

Estableix $u_m = \prod_{k=1}^K \frac{P_k^*(x_k)}{P_{k-1}^*(x_k)}$, on $P_k^*(\cdot)$ és una funció de distribució marginal no normalitzada de R_k .

end for

La estimació de $\frac{Z_K}{Z_0}$ és $\frac{1}{M} \sum_{m=1}^M u_m$.

Després d'analitzar la primera forma ara en centrarem en la precisió de classificació i altres mesures de la màquina per a la seva avaluació.

És evident a partir dels treballs de recerca esmentats anteriorment utilitzant xarxes neurals profundes construïdes a partir de la pila de RBMs (Salakhutdinov, 2009b; Hinton and Salakhutdinov, 2006) que les probabilitats d'activació ocultes de RBM entrenades en el conjunt de dades podrien millorar la precisió de classificació en comparació amb classificar el conjunt de dades basat en les seves funcions en brut. Tanmateix, aquests enfocaments solen requerir l'afinació discriminativa que destrueix la generativitat de RBM.

Afortunadament, els documents recents suggereixen que les probabilitats d'activació oculta de RBM que es van entrenar sense supervisió també ajuden a la tasca de classificació. Krizhevsky (2009) va utilitzar amb èxit un Gauss-Bernoulli RBM per extreure característiques d'imatges que ajuden a obtenir una alta precisió de classificació. A més, es va demostrar que les formes més sofisticades de RBM introduïdes recentment (Ranzato and Hinton, 2010; Ranzato et al., 2010; Osindero and

Hinton, 2008) van poder extreure característiques que són més útils per a la tasca de classificació.

A més, Coates et al. (2010) va mostrar que les característiques extretes pels models probabilístics apreses d'una manera no supervisada superen les contraparts supervisades, com ara xarxes neuronals convolucionals (LeCun et al., 1998) i la xarxa de conviccions profundes (Krizhevsky, 2010). Per tant, és convenient utilitzar la precisió de classificació del RBM format com a mesura de rendiment.

Addicionalment, gràcies a la seva estructura bipartida i al mostreig de Gibbs capaç, l'error de reconstrucció també es pot utilitzar com a mesura per a l'avaluació del rendiment (Hinton, 2010). Es defineix un error de reconstrucció

$$\mathcal{E}(X) = \|x - x_1\|_2$$

Tanmateix, aquestes mesures no reflecteixen directament la veritable qualitat de RBM, ja que l'entrenament no maximitza ni minimitza cap d'aquestes mesures. Per tant, per a la resta d'aquesta tesi, els experiments valoren principalment el RBM format per la probabilitat del model i les probabilitats de les mostres de prova donades el model.

Finalment només ens queda analitzar la visualització directa i inspecció de paràmetres. Una manera d'analitzar la qualitat d'un model entrenat és mirar les característiques (les ponderacions w_{ij}) i els termes de polarització c_j corresponents a diferents neurones ocultes de la RBM modelada. Especialment ajuda quan es formen mostres de dades consisteixen en imatges que es poden visualitzar fàcilment.

Per exemple, es poden visualitzar funcions de RBM formats en dígit manuscrits. Cada característica o filtre s'assembla a una part de dígit o una combinació de parts de dígit. Quan l'aprenentatge falla, és fàcil observar característiques degenerades que són sorolloses funcions globals.

En cas de biaixos ocults, els valors mateixos suggereixen si cada neurona oculta contribueix a la capacitat de modelització de RBM. Les neurones que tenen un gran biaix c_j són la major part del temps actiu, i no són molt útils, ja que els pesos associats a ells es poden incorporar al biaix b . D'altra banda, les neurones ocultes que són majoritàriament inactives (p.Ex., Amb grans biaixos negatius c_j) o les activacions dels quals són independents de les dades també són inútils, ja que la capacitat d'aprenentatge de la RBM no canvia encara que s'eliminin.

Igual que altres mesures indirectes presentades anteriorment, la visualització i la inspecció dels valors dels paràmetres s'haurien de realitzar acuradament. No hi ha cap mesura objectiva per la qualitat de les funcions visualitzades, i les característiques visualitzades i els valors dels biaixos poden evolucionar lentament a través de l'entrenament.

4.2.3 Dificultats i solucions convencionals

El fet que la funció de destinació no es pugui calcular exactament durant l'aprenentatge dificulta l'entrenament de les RBM. És computacional inviable de saber quan l'aprenentatge ha convergit, o fins i tot no és fàcil dir si l'aprenentatge està passant realment. A més, no és possible utilitzar cap mètode de degradat avançat com el gradient de conjugat no lineal.

Atès que l'aprenentatge es realitza utilitzant actualitzacions de gradients estocàstics, convergeix a una solució local.

El problema és que no és factible comparar les diferents solucions analíticament, i triar el millor d'entre elles. Schulz et al. (2010) i Fischer and Igel (2010) han mostrat recentment que, depenent de la inicialització i els paràmetres d'aprenentatge, els RBM resultants varien molt, fins i tot en els petits conjunts de dades de joguines.

Més problemàticament, els enfocaments més aproximats presentats en els apartats anteriors s'han demostrat que difereixen, si els paràmetres d'aprenentatge no s'han triat de manera adequada. L'ús d'un millor mètode de mostreig MCMC, p. Ex. el temperament paral·lel s'ha demostrat que evita millor el comportament divergent, però a llarg termini sense utilitzar la programació de tarifes d'aprenentatge apropiada, la probabilitat del registre fluctua molt (Desjardins et al., 2010b, 2009), cosa que no és desitjable.

S'ha demostrat que RBM és un aproximador universal de manera que amb prou nombre de neurones ocultades pot modelar qualsevol distribució de probabilitat valorada discreta.

Tanmateix, a la pràctica, el nombre de neurones ocultes sempre és limitat i, depenent dels procediments d'aprenentatge, no totes les neurones ocultes contribueixen al poder de representació de RBM.

Per exemple, aquestes neurones ocultes sempre actives no tenen sentit, ja que els pesos associats a ells es poden incorporar a biaixos. A més, qualsevol neurona oculta inactiva sempre no té sentit, des de llavors, l'eliminació de la neurona oculta no afecta la capacitat de modelatge de RBM (vegeu la secció 2.3.3 per obtenir informació sobre la determinació de neurones ocultades sense sentit).

Idealment, cada neurona oculta hauria de representar una característica "significativa" diferent, per exemple, una part típica de la imatge. Tanmateix, ens hem adonat que sovint les neurones amagades tendeixen a aprendre característiques que s'assemblen al biaix visible b . Aquest efecte és més destacat en la fase inicial d'aprenentatge i en el conjunt de dades en què els bits visibles són majoritàriament actius, com ara 1-MNIST on es va desplegar cada bit de conjunt de dades manuscrits MNIST.

Es van obtenir RBM amb 36 neurones ocultes en 1-MNIST, que és un conjunt de dades molt dens en comparació amb el MNIST original. 18 neurones ocultes no van poder aprendre cap característica útil, i són majoritàriament inactives.

Les altres 18 neurones són, en la seva majoria, actives i, tal com s'esperava, s'han après unes característiques globals que s'assemblen una mica al biaix visible.

Fins i tot quan les dades de formació no són denses, amb la petita quantitat de neurones ocultades, l'elecció inapropiada dels paràmetres d'aprenentatge i l'elecció inapropiada d'inicialització dels paràmetres, moltes neurones ocultes seran inútils. La visualització dels filtres apresos per RBM amb 36 neurones ocultes entrenades a MNIST amb la taxa d'aprenentatge constant 0.1 i els pesos inicials mostrats de la distribució uniforme entre -1 i 1 són. No obstant això, encara existeixen aquestes neurones que són majoritàriament actives o majoritàriament inactives.

4.3 Algorisme seleccionat a implementar en codi

Un cop ja sabem quin algorisme hem d'utilitzar per tal de poder desenvolupar el nostre problema, ara hem de decidir com serà implementat, és a dir, si crearem nosaltres l'algorisme des de zero o utilitzarem alguna llibreria o algorisme ja conegut.

En aquest cas i després de fer molta investigació, en primer lloc es va decidir utilitzar el llenguatge de programació Python per tal de desenvolupar les parts essencials de l'algorisme. Python ens ofereix una gran quantitat de recursos per al machine learning i actualment juntament amb Scala són els principals llenguatges utilitzats per al desenvolupament de màquines.

Així doncs un cop escollit el llenguatge ara falta decidir si implementar un RBM des de 0 o utilitzar alguna llibreria que ens permetés usar una implementació ja desenvolupada.

Crear una màquina de zero suposa un gran esforç i treball i pot suposar que la màquina no estigui ben dissenyada o que els algorismes aplicats i fórmules siguin incorrectes. Així doncs es va decidir utilitzar una llibreria la qual ens permetia tenir un RBM ben creat i amb les característiques que buscàvem. La màquina que es va decidir utilitzar seria de la llibreria *sklearn* la qual ens proporcionaria una màquina RBM anomenada *BernoulliRBM*.

Aquesta màquina es tracta d'una màquina de Boltzmann Restrictiva amb unitats visibles binàries i unitats ocultes binàries. Els paràmetres es calculen utilitzant la Probabilitat màxima estocàstica (SML), també coneguda com Persistent Contrastive Divergence (PCD). La complexitat del temps d'aquesta implementació és de $O(d * 2)$ si assumim que el nombre de features és igual al nombre de components.

A més aquesta implementació ens permet utilitzar diferents mètodes i funcions per tal d'obtenir resultats i/o entrenar la màquina. Aquest són els mètodes i funcions

principals d'aquesta implementació:

- *fit*(*X*[, *y*]): ens permet entrenar la màquina a partir de les dades *X*.
- *fit_transform*(*X*[, *y*]): ens permet entrenar la màquina a partir de les dades *X* i després transformar-la.
- *get_params*([*deep*]): obtenim els paràmetres del model que hem generat.
- *gibbs*(*v*): s'executa un mostreig de Gibbs.
- *partial_fit*(*X*[, *y*]): ens permet entrenar la màquina a partir de les dades *X* que han de contenir una part segmentada de les dades.
- *score_samples*(*X*): computació de la probabilitat de *X*.
- *set_params*(** *params*): estableix els paràmetres del model.
- *transform*(*X*): computació de les probabilitats d'activació de la capa oculta.

Un exemple del ús d'aquesta màquina es troba en l'obtenció de features per la classificació de dígit. A continuació mostrarem un exemple de la creació d'un RBM utilitzant les funcions mencionades anteriorment.

```
X = np.array([[0, 0, 0], [0, 1, 1], [1, 0, 1], [1, 1, 1]])
model = BernoulliRBM(n_components=2)
model.fit(X)
#Sortida del resultat per la consola
BernoulliRBM(batch_size=10, learning_rate=0.1,
n_components=2, n_iter=10, random_state=None, verbose=0)
```

Com podem observar creem una llista de diferents elements els quals tots aquests són binaris, ja que la màquina només pot rebre entrades binàries. A continuació creem la màquina i li assignem el nombre de components i l'entrenem amb les dades d'entrada. A partir d'aquí podríem aplicar els altres mètodes per obtenir resultats o analitzar el seu funcionament.

Com podem observar al crear la màquina podem passar-li, especificar-li alguns paràmetres d'entrada. Aquests, definiran el seu comportament. Aquests paràmetres són els següents:

- *n_components*: Estableix el nombre de unitats binàries ocultes.
- *learning_rate*: La velocitat d'aprenentatge de les actualitzacions de pes. Es recomana sintonitzar aquest hiperparametre. Els valors raonables es troben en el rang de $10^{**} [0., -3.]$.
- *batch_size*: Nombre d'exemples per minibatch.
- *n_iter*: Nombre d'iteracions sobre el conjunt de dades de formació que es realitzarà durant la formació.

- *verbose*: El nivell de verbositat. El valor predeterminat, zero, significa el mode silenciós.
- *random_state*: Una instància de generador de números aleatoris per definir l'estat del generador de permutacions aleatòries. Si es dóna un enter, corregeix la llavor. Valors predeterminats del generador de nombres aleatoris global de *numpy*.

Aquests paràmetres no són obligatoris, es pot crear una màquina sense la necessitat de definir tots els paràmetres. En cas de no definir-ne cap, s'aplicaran els valors predeterminats.

Un cop tenim la implementació que usarem ara només ens queda preparar les dades per la entrada a la màquina. En els següents apartats anirem explicant el desenvolupament de la màquina creada sobre les dades que li proporcionem i el seu funcionament.

5 Desenvolupament de l'algoritme aplicat al problema

Un cop ja tenim tots els elements que componen el nostre problema només ens falta explicar el desenvolupament d'aquest utilitzant les dades que ens proporcionen i com utilitzem aquestes per a crear una màquina i a generara resultats.

A continuació analitzarem les dades que rebem de l'aplicació, com es transformen per tal de ser aptes per la màquina, com funciona i com definim la màquina i com són les dades de sortida i com les transformem per tal de poder-les tornar a llegir amb l'aplicació.

5.1 Dades obtingudes de l'eina

El primer pas com hem mencionat anteriorment és la generació de moodboards a partir de l'eina creada. Les graelles resultants són imatges formades a partir d'altres imatges, són representacions visuals que en aquest cas no podem entrar directament a la màquina ja que les hem de descompondre d'alguna manera per tal que ens sigui més fàcil entendre la seva composició.

Així doncs explicarem el format d'aquestes dades i quines dades hem obtingut en el nostre projecte.

5.1.1 Format de les dades

Com hem dit, cada imatge que forma un moodboard representa un objecte, en aquest cas un moble. Així doncs, cada moble ha de ser identificable d'alguna manera. La forma que tenim per identificar els mobles és un ID que ens proporciona la base de dades per a cada un dels mobles disponibles que tenim, d'aquesta forma podem identificar els diferents mobles que componen una graella.

Aixo ens permet obtenir un diccionari o una llista dels diferents ID que formen un moodboards que el podrem exportar d'una forma senzilla en un fitxer. Per tal d'obtenir aquesta llista, només necessitem recórrer el moodboard, començant des de la primera cel·la, i obtenir els ID d'aquelles objectes que es trobin a les caselles. En cas que no hi hagi cap objecte a la casella, posar un 0 per tal d'indicar que allà no hi ha cap objecte posat. L'algorisme 3 ens mostra el procediment d'aquest algoritme explicat.

Un cop apliquem l'algorisme descrit anteriorment, algorisme 3, obtindrem una llista de ID que abans eren simples imatges i no ens proporcionaven res d'informació a part de la visual. Ara tenim una llista amb els ID el qual cada un d'ells és un objecte que s'ha col·locat al moodboard. Així doncs de moment hem passat d'una representació visual del objecte a una representació descriptiva formada per la llista dels identificadors dels objectes. A continuació es mostra un exemple d'un resultat

Una forma de resoldre aquest problema seria descomponent cada graella del diccionari en múltiples graelles que continguin un identificador per element. L'únic problema a resoldre aquí seria el cas del que les longituds són diferents ja que no tots els objectes tenen el mateix nombre de característiques. Així doncs hem de trobar una forma de solucionar aquest problema. La solució més senzilla i més ràpida d'aplicar és buscar l'objecte amb més nombre de característiques. Un cop tenim aquest valor, aquest ens definirà el nombre de graelles les quals hem de descompondre el diccionari, és a dir, suposem que l'objecte amb més característiques en té 10. Això vol dir que el màxim nombre de graelles noves que hem de obtenir és 10 per cada moodboard generat, ja que si en fem menys no hi hauran totes els característiques d'alguns objectes.

Per tal d'obtenir aquest valor aplicarem un algorisme de cerca en tot el diccionari. Fins ara només hem suposat que tenim un moodboard generat al diccionari però podem tenir varis moodboards en ell, per tant, haurem de fer una recerca entre tots els elements del diccionari i anar guardant la longitud de cada cel·la fins a trobar la longitud màxima. Per dur a terme aquest algorisme, primer transformarem el nostre diccionari en una llista ben construïda.

Per tal de realitzar la transformació utilitzarem un *Dataframe* a partir del qual ens organitzarà els elements en forma de taula. D'aquesta taula agafarem les *values*, que correspondran a cada cel·la i les afegirem a una llista així obtindrem una llista on cada element serà una llista i aquesta tindrà com a elements les diferents característiques de cada objecte.

$$input = [[id_1^1, id_{2,i}^1, d_3^1, \dots, id_{255}^1], [id_1^2, id_{2,i}^2, d_3^2, \dots, id_{255}^2], \dots, [id_1^{48}, id_{2,i}^{48}, d_3^{48}, \dots, id_{255}^{48}]]$$

Un cop ja tenim la llista apunt només necessitem aplicar el següent algorisme:

```
def getMaxLengthForElements(vector):
    maxLength = 0
    for i in range(len(vector)):
        tmp = vector[i]
        for j in range(len(tmp)):
            if (len(tmp[j]) > maxLength):
                maxLength = len(tmp[j])
    return maxLength
```

Aquest algorisme ens retornarà el numero màxim de graelles que hem de generar de nou per tal d'obtenir a cada posició de cada element un identificador diferent.

Per a realitzar aquesta transformació hem de tenir en compte varis factors. El primer de tots hem de saber el nombre de moodboards que hem generat ja que haurem d'iterar sobre cada moodboard generat per transform-lo correctament. I el segon punt i el més important hem de tenir en compte com tractarem aquelles cel·les les quals, la seva longitud sigui menor que la longitud màxima. En aquests casos la solució plantejada és afegir un valor nul, és a dir, un valor que no porti informació, en aquest cas el valor 0. Això provocara que el resultat contingui llistes en les

quals molts elements són 0 però no ens importa ja que només ens centrarem en els identificadors que no són 0. L'algorisme aplicat per obtenir aquesta transformació és el següent:

```
def transformInputVector(vector, maxLength, length):
    output = []
    for i in range(length):
        v = vector[i]
        for j in range(maxLength):
            tmp = []
            length_v = len(v)
            for k in range(length_v):
                if j >= len(v[k]):
                    tmp.append(0)
                else:
                    tmp.append(v[k][j])
            output.append(tmp)
    return output
```

Com podem observar només necessitem passar-li el vector *input* descrit anteriorment i obtindrem un vector *outuput* que tindrà el següent format:

$$output = [[id_1^1, id_1^2, \dots, id_1^{48}], [id_2^1, id_2^2, \dots, id_2^{48}], \dots, [0, 0, \dots, id_y^{48}]]$$

Un exemple d'una graella generada a partir de la descomposició en vàries graelles de les dades d'entrada podria ser la següent:

```
[ 25 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 25 25 25 178 24 56 25 25 25 25 46 56 25 99 0 25 26 25 0 0 0 25 25]
```

Figura 19: Exemple d'una de les graelles obtingudes al realitzar la descomposició.

El número de llistes que obtindríem com aquestes seria la longitud màxima que hem obtingut aplicant l'algorisme anterior multiplicat pel nombre de moodboards que hem generat. D'aquesta forma, realitzant tot el procés obtindríem el format correcte de les nostres dades per tal d'entrenar la màquina.

Resumint, l'objectiu de la primera part del tractament de les dades és convertir les imatges obtingudes a identificadors que ens permetin classificar els objectes segons les seves característiques. Per tal d'aconseguir això primer hem de transformar aquestes imatges, en aquest cas utilitzant l'eina, en una llista de identificador els quals ens permeten identificar cada objecte dintre la base de dades. El següent pas és a partir d'aquests identificadors obtenir un diccionari on cada element estigui format per una llista, que són les 48 cel·les del moodboard, la qual cada element contingui una llista de tots els identificadors de les característiques del objecte que es trobava a la cel·la i així obtenir un format de dades més precís i descriptiu. I l'últim pas a realitzar és descompondre aquest diccionari en llistes, de forma que obtinguem un conjunt de llistes on cada element sigui un identificador d'una característica d'un objecte, és a dir, descompondre cada moodboard en tantes llistes, graelles, com sigui necessari fins a obtenir un identificador a cada element.

5.1.2 Dades obtingudes del projecte

Ara que ja sabem com és el format de les nostres dades explicarem quines són les dades que hem obtingut al utilitzar l'eina durant el procés de desenvolupament del projecte.

En primer lloc es va demanar als experts la creació d'alguns moodboards d'un estil concret. En aquest cas l'estil s'anomenava Popular Modern. Es van crear 8 moodboards totalment diferents.

Utilitzant l'eina es va obtenir fitxer on hi havia els 8 moodboards transformats a llistes de identificadors. Seguidament com hem explicat anteriorment i aplicant els algorismes descrits es van obtenir els identificadors de les característiques dels objectes i es va crear un diccionari el qual cada element era un moodboard fet de llistes de identificador a cada element, a cada cel·la. En aquest punt es va buscar la llargada màxima de característiques que va resultar ser 15. Així doncs sortiren 120 graelles noves ja que per cada moodboard, aquest es descompondria en 15 graelles on cada element, cada cel·la, contindria un identificador.

Finalment, les dades resultants després d'aplicar la primer transformació és una matriu de mida 120 per 48 on cada element seria una llista, una graella, amb un identificador d'una característica. En aquest cas, les dades proporcionades eren poques ja que amb 120 elements no seria necessari per entrenar una màquina.

Per solucionar aquest problema vam decidir duplicar algunes dades ja que en realitat era com generar un moodboard igual a un altre i per tant les dades no es veurien afectades. L'altre solució era demanar als experts que generessin 500 moodboards però per falta de temps i coordinació no va ser possible. Així doncs necessitàvem obtenir una mida de dades més gran que 5000, per tant, vam duplicar les dades 51 vegades i vam obtenir una matriu de mida 6120 per 48. Ara ja teníem una mida raonable per entrenar una màquina i podíem procedir a realitzar els següents passos.

5.2 Dades d'entrenament

Cada màquina necessita un format específic per tal de llegir correctament les dades i poder aprendre d'elles per tant necessitem transformar les dades actuals en dades llestes per la ingesta. A continuació explicarem com són aquestes dades i quines dades hem obtingut.

5.2.1 Format de les dades

Un cop ja tenim les dades d'entrada transformades i llestes tal i seguin el format establert prèviament necessitem comprovar que la màquina que crearem és capaç de llegir-les. En el nostre cas crearem un RBM i segons la descripció anterior el format de les nostres dades és el correcte però com que utilitzarem una implemen-

tació d'una llibreria, aquesta màquina té una sèrie de restriccions sobre les dades d'entrenament per a la màquina.

La més important és que les dades han de ser binàries, és a dir, cada element, cada ID ha de estar en forma de llista de binaris. Tot i que el format de les nostres dades podria funcionar en altres màquines que utilitzen el mateix algorisme, en el nostre cas necessitem transformar-les per tal que la nostra implementació funcioni correctament. En aquest punt, les dades de les quals disposem no es troben en aquest format per tant necessitem trobar alguna forma de convertir-les.

En el nostre cas, les dades que tenim són dades categòriques, és a dir, són variables que contenen valors d'etiqueta, a cada categoria li assignem un número. Així doncs, una solució al problema seria codificar cada un dels ID que tenim en binari però aquesta solució no seria òptima ja que hauríem de fer una màquina que pogués descodificar cada un d'ells per tal de poder entendre el valor que li estem entrant i inclús en alguns casos no podríem entrar certes dades si el número a codificar és molt llarg.

La solució més òptima i més assequible és utilitzar els denominats *One Hot Vectors*. Aquests tipus de vectors són un grup de bits entre els quals les combinacions vàlides de valors són només aquelles amb un sol bit alt(1) i tots els altres valors amb bits baixos(0). És a dir, ens permet codificar qualsevol número en forma d'una llista que serà tant llarga com el número més alt i que contindrà tots els valors 0 excepte la posició en la qual representi el valor a codificar.

Un exemple fàcil per a veure el funcionament d'aquests vectors és aplicar al nostre cas. Disposem de 262 característiques diferents a la nostra base de dades per tant si volem codificar un identificador d'una característica concreta, per exemple l'identificador 10, el codificarem de la següent manera:

$$10 = [0_1, 0_2, \dots, 1_{10}, 0_{11}, \dots, 0_{262}]$$

D'aquesta forma podrem codificar tots els identificadors de forma binària. Per transformar cada identificador podem crear un algorisme que ens converteixi un número en un vector o també podem utilitzar eines que ens proporcionen algunes llibreries com és el cas de la llibreria *sklearn* que ens proporciona un codificador per codificar les nostres dades. Aquest codificador s'anomena *OneHotEncoder*.

Aquest codificador ens permet transformar les nostres dades d'entrada en una matriu de on cada columna correspon a un possible valor d'una característica. Al crear un codificador podem establir alguns paràmetres d'entrada per tal de definir com serà i com seran les dades que li entrem i volem obtenir.

De tots els paràmetres que podem definir els més importants són:

- *n_values*: Nombre de valors per característica. En aquest cas el nombre de identificadors màxims que hi pot haver. Com hem dit anteriorment, aquest valor és 262.

- *sparse*: Boolea que ens indica si utilitzarem matrius disperses, *sparse matrix*, o matrius denses i el format del resultat.

A part d'aquests paràmetres també té mètodes que ens permetran realitzar certes accions. El mètodes més importants i que més endavant utilitzarem són els següents:

- *fit*($X[, y]$): ens permet entrenar el codificador amb les dades X .
- *fit_transform*($X[, y]$): ens permet entrenar el codificador a partir de les dades X i després transformar-la.
- *transform*(X): ens permet transformar unes dades X en un one hot vector.

Així doncs, un cop creat el codificador i entrenat amb les dades que obtenim de la primera transformació podem obtenir les nostres dades llestes per la ingesta de la màquina. La creació d'aquestes dades provocarà que la mida de les nostres dades creixi sobretot en les files de la matriu, en aquest cas, en la llista d'elements de cada cel·la ja que cada cel·la contindrà una successió de 0 i 1 i aquesta serà tant llarga com el nombre de valors per característica que li haguem definit.

5.2.2 Creació del codificador i dades obtingudes

Ara ja sabem què necessitem per transformar les nostres dades per realitzar la ingesta a la màquina i així entrenar-la. A continuació explicarem com hem creat el nostre codificador, quins paràmetres hem definit i quines són les dades obtingudes.

En primer recordem que de la primer part de les dades vam obtenir una matriu de mida 6120 per 48 que contenia les graelles amb els identificadors de les característiques dels objectes del moodboard. El primer que hem de definir és el nombre de valors per característica, en aquest cas disposem de 262 característiques diferents, el que és el mateix, que 262 identificadors diferents per tant, el paràmetre *n_values* serà igual a 262.

La segona cosa i la que més ens afectarà alhora de rendiment és el booleà *sparse*. Com hem dit aquest valor ens indica quin tipus de matriu farem servir i obtindrem, si *sparse matrix* o matriu denses. Per entendre bé la decisió a prendre necessitem saber les diferències entre els dos tipus.

Les *sparse matrix* només emmagatzemen les entrades que no són zero, aquelles entrades de la matriu que tenen algun valor mentre que les denses emmagatzemen tots els valors. Així doncs la diferència es troba alhora de treballar amb elles, és a dir, a l'hora de recórrer les matrius o de comprimir-les. En aquests cas és molt millor treballar amb *sparse matrix* ja que el procés serà més ràpid i sobretot quan es tracten de matrius molt grans. En cas contrari, si es tracten de matrius petites es poden utilitzar matrius denses.

Aplicant això al nostre cas podem fer servir les dos formes ja que la matriu és

gran però no exageradament. De tot formes a continuació mostrarem dos exemples, Figura 20, de la creació del nostre codificador i el resultat que dona, un utilitzant *sparse matrix* i l'altre utilitzant matrius denses.

```
%%time
oneHotEncoder = OneHotEncoder(262, sparse=False).fit(f_data)
oneHotData = oneHotEncoder.transform(f_data)
print(oneHotEncoder)
print(oneHotData.shape)

OneHotEncoder(categorical_features='all', dtype=<class 'numpy.float64'>,
              handle_unknown='error', n_values=262, sparse=False)
(6120, 12576)
Wall time: 341 ms
```



```
%%time
oneHotEncoder = OneHotEncoder(262, sparse=True).fit(f_data)
oneHotData = oneHotEncoder.transform(f_data)
print(oneHotEncoder)
print(oneHotData.shape)

OneHotEncoder(categorical_features='all', dtype=<class 'numpy.float64'>,
              handle_unknown='error', n_values=262, sparse=True)
(6120, 12576)
Wall time: 59.8 ms
```

Figura 20: Creació del codificador de les dos forma. A dalt amb matrius denses i a baix amb *sparse matrix*

Com podem observar, el resultat final és una matriu del mateix nombre de files que la matriu anterior però ara els elements són més grans ja que multipliquem cada element per 262. També podem observar que utilitzant *sparse matrix* el temps es redueix. En aquest cas el temps és molt petit però es pot observar una diferència. Així doncs el millor mètode és fer servir *sparse matrix*.

El contingut de la nostra matriu serà per a cada fila una successió de 0 i 1 per al tractar-se d'una *sparse matrix* si la mostrem per pantalla només veurem aquelles posicions que contenen un valor diferent a 0. Ara ja tenim les dades llestes per entrar a la màquina i entrenar-la però primer de tot hem de crear-la d'acord a les dades que tenim.

5.3 Creació i entrenament del RBM

Un cop ja tenim les dades transformades per l'entrenament de la màquina només ens queda crear la màquina i entrenar-la per tal d'obtenir resultats. A continuació explicarem la creació de la nostra màquina i analitzarem el seu comportament.

5.3.1 Creació del RBM a partir de les dades

En apartats anterior hem vist com crear un RBM estàndard utilitzant la llibreria. Però ara ja tenim les dades i necessitem trobar els paràmetres adients per tal que la creació d'aquesta màquina sigui la correcte o la més adient per al nostre problema. El principals camps que definirem seran el *n_components*, *learning_rate* i *n_iter*. I altres camps que modificarem però els seus valors són valors ja establerts però que els hem definit alhora de la creació tal i com hem trobat a la documentació de la llibreria.

Per començar hem de definir el nombre de unitats ocultes que tindrà la nostra màquina. Hi ha varies formes de trobar el número perfecte de unitats ocultes aplicant formules però en el nostre cas hem decidit que aquest nombre serà el valor total de les característiques disponibles que en aquest cas es 262, així disposem d'una unitat oculta per a cada un dels valors de les dades.

Un cop ja tenim el nombre de components decidit hem de decidir la velocitat d'aprenentatge. Tal i com es va explicar anteriorment ha d'estar en el rang de $10^{**}[0., -3.]$, així doncs a partir de la documentació de la llibreria i partir d'alguns exemples que ens han proporcionat vam poder establir un valor relativament correcte per a la nostra màquina.

Es va establir un valor de 0.01. Aquest valor és una funció que el que defineix és com de ràpid canvien el pesos dels valors al llarg del aprenentatge. Si aquest valor és massa alt, és possible que es perdi el punt 0 del pendent i si es massa baix pot provocar que es tardi molt a arribar aquest punt.

Finalment queda definir el nombre d'iteracions. Ja es sap que en una màquina d'aquest tipus cada iteració pot durar molt de temps per tant no volíem que tardes molt a realitzar totes les iteracions. Per tal de començar vam establir unes 10 iteracions per tal de comprovar el temps i la seva eficiència. Finalment, a partir dels resultats es va establir un valor de 20 iteracions.

Així doncs, un cop ja tenim tots els valors definits només ens queda crear la màquina. A continuació, a la figura 21 es mostra com s'ha realitzat la creació de la màquina.

```
RBM_Machine = BernoulliRBM(n_components=262, learning_rate=0.01, n_iter=20, random_state=0, verbose=True)
print(RBM_Machine)

BernoulliRBM(batch_size=10, learning_rate=0.01, n_components=262, n_iter=20,
             random_state=0, verbose=True)
```

Figura 21: Creació del RBM amb els paràmetres definits anteriorment

5.3.2 Entrenament del RBM

Un cop ja creada la màquina necessitem entrenar-la per tal de, més endavant, poder generar noves dades i així poder analitzar el seu funcionament amb les dades que li passem. Per tal d'entrenar-la només cal cridar el mètode *fit* de la màquina.

Al cridar aquest mètode li hem de passar les dades les quals volem que siguin les utilitzades per l'entrenament. En aquest cas li hem de passar la *sparse matrix* que hem obtingut anteriorment.

A mesura que es vagin realitzant les iteracions es mostrarà per pantalla la iteració la qual s'acaba de realitzar, la *pseudo-likelihood* i el temps que ha tardat. La *pseudo-likelihood* és una estimació bastant decent sobre el rendiment del model actualment encaixant les dades de formació. Com més baix sigui millor. Així doncs al entrenar la nostra màquina creada obtenim un resultat com aquest:

```
RBM_Machine.fit(oneHotData)
[BernoulliRBM] Iteration 2, pseudo-likelihood = -78.52, time = 57.54s
[BernoulliRBM] Iteration 3, pseudo-likelihood = -80.80, time = 60.48s
[BernoulliRBM] Iteration 4, pseudo-likelihood = -74.14, time = 55.01s
[BernoulliRBM] Iteration 5, pseudo-likelihood = -77.79, time = 52.40s
[BernoulliRBM] Iteration 6, pseudo-likelihood = -75.31, time = 53.13s
[BernoulliRBM] Iteration 7, pseudo-likelihood = -74.91, time = 52.39s
[BernoulliRBM] Iteration 8, pseudo-likelihood = -77.24, time = 52.09s
[BernoulliRBM] Iteration 9, pseudo-likelihood = -75.42, time = 52.31s
[BernoulliRBM] Iteration 10, pseudo-likelihood = -75.18, time = 56.31s
[BernoulliRBM] Iteration 11, pseudo-likelihood = -81.44, time = 61.02s
[BernoulliRBM] Iteration 12, pseudo-likelihood = -73.93, time = 55.00s
[BernoulliRBM] Iteration 13, pseudo-likelihood = -71.03, time = 66.80s
[BernoulliRBM] Iteration 14, pseudo-likelihood = -79.37, time = 61.06s
[BernoulliRBM] Iteration 15, pseudo-likelihood = -76.83, time = 62.95s
[BernoulliRBM] Iteration 16, pseudo-likelihood = -73.65, time = 60.16s
[BernoulliRBM] Iteration 17, pseudo-likelihood = -76.19, time = 66.38s
[BernoulliRBM] Iteration 18, pseudo-likelihood = -76.36, time = 64.10s
[BernoulliRBM] Iteration 19, pseudo-likelihood = -73.28, time = 61.93s
[BernoulliRBM] Iteration 20, pseudo-likelihood = -75.75, time = 57.33s
Wall time: 19min 22s
```

Figura 22: Resultat del entrenament de la màquina

Com podem observar el temps de cada iteració es similar i la *pseudo-likelihood* va variant a mesura que passen les iteracions. Aquesta informació ens permet analitzar el comportament de la nostra màquina, és a dir, si volem canviar el valor la *pseudo-likelihood* per tal que sigui més baix i millor necessitem anar realitzant proves i anar canviar els diferents paràmetres de la màquina. En aquest cas, una solució seria canviar el *batch_size* però això comportaria més temps per iteració i a més tenint en compte que el nombre de dades actual és baix en comparació a altres màquina i en un futur podria augmentar considerablement.

La qüestió final és fer proves fins a trobar una implementació que ens sigui assequible, que s'adapti a les nostres necessitats i a les del problema proposat. Per tal de trobar aquesta implementació es van realitzar diferents proves amb diferents valors però els valors de temps eren molt elevats i la *pseudo-likelihood* era molt alta en comparació a les altres implementacions.

Finalment un cop acaben totes les iteracions ja tenim la màquina creada per tal d'obtenir valors nous. Per tal d'evitar entrenar la màquina cada cop que volem generar valors nous es va decidir guardar tant les dades d'entrenament com la màquina en un format de fitxer *pickle*. Aquest format ens permet guardar la nostra màquina i les nostres dades sense la necessitat d'ocupar molt d'espai i també ens permet més endavant carregar de nou la màquina, ja entrenada, i les nostres dades.

5.4 Dades de sortida

Després d'haver entrenat la màquina només ens queda una pas i és generar moodboards. Però per tal de generar moodboards hem de saber com obtenir valors nous a partir de la màquina i en quin format es troben i quin format necessitem per visualitzar-los a la eina.

5.4.1 Dades obtingudes del RBM

Un cop la màquina està entrenada necessitem generar dades noves. Per tal de generar dades noves la llibreria amb la qual hem implementat la màquina ens proporciona un mètode per tal de generar aquestes dades. Aquest mètode es tracta del mostreig de Gibbs, $gibbs(v)$. El mostreig de Gibbs és un algorisme per generar una mostra aleatòria a partir de la distribució de probabilitat conjunta de dues o més variables aleatòries. A partir d'un vector de unitats visibles, de valors visibles i vàlids genera un altre vector d'unitats visibles mitjançant el mostreig de Gibbs.

Així doncs, aquesta funció ha de rebre com a paràmetre un *one hot vector*. En aquest cas, aquest vector ha de correspondre a una dada de les d'entrenament ja que són les aquestes són les úniques dades vàlides. En molts casos, es té unes dades d'entrenament i unes dades de test, però en aquest cas al tenir poques dades no s'ha dividit en dos tipus de dades.

D'aquesta forma, al passar un vector format de 0 i 1 a la funció *gibbs*, aquesta ens retornarà un vector de unitats visibles de la màquina.

Aquest vector que obtindrem estarà no estarà format per 0 i 1 sinó que estarà format per booleans a cada posició. A més la seva llargada no serà de 48 elements com la mida d'un moodboard sinó que la mida serà de 12576 com la mida de cada vector de la matriu d'entrada a la màquina. Així doncs el format d'aquest vector resultant serà:

$$[False_0, False_1, \dots, True_{125}, \dots, False_{12576}]$$

Això ens suposa alguns problemes ja que nosaltres volem un vector que tingui una mida de 48 i on a cada posició tingui un o més d'un identificador de característiques. El primer problema a observar és que haurem de transformar el vector obtingut per obtenir identificador i el segon problema és que a si només obtenim un vector només tindrem un identificador per cel·la o poder cap. Per tant haurem de realitzar el mostreig de Gibbs varis cops fins que tinguem un vector correcte.

Primer de tot hem de transformar el vector per obtenir un vector de 0 i 1. Per fer això només hem de recórrer el vector i substituir tots els valors *False* per 0 i els valors *True* per 1.

Un cop realitzat aquest procés ja tenim un vector amb el següent format:

$$[0_0, 0_1, \dots, 1_{125}, \dots, 0_{12576}]$$

El següent pas és convertir aquest vector en un de mida 48 i amb identificadors a cada posició on cada identificador pot tenir un valor entre 1 i 262 que són els identificador de les característiques que tenim. També poden tenir un valor 0 el qual ens indica que en aquella posició no hi ha cap objecte. Per realitzar aquesta transformació aplicarem el següent algorisme: Així doncs, aplicant l'algorisme ante-

Algorithm 5 Generació d'un vector amb identificadors a cada posició a partir d'un vector de 0 i 1

Vector V tal que $V = [i_0, i_1, \dots, i_{48}]$ on $0 \leq i \leq 262$.

Vector D tal que $D = [j_0, j_1, \dots, j_{12576}]$ on $j = 0$ o $j = 1$.

for $i = 1 \dots 48$ **do**

Obtenir el vector K tal que $K = D[(262 * i) : ((i + 1) * 262)]$

if K no conté 1 **then**

Afegir 0 a V

else

Afegir $K.index(1)$ a V

end if

end for

V conté a cada posició un *int* que és l'identificador d'una característica d'un objecte.

rior obtindríem un vector amb els identificadors corresponents a les característiques. Però, tot i això encara ens falta solucionar el problema d'obtenir més d'un valor per cada element. Per tal d'aconseguir el format final correcte necessitem tenir més d'un identificador a cada element, és a dir, una llista de identificador a cada posició tal i com obteníem les dades després de transformar-les al sortir de l'eina.

Per aconseguir aquest format necessitem generar diferents vectors utilitzant el mostreig de gibbs, és a dir, utilitzant les dades d'entrenament agafar vectors aleatoris i realitzar un mostreig de gibbs. Per realitzar això només cal que definim un numero de iteracions, en aquest cas, vam definir 50 ja que no era un numero molt elevat ni molt baix i realitzar un mostreig de Gibbs i transformar aquest vector resultant a 0 i 1 a cada iteració. Així aconseguim un vector que conté 50 elements on cada

Posar en el document les graelles generades per la RBM, alguna graella random i alguna graella original. Explicar quins aspectes son bons i quins dolents de les graelles generades. Per exemple, demanar al expert que, de les graelles RBM generades quines coses 'no li agraden'.

Posar imatge de la eina

Interficie grafica de la imatge.

- Que aporta la eina - Temps que triguen els humans a generar una graella amb photoshop - Temps que triguen els humans a generar una graella amb eina - Temps que triguen els humans a generar una graella amb RBM - Podries vendre-ho com una "help tool" que a un disenyador li genera 25 graelles i ell sols ha de triar quina li agrada més. I si vol variar alguna cosa de la graella.

Referències

- [1] A Beginner's Tutorial for Restricted Boltzmann Machines,
<https://deeplearning4j.org/restrictedboltzmannmachine>.
- [2] Eric Yuan; Restricted Boltzmann Machine,
<http://eric-yuan.me/rbm/>, 2014.
- [3] Restricted Boltzmann Machines (RBM),
<http://deeplearning.net/tutorial/rbm.html>.
- [4] Geoffrey Hinton; Advanced Machine Learning, Lecture 4, Restricted Boltzmann Machines,
<https://www.cs.toronto.edu/~hinton/csc2535/notes/lec4new.pdf>.