Modelització de dades òmiques amb autoencoders.

Jordi Masip Masip
Màster universitari de Bioinformàtica i Bioestadística (UOC-UB)
Bioinformàtica i Bioestadística Area 5

Nom del Consultor:
Ferran Reverter Comes i Estevan Vegas Lozano
Nom dels Professors responsables de l'assignatura:
Alexandre Sánchez Pla
Carles Ventura Royo

13 de juny de 2019

Reconeixement-CompartirIgual
CC BY-SA
Esta obra está sujeta a una licencia de Reconocimiento-CompartirIgual
3.0 España de Creative Commons

FICHA DEL TRABAJO FINAL

Títol del treball	*Modelització de dades òmiques amb autoencoders.*
Nom de l'autor	*Jordi Masip Masip*
Nom del consultor	*Ferran Reverter Comes* *Estevan Vegas Lozano*
Nom del PRA:	*Alexandre Sánchez Pla* *Carles Ventura Royo*
Data d'entrega (mm/aaaa):	06/2019
Titulaciò::	*Màster universitari de Bioinformàtica i Bioestadística (UOC-UB)*
Àrea del Treball Final:	*Bioinformàtica i Bioestadística Area 5*
Idioma del treball:	Català
Paraules clau	*Stacked denoising auto-encoder, omic data, omic data, deep learning.*

Resum del Treball (màxim 250 paraules)*:*

Finalitat: reproduir i explorar l'ús xarxes neuronals basades en autoencoders apilats per a la modelització de dades òmiques.

Context aplicació: Donada la seva gran dimensionalitat, per l'anàlisi de dades òmiques s'han hagut de desenvolupar noves tècniques. Els mètodes de deep learning són unes de les eines que en els últims anys han suposat millores en el seu tractament. El treball de Xie et al. (1) mostra una superioritat d'un model de perceptró multicapa basat en autoencoders apilats (MLP-SAE) per estudiar la relació del patró de polimorfisme genètic de nucleòtids simples (SNP) i el recompte d'expressió genètica per tècniques de seqüenciació d'alt rendiment (HTSeq), respecte a altres tècniques d'anàlisis.

Metodologia: Creació, ajust i avaluació del rendiment d'un model MLP-SAE basat en el treball de Xie et al. Es desenvolupa en R, amb l'ús dels paquets Keras i Tensorflow. Es fan ús de les dades utilitzades en el treball de referència. Es planifica cercar nous sets de dades obertes aptes per a la rèplica de l'anàlisi.

Resultats: El model reproduït presenta pitjor rendiment que el model original. Tot i tenir un error quadràtic mig millor (0.1956 vs 0.2890), no obté cap gen amb un R^2 superior a 0.2, envers 890. La cerca de noves dades no es va dur a terme per centrar-se en la millora i anàlisis del model creat.

Conclusió: L'ús de xarxes neuronals basades en autoencoders és factible, però obtenir un predictor amb bon rendiment és complex i és tècnicament exigent amb la

infraestructura utilitzada.

Abstract (in English, 250 words or less):

Objective: replicate and explore the use of a stacked auto-encoder based artificial neural network to model omic data.

Application context: omic data analysis has required the development of new analysis tools, given their great dimensionality. Deep learning tools have brought improvements to this analysis challenge. The work of Xie, et al. (1) showed the superiority of a multi-layer perceptron based on stacked autoencoders model (MLP-SAE) to study the relationship between single nucleotide polymorphisms (SNP) and genetic expression (measured using high-throughput sequencing [HTSeq]), when compared with other available methods.

Methodology: implementation, fine-tuning and evaluation of an MLP-SAE model based on Xie, et al. work. Keras and Tensorflow packages for R are used. The original paper data set were used for first evaluation of created model. A search for suitable alternative data set was planned to be performed.

Results: The created model outcome showed to be worse than the original one. Despite having better mean squared error (0.1956 vs 0.2890), it failed to have a single gene prediction with an R^2 over 0.2, compared with 890 genes over that value that got the original work. Efforts on improving and understanding the created model were prioritized, and no search for the new data set took place.

Conclusion: Using autoencoder based artificial neural network to model omic data is feasible, but get a good performance as a predictor is complex and is resource demanding to the used hardware.

Índex

1 Introducció....1
1.1 Context i justificació del Treball....1
1.2 Objectius del Treball....3
1.3 Orientació i mètode seguit....3
1.4 Planificació del Treball....6
1.5 Breu sumari dels productes obtinguts....9
1.6 Breu descripció dels altres capítols de la memòria....10
2 Model MLP-SAE....12
2.1 Breu descripció dels autoencoders....12
2.1.1 Problema de la dimensionalitat i el feature engineering....12
2.1.2 L'ecosistema dels autoencoders....13
2.1.3 Denoising autoencoder....14
2.1.4 MLP-Satcked autoencoders....14
2.2 Planificació del desenvolupament....15
2.2.1 Segmentació dades....15
2.2.2 Estructura del codi i directoris....15
2.3 DAE....16
2.3.1 Preprocessat de les dades....16
2.3.2 Creació AE: model....21
2.3.3 Creacio DAE:....25
2.3.4 Ajust hiperparàmetres....26
2.3.5 DAE final:....27
2.3.6 Stacked Autoencoder....28
2.4 MLP-SAE....29
2.4.1 Creació model MLP-SAE....29
2.5 Avaluació del model: 5-fold cross-validation....31
2.6 Resultats....32
2.7 Discussió....34
3 Conclusions....39
3.1 Conclusions del treball....39
3.2 Assoliments dels objectius....39
3.3 Seguiment, planificació i metodologia....39
3.4 Punts pendents....41
4 Glossari....42
5 Bibliografia....42
6 Annexos....44
6.1 Codi font....44
6.1.1 preprocessament_i_imputacio.R....44
6.1.2 mlp_sae_5fold_CV.R....46
6.1.3 crear_entrena_avalua.R....52
6.1.4 crear_model_mlpsae.R....53
6.1.5 entrenar_MLPSAE.R....55
6.1.6 avaluar_MLPSAE.R....57
6.1.7 funcions_suport.R....58
6.1.8 directoris_i_funcions.R....60

Llista de figures

Logo UOC.....1
Llicència Reconeixement-CompartirIgual CC BY-SA.....2
Diagrama de Gantt de la planificació original.....8
Diagrama de Gantt de la planificació modificada.....9
Esquema de recodificació dels patrons SNP.....17
Q-Q plots de les expressions mitges dels gens.....18
Anàlisis PCA de les observacions amb valors complets.....19
Diagrama de mitjanes ordenades de l'expressió dels gens.....19
Relació entre proporció de valors perduts i mitja d'expressió del gen.....20
Esquema autoencoder.....24
Historial d'entrenament del AE seleccionat.....25
Esquema del mètode per afegir soroll a les dades.....26
Esquema model DAE i SAE.....28
Historial entrenament del model SAE.....29
Esquema entrenament model MLP-SAE.....31
Distribució dels gens segons rang de R2.....33
Expressió dels gens: observat i predit.....34
Expressió predita i observada en els 200 gens amb major correlació.....34
Densitat de distribució de les mitges dels gens, predites i observades.....36
Núvol de punts de predit vs observat.....36
Q-Q plot de mitges d'expressió vs predites.....37
Núvol de punts predit vs observat de gens amb expressió no central.....38
Núvol de punts de gens d'expressió centrada.....38
Relació R2 i mitja d'expressió.....38
Esquema del flux d'execució del codi.....44

1 Introducció

1.1 Context i justificació del Treball.

Els estudis *genome-wide association studies (GWAS)* són estudis observacionals que estudien les diferències individuals en les variants o expressió genètica al llarg de tot el genoma. El seu objectiu és detectar possibles associacions amb alguna característica particular dels individus. La naturalesa de les característiques poden ser molt variades, com per exemple la presència d'una determinada condició patològica, exposicions a un determinat agent o condició ambiental, o com en el cas concret d'aquest treball, el patró de polimorfisme de nucleòtid simple (single *nucleotide polymorphisms o SNP).*

Per l'estudi de l'expressió genètica s'han desenvolupat al llarg dels anys diferents tècniques. Entre elles, com un dels referents actuals, s'inclouen els estudis de seqüenciació d'alt rendiment (High-Throughput Sequencing assays o *HTSeq)*. Aquestes tècniques consisteixen de forma resumida a amplificar massivament el transcriptoma de la cèl·lula per tècniques de PCR. D'aquesta manera s'aconsegueixen microsegments genètics que són seqüenciats. El conjunt de microlectures obtingudes d'aquest procés, són seguidament alineades sobre una base de dades del genoma conegut de l'espècie estudiada.

En obtenir l'alineació de les microlectures, es pot anotar el gen al qual pertanyen i fer un recompte del nombre de lectures pertanyents a cada gen. Aquest recompte és utilitzat com el subrogat de l'activitat d'expressió genètica de cada gen. S'havien proposat diferents mètodes de correcció de l'expressió per diferents factors, com per exemple corregint les lectures segons la mida del gen en parell de bases, i d'altres amb consideracions anàlogues, però s'han mostrat menys precises que el recompte simple (2).

En l'escenari dels estudis GWAS, i en general en l'estudi de dades òmiques, el problema de la dimensionalitat de les dades i la relativa baixa mostra amb què se sol poder treballar, en relació al nombre de variables generades, és un dels principals reptes (3).

Aquest treball posa l'atenció en un treball de Xie et al. (1) en el que es proposava un model de perceptró multicapa basat en denoising autoencoders (MLP-SAE) per a la predicció del patró HTSeq a partir del patró de SNP coneguda. A la proposta l'acompanyava una comparació del rendiment amb altres tècniques realitzades fins aleshores per realitzar aquest tipus l'estudi.

En l'apartat d'introducció de l'article es presenta un resum de l'evolució i l'estat de les tècniques aplicades fins al moment de la publicació, a finals de 2017. Aquest tipus d'estudi GWAS són també anomenats estudis eQTL («*expression Quantitative trait locus*»), i han abastat clàssicament en els seus inicis estudis multivariants amb les seves modificacions progressives, fins als mètodes de *machine learning*, i *Deep*

*Learning. A*quests últims s’han aplicat amb millors resultats que els eQTL tradicionals (4).

Aquest tipus d’estudi presenten entre d’altres el problema de la dimensionalitat de les dades, com ja s’ha comentat anteriorment. A l’hora de modelar les dades per poder utilitzar-les per a models predictius o de classificació, els models basats en xarxa neuronals s’han mostrat superiors a altres mètodes clàssics com l'anàlisi de components principals per a la reducció de dimensionalitat i l’obtenció d’una representació en un subespai reduït que pugui ser utilitzada com a font de dades per a prosseguir anàlisis no realitzables amb la informació expressada en el seu espai original (5).

Obtenir un bon model per estudiar la influència dels SNP en els estudis GWAS té gran interès, citant entre altres possibles aplicacions les següents:

- Estudis eQTL pròpiament dits: determinar quins gens estan més influenciats per determinats conjunts de característiques del patró SNP
- Prediccions individuals de la intensitat d’expressió d’uns determinats gens amb la informació del seu patró SNP.
- En estudis per determinar l’efecte d’un factor determinat sobre l'expressió genòmica, poder ajustar l’efecte observat a l’expressió esperable segons els patrons SNP individuals.

Tot i que en l’article citat, el mètode es mostrava superior als altres, es tracta d’un únic estudi, en una única mostra, d’una única espècie. Donat el gran nombre de factors coneguts, i probablement també de desconeguts en aquest moment, que poden modificar l’expressió genètica, s’hauria de comprovar que el mètode manté les seves bondats en la capacitat de resumir i extreure la informació dels SNPs en un àmbit més general.

Per tal fita, s’hauria d’aplicar a altres set de dades disponibles i constatar que manté les bondats descrites. Però el citat estudi, tot fer una introducció interessant de l’evolució dels estudis eQTL, tant els clàssics com els nous, i d’una explicació detallada de les bases tècniques dels denoising autoencoders, tan sols fa una descripció molt superficial del model de MLP-SAE utilitzat, sense entrar en detalls de la seva geometria, dels hiperparàmetres utilitzats (a excepció del learning rate), ni tampoc aportar el codi font per a la reproducció de l’estudi.

Per aquest motiu s’ha trobat interessant recrear un model MLP-SAE per intentar reproduir el treball, i obtenir una eina pròpia per aplicar a altres sets de dades per a estudiar l’aplicabilitat més general de la metodologia.

1.2 Objectius del Treball

1. Reproduir el model de l'article, creant un model MLP-SAE propi. Aquest es segmenta en tres fites individuals.
   1.1. Crear un model *denoising auto-encoder* per a representació de patrons de *SNP*.
   1.2. Crear un perceptró multicapa amb l'ús de denoising auto-encoder pre-entrenat en les capes internes (*Multilayer Perceptron and Stacked Denoising Auto-Encoder* [*MLP-SAE*]) per a la predicció del patró de *HTSeq counts* amb el patró de *SNPs*. Implementar el mètode de *drop-out* com a mitjà de control del *overfitting*.
   1.3. Reproduir el model predictiu amb les dades de l'article original i crear el mètode d'avaluació.
2. Seleccionar altres set de dades apropiats per reproduir l'estudi: Identificar-los entre les fonts open-data segons si fossin aplicables a models de *deep-learning* d'estudis *GWAS* entre el patró de *SNPs* i l'expressió genètica en forma de *HTSeq counts.*
3. Aplicar el model generat al nou grup de dades identificades i avaluar si es manté la capacitat predictora mostrada en les dades de l'estudi original.

1.3 Orientació i mètode seguit.

Estratègia de desenvolupament:

L'estratègia per la realització del treball s'ha basat en la separació del mateix en 3 fases diferenciades, que corresponen als tres objectius marcats: reproduir el model MLP-SAE, localitzar nous set de dades i aplicar el model en les noves dades per a valorar la generalització del mètode.

De les nombroses estratègies de desenvolupament d'un model informàtic, els factors que s'ha tingut en compte a l'hora de plantejar una estratègia de desenvolupament han estat les següents:

- Estructura de l'equip desenvolupador: hi ha estratègies de desenvolupament en equip, com per exemple la metodologia *Scrum*, i altres adaptables a treball individual. En ser aquest un projecte desenvolupat de forma personal, el mètode ha de ser apropiat per l'ús individual.
- Background i experiència del desenvolupador: l'experiència com a desenvolupador, i el possible coneixement d'eines i frameworks es limiten en aquest cas a les utilitzades durant el seguiment del programa de Màster. Tot i existir eines que poden aportar beneficis al desenvolupament del projecte, la corba d'aprenentatge requeriria de part dels temps disponible per l'execució del projecte. Així doncs s'han reduït el nombre d'eines i marcs de treball nous als imprescindibles, per controlar el temps no destinat de forma directa a l'assoliment dels objectius.

Tenint en compte aquests darrers punts, s'ha optat per la següent estratègia de treball:

La primera fase es dedica a la rèplica d'un model informàtic per estudis *GWAS* amb l'ús d'autoencoders similar al descrit en l'article de referència.

El desenvolupament de la reproducció del *Multilayer Perceptron-Stacket Denoising Auto-Encoder (MLP-SAE)* de l'estudi, es realitzarà per fases de forma incremental. El primer pas preparatori de pre-processar les dades, és la implementació d'un *auto-encoder* basat en xarxes neuronals per a la reproducció dels patrons de *SNP*.

En l'ús del model MLP-SAE per a estudis GWAS hi ha l'objectiu d'aprofitar les capacitats dels autoencoders de reduir la dimensionalitat de les dades, tot preservant les característiques més informatives d'aquestes. Sense menystenir l'interès de les aplicacions dels estudis *GWAS*, l'interès de l'autor del treball se centra en la utilitat dels autoencoders per a obtenir models de dimensió reduïda de les dades òmiques. Aquest interès es dóna per les múltiples possibles aplicacions d'aquests models, més enllà dels estudis *GWAS*. Per aquest motiu, l'obtenció d'un model de *denoising-autoencoder* s'ha marcat com la primera fita.

Per aconseguir el model MLP-SAE s'utilitzarà una metodologia incremental de complexitat. S'aniran afegint els diferents components del model pas a pas. D'aquesta manera es poden identificar i corregir més fàcilment les errades comeses.

D'altra banda, tenint en compte que durant la realització del treball s'ha hagut de familiaritzar-se amb noves eines i entorns per al seu desenvolupament, el model basat en increments cíclics de complexitat permet també desglossar el treball d'aprenentatge en tasques més senzilles. Així es podrien realitzar de forma més àgil cicles de documentació-implementació-depuració-millora, amb assoliment de fites intermèdies corresponents amb models parcials funcionats.

La segona i tercera fase estaven més relacionades entre si, ja que la realització de l'última depenia dels assoliments de la segona. En conjunt, durant la segona i tercera fase s'haurien de localitzar altres set de dades per reproduir l'estudi, obtenir un model amb aquestes dades, i comparar els resultats obtinguts dels diferents sets. Les tasques s'havien separat en dos per la diferent naturalesa de les subtasques i la possibilitat de compaginar-les.

La segona fase era de recerca de sets de dades alternatives. L'estratègia plantejada era l'ús de portals de dades òmiques obertes, i si no s'obtenien resultats, es faria una recerca manual d'articles d'estudis *GWAS*. Aquesta tasca no requeria ni de la finalització de la primera, ni era imprescindible la seva compleció per iniciar la tercera, un cop s'haguessin recuperat algun set de dades alternatius per iniciar-la. També permetia optimitzar el temps en què no es tingui accés a l'equip on es realitzava el desenvolupament del codi i l'entrenament del model, ja que es podia realitzar amb qualsevol dispositiu amb accés Web. Però donada que la seva utilitat estava supeditada a poder aplicar les dades recuperades sobre un model MLP-SAE, no es plantejava iniciar-la fins a haver aconseguit un primer model MLP-SAE funcionant.

En la tercera fase es preprocessaven els sets de dades obtinguts per poder-los utilitzar en el model realitzat. El processament de dades s'iniciaria un cop finalitzada la primera fase de reproducció del model de l'estudi. D'aquesta manera se centraria la prioritat a obtenir el model en cas de no complir-se el calendari per qualsevol motiu, perquè no tenia sentit iniciar la preparació d'altres dades si posteriorment no es poguessin aplicar.

Entorn de desenvolupament:

Les eines elegides pel desenvolupament i entrenament de les xarxes neuronals són aquelles ofertes per l'entorn Keras, utilitzant les funcionalitats de Tensorflow, combinació amb un gran desenvolupament, corba d'aprenentatge curta i amb bon rendiment amb l'ús de GPU de classe Nvidia. Donat que són de forma nativa eines basades en Python, i la manca de coneixement suficient d'aquest llenguatge, s'utilitzarà el front-end ofert pel paquet Keras per a R, amb l'entorn virtual per a Python instal·lat per defecte, especificant l'opció d'entorn per a GPU.

L'entorn de desenvolupament ha estat basat en:

- Sistema operatiu: Ubuntu Linux 18.04.1, x86_54,.kernel 4.18.0-20-generic
- Llenguatge R versió 3.6.0 (2019-04-26)
- R-Studio 1.1.463
- Paquet Keras per R versió 2.2.4
- Paquet Tensorflow per R versió 1.10
- Reticulate versió 1.11.1, executant l'entorn virtual estàndard «r-tensorflow» per a GPU, instal·lat per defecte pel paquet Keras. Aquest utilitza Python 2.76.
- OpenSSH versió 7.6: client-servidor per a l'operació remota de la màquina de desenvolupament. S'utilitza per poder entrenar models de proves durant el desenvolupament, quan per qualsevol motiu no es tingui accés físic a l'equip on es realitza el treball.
- Llibreries CUDA 9.0.
- Gestor de control de versions: paquet «git» per Linux versió 2.17.1, gestionat pel connector del gestor de projectes de R-Studio. Aquesta elecció més simple del control de versions i còpies de seguretat s'ha triat sobre sistemes amb sincronització al núvol atès el criteri de reduir al màxim el temps dedicat a aprendre l'ús de noves eines de desenvolupament. També es té en compte que hi ha un sol desenvolupador i que el treball es du a terme sobre una mateixa màquina, no sent necessaris sistemes de control de versions amb més prestacions. Respecte a la còpia de seguretat per possibles fallades de l'equip, s'utilitza la funcionalitat de sincronització d'arxius al núvol de l'empresa DropBox, amb el seu connector per Nautilius de Linux.
- LibreOffice 6.0.7: paquet d'ofimàtica per a la redacció de les memòries, realització d'esquemes i preparació de la presentació.

- Kdenlive 17.12.3: per a la preparació i renderitzat del vídeo de la memòria.
- Kazam 1.4.5: per a la captura de pantalla per al vídeo de presentació.
- GIMP 2.10.10: per a petits retocs d'imatge i ajuda a la composició de les animacions de la presentació.

Maquinari:

- CPU: AMD Ryzen Threadripper 1920X
- Nuclis / fils d'execució: 12 / 24
- Memoria del sistema: 64 Gb.
- Emmagatzematge: SSD NVMe PCIe 2.0 1 Tb.
- GPU: Nvidia GTX 1070
- Memòria GPU: 8 Gb.
- Nuclis CUDA de GPU: 1920

1.4 Planificació del Treball

Donada la combinació del treball amb la jornada laboral, s'havien planificat les tasques assumint una dedicació mitjana de 4 hores diàries.

Planificació de tasques:

- PAC0 - Definició dels continguts del treball (13 dies)
- PAC1 - Pla de treball (14 dies)
- PAC2 - Desenvolupament fase 1 (37 dies)
  - Objectiu 1: reproducció de l'estudi (37 dies)
    - Pre-porcessat dades originals (1 dia)
    - *Denoising-Autoencoder* (5 dies)
    - Primer model *MLP-SAE* (10 dies)
    - Pre-entrenament *SAE* (5 dies)
    - *Drop-Out* (5 dies)
    - TEST i valoració model definitiu (5 dies)
    - Imprevistos (6 dies)
  - Objectiu 2: Localització nous conjunts de dades (30 dies) Inici.
- PAC3 - Desenvolupament fase 2 (26 dies)
  - Objectiu 2: Localització nous conjunts de dades (30 dies) Final.

- Objectiu 3: Avaluació amb dades alternatives (18 dies)
  - Creació *pipeline* pre-processat (6 dies)
  - Test de dades noves *Denoising-Autoencoder* (5 dies)
  - Test de dades noves *MLP-SAE* (5 dies)
  - Valoració rendiment en conjunt (2 dies)

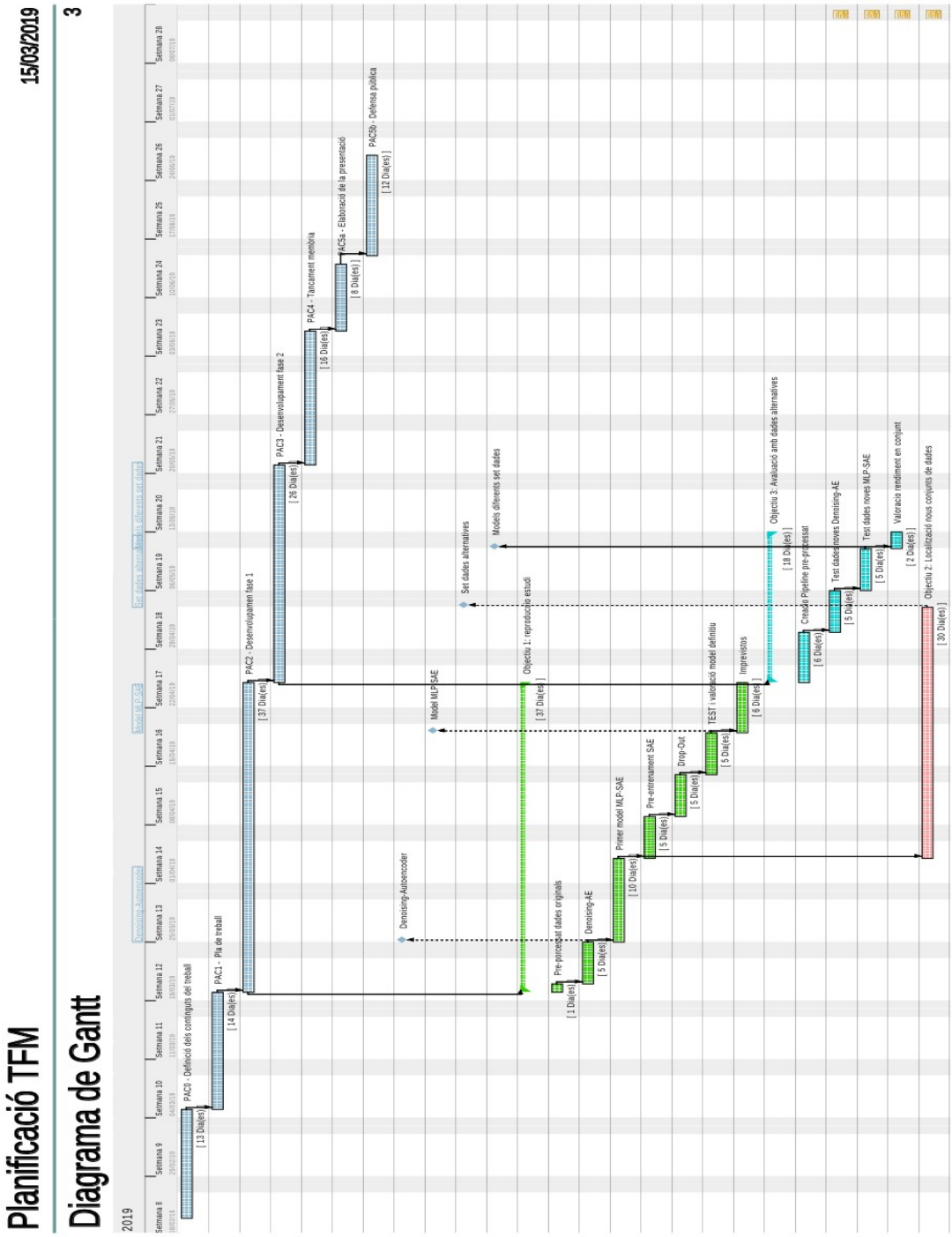
- PAC4 - Tancament memòria (16 dies)
- PAC5a - Elaboració de la presentació (8 dies)
- PAC5b - Defensa pública (12 dies)

Fites:

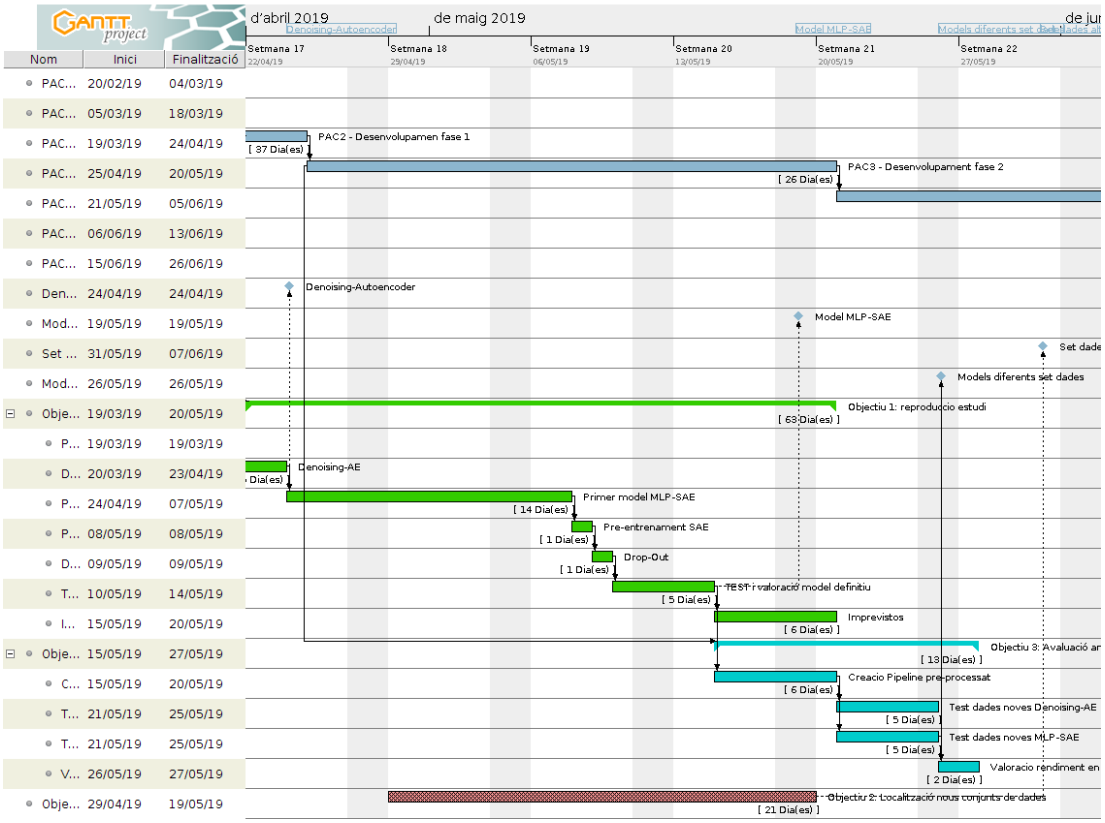
- Definició general dels objectius. **PEC0**: entregat el 4/3/2019
- Pla de treball. **PEC1**: 18/3/2019
- Un **model *Denoising-Autoencoder***: 25/3/2019
- Un **model *MLP-SAE* definitiu** amb les dades originals: 19/4/19
- Memòria seguiment **PEC2**: 24/4/2019
- Un conjunt de **dades alternatives**: 4/5/2019
- Conjunt de **models *MLP-SAE* alternatius**: 11/5/19
- Memòria seguiment **PEC3**: 20/5/2019
- Entrega de la **memòria TFM** PEC4: 5/6/2019
- Elaboració de la presentació. **PEC5a**. 13/6/2019
- Defensa pública. **PEC5b**. Fins al 26/6/2019

Calendari:

El calendari inicialment planificat es el mostrat en la figura 1. Donats els endarreriments produïts durant el desenvolupament de la primera fase del projecte, es va replanificar el treball tal com es mostra en la figura 2. Finalment les fases 2 i 3 han estat cancel·lades pels motius que seran exposats posteriorment.



Il·lustració 1: Esquema Gantt de la planificació original del projecte



Il·lustració 2: Diagrama de Gantt del cronograma modificat del projecte

1.5 Breu sumari dels productes obtinguts

- **Memòria del treball:** El document present, elaborat amb el paquet LibreOffice 6.0.3, i convertit a PDF amb la mateixa eina.
- **Fitxer amb el codi i resultats:** Fitxer comprimit que conté els següents components:
  - *Codi R:* diversos fitxers interdependents. Un conté el codi pel preprocessament del fitxer de dades facilitades pels autors de l'article. Els altres fitxers són necessaris per a generar el 5-fold cross-validation del model, i generar els gràfics i càlcul de les dades exposades en la memòria.
  - *Dades originals*: recopilades del repositori de l'article original.
  - *Resultats:* matriu d'expressió HTSeq original, depurada i amb les imputacions dels valors perduts, i la matriu d'expressió estimada pel model. També inclou els gràfics generats i el fitxer Rdata amb els resultats dels càlculs exposats en la memòria.
- **Presentació virtual**

1.6 Breu descripció dels altres capítols de la memòria.

2 Model MLP-SAE:

2.1 Breu descripció dels autoencoders.

Introducció als autoencoders, el seu ús i els models MLP-SAE. Conté els següents subapartats.

2.1.1 Problema de la dimensionalitat i el feature engineering.

2.1.2 L'ecosistema dels autoencoders.

2.1.3 Denoising autoencoder.

2.1.4 MLP-Satcked autoencoders.

2.2 Planificació del desenvolupament.

Descripció de la planificació inicial de desenvolupament del projecte.

2.2.1 Segmentació dades.

2.2.2 Estructura del codi i directoris.

2.3 DAE

Descripció del desenvolupament del model de denoising autoencoder. Consta dels següents subapartats.

2.3.1 Preprocessat de les dades.

2.3.2 Creació AE.

2.3.3 Creació DAE.

2.3.4 Ajust hiperparàmetres.

2.3.5 DAE final.

2.3.6 Stacked Autoencoder.

2.4 MLP-SAE

Creació del model MLP-SAE. Conté el subapartat següent:

2.4.1 Creació model MLP-SAE.

Descripció de la creació i ajust del model final de MLP-SAE.

2.5 Avaluació del model: 5-fold cross-validation.

Descripció de la creació de l'avaluació per validació creuada amb 5 iteracions.

2.6 Resultats

Exposició dels resultats del model.

2.7 Discussió

Discussió dels resultats.

3 Conclusions.

Exposició de les conclusions del treball i autoavaluació de l'execució del projecte. Se separa en els següents subapartats:

3.1 Conclusions del treball.

3.2 Assoliments dels objectius.

3.3 Seguiment, planificació i metodologia

3.4 Punts pendents.

Tasques pendents i suggeriments de continuació del treball.

4 Glossari.

5 Bibliografia.

6 Annexos

S'inclouen els diferents fitxers de codi font utilitzat durant el projecte.

6.1 Codi font.

6.1.1 preprocessament_i_imputacio.R.

6.1.2 mlp_sae_5fold_CV.R.

6.1.3 crear_entrena_avalua.R.

6.1.4 crear_model_mlpsae.R.

6.1.5 entrenar_MLPSAE.R.

6.1.6 avaluar_MLPSAE.R.

6.1.7 funcions_suport.R.

6.1.8 directoris_i_funcions.R.

2 Model MLP-SAE

2.1 Breu descripció dels autoencoders

Per al desenvolupament del treball, a banda dels coneixements generals que es puguin tenir en el camp del machine learning i deep learing, s'han d'assolir un coneixement mínim de l'eina que es triï per al desenvolupament del model de MLP-SAE, i dels conceptes generals dels autoencoders.

Per a assolir uns coneixements suficients per desenvolupar-se en el nou entorn i la forma de treballar de l'eina Keras és recomanable una obra específica. En aquest cas, s'ha utilitzat un llibre d'un dels desenvolupadors de Keras, que no solament ofereix una introducció progressiva a les capacitats de l'entorn, sinó que serveix també per revisar els problemes que un se sol trobar en el processament de dades d'alta dimensionalitat com el cas de les dades òmiques (6).

El llibre és centrat en les xarxes neuronals en general. Per obtenir una visió suficientment ampla dels autoencoders, les seves característiques, usos i limitacions, ens podem centrar en una revisió de D. Charte (7), i en l'estudi sobre les aplicacions per la reducció de la dimensionalitat de P. Vincente i G.E. Hinton (8)(5).

A continuació es resumeixen els conceptes útils per al desenvolupament d'aquest treball tractats en les obres citades (5–8).

2.1.1 Problema de la dimensionalitat i el feature engineering

El problema de la dimensionalitat (o maledicció de la dimensionalitat) és generalment atribuït al matemàtic nord-americà Richard E. Bellman. En els treballs que desenvolupà sobre programació dinàmica i mètodes numèrics per a resolució de problemes, va constatar com a problema important l'increment de dimensions. Aquest fa créixer el volum de l'espai de solucions de forma exponencial, i fa necessari incrementar el nombre d'observacions i el temps de càlcul.

Aquest augment del volum de l'espai de representació també porta a una matriu de representació de les observacions en la que el recompte d'observacions de cada categoria és escassa i conté moltes categories sense cap observació. Les matrius amb aquestes característiques s'anomenen *sparse matrix*, i comporten una sèrie de problemes en el seu tractament matemàtic que queden fora de l'abast d'aquesta introducció.

Les variables o característiques de les observacions s'anomenen genèricament amb el terme anglès *feature*. El problema de la dimensionalitat de les dades i l'aparició de les *sparse matrix* es pot mitigar construint una representació millorada de les observacions.

El *feature engineering* és el conjunt de mètodes per a transformar aquestes representacions en noves, mitjançant la manipulació dels *features*. Aquesta manipulació pot ser manual, guiada pel criteri dels experts, o automatitzada per algun algoritme, com són capaces de fer les xarxes neuronals.

En resum, els *features* d'una observació, es podrien seleccionar eliminant aquells que aportin informació irrellevant o repetida (*feature selection*), es poden representar d'una forma nova mitjançant la combinació de diversos *features*, amb tècniques diverses com l'anàlisi de components principals entre d'altres. Aquests mètodes poden ser tècniques supervisades o no supervisades (*feature extraction*), i es poden combinar per mètodes no lineals i altres algoritmes d'aprenentatge de màquina per eliminar informació redundant (*feature fusion*). El procés de la informació de les xarxes neuronals entraria dintre d'aquesta última definició.

Del procés de selecció o combinació dels *features* per mètodes de *machine learning*, també se'ls anomena en la literatura *feature learning*, o *representation learning*.

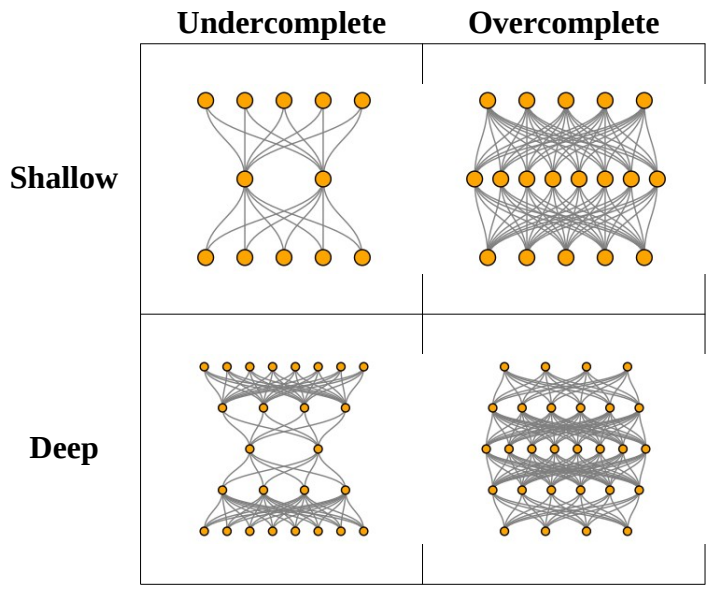
2.1.2 L'ecosistema dels autoencoders.

Els autoencoders són metodologies per reduir la dimensionalitat de les dades conservant-ne la informació. Tot i que no tots els autoencoder són xarxes neuronals, sí que les xarxes neuronals són una de les estructures de les més utilitzades per a la seva creació.

En el cas dels autoencoders basats en xarxes neuronals, la seva estructura es basa en una o més capes de neurones en les quals la capa de sortida ha de ser idèntica a les dades d'entrada. Si la funció d'activació fos lineal, el sistema no diferiria massa d'un sistema algebraic lineal. Però l'ús de funcions d'activació no lineals permet al model ajustar-se a distribucions de les dades en espais no lineals millor que aquests darrers.

En les capes internes del model s'hi conforma una representació de les relacions dels diferents *features* de les dades d'entrada, produint-se de forma automatitzada i no supervisada un procés de *feature selection* i *feature fusion*.

Els models es poden classificar d'una forma genèrica en dos eixos: el nombre de capes del model, sent un model «*shallow*» si existeix una sola capa oculta, i «*deep*» si hi ha més d'una capa oculta. L'altre eix és la mida de les capes ocultes internes. Si aquestes presenten menys unitats del nombre de dimensions del tensor d'entrada, es considera un model «*undercomplete*», mentre que els que un nombre major d'unitats són anomenats models «*overcomplete*».



Taula 1: Esquema de la classificació de les autoencoders basats en xarxes neuronals segons les seves dimensions.

2.1.3 Denoising autoencoder

El denoising auto-encoder és un model en el qual les dades d'entrada són corrompudes amb l'addició d'algun tipus de senyal aleatòria, previ a ser processades pel model. Aleshores el model és ajustat per a reproduir les dades d'entrenament sense modificar.

Aquest senyal és el soroll afegit a les dades, que dificulta que el sistema pugui entrenar-se basat en peculiaritats de les dades d'entrenament, i facilita el seu ajust a característiques més generals d'aquestes. És un mètode útil per al control del sobreajust.

2.1.4 MLP-Satcked autoencoders

Les xarxes neuronals profundes, amb diverses capes internes, tenen a priori més capacitat de processar les dades i assolir representació de nous *features* més útils. Però el problema que presenten és que a major mida del model, més depèn de què els pesos inicials d'aquest siguin més favorable per la representació de les dades (5).

El disseny dels autoencoders els permet obtenir una representació interna de la informació de les dades. La utilitat d'aquesta és el seu possible ús com a representació alternativa de la informació de les dades originals.

En els autoencoders apilats, aquesta representació alternativa d'un primer autoencoder, és utilitzada com a dades d'entrada per l'entrenament d'una segona versió d'autoencoder que s'ajusta a reproduir el conjunt de dades originals a partir de la representació alternativa obtinguda del primer autoencoder.

El que permet aquest model és realitzar un entrenament progressiu, obtenint uns pesos de les capes corresponent al primer autoencoder, convergeixin a una solució

més bona que si l'entrenament fos amb tot el model sencer. Progressivament es van entrenant les successives capes, preservant en un primer moment els pesos aconseguits per la fase anterior del procés d'ajust del model.

D'aquesta manera es pot escapar progressivament d'entrenar un model tipus «*deep*» amb un conjunt de pesos inicials que convergissin a un mínim local lluny de la representació òptima de la informació.

Un perceptró multicapa és un model de xarxa neuronal artificial amb una funció d'activació no lineal, i més d'una capa interna. En el model MLP-SAE, aquestes capes primeres capes del model estan compostes per un model d'autoencoders apilats, i una o més capes de sortida que utilitza la representació final dels autoencoders com entrada. Això permet entrenar un model de perceptró de forma progressiva, evitant el problema de la dependència de l'estat inicial per a convergir a un estat amb un bon rendiment.

Com s'ha anomenat anteriorment, el treball de Xie et al. ha utilitzat aquest últim model aplicant-lo a un estudi GWAN per la predicció de l'expressió genètica en funció del patró de SNPs. L'objectiu d'aquest treball és la reproducció d'un model similar i avaluar-ne l'aplicabilitat més general.

2.2 Planificació del desenvolupament

2.2.1 Segmentació dades

Per al desenvolupament del treball s'ha realitzat la segmentació de les dades amb dos sistemes diferents, segons la fase del treball:

- **Desenvolupament**: duran la fase de desenvolupament s'ha creat una segmentació aleatòria de les observacions en **3 grups**: **entrenament** dels models parcial,, **validació** durant els cicles d'entrenament, i un grup de **test**, per a l'avaluació final del rendiment. De les 112 observacions, 22 es van destinar al grup de test, 16 al de validació i 72 per a l'entrenament.
- **Avaluació**: l'avaluació s'ha realitzat amb tot el conjunt d'observacions disponibles del model, utilitzant uns processos de ***5-fold cross validation***. Per a cada iteració s'entrena de forma completa el model, i s'obtenen els paràmetres del rendiment i la predicció dels valors que s'han deixat fora del grup d'entrenament. D'aquesta manera s'ha pogut estimar els paràmetres del rendiment amb un grup d'entrenament més gran i reconstruir la matriu d'observacions de forma completa.

2.2.2 Estructura del codi i directoris

La planificació inicial del treball contemplava segmentar el treball en subtasques. Cada subtasca es desenvoluparia en un subdirectori propi per mantenir el treball ordenat. Al final del procés de desenvolupament, la línia de treball consistiria a executar de forma ordenada el codi de cada subdirectori.

Aquesta planificació ha estat variada a causa del problema «O*ut Of Memory – OOM*». En l'entrenament repetit de diversos models, es produïa un error de falta de memòria en l'entorn de Python, tot i esborrar els grafs de representació del model Keras. L'única solució trobada per a evitar aquest error ha estat inicialitzar l'entorn de Python cada cop que s'ha d'entrenar un model. Per tal de fer això, s'ha organitzat el codi que crea i entrena el model en uns fitxers de codi separats, i s'executa en un procés nou de R. L'intercanvi de dades es fa a través de fitxers temporals.

Tot i que s'hauria pogut mantenir l'estructura inicial del projecte, el sistema de resoldre l'error portava a una segmentació del codi, complicant la seva lectura. Això queda lluny de la idea de la divisió en subtasques, que era la de simplificar els arxius del codi per a que fossin més fàcils de llegir i interpretar quina tasca realitzaven.

Tot i que el treball s'ha desenvolupat en aquesta estructura de directoris per l'avantatge de poder mantenir aïllades cada subtasca durant les proves, el codi final presentat es manté en un únic directori.

El codi final s'organitza en un sol directori de la següent forma:

- **dades_originals**: directori on es contenen les dades de l'article original, tal com es poden obtenir del repositori de Github de l'autor. També s'hi desaran les dades preprocessades que es generen, per a un ús posterior.
- **directoris_i_funcions.R**: fitxer auxiliar on es defineixen els noms dels fitxers i directoris utilitzats, i carrega el conjunt de funcions que es reutilitzen en els diferents fitxers de codi.
- **funcions_suport.R**: fitxer amb funcions auxiliars.
- **preprocessament_i_imputacio.R**: preprocessat de les dades originals. Genera els arxius amb les dades preparades per a l'entrenament del model.
- **mlp_sae_5fold_CV.R**: realitza el 5-fold cross-validation del model. Realitza i desa els càlculs de l'avaluació del model i els gràfics per a la memòria.
- **crear_entrena_avalua.R**: arxiu per crear, entrenar i avaluar cada una de les iteracions del model.
- **crear_model_mlpsae.R**: crea amb Keras el graf que representa el model MLPSAE, i els models parcials necessaris per al preentrenament.
- **entrenar_MLPSAE.R**: realitza l'entrenament del model.
- **avaluar_MLPSAE.R**: calcula el rendiment i la matriu predita del model entrenat.

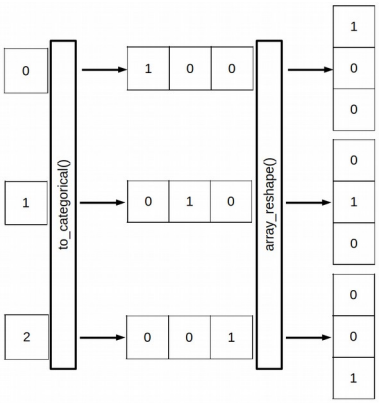
2.3 DAE

2.3.1 Preprocessat de les dades

Preprocessat dels SNPs

Es carrega la taula original, eliminant la primera columna identificativa. Cada variant es codifica com 1 de 3 nivells d'un factor (0-2). S'obté un tensor de dimensions (112,2956). Per a l'ús en el model es fa una recodificació mitjançant la funció de Keras «*to_categorical()*», a variables *dummy*, un per a cada categoria del factor, codificant la presència o absència de l'al·lel. Finalment, aquest nou tensor de

representació de les variants SNPs, de dimensions (112,2956,3), es recodifica reduint-ne les dimensions a 2, amb un tensor final de dimensions (112,8868).



Il·lustració 3: Recodificació dels patrons SNPs

Preprocessament de les dades *HTSeq counts i tractament dels NA*.

Les dades HTSeq counts estan expressades, segons l'apartat de mètodes del treball original que les va generar (9), com el logaritme normalitzat de la ratio entre el número de recompte de còpies d'un gen determinat per la lectura de la mostra de referència:

$$\log_2\left(\frac{MOSTRA}{REFERENCIA}\right)$$

Així doncs són dades que ja han estat normalitzades, i no s'ha realitzat cap altre processat en aquest punt.

Respecte als valors perduts, hi ha prop d'un 11% de valors perduts en les lectures HTSeq. D'aquestes, una bona part pertanyen a gens dels que no hi ha cap dada disponible, i que han set conseqüentment eliminats.

Tenint en compte els gens, la mediana de valors perduts és del 1.8%, mentre que si tenim en compte les mostres, aquestes presenten una mediana de valors perduts és del 9,7%.

El tractament dels valors perduts és un camp ampli i lluny de tenir una resposta única. Dels nombrosos texts que es poden prendre de referència, s'ha triat un blog especialitzat (10), que presenta un resum bastant actualitzat de les principals solucions. També explica els seus requeriments i limitacions. El text no entra detalls, però ofereix prou informació per a poder cribrar les diferents opcions, estalviant per tant, tecnicismes innecessaris.

Per poder tractar els valors perduts, s'ha hagut d'explorar les dades per obtenir resposta a les següents preguntes, per triar l'aproximació més adient:

- Tenen els valors perduts una distribució aleatòria i uniforme?
- Té la variable una distribució normal?

- Són els individus un grup homogeni on es pugui assumir igualtat de variàncies?

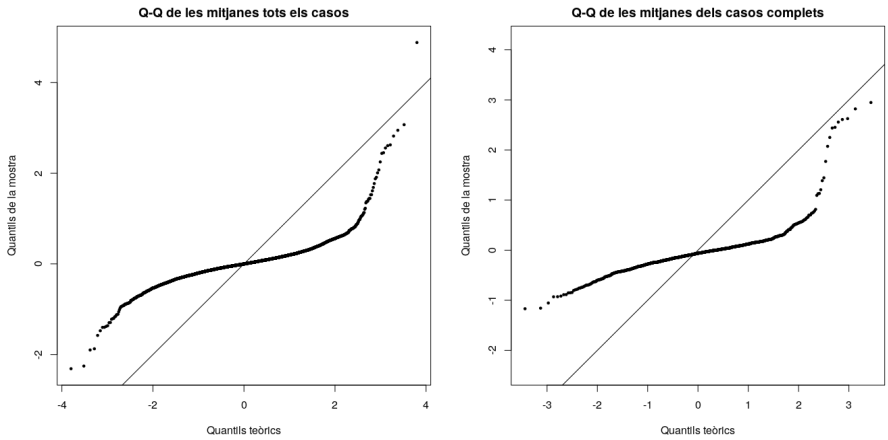
El fitxer de dades, original, de 7085 gens, 1706 presentaven tots els 112 valors, i 191 no en tenien cap de registrat.

Quan es tenen en compte el conjunt de lectures independentment del gen, i es calcula el *Skewness* i la *kurtosis*, s'obté els següents resultats:

Tots els gens: Kurtosis=27.4985 Skewness=0.9146.

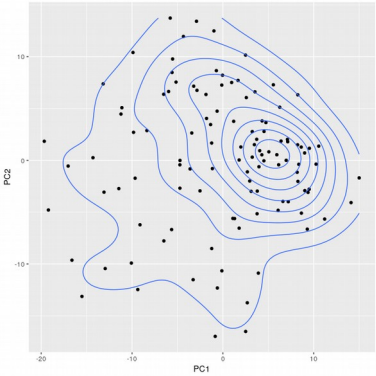
Sol gens amb tots els valors: Kurtosis=38.67 Skewness=3.0408.

Les dades de tots els gens presenten una marcada *kurtosis,* amb un excés de valors en els extrems de la distribució, com es pot observar en el gràfic Quantil-Quantil de les mitjanes d'expressió de cada gen (il·lustració 4):



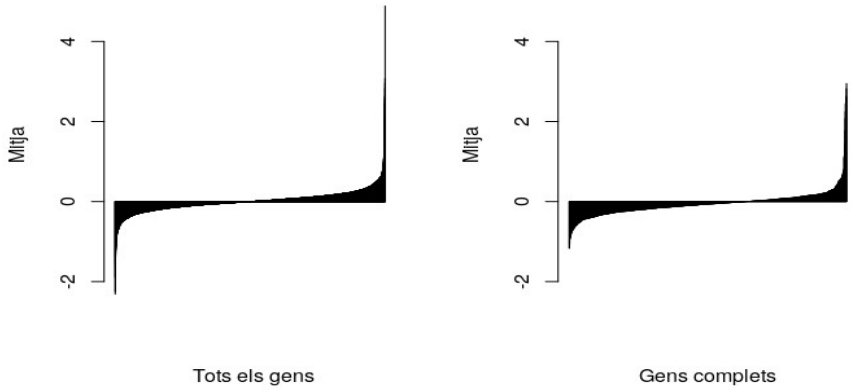
Il·lustració 4: Gràfics Quantil-Quantil de les mitges d'expressió de cada gen. Es mostra el de tots els gens, i el dels gens que no tenien cap valor perdut. Es pot observar la perdua de simetria de la distribució quan sol es representen els gens amb cap valor perdut.

Si es realitza un test de distribució normal multivariant, també es descarta la distribució normal, sobretot en el cas que se seleccionin els gens amb tots els valors. En aquest cas, realitzant una projecció de les mostres sobre $\Re^2$, mitjançant els components principals dels gens, s'observa que la distribució no és uniforme, presentant un grup central d'expressió, però amb una dispersió d'individus asimètrica.

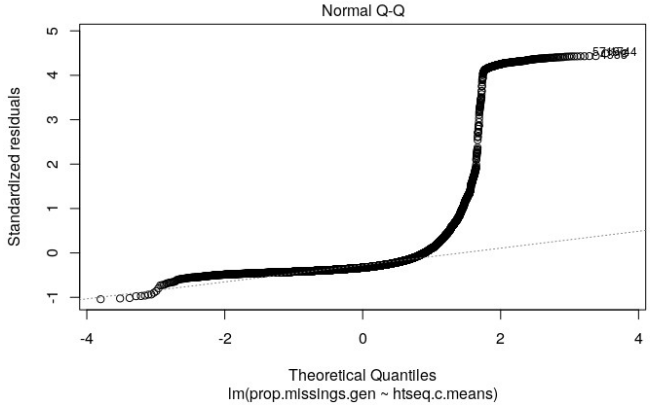


Il·lustració 5: Anàlisi PCA de les mostres amb els gens amb tots els valors complerts. Presenten una distribució asimètrica.

Si es comparen les gràfiques de les mitjanes ordenades dels HTSeq de tots els gens envers els gens complets, es manifesta amb claredat que la probabilitat de tenir un valor incomplet no es distribueix de forma uniforme (Il·lustració 6). En un model lineal de percentatge de valors perduts en funció de la mitjana d'expressió, es pot observar en el gràfic Q-Q una desviació de la distribució normal en l'anàlisi de residus estandaritzats (il·lustració 7).



Il·lustració 6: Mitjana ordenada de l'expressió dels gens. A la esquerra la representació de tots els gens. A la dreta la representació dels gens amb tots els valors. S'observa que varia la proporció de gens segons l'expressió major o menor de 0.



Il·lustració 7: Gràfic Q-Q dels valors dels residus estandarditzats d'un model de regressió lineal entre la proporció de valors perduts d'un gen i la seva mitja d'expressió. Es pot observar una desviació asimètrica de la distribució normal molt important en els valors més alts de les mitjanes.

D'aquestes anàlisis en podem concloure:

- Els individus observats presenten 1 grup principal d'expressió, però amb distribució asimètrica, no normal dels individus que se separen de la centralitat.
- No es pot assumir que els valors perduts es distribueixin de forma aleatòria i uniforme.
- Les variables no presenten una distribució normal.

En conseqüència:

- L'opció d'eliminar els gens amb valors perduts i realitzar l'anàlisi amb la resta pot introduir un biaix en el model, i independentment de la pèrdua d'informació, no és una solució acceptable. El més apropiat és una imputació de valors.
- El sistema d'imputació de valors ha de ser insensible a la distribució no homogènia dels valors perduts.
- El sistema d'imputació de valors no ha de dependre de conèixer la distribució de les variables a imputar.

Dels sistemes d'imputació, que s'han valorat, s'ha elegit finalment un sistema basat en random forest que ha mostrat certa superioritat (11) davant altres sistemes, tot i ser un sistema de certa exigència quant a temps de computació. Aquest sistema està disponible en forma del paquet de R *missForest*.

Aquest sistema es basa a crear un eixam de models predictors, en aquest cas arbres de decisions, creats a partir de la selecció aleatòria d'una submostra de la resta de variables. Aquestes prediccions s'incorporen a la matriu de dades, i es torna a fer una nova predicció dels valors perduts.

D’aquesta manera s’entra en un cicle de construcció d’arbres i generació de noves prediccions, que s’espera que convergeixin progressivament cap a una predicció més precisa. Aquests cicles es repeteixen fins a un número límit que s’hagi marcat, o fins que les prediccions deixen de millorar.

La utilització d’un sistema basat en random forest presenta com a avantatge addicional, que el sistema pot fer una estimació de l’error comès en la predicció, tant de forma global com en cada variable de forma individual.

Com últim comentari sobre aquest sistema, és que, com es mostra en la publicació de Steckhoven et al (11), la bondat del sistema redueix amb la proporció de valors perduts que s’ha d’imputar, sent pobre per imputacions per sobre del 30% de les observacions.

Així doncs la imputació del model s’ha dut a terme de la següent manera:

1. Eliminació dels gens amb més d’un 30% de valors perduts.
2. Imputació dels valors perduts amb el paquet *missForest*, amb la capacitat de paral·lelitzar la construcció dels *random forest* que porta ja implementada *missForest.* Per això es requereix el paquet *parallel.* Les opcions utilitzades foren les següents:

```
missForest(
  matriu.expressio.original.missings,
  variablewise = TRUE,
  verbose = TRUE,
  parallelize = "variables",
  maxiter = 24)
```

3. Eliminació dels gens que presentessin un error en l’estimació major d'1 desviació estàndard.

2.3.2 Creació AE: model

El primer pas triat cap a la construcció d’un model MLP-SAE, va ser la creació d’un model d’autoencoder senzill funcionant, i avaluar-ne els paràmetres millors.

Respecte a la geometria del model d’autoencoder, es va decidir iniciar amb el model més clàssic de *undercomplete* autoencoder, que és el presentat pel model de l’article de referència. Altres configuracions es van provar en una fase més avançada del desenvolupament, però no van ser factibles per motiu que la GPU utilitzada no disposava de suficient memòria RAM.

Un dels paràmetres més importants a considerar, és la mida de la capa oculta que suposarà. Com ja s’ha comentat, el model d’autoencoder intenta regenerar la informació d’entrada en la seva sortida. En els models undercomplete, la capa interna de menor mida representa un espai latent de menor dimensió, on la informació que contenen les dades d’entrada s’hi representa amb la menor pèrdua d’informació possible. Aquesta reducció forçada de dimensionalitat és el que porta a fer que es produeixi el *feature fusion.*

Per tant, la selecció de la mida de les capes internes, en especial de la de menor mida és especialment important, ja que ha de ser suficientment gran com per poder representar la informació continguda en l'entrada.

La representació d'entrada del model, per qüestions pràctiques de construcció del model i la seva base de funcionament d'acord amb el càlcul matricial, ha estat expandida durant el procés de preparació de les dades. Així doncs, la informació d'un locus de polimorfisme, que era contingut en una unitat d'informació mitjançant un factor de valors 0-1-2, ha estat expandida a l'ús de 3 unitats d'informació. Intuïtivament s'entén la possible reducció de la dimensionalitat d'almenys en un factor 3. Però decidir la representació mínima on podem suposar que encara es pot mantenir la majoria d'informació és un tema complicat. La teoria de la informació és tota una rama de les matemàtiques que s'escapa de les capacitats de l'autor d'aquest treball.

Per això s'ha intentat acotar els valors d'una forma més pràctica i empírica. Per acotar la mida de la capa mínima, s'ha utilitzat el programa *gzip* versió 1.6 per Linux. S'ha desat en un fitxer el tensor de representació dels SNP, en forma d'una cadena de caràcters. Aquest fitxer s'ha comprimit amb *gzip* amb l'opció «--best» El resultat ha estat d'una reducció de la mida del fitxer a un 4.4% de la mida original.

Tot i que l'objectiu de *gzip* és diferent del de la nostra xarxa, ja que sol intenta representar la informació explícita de les dades administrades, i no pas cap tipus d'informació implícita de la naturalesa d'aquestes, sí que serveix per acotar el valor inferior de representació. És difícil pensar que es podrà representar la informació en el model d'autoencoder en un espai menor sense pèrdua d'informació o un problema d'overfitting significatiu. D'altra banda s'ha de considerar que *gzip* intenta representar la informació de forma exacta, per tant pot estar consumint cert espai emmagatzemant informació que puntualment no segueixi la potencial estructura interna de les dades, i per tant consumint espai per assolir una representació perfecta.

Considerat això, es va realitzar un estudi empíric creant diferents models, amb diferent geometria, tant en nombre de capes (models «shallow» vs «deep»), com en la mida d'aquestes, presentant el límit inferior en 138 unitats, segons les consideracions exposades anteriorment.

Els millors resultats es mostren en la taula 2.

Descripció del mode d'autoencoder	Accuracy	Loss
Entrada 1, mínim 554, sortida 1	80,6%	0,431
Entrada 1, mínim 1108, sortida 1	80,5%	0,431
Entrada 1, mínim 2217, sortida 1	80,5%	0,427
Entrada 1, mínim 277, sortida 1	80,1%	0,440
Entrada 1, mínim 138, sortida 1	79,6%	0,446
Entrada 2, mínim 2217, sortida 1	78,3%	0,455
Entrada 2, mínim 138, sortida 1	78,1%	0,454
Entrada 2, mínim 1108, sortida 1	77,8%	0,452
Entrada 2, mínim 554, sortida 1	77,8%	0,453
Entrada 2, mínim 277, sortida 1	77,2%	0,458
Entrada 2, mínim 1108, sortida 2	77,2%	0,478
Entrada 3, mínim 1108, sortida 1	76,5%	0,468
Entrada 2, mínim 554, sortida 2	76,2%	0,468
Entrada 3, mínim 554, sortida 1	76,2%	0,465
Entrada 2, mínim 277, sortida 2	76,0%	0,456

Taula 2: Models d'autoencoder. Es mostren els 15 models amb millor accuracy. En la descripció s'indica el nombre de capes entre l'entrada o la sortida i la capa de representació mínima, així com el nombre d'unitats que componen aquesta última. Les unitats de la resta de capes es calculen distribuint-les uniformement des del nombre d'unitats del tensor d'entrada, fins al nombre d'unitats de la capa mínima.

Els models testats s'han creat i entrenat de forma automatitzada. Per la seva construcció s'utilitzava la funció d'activació RELU, la funció «binary cross-entropy» com a valor del paràmetre Loss, i s'han avaluat amb la exactitud de la predicció (*Accuracy)*. En aquesta fase no s'han aplicat modificacions en l'entrenament, com funcions «*callback*» ni funcions de regularització.

D'aquests resultats podem extreure unes guies generals:

- Els models petits obtenen millor resultat.
- La mida de la capa mínima no influeix clarament amb el rendiment fins que no es baixa molt el nombre d'unitats.
- Augmentar el nombre de capes de sortida sembla anar en contra del rendiment.

Aquest últim punt va en contra de crear un model d'autoencoder clàssic, que té una estructura simètrica. Es tractaria probablement del conjunt de problemes clàssics de les xarxes neuronals profundes (*gradient vanishing, weight explosion*, ...), amb peculiaritats de l'aplicació de la funció RELU en aquest cas concret.

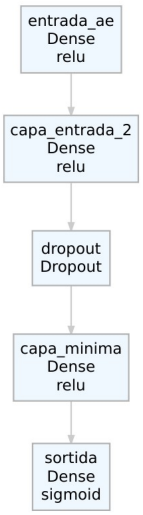
S'han provat altres funcions d'activació per a la construcció del model. Hi ha autors que advoquen per altres funcions d'activació diferents de les clàssiques com el RELU, atribuint-les-hi superioritat (7). Però en les proves realitzades, la funció RELU ha estat quelcom superior a les altres, o com a mínim, no se n'ha observat inferioritat. Això podria estar explicat pel mateix argument que sustenten els avantatges dels altres mètodes.

La funció RELU defineix el valor de sortida segons la següent expressió:

$$f(x)=\begin{cases} 0 \; per \; x<0 \\ x \; per \; x \geq 0 \end{cases}$$

Així qualsevol lectura negativa d'una variable perd informació, ja que tots els valors negatius es representen pel valor 0. Però atès en el treball actual les dades d'entrada tan sols prenen dos valors no negatius, per tots els conjunts possibles de pesos del model amb tots els pesos positius, aquesta funció actuaria com a la funció identitat, que el conjunt com un sistema lineal. Així doncs, ajustant els pesos a valors positius no hi ha més pèrdua d'informació que utilitzant altres funcions d'activació.

El model finalment triat per a continuar amb el desenvolupament del MLP-SAE és el següent:

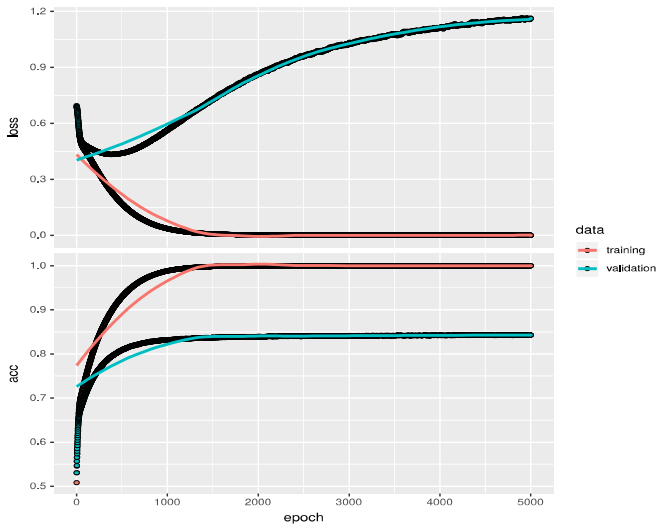


Il·lustració 8: Esquema de l'autoencoder elegit.

- Dues capes prèvies a la capa mínima del model: En el mètode d'entrenament dels autoncoders apilats, hi ha la possibilitat de fer un pre-entrenament del primer autoencoder, i entrenar el segon congelant-ne tots els pesos durant la primera fase d'entrenament. Això evita la destrucció de la informació representada en els pesos ja entrenats del primer autoencoder per propagació d'errors grans en la primera fase d'entrenament del segon autoencoder. Però en la segona fase, es descongelen part del primer autoencoder. D'aquesta manera, la part congelada manté l'inici d'extracció de les característiques o *features* de les dades d'entrada, i la segona meitat s'ajusta (es realitza un *fine-tunning*) a la utilitat concreta del model que s'està creant.

- Capa mínima de 739 unitats: atès que el model es continuarà amb un segon autoencoder, no es necessari reduir completament la dimensió en aquest model, permetent una representació més flexible, i deixant per al segon autoencoder la finalització del procés de *feature extraction*, i *feature fusion*.

- Una capa de sortida: tot i no ser el model clàssic d'autoencoder, al trencar la simetria teòrica, ha mostrat millor rendiment. Les possibles explicacions són per una banda la disminució de la profunditat del model, estalviant-nos la necessitat de la implementació de sistemes per controlar els problemes que comporten. D'altra banda, com ja s'ha comentat, l'ús de la funció RELU en aquest cas particular, es comportaria de forma similar a un sistema lineal. Per tant afegir més capes pot ser redundant, podent-se «reduir» l'expressió de productes d'expressions lineals a una sola formulació.



Il·lustració 9: Model AE seleccionat. Entrenament amb «drop-out», «kernel regularization L2» i ajust de «learning rate»: 2 capes d'entrada, representació mínima de 739 unitats, 1 capa de sortida. Representació de la accuracy i el valor de la funció loss en funció de l'època d'entrenament per als grups de entrenament i de validació.

- Funció d'activació RELU: pels motius ja explicats.

- Funció d'optimització RMS*prop*. La funció d'optimització del model és un tema clau, fins al punt que va ser la limitació del desenvolupament dels models de xarxes neuronals durant anys. Però també és un tema obert. S'ha elegit aquesta funció donada que és una opció molt utilitzada per el seu comportament en dades d'alta dimensionalitat que puguin tenir gradients de magnitud molt diferent. Una resum de l'estat actual de la situació es pot consultar en la publicació de Bottou de 2018 (12). S'ha elegit el RMSprop per ser una variant del s*tochastic gradient descent* utilitzat en el model original.

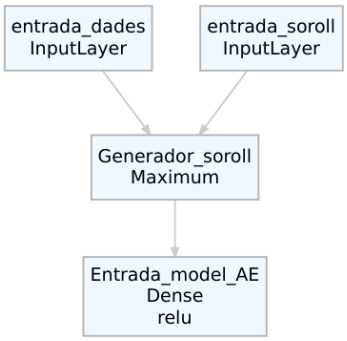
2.3.3 Creacio DAE:

Com ja s'ha comentat, els denoising autoencoders es van desenvolupar per la necessitat de controlar el problema del sobreajust del model. En la seva creació, s'ha de decidir el mètode d'afegir soroll a les dades i en quina quantitat.

Selecció del model de soroll.

En aquest model, les dades d'entrada i sortida estan codificades de forma binària, 0-1, que per triplets contenen la informació d'un al·lel. Per això s'ha considerat més adient no utilitzar un generador de soroll gaussià, que són dels més utilitzats, ja que no correspon a la naturalesa ni la forma de representació de la informació.

S'ha adoptat un sistema basat en un generador de successos binomial, amb el que s'ha imputat el valor 1 de forma aleatòria a una certa proporció de la mostra.



Il·lustració 10: Esquema del generador de soroll. L'entrada de dades es combina amb un generador de soroll basat en la distribució binomial, mitjan una funció "MÀXIM". Aquest nou tensor es passat al model d'autoencoder elegit prèviament.

Aquesta imputació s'ha realitzat per la unió del tensor generador de zeros i uns, amb el tensor de representació dels SNP, amb una capa que aplica la funció «màxim». D'aquesta manera el resultat és el tensor de representació de SNP amb una proporció de valors 1 afegits, que es reparteix aleatòriament.

La selecció de la proporció del soroll és important per a obtenir l'efecte de control del sobreajust. Un excés de soroll pot impedir al model extreure la informació de la distribució de les dades. Una proporció insuficient pot no evitar que el model s'entreni en la «memorització» dels detalls dels casos d'entrenament.

Per la seva elecció s'ha realitzat una sèrie de models amb diferents proporcions de soroll. Els millors resultats han estat generant una proporció d'1 respecte de 0 de 1/30. Atès que cada al·lel està codificat per 3 posicions del tensor, això equival afegir soroll a aproximadament a una desena part de les representacions dels al·lels. (Una mica menys si es considera la possibilitat d'afegir un 1 sobre un 1, o que s'afegeixi soroll dos cops sobre un mateix al·lel).

2.3.4 Ajust hiperparàmetres

Per millorar el rendiment del model DAE s'han utilitzat i ajustat una serie de tècniques addicionals que resumeixo seguidament:

Drop-out: és una tècnica per controlar el sobreajust. Es basa en afegir una o diverses capes en diferents punts del model que eliminen aleatòriament una proporció determinada de les dades que arriba a la capa, ajustant el seu valor a 0. L'efecte es similar al del soroll, amb la diferència que en el cas del soroll es preserva la informació emmascarada, i el cas del drop-out s'elimina la informació.

El **learning rate**, en la funció d'optimització RMSprop regula la magnitud de la variació dels pesos en funció de l'error marcat per la funció Loss, en la fase de retropropagació de l'error. El comportament del model, donat el nombre tan elevat de paràmetres, és d'un model on amb *learning* rates alts s'arriba ràpidament a una situació de no convergència. Però en l'ajust d'aquest, s'ha de permetre un temps llarg d'entrenament per a decidir si hi ha millora dels paràmetres d'avaluació, atès que la variació dels mateixos amb *learning rates* petits són lents.

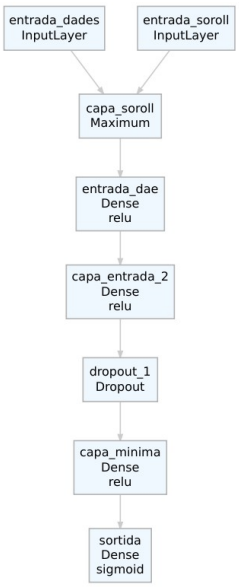
Kernel regularizer L2: En els models profunds, la tendència de la taula de pesos és a incrementar-se per l'efecte additiu de les multiplicacions i addicions successives de matrius. Els mètodes de regulació del nucli busquen reduir aquest fenomen mitjançant la inclusió d'un valor que representi la magnitud dels pesos a la funció L*oss*. D'aquesta manera durant l'entrenament es penalitzen les solucions amb pesos més elevats. Igual que en els casos anteriors, obtenir un valor apropiat és important:

excessivament elevat causaria que dominés al valor de la funció *loss*, i excessivament baix no aconseguiria la funció esperada. En aquest cas, igualment que en l'ajust dels altres paràmetres, el valor òptim és relativament baix, donat el gran nombre de paràmetres del model.

Finalment s'han implementat 2 funcions de *callback*. Aquestes són funcions que s'apliquen entre cada època de l'entrenament per ajustar dinàmicament com es realitza aquest. De les funcions implementades, una serveix per ajustar dinàmicament el *learning rate*, i la segona atura l'entrenament de forma prematura en el cas que hagi deixat d'haver-hi millora en els resultats model.

Donades les característiques del model, es tarden algunes èpoques en veure canvis de tendència. Per aquest motiu els valors que regulen el ràpid que la funció *callback* modifica els paràmetres del model s'han d'ajustar per a adaptar-se a aquesta lentitud. Així tant el paràmetre de «patiente», que indica el nombre d'èpoques sense millora per aplicar el canvi de *learning rate*, com el de «*colddown*», que indica el nombre d'èpoques a esperar després d'un canvi a tornar a aplicar la funció, s'ha ajustat a 100 èpoques. En conseqüència els canvis dels ajustos del model es realitzaran com a poc cada 200 èpoques.

2.3.5 DAE final:

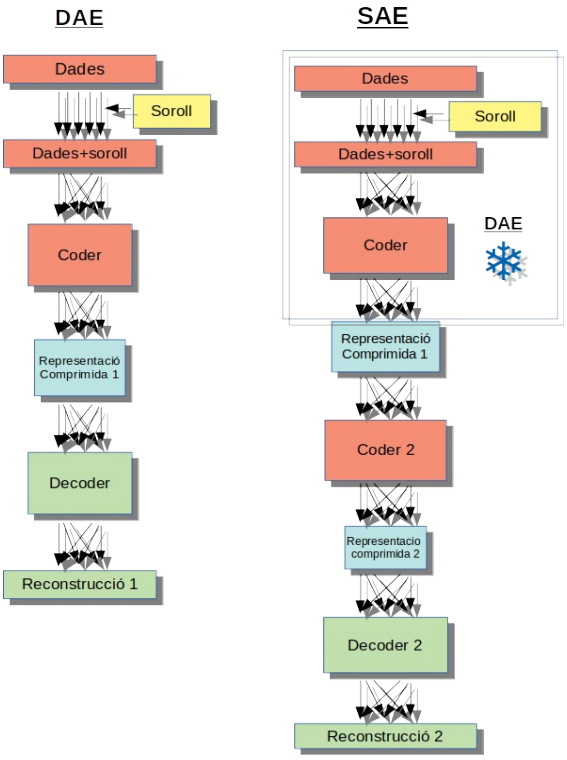


Il·lustració 11: Esquema del model DAE final.

En conseqüència a l'exposat, el model creat per al model DAE està format:

- Estructura del DAE tipus *«deep undercomplete»*
- Per un generador de soroll de tipus binomial.
- Agregador del soroll mitjançant una capa de funció «MÀXIM».
- Dues capes de l'encoder, de tipus «dense» i amb la funció d'activació RELU.
- Una capa amb la funció «*drop-out*».
- Una capa amb la representació mínima de la informació.
- Una única capa de tipus «dense» de sortida amb la funció d'activació SIGMOID.
- Call-back de per la fase d'entrenament: funció per aturar l'entrenament de forma precoç si no hi ha millora en el rendiment del model (*callback_early_stopping*), i per controlar el *learning rate* (*callback_reduce_lr_on_plateau*)

2.3.6 *Stacked Autoencoder*



Il·lustració 12: Models de denoising autoencoder (DAE) i stacked denoising autoencoder (SAE). El SAE utilitza la partpart codificant del model DAE pre-entrenada. En una primera fase els pesos del encoder del DAE són "congelats" per evitar-ne la modificació durant l'entrenament del model SAE. En la segona fase de "fine-tunning", son parcial o totalment descongelats per finalitar l'ajust dels pesos del model complet.

Un cop generat el model de DAE, es procedeix a crear el model d'autoencoders apilats, segons l'esquema de la il·lustració 12, on es mostra un esquema del DAE, i de la construcció del SAE a partir d'aquest.

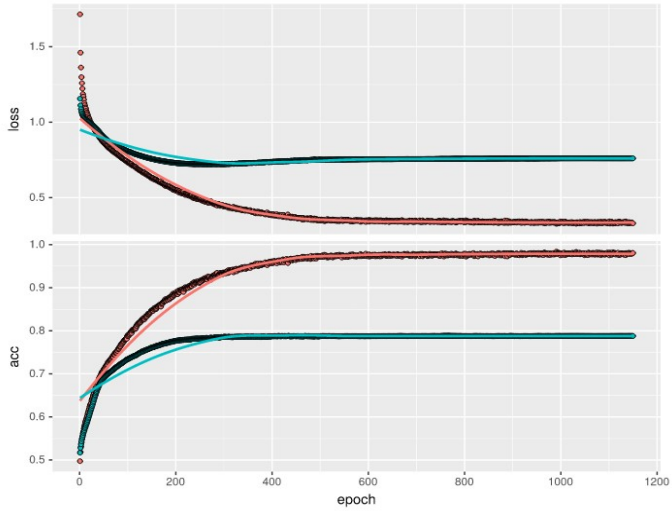
Com ja s'ha explicat, l'entrenament directe d'un model profund pot portar fàcilment a la no convergència cap a un estat proper a l'òptim. En aquest model això es tracta per la realització de pre-entrenaments dels components del model.

S'han tornat a realitzar proves amb diferents models, bàsicament testant els models del segon autoencoder dintre dels models shallow undercomplete, o deep undercomplete. La capa mínima s'ha creat amb 184 unitats, que és troba en el límit empíric que s'ha argumentat amb anterioritat.

Finalment s'ha optat per una model shallow undercomplete, amb una capa interna oculta, una capa de drop-out i la capa de mínima representació de la informació de 184 unitats.

Per a la reconstrucció dels patrons de SNP s'ha utilitzat un única capa de sortida amb funció d'activació «sigmoid».

Tot i el teòric benefici de la representació dels models SAE, en aquest cas, i amb aquest set d'entrenament, no s'ha aconseguit un ajust dels hiperparàmetres que millorés al primer model DAE. De fet, el rendiment és lleugerament inferior com es mostra en la taula 3 . La il·lustració 13 mostra l'evolució dels resultats del model durant el seu entrenament.



Il·lustració 13: Historial d'entrenament del model SAE

	Accuracy	Loss
Primer model DAE	0.808	0.601
Primer model SAE	0.797	0.745

Taula 3: Rendiment del model SAE

2.4 MLP-SAE

2.4.1 Creació model MLP-SAE

S'anomena *multilayer perceptron* (MLP) a un model de xarxa neuronal amb un mínim de 3 capes i funcions d'activació no lineals (excepte en la capa d'entrada).

En el model MLP-SAE, els autoencoders apilats actuen de capes funcionals internes d'entrada al model. Falta afegir una o més capes que facin la predicció de l'expressió HTSeq a partir de la informació processada del patró SNP pels SAE.

Per a completar el model, s'han hagut de definir els següents punts.

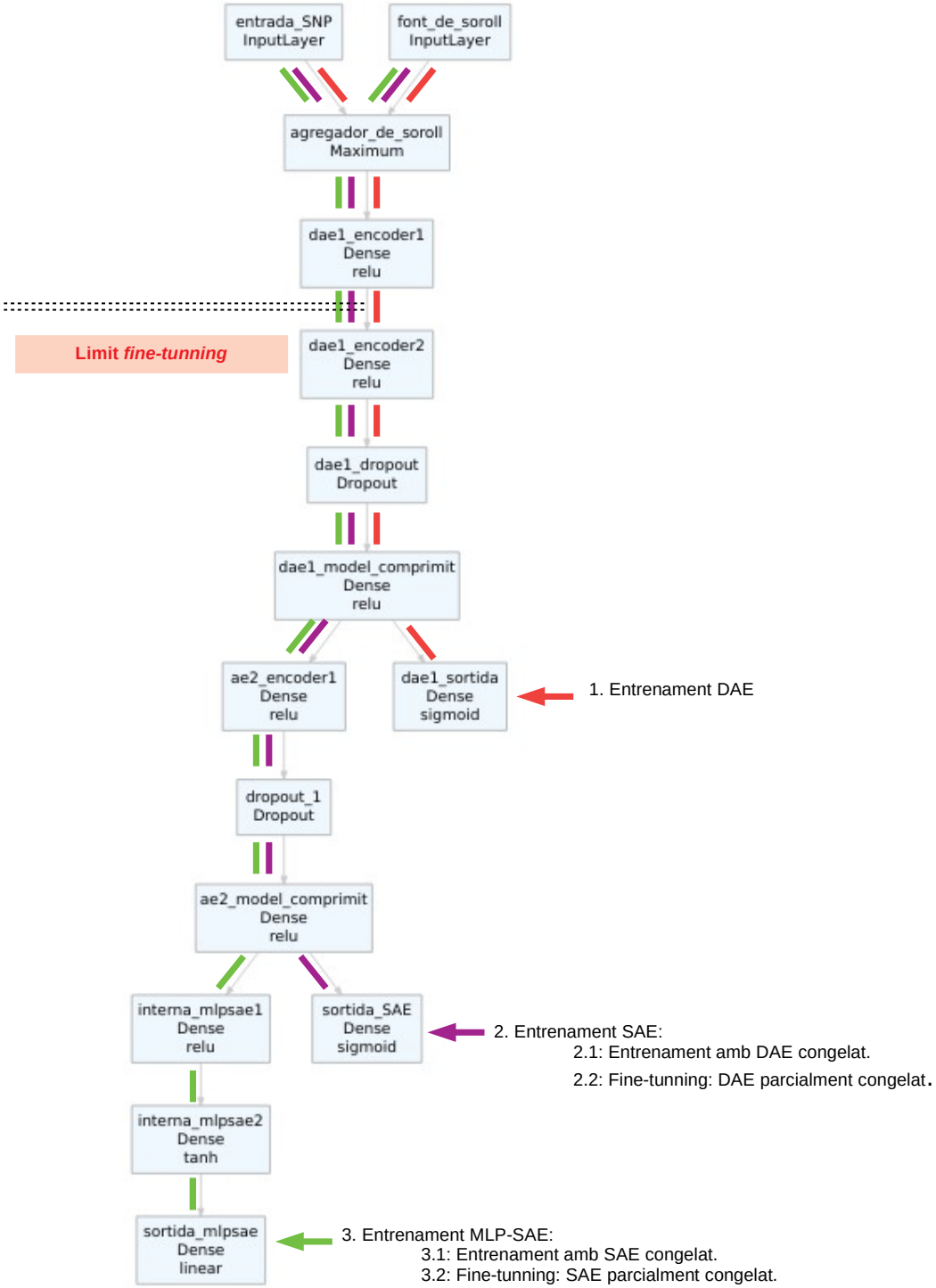
- nombre de capes de sortida.
- Mida de les capes
- Funcions d'activació: atès que s'intenta predir els valors de HTSeq, que contenen valors positius i negatius, s'ha fet la hipòtesi que l'aplicació d'una funció d'activació diferent de RELU, que puguin representar ja valors positius o negatius, podria tenir un avantatge respecte a deixar el modelatge del signe de la predicció a l'última capa de sortida. Per això s'ha comparat «RELU» amb la funció tangent hiperbòlica.

Construint diferents models amb els criteris anteriors, s’ha pogut concloure:

- Entre el SAE i la capa de sortida, s’han interposat una o diverses capes densament connectades. Hi ha un benefici evident entre 1 i 2 capes, no observant-se evidència de millora entre 2 i 3, i començant a presentar tendència a pitjor rendiment a partir de 4 capes.
- El model amb una sola capa presenta millora augmentant la mida d’aquesta fins a les 4000 unitats. El model amb 2 capes, presenta un rendiment similar al d'1 capa amb 4000 unitats per capa. Amb dues capes o més, no presenta millora l’ús de més de 2000 unitats.
- L’ús de la funció tangent hiperbòlica com a funció d’activació de l’última capa prèvia a la capa de sortida, millora el rendiment del model. L’ús d’aquesta funció en capes prèvies no millora el resultat, alentint el procés d’entrenament.

Així doncs, s’ha creat el model afegint 2 capes internes més, de 2000 unitats, la primera amb la funció d’activació *relu*, i la segona amb la funció *tanh*, i una última capa de sortida amb funció d’activació lineal.

En la il·lustració 14 s’observa el graf complet del model i les vies per realitzar l’entrenament d’aquest.



Il·lustració 14: Esquema del model complerta MLP-SAE, amb les sortides addicionals per realitzar el pre-entrenament de les porcions d'autoencoder. L'entrenament es produeixi en l'ordre indicat. Els últims dos passos es realitzen en 2 fases: pre-entrenament amb part de l'autoencoder congelat, i fine-tunning. En la fase de fine-tunning el model sol s'entrena fins al límit marcat com a capa "dae1_encoder2".

2.5 Avaluació del model: 5-fold cross-validation.

En l'avaluació dels models parcials construïts s'han utilitzat una partició fixa de les dades en 3 conjunts: entrenament, validació i test. Això presenta dos problemes.

El primer és que els paràmetres de rendiment del model poden estar sobre o infraestimats per una selecció a l'atzar d'agrupacions de les mostres favorables o desfavorables pel model.

En segon lloc, s'ha de tenir en compte que es disposen d'un grup reduït d'observacions. Idealment, per millorar el rendiment del model s'hauria d'entrenar amb el nombre màxim d'observacions. Però continua se necessari reservar un percentatge d'observacions per la validació prou gran com per tenir un mínim de precisió en l'avaluació.

El sistema de la validació creuada és una aproximació senzilla però molt adequada per casos com el present. Per realitzar l'avaluació final del model s'ha implementat una validació creuada en 5 grups. S'han generat 5 agrupacions aleatòries de les observacions, amb un 20% de les observacions cada una, i s'han generat i entrenat 5 models, utilitzant en cada iteració un 80% de les observacions per entrenar el model, i un 20% per avaluar el rendiment. També s'han desat les prediccions del model sobre aquest 20% de dades que no participen de l'entrenament en cada iteració, per reconstruir una matriu d'observacions predites amb la que avaluar el rendiment.

En aquest apartat torna a presentar el problema de l'error «OOM» de l'entorn Python en els entrenaments repetits de models. També s'ha optat per la solució de preparar les dades a utilitzar en cada iteració de la validació creuada en un fitxer temporal, executar en un entorn de R nou el codi que s'encarrega d'entrenar i avaluar el model a partir de les dades incloses en l'arxiu temporal. Finalment s'ha reconstruït la matriu predita i els valors de rendiment retornats pel procés d'entrenament, també a través d'arxius temporals.

2.6 Resultats

En el model creat, per a la predicció del HTSeq s'han seleccionat 6329 gens dels disponibles en les dades inicials. La predicció es basa en la informació que aporten els 2956 SNPs disponibles per cada observació.

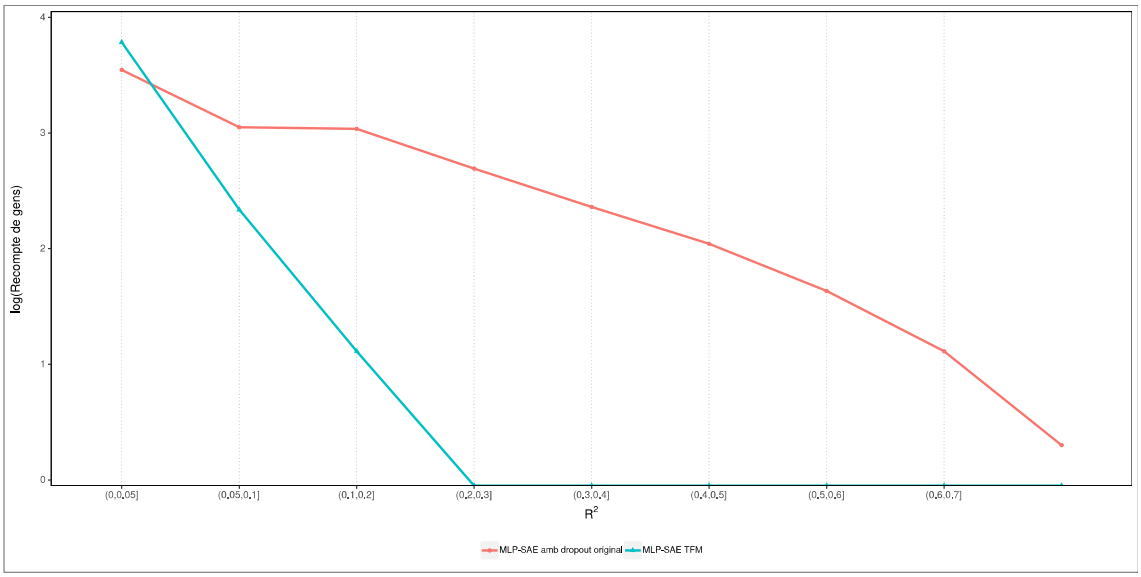
Tot i que el model creat en aquest treball presenta un error quadràtic mig (EQM) de 0.1956, inferior respecte el 0.2890 del model creat en l'article original, el seu rendiment és inferior al d'aquest últim.

Calculant la correlació global de les matrius d'expressió predita i l'observada, el coeficient de correlació és de 0.410 (IC 95% 0.408-0.412), amb un valor de $p<<10^{-4}$. El coeficient R^2 global és de 0.168.

Valorant gen a gen, la distribució comparada amb la de l'article original és mostra a la taula 4 i la il·lustració 15 . La distribució dels coeficients de correlació R^2 per gen és clarament inferior, no obtenint cap gent amb un R^2 superior a 0.2, enfront dels 890 gens amb un R^2 majors de 0.2 del model del treball original.

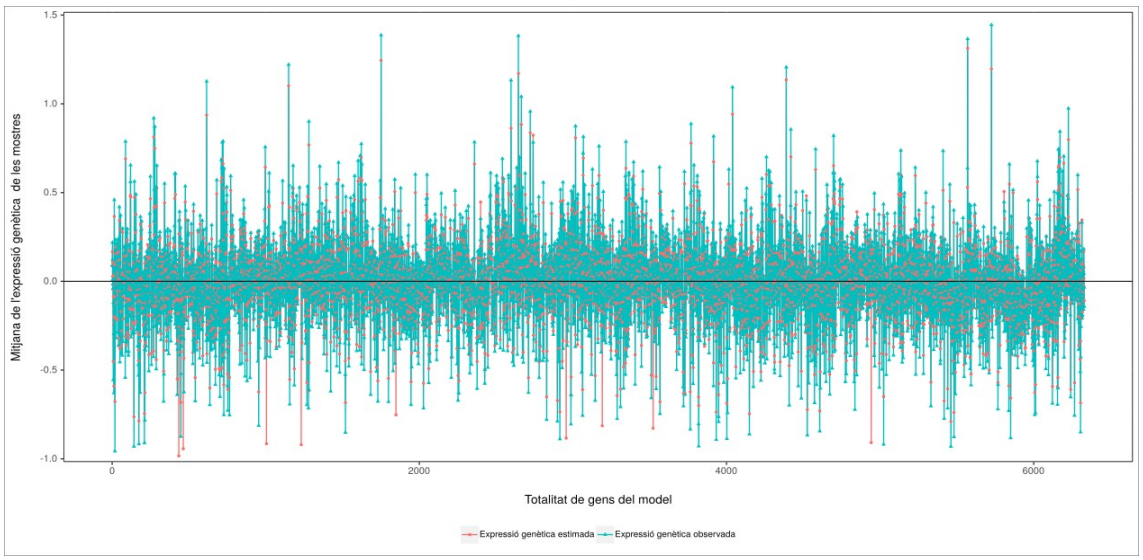
R^2	MLP-SAE creat.	MLP-SAE original
(0,0.05]	6099	3507
(0.05,0.1]	217	1121
(0.1,0.2]	13	1086
(0.2,0.3]	0	493
(0.3,0.4]	0	229
(0.4,0.5]	0	110
(0.5,0.6]	0	43
(0.6,0.7]	0	13
(0.7,0.8]	0	2
Total gens avaluats	6329	6604

Taula 4: Comparació del nombre de gens segons rang de R^2: model creat vs model original

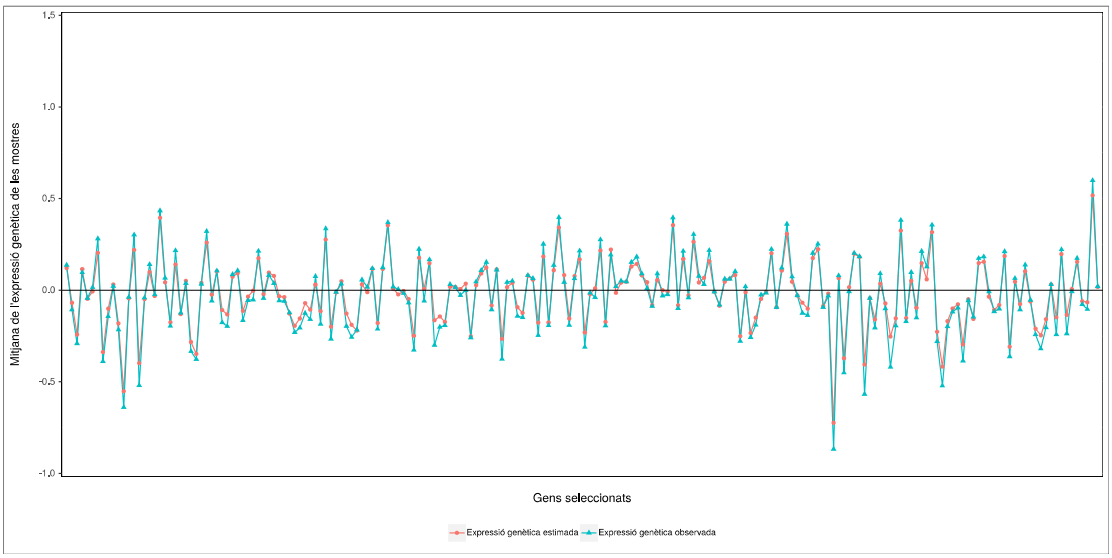


Il·lustració 15: Distribució dels gens segons rang de R^2 resultant. Es comparen esl models MLP-SAE amb drop-out creat en l'article original amb el creat en aquest treball.

La comparació entre l'expressat i el predit pel model del conjunt de gens es pot observar en el la il·lustració 16, mentre que la il·lustració 17 mostra solament l'expressió dels 200 gens amb una coeficient de correlació de Pearson més elevada.



Il·lustració 16: Gràfic d'expressió de tots els gens inclòsos en el model. Es mostra les mitges dels valors observats i els redits.



Il·lustració 17: Expressió predita vs observada en els 200 gens amb millor correlació.

Donat que no es va assolir un model MLP-SAE equiparable al de l'estudi de referència, no es va dur a terme les dues següents fases planificades de recerca i aplicació del model a noves dades. Es va centrar l'esforç en la millora del model i en l'anàlisi del seu comportament.

2.7 Discussió

En ambdós treballs s'ha obtingut una proporció de gens amb una correlació no significativa o amb una correlació baixa important. Això és en principi esperable. Sol esperaríem que una certa part de l'expressió genètica estigués condicionada de forma exclusiva, o de forma suficientment forta, al patró de SNP. Precisament no s'ha de perdre de vista, que l'origen dels treballs que s'han pres com a referència, parteix de

l'exploració de tècniques per millorar els estudis *eQTL*, orientats a detectar les correlacions, però no a determinar-ne la magnitud.

D'altra banda hi ha altres factors que poden influir en l'expressió genètica que no estarien reflectits en el patró de SNP. Aquest podrien no haver estat controlats en el treball que originalment va generar el conjunt de dades analitzat i seria impossible de terminar-los en aquest moment.

Per tant, com que R^2 mesura la proporció de la variància que depèn de la variable predictora, i la variable predita té múltiples influències, R^2 ha de ser necessàriament menor d'1, en una magnitud que no es pot estimar. Per poder fer-ho hauríem de poder conèixer *la realitat* en un sentit més filosòfic, i saber de forma certa tots els factors influents i el pes exacte que tenen en la resposta observada. Tan sols així es podria construir un model que aspirés a apropar-se a 1.

Però tot i la incertesa de quin és el valor màxim de R^2 assolible per cada gen, tenint en compte únicament els patrons de SNPs, el model creat presenta pitjors resultats que el publicat anteriorment. Aquests últims representen per tant la cota mínima assolible. Aquestes diferències poden tenir causes diverses que exposarem seguidament.

El primer punt d'incertesa és la composició del model. En l'article original no es mostra l'estructura, els paràmetres ni les característiques del model creat. Tampoc hi ha enllaços al codi utilitzat. Les referències que se'n fan són molt vagues. Amb la informació que proporcionen es pot deduir que han fet servir un model amb l'apilament de dos autoencoders de tipus *shallow undercomplete*, per finalment utilitzar una sortida amb un nombre indeterminat de capes densament connectades.

Però no hi ha referència a la o les funcions d'activació utilitzades pel seu model, ni el nombre d'unitats de cada capa, o de si han fet servir alguna tècnica diferent del *drop-out* i del model «*dennoising autoencoder*» per a controlar el sobreajust, o els problemes dels models de xarxes neuronals profundes.

Arribats aquest punt, conèixer les diferències en els models seria cabdal per poder aportar més llum als factors que influencien la millor predicció dels HTSeq counts, i en conseqüència en el comportament i distribució d'aquest tipus de dades.

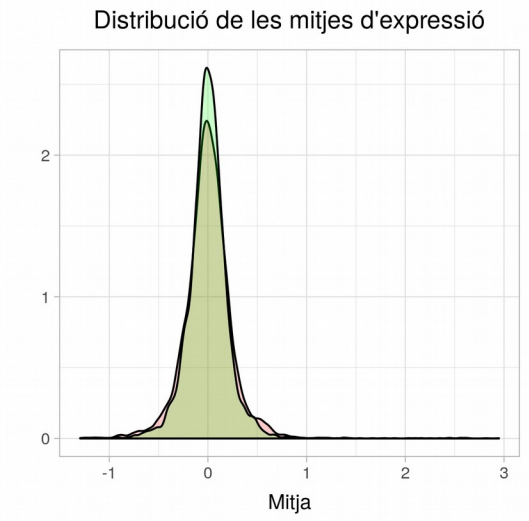
Probablement pel fet que R^2 presenta un alt grau d'incertesa respecte a la cota que per cada gen podríem considerar un bon resultat, ja que desconeixem la proporció de variància explicada pel patró de SNPs en cada gen, en l'article original es mesura el rendiment amb l'error quadràtic mig. El model creat presenta un MSE mig de 0.1956, mentre que el del model de l'article era 0.3093. Segons aquest criteri el model creat és superior al de l'article, punt que ja s'ha mostrat que no és cert. Quina és doncs la causa d'aquesta discrepància?

El primer element a pensar és en el sobreajust. El problema dels models basats en xarxes neuronals (o en altres sistemes de machine learning), és el control del sobreajust. Els models de machine learning fàcilment s'adapten a reproduir les dades d'entrenament per l'aprenentatge d'aquestes, i no pas per l'aprenentatge de la distribució subjacent d'aquestes. Atès que s'han ajustat els paràmetres de l'entrenament per evitar aquest fenomen, em d'entendre que el sobreajust que encara es pogués produir seria causat per la mateixa naturalesa, estructura i dimensió del

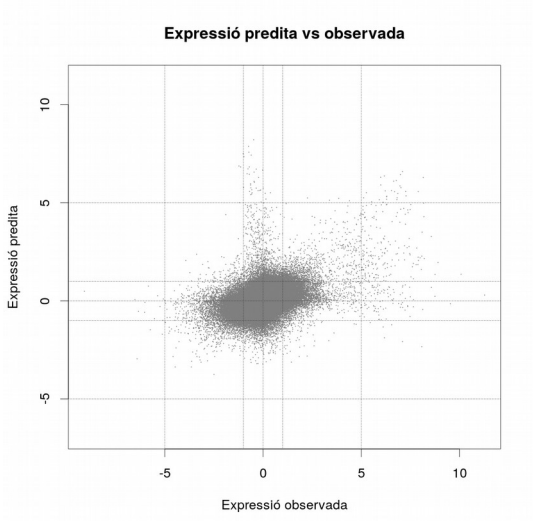
model. Novament no podem comprovar aquest punt per la manca d'informació del model de l'article original.

Com s'ha d'entendre aquesta discrepància entre el rendiment real i la suposada superioritat segons el MSE mig? El sobreajust pot explicar-ho, però pot haver-hi altres causes?

Si ens fixem quina és la relació entre els valors predits i observats del model creat, trobarem certes anomalies que poden apuntar a una altra explicació.



Il·lustració 18: Densitat de distribució de les mitges d'expressió genètica per a les observacions (vermell) i les prediccions (verd).

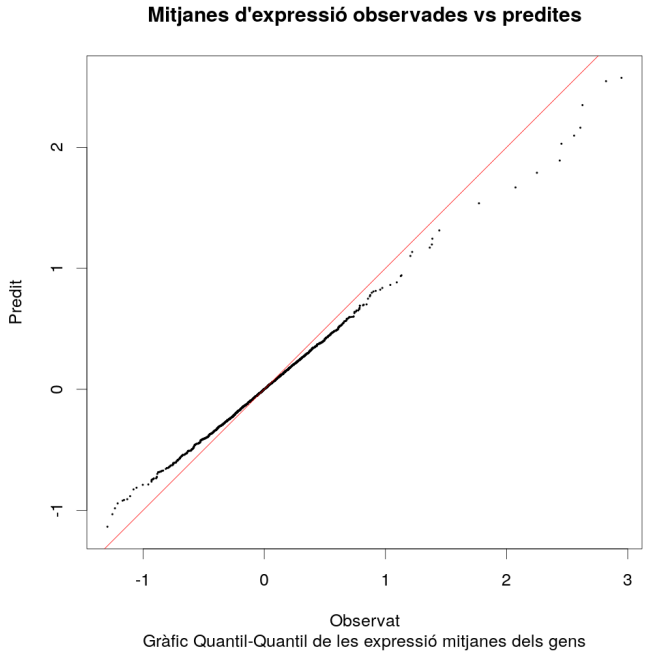


Il·lustració 19: Diagrama amb el núvol de punts valors d'expressió genètica predita vs la observada del total de les gens.

La il·lustració 18 mostra la densitat de distribució de les la mitja d'expressió. La il·lustració 19 mostra el núvol de punts dels valors predits vs els observats per a tots els gens. El que s'observa és que els valors predits presenten una concentració més important al voltant del valor central del conjunt de valors de la matriu. Observant el núvol de punts es podria destacar:

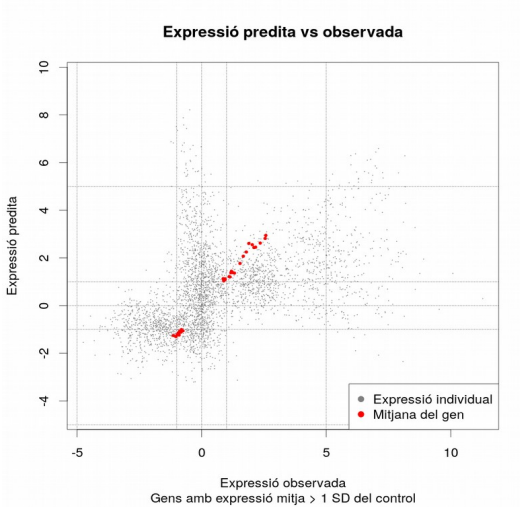
1. Existeix una aparent sobreposició de més d'una distribució:
    - Una distribució binomial al voltant del centre, amb cert grau d'autocorrelació. Podrien correspondre al conjunt de gens no relacionats amb el SNPs.
    - Una distribució aproximadament lineal dels punts que s'allunyen del centre. Aquest podrien correspondre a les observacions dels gens relacionats amb el SNPs. Identificar-los és l'objectiu dels estudis eQTL.
    - Una distribució de punts centrals, amb valors predits molt elevats però valors observats baixos, que podrien correspondre a errades del model en la seva predicció.
2. Els valors predits estant en una escala inferior. Els valors estan distribuïts d'una forma similar, però el seu valor absolut és inferior. Així doncs, el model sembla tendir a predir valors el més centrats possible.

Si tenim en compte les mitjanes d'expressió de cada gen, la correlació és molt bona, amb una R^2 ajustada de 0.9852. (vegeu il·lustració 20). Les il·lustracions 21 i 22 mostren el núvol de punts de les prediccions en funció del valor predit segons si el gen presentava una expressió observada superior o no a ±1 SD. En elles es veu com el primer grup, d'expressió no alterada, presenta una distribució més aleatòria. El segon grup presenta una distribució més lineal. Però la distribució de les mitges en els dos grups segueixen una distribució lineal en cada grup.

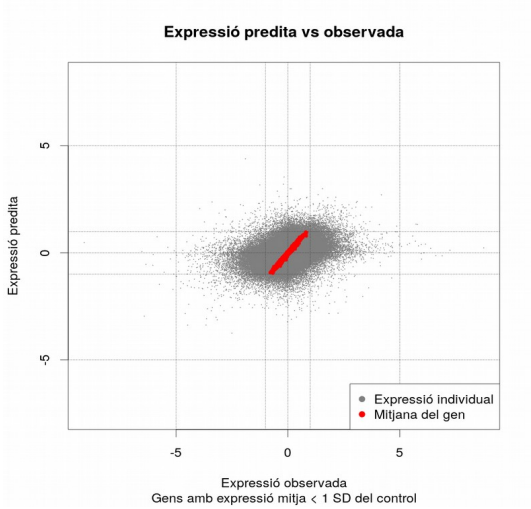


Il·lustració 20: Gràfic Quantil-Quantil de les mitges d'expressió predites vs les observades. S'observa una correlació molt important linealment, amb un certa disminució en l'escala dels valors extrems.

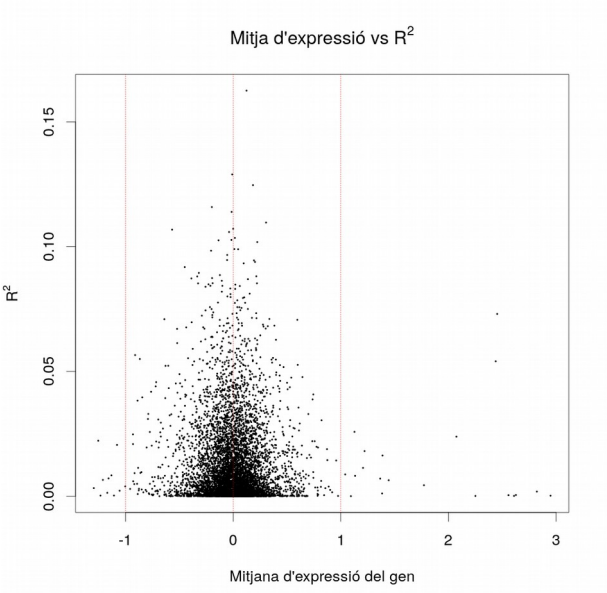
Aparentment el model tendeix a la centralitat del global de les prediccions. Si representem el valor de R^2 obtingut en funció de la mitja d'expressió (il·lustració 23), es pot observar que són els valors centrals (gens ni sobreexpresats, ni infraexpresats) els que presenten una correlació major.



Il·lustració 21: Núvol de punts dels gens amb expressió mitja > 1SD



Il·lustració 22: Núvol de punts dels gens amb expressió mitja < 1SD



Il·lustració 23: Relació del valor R^2 en relació a la mitja d'expressió observada del gen.

Això pot portar a pensar que l'elecció del MSE com a funció *Loss* del model no és l'ideal. Sembla que pogués causar que el model s'ajusti a la centralitat de la distribució multivariada de les dades, i tendeixi a una pitjor representació de la variabilitat de les observacions per gens individuals i de mantenir la magnitud de l'escala del valor de l'expressió genètica. Per solucionar això es podria construir un model individual per cada gen, però seria computacionalment molt costós i es perdria la informació de les correlacions de les expressions genètiques.

Així doncs, el que seria possible fer és estudiar alternatives a la funció MSE com a funció *Loss*, que solucioni aquests problemes.

3 Conclusions.

3.1 Conclusions del treball.

Els models MLP-SAE, com ja es va mostrar en l'article original, són factibles per estudis eQTL. Però tot i això, el seu desenvolupament més enllà de la detecció de gens relacionats amb els SNPs, com seria l'ús com a predictor de la magnitud de l'expressió, presenta interrogants importants.

Les característiques del model de xarxa neuronal són determinants en el rendiment que se n'obté. Per tant s'haurien d'incorporar als resultats dels treballs publicats per assegurar la reproductibilitat del treball requerida pel mètode científic.

La selecció de la funció MSE com a funció *Loss* de l'entrenament del model de MLP-SAE probablement no és l'opció més adient, ja que solament assegura que el model anirà a reproduir els valors centrals de la distribució de les dades, i no assegura que es mantingui l'amplitud de la distribució ni l'escala dels magnituds, en especial en aquells gens que presentin una expressió diferenciada respecte a l'expressió mitjana del conjunt de gens.

3.2 Assoliments dels objectius.

Dels objectius planificats, s'ha assolit la creació d'un model funcionant cd MLP-SAE. Però l'objectiu de cercar nous conjunts de dades i l'avaluació del model amb les dades noves no s'ha dut a terme.

Els motius són bàsicament dos:

- El retard del desenvolupament del projecte, que es comenta en el següent punt.
- El baix rendiment del model creat: comparant el model creat amb el rendiment publicat, presenta un rendiment bastant més baix que aquell. Per tant, prosseguir amb l'aplicació sobre altres sets de dades no podria aportar llum a la motivació inicial. Aquesta no era altra que comprovar si l'ús de MLP-SAE és generalitzable a altres especies en aquest tipus concret d'estudis GWAN. Per tant es va considerar més interessant intentar millorar el model en un primer model, i analitzar els resultats per esbrinar perquè no generava una bona predicció al final, de cara a obtenir algun aprenentatge aplicable a una futura construcció d'un nou model.

3.3 Seguiment, planificació i metodologia

Els canvis que hi ha hagut en la metodologia han vingut forçats pels imprevistos de funcionament de l'entorn de desenvolupament, que ha obligat a canviar l'estructura planejada d'organització del codi. Però pel que fa a la metodologia de desenvolupament ha complert correctament amb els requeriments del treball.

La planificació del treball ha hagut de sofrir importants modificacions, basats en dos aspectes:

Primer, com ja ha estat comentat, el baix rendiment del model obtingut en comparació al del treball original condicionava l'execució de la segona i tercera fase de l'estudi. L'objectiu de valorar si el model mantenia el rendiment amb dades de diferent origen, no era possible si no s'assolia un rendiment similar al de l'estudi amb les dades originalment utilitzades. Per aquest motiu, en la fase final de l'estudi s'ha renunciat definitivament a l'anàlisi de noves col·leccions de dades per centrar-se en l'anàlisi dels resultats del model, la qual ha estat presentada en l'apartat de discussió.

El segon aspecte que va dur a la modificació del pla de treball en una fase més primerenca que l'anterior, que es mostrat en la il·lustració 2, va ser una estimació massa optimista la corba d'aprenentatge de Keras. No es va tenir en compte que no es tracta d'un paquet desenvolupat nativament per R, sinó un front-end per a Keras per a Python.

Aquest aspecte fa que el conjunt de l'eina sigui més complexa del que habitualment són els paquets natius de R. Al coneixement de Keras per R, s'hi afegeix la complexitat d'integrar codi de Pyhton a través del paquet *reticulate* que utilitza Keras. Per altra banda, al no tenir coneixements de Python i no tenir instal·lat un entorn de Python fora del que instal·la el sistema operatiu per a l'ús propi, fa que es perdi el control i la capacitat d'esbrinar que està passant en aquests l'entorn. En cas d'error, tot i els sistemes d'informació d'errors enllaçat amb R, és més complicat de traçar on s'ha produït aquest.

Tot i l'enorme esforç que hi ha darrere d'aquest projecte d'integració de l'entorn virtual de Python amb el que treballa Keras, aquest en el cas d'aquest projecte ha donat problemes de gestió de memòria. Això no solament ha causat problemes de temps en la re-escriptura i re-organització de part del codi, sinó que ha suposat una pèrdua de temps prèvia, per esbrinar si l'error era causat per un error propi, o bé de la configuració de l'entorn virtual.

A aquest problema també s'hi afegeix un dèficit de documentació. La documentació de Keras per R és bastant més reduïda que la de Keras per Python. Aquest desavantatge s'observa sobretot en les pàgines de referències. Comparat amb altres paquets natius de R, Keras presenta un dèficit important en la documentació. Les pàgines són molt reduïdes, sense exemples, i molts cops donen per suposat el coneixement de la funcionalitat que realitza la seva funció corresponent de Python.

Tot i que per contra, hi ha tutorials molt detallats on s'introdueixen des dels conceptes més bàsics, fins als més avançats, el seu ús és el propi d'un tutorial, i no estan preparats per poder ser utilitzades com a text de referència ràpida per resolució de dubtes.

Tot plegat ha causat un retard en l'inici efectiu de creació de codi funcionant, i va obligar a un replantejament més realista del temps de desenvolupament atès que la fase de documentació podia ser més llarga que l'experiència que s'havia tingut fins al moment amb eines desenvolupades íntegrament en R.

L'aprenentatge que un pot endur-se'n, és la de concloure que si bé l'entorn virtual que instal·la per defecte Keras per R és adequat per ús ocasional, o per projectes senzills, l'ideal és tenir un entorn propi de Python instal·lat i funcionant. D'aquesta manera es té control de com és l'entorn, de si les llibreries estan actualitzades, i s'hi té més accés en cas de produir-se un error. Això amb independència que després es desenvolupi el projecte en Keras sobre R.

3.4 Punts pendents

El punt pendent més evident seria el d'aconseguir un rendiment millorat del model. A priori, s'hauria de poder aconseguir modificant les dimensions d'aquest, ja que no es comenta en l'article original l'ús de cap altre element o tècnica no aplicada en el model presentat. Així s'ha d'assumir que la principal diferència és en el nombre o mida de les capes, o l'elecció de la funció d'activació o del mètode per afegir soroll.

Durant el desenvolupament del model, es va tenir present la possibilitat d'utilitzar altres estructures del model: per exemple l'augment de la profunditat del model, o bé l'ús d'una autoencoder per a l'entrada de tipus *deep overcomplete*. Tot i no haver-se exposat en l'apartat de desenvolupament, durant el transcurs del treball s'havia creat algun model de prova d'autencoder *deep overcomplete* que obtenia millor *accuracy* que l'utilitzat. Però no es va poder incloure en les proves del model MLP-SAE, ja el model resultant era excessivament gran com per ser entrenat en la memòria disponible en la GPU de l'equip utilitzat.

Finalment, també quedaria pendent la pregunta que sorgeix d'una de les conclusions del treball: quina seria la funció *Loss* millor per aquesta aplicació? Aquest requeriria un treball més profund, amb un intens treball de recerca de la teoria matemàtica i un desenvolupament de models de prova. L'esforç requerit probablement excedeix el d'un treball de final de màster, i podria ser motiu d'un projecte d'investigació més extens.

4 Glossari.

- AE: autoencoder
- DAE: denoising auto-encoder
- eQTL: expression Quantitative trait locus
- GPU: unitat de procés gràfic
- GWAS: genome-wide association studies
- HTSeq counts: High-Trhoughput Sequencing counts
- MLP: multilayer perceptron
- MLP-SAE: multilayer perceptron - stacked denoising autoencoder
- EQM: error quadràtic mig.
- MSE: mean squared Error; error quadràtic mig.
- PCR: polymerase chain reaction
- Q-Q: en gràfics, quantil-quantil
- SAE: stacked (denoising) autoencoder
- SNP: Single-nucleotide polymorphism

5 Bibliografia.

1. Xie R, Wen J, Quitadamo A, Cheng J, Shi X. A deep auto-encoder model for gene expression prediction. BMC Genomics [Internet]. 2017 Nov 17 [cited 2019 Mar 3];18(S9):845. Available from: https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-017-4226-0

2. Dillies MA, Rau A, Aubert J, Hennequet-Antier C, Jeanmougin M, Servant N, et al. A comprehensive evaluation of normalization methods for Illumina high-throughput RNA sequencing data analysis. Brief Bioinform. 2013;

3. Mirza B, Wang W, Wang J, Choi H, Chung NC, Ping P. Machine Learning and Integrative Analysis of Biomedical Big Data. Genes (Basel). 2019;

4. Michaelson JJ, Alberts R, Schughart K, Beyer A. Data-driven assessment of eQTL mapping methods. BMC Genomics [Internet]. 2010 Sep 17 [cited 2019 Mar 3];11:502. Available from: http://www.ncbi.nlm.nih.gov/pubmed/20849587

5. Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. Science [Internet]. 2006 Jul 28 [cited 2019 Jun

3];313(5786):504–7. Available from: http://www.ncbi.nlm.nih.gov/pubmed/16873662

6. Chollet F, Allaire JJ. Deep learning with R [Internet]. [cited 2019 Jun 3]. 335 p. Available from: https://www.manning.com/books/deep-learning-with-r

7. Charte D, Charte F, García S, del Jesus MJ, Herrera F. A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines. Inf Fusion. 2018;

8. Vincent P, Larochelle H, Bengio Y, Manzagol P-A. Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on Machine learning - ICML ’08 [Internet]. New York, New York, USA: ACM Press; 2008 [cited 2019 Mar 13]. p. 1096–103. Available from: http://portal.acm.org/citation.cfm?doid=1390156.1390294

9. Brem RB, Kruglyak L. The landscape of genetic complexity across 5,700 gene expression traits in yeast. Proc Natl Acad Sci U S A [Internet]. 2005 Feb 1 [cited 2019 May 10];102(5):1572–7. Available from: http://www.ncbi.nlm.nih.gov/pubmed/15659551

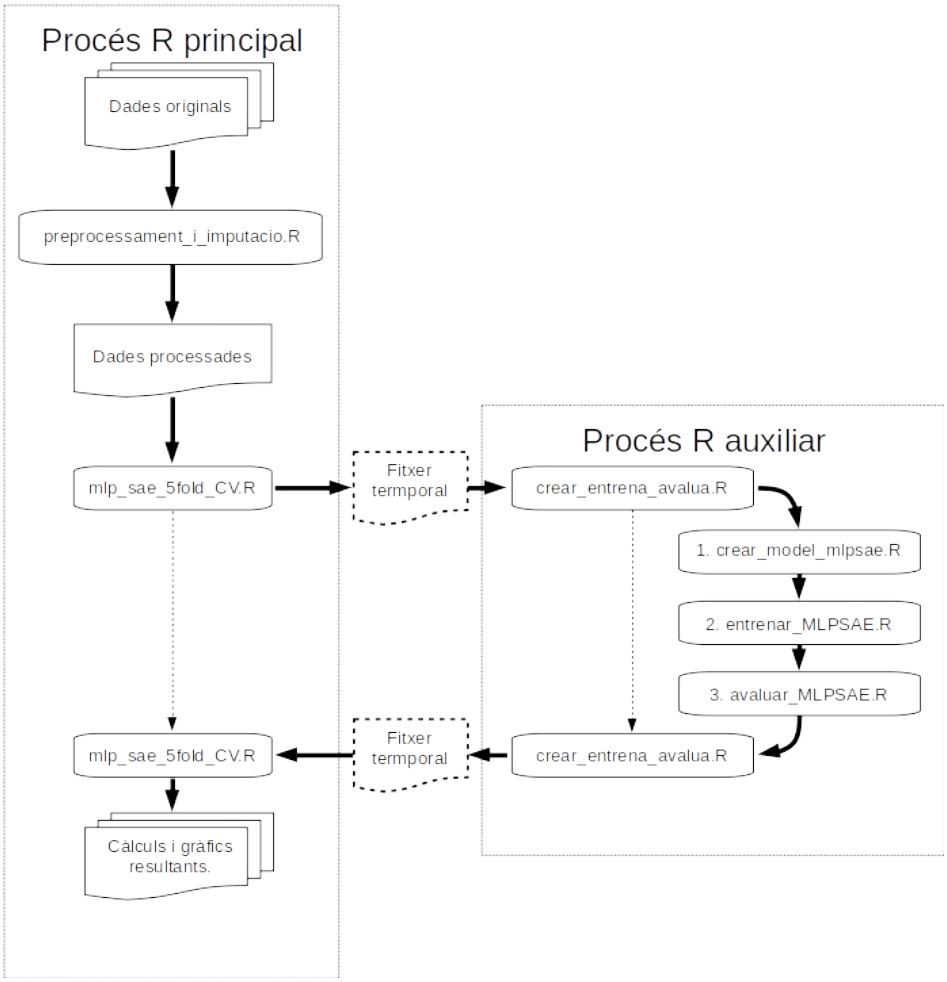
10. Mekala H. Dealing with Missing Data using R – Coinmonks – Medium [Internet]. 2018 [cited 2019 May 28]. Available from: https://medium.com/coinmonks/dealing-with-missing-data-using-r-3ae428da2d17

11. Stekhoven DJ, Buhlmann P. MissForest--non-parametric missing value imputation for mixed-type data. Bioinformatics [Internet]. 2012 Jan 1 [cited 2019 May 7];28(1):112–8. Available from: http://www.ncbi.nlm.nih.gov/pubmed/22039212

12. Bottou L, Curtis FE, Nocedal J. Optimization Methods for Large-Scale Machine Learning. SIAM Rev [Internet]. 2018 Jan [cited 2019 May 30];60(2):223–311. Available from: https://epubs.siam.org/doi/10.1137/16M1080173

6 Annexos

6.1 Codi font.



Il·lustració 24: Esquema del flux d'execució del codi en els dos processos d'entorn de R del sistema.

6.1.1 preprocessament_i_imputacio.R

Aquesta secció del codi carrega els fitxers de dades que posa a disposició els autors de l'article de Xie et al. (1), realitza el processat, aplica el procés d'imputació de les dades i les desa en un arxiu en el format necessari per al seu ús en la creació i l'entrenament del model per part de *«mlp_sae_5fold_CV.R»*

```
# Jordi Masip Masip - Treball de final de màster.
# Màster universitari de Bioinformàtica i Bioestadística (UOC-UB)
# Llicència Creative Common. Reconeixement-CompartirIgual CC BY-SA
# 3.0 España de Creative Commons
# https://creativecommons.org/licenses/by-sa/3.0/es/


#Preprocessament de les dades:
```

```
 9
10 #SNP: carrega, conversió a categòrica, conversió
11 #   a tensor amb variables "dummy" per cada categoria de cada variant de SNP
12 #   inicialitza els noms de directoris i fitxers i carrega de llibreries
13
14 #HTSeq counts: carrega. Eliminació gens amb >30% de NA. Imputació de la resta de
15 #   NA amb l'algoritme de missForest
16
17
18 source("directoris_i_funcions.R")
19
20 # Carregar fitxers -------------------------------------------------------
21
22 #SNPs
23 #Lectura SNP
24 snps.original <- read.table(
25   file.path(dir.dades.originals,"matrix_genotypes.txt"),
26   header = TRUE,
27   sep = "\t",
28   stringsAsFactors = FALSE
29 )
30
31 #Lectura HTSeq conunts
32 matriu.expressio.original <- read.table(
33   file.path(dir.dades.originals,"expression_matrix.txt"),
34   header = TRUE,
35   sep = "\t",
36   row.names = 1,
37   stringsAsFactors = FALSE,
38   na.strings = "NULL"
39 )
40
41
42 # Processar HTSeq ---------------------------------------------------------
43
44
45 #Convertir a numeric
46 matriu.expressio.original <-
47   data.frame(lapply(matriu.expressio.original, as.numeric))
48
49 #Eliminar gens amb més de 30% de missings.
50 htseq.missing <- is.na(matriu.expressio.original)
51 p.missings.per.gen <- rowSums(htseq.missing)/ncol(htseq.missing)
52 matriu.expressio.original<-matriu.expressio.original[p.missings.per.gen<0.3,]
53
54 #Queden 6383 gens, amb un global de 3.823633% de valors perduts.
55 #missForest per imputació valors perduts.
56
57
58 matriu.expressio.original <- t(matriu.expressio.original)
59
60 if (file.exists(f.htseq.imp)) {
61   load(file = f.htseq.imp)
62 } else {
63   library(missForest)
64   library(doParallel)
65   registerDoParallel(cores = detectCores()-1)
66   matriu.expressio.imp <-
67     missForest(
68       matriu.expressio.original,
69       variablewise = TRUE,
70       verbose = TRUE,
71       parallelize = "variables",
72       maxiter = 24
73     )
74
75   save(matriu.expressio.imp, file = f.htseq.imp)
76 }
77
78 #refusar gens en el que el NRMSE >= SD
```

```
htseq.refusats.NRMSE.alt <- which(matriu.expressio.imp$OOBerror>=1)
T_HTSeq <- data.frame(matriu.expressio.imp$ximp[,-htseq.refusats.NRMSE.alt])



# Processar SNPs -------------------------------------------------------


#Eliminar primera columna de noms
snps.original <- data.frame(snps.original[,-1])

#Pasar a factor
snps.original <- converteix.a.factor(snps.original)

#Reordenar les observacions per a que corresponguin al mateix individuo que els
#HTSeq counts.
snps.original <- snps.original[rownames(T_HTSeq),]
T_SNP <-to_categorical(as.matrix(snps.original))

#canvia les observacions, de 2D-tensor a 1D-tensor
n.observacions <- nrow(snps.original)
input_shape <- prod(dim(T_SNP)[-1])
T_SNP <- array_reshape(T_SNP, c(n.observacions, input_shape))
rownames(T_SNP) <- rownames(snps.original)


# Desar les sortides ---------------------------------------------------

save("T_SNP", file = f.SNP)
save("T_HTSeq", file = f.htseq.c)
```

6.1.2 *mlp_sae_5fold_CV.R*

Aquest fitxer de codi coordina la part central del treball. Recupera les dades en el fitxer i realitza una validació creuada en 5 iteracions, reconstruint la matriu de prediccions. Finalment realitza els càlculs i gràfics utilitzats per a realitzar aquesta memòria. Cada iteració és executada en un procés diferent per el fitxer «*crear_entrena_avalua.R*», utilitzant un fitxer temporal per l'intercanvi de les dades necessàries.

```
# Jordi Masip Masip - Treball de final de màster.
# Màster universitari de Bioinformàtica i Bioestadística (UOC-UB)
# Llicència Creative Common. Reconeixement-CompartirIgual CC BY-SA
# 3.0 España de Creative Commons
# https://creativecommons.org/licenses/by-sa/3.0/es/


# Realitzar 5-fold Cross Validation del model MLS-SAE creat.
# S'obté la matriu de HTSeq counts predita, i les mètriques dels autoencoders.

#inicialitza els noms de directoris i fitxers i carrega de llibreries
source("directoris_i_funcions.R")


require(keras)
require(caret)

FLAGS <- recupera.FLAGS()

#Carrega Tensors
carrega.dades()

```

```
23 #Definex els paràmetres d'entrada
24 n.obs <- nrow(T_SNP)
25 index.obs <- seq(n.obs)
26 snp_shape <- ncol(T_SNP)
27 htseq_shape <- ncol(T_HTSeq)
28
29 #Crear els folds
30 grups.cv <- createFolds(index.obs, k = 5)
31
32 #Crea-entrena-avalua cada "fold"
33
34 # Resultats de les prediccions dels models.
35 matriu.predita <- matrix(NA,ncol = htseq_shape, nrow = 0) # Prediccions dels models.
36
37 #metriques d'avaulació de les diferents parts del model.
38 mse.mlpsae.parcials <- acc.dae.parcials <- acc.sae.parcials <- numeric(0)
39 i <- 1 # fold index.
40 for (i.VALIDATE in grups.cv){
41   print(paste0("Entrenant fold: ",i))
42   i <- i + 1
43
44   #crea index TRAIN.
45   i.TRAIN <- index.obs[-i.VALIDATE]
46
47   #crea els tensors de dades
48   segrega.dades(
49     T_SNP = T_SNP,
50     htseq.c = T_HTSeq,
51     i.TRAIN = i.TRAIN,
52     i.VALIDATE = i.VALIDATE,
53     i.TEST = NULL)
54     #Retorna T_snp_* i T_htseq_*
55
56   #Desa les dades per l'entrenament.
57   save(T_htseq_TRAIN, T_htseq_VALIDATE, T_htseq_TEST,
58        T_snp_TRAIN, T_snp_VALIDATE, T_snp_TEST,
59        FLAGS, dir.source,
60        file = f.temporal)
61
62   #Crear, entrena i evalua el model MLPSAE
63   ordre <- paste0('Rscript --vanilla "',
64                   file.path(dir.source, "crear_entrena_avalua.R"),'"')
65   system(ordre)
66
67   #Recupera resultats
68   load(file = f.temporal)
69   file.remove(f.temporal)
70
71   #valors predits
72   matriu.predita <- rbind(matriu.predita,
73                           valors.predits)
74   mse.mlpsae.parcials <- c(mse.mlpsae.parcials,
75                            mlpsae.val.mse)
76   acc.dae.parcials <- c(acc.dae.parcials,
77                         dae.val.acc)
78   acc.sae.parcials <- c(acc.sae.parcials,
79                         sae.val.acc)
80
81 } #fi "for" de CV
82
83 #Calcular R^2
84
85 #Matriu de observacions reals
86 m_HTSeq <- as.matrix(T_HTSeq)
87
88 tests.correlacions <- lapply(seq(htseq_shape),
89                        function(i){cor.test(matriu.predita[,i], m_HTSeq[,i])})
90
91       #funcio auxiliar
92       extreu.parametre.llista <-function(llista,parametre){
```

```
        unname(unlist(lapply(llista, function(l){l[[parametre]]})))}

correlacions <- extreu.parametre.llista(tests.correlacions, "estimate")
p.valors <- extreu.parametre.llista(tests.correlacions, "p.value")
R2.tots <- correlacions^2
R2 <- mean(R2.tots)

#Replica de corr.R
i.200.majors.R2 <- order(correlacions, decreasing = TRUE)[1:200]


# Replica de grafics de la publicació -------------------------------------


#El seguents segments de codi són una adaptació del codi original del treball de
#Xie R, Wen J, Quitadamo A, Cheng J, Shi X.
#A deep auto-encoder model for gene expression prediction.
#BMC Genomics [Internet]. 2017 Nov 17 [cited 2019 Mar 3];18(S9):845.
#Available from: https://bmcgenomics.biomedcentral.com/articles/10.1186/s12864-017-4226-0

#Codi original disponible a:
#https://github.com/shilab/MLP-SAE

#Sota la llicencia "Creative Commons Attribution 4.0"
#https://creativecommons.org/licenses/by/4.0/

library(reshape)
library(ggplot2)

#Gràfic de la comparació de les mitjanes expresades i predites de tots els gens del model.
png(filename = "Comparacio_expression_predites_observadest%03d.png", width = 14, height = 7,
    units = "in", pointsize = 50, res = 100)

#preparació dades
me <- cbind(colMeans(matriu.predita), colMeans(m_HTSeq))
me.df <- melt(me)
me.df$X1 <- as.numeric(me.df$X1)
me.df$X2 <- as.factor(me.df$X2)

ps <- ggplot(me.df,aes(x=X1, y=value)) +
  geom_line(aes(group=X2,color=X2)) +
  geom_point(aes(shape=X2,color=X2), size = 1) +
  geom_hline(yintercept = 0,size=0.4) +
     scale_color_discrete(name="",labels=c("Expressió genètica estimada","Expressió genètica observada")) +
    scale_shape(name="",labels=c("Expressió genètica estimada","Expressió genètica observada")) +
  scale_y_continuous(breaks = seq(-1.0,1.5, by = 0.5),limits = c(-1.0, 1.5),expand = c(0.005,0.005)) +
  xlab('\nTotalitat de gens del model') + ylab("Mitjana de l'expressió genètica de les mostres\n") +
  theme(legend.text = element_text(size = 8),legend.background = element_rect(fill = 'white'), legend.position = "bottom", legend.title = element_blank()) +
  theme(panel.background = element_rect(fill = "white",colour = "grey50")) +
  theme(panel.border = element_rect(linetype = "solid", fill = NA)) +
  theme(plot.background = element_rect(fill = "white", colour = "grey50")) +
  theme(plot.caption=element_text(size=8, hjust=1.0, margin=margin(t=10)));
ps


#Gràfic de la comparació de les mitjanes expresades i predites dels 200 gens amb
#una R^2 més gran.
me <- cbind(colMeans(matriu.predita[,i.200.majors.R2]), colMeans(m_HTSeq[,i.200.majors.R2]))
me.df <- melt(me)
me.df$X1 <- as.numeric(me.df$X1)
me.df$X2 <- as.factor(me.df$X2)
```

```
ps <- ggplot(me.df,aes(x=X1, y=value)) +
  geom_line(aes(group=X2,color=X2)) +
  geom_point(aes(shape=X2,color=X2)) +
  geom_hline(yintercept = 0,size=0.4) +
     scale_color_discrete(name="",labels=c("Expressió genètica estimada","Expressió genètica observada")) +
    scale_shape(name="",labels=c("Expressió genètica estimada","Expressió genètica observada")) +
  scale_y_continuous(breaks = seq(-1.0,1.5, by = 0.5),limits = c(-1.0, 1.5),expand = c(0.005,0.005)) +
  scale_x_continuous(breaks = seq(800,880, by = 10),expand = c(0.005,0.005)) +
  xlab('\nGens seleccionats') + ylab("Mitjana de l'expressió genètica de les mostres\n") +
   theme(legend.text = element_text(size = 8),legend.background = element_rect(fill = 'white'), legend.position = "bottom", legend.title = element_blank()) +
  theme(panel.background = element_rect(fill = "white",colour = "grey50")) +
  theme(panel.border = element_rect(linetype = "solid", fill = NA)) +
  theme(plot.background = element_rect(fill = "white", colour = "grey50")) +
  theme(plot.caption=element_text(size=8, hjust=1.0, margin=margin(t=10)));
ps


#Reproducció de la figura Figura 4, comparant el log(nombre de gens) per rang de R^2
R2.bin <- table(cut(R2.tots, breaks=c(0,0.05,seq(0.1,0.8, by=0.10))))
R2.bin.article.TFM<- data.frame (corr.square = factor(names(R2.bin)),
                                 Original = log10(c(3507, 1121, 1086, 493, 229, 110, 43, 13, 2)),
                                TFM = log10(as.vector(R2.bin)))

#Dades des de la taula 2 de l'article original.
me.df <- melt(R2.bin.article.TFM, variable_name = "Origen")


ps <- ggplot(me.df,aes(x=corr.square, y=value)) +
  geom_line(aes(group=Origen,color=Origen),size=1) +
  geom_point(aes(shape=Origen,color=Origen),size=1.5) +
     scale_color_discrete(name="",labels=c("MLP-SAE amb dropout original","MLP-SAE TFM")) +
  scale_shape(name="",labels=c("MLP-SAE amb dropout original","MLP-SAE TFM")) +
     scale_y_continuous(breaks = seq(0,4, by = 1),limits = c(0, 4),expand = c(0.01,0.01)) +
  scale_x_discrete(breaks = unique(me.df$corr.square)[seq(1,8,1)]) +
  xlab(bquote("R"^"2")) + ylab('log(Recompte de gens)\n') +
   theme(legend.text = element_text(size = 8),legend.background = element_rect(fill = 'white'), legend.position = "bottom", legend.title = element_blank()) +
   theme(panel.grid.major = element_line(colour = "grey", linetype = "dotted",size = 0.3), panel.grid.major.y = element_blank(),panel.grid.minor.y = element_blank()) +
  theme(panel.background = element_rect(fill = "white",colour = "grey50")) +
  theme(panel.border = element_rect(linetype = "solid", fill = NA)) +
  theme(plot.background = element_rect(fill = "white", colour = "grey50")) +
  theme(plot.caption=element_text(size=8, hjust=1.0, margin=margin(t=10)));
ps
dev.off()

#Grafics matriu predita vs observada segons expressió mitja del gen.

png(filename = "scatterplot_predit_vs_obserbat_segons_expressio_genetica%03d.png",
    width = 1000, height = 1000, pointsize = 25)

#Gràfic gens amb expressió < 1 SD respecte a l'expressió base del control
tmp <- abs(colMeans(m_HTSeq)) < 1
plot(
  m_HTSeq[, tmp],
  matriu.predita[, tmp],
  pch = 19,
  cex = 0.1,
  asp = 1,
  col = gray(0.5),
  main = "Expressió predita vs observada",
```

```
  sub = "Gens amb expressió mitja < 1 SD del control",
  ylab="Expressió predita",
  xlab="Expressió observada"
)
abline(v=c(-5,-1,0,1,5), h=c(-5,-1,0,1,5), lty=3, cex = 0.5, col=gray(0.2))
points(
  colMeans(matriu.predita[, tmp]),
  colMeans(m_HTSeq[, tmp]),
  pch = 19,
  cex = 0.5,
  asp = 1,
  col = "red"
)
legend(
  x = "bottomright",
  legend = c("Expressió individual", "Mitjana del gen"),
  col = c(gray(0.5), "red"),
  pch = 19
)

#Gràfic gens amb expressió > 1 SD respecte a l'expressió base del control
tmp <- abs(colMeans(m_HTSeq)) > 1
plot(
  m_HTSeq[, tmp],
  matriu.predita[, tmp],
  pch = 19,
  cex = 0.1,
  asp = 1,
  col = gray(0.5),
  main = "Expressió predita vs observada",
  sub = "Gens amb expressió mitja > 1 SD del control",
  ylab="Expressió predita",
  xlab="Expressió observada"
)
abline(v=c(-5,-1,0,1,5), h=c(-5,-1,0,1,5), lty=3, cex = 0.5, col=gray(0.2))
points(
  colMeans(matriu.predita[, tmp]),
  colMeans(m_HTSeq[, tmp]),
  pch = 19,
  cex = 0.5,
  asp = 1,
  col = "red"
)
legend(
  x = "bottomright",
  legend = c("Expressió individual", "Mitjana del gen"),
  col = c(gray(0.5), "red"),
  pch = 19
)

plot(
  m_HTSeq,
  matriu.predita,
  pch = 19,
  cex = 0.1,
  asp = 1,
  col = gray(0.5),
  main = "Expressió predita vs observada",
  ylab="Expressió predita",
  xlab="Expressió observada"
)
abline(v=c(-5,-1,0,1,5), h=c(-5,-1,0,1,5), lty=3, cex = 0.5, col=gray(0.2))

dev.off()


#Grafic mitjana expressio vs R^2
png(filename = "mitjana_expresio_vs_R2.png",
    width = 1000, height = 1000, pointsize = 25)
MEAN.gens <- colMeans(m_HTSeq)
```

```
MEAN.gens.predit <- colMeans(matriu.predita)
plot(MEAN.gens, R2.tots,
     main = bquote("Mitja d'expressió vs R"^"2"),
     ylab= bquote("R"^"2"),
     xlab="Mitjana d'expressió del gen",
     pch=19,
     cex = 0.2
     )
abline(v=c(-1,0,1), lty=3, col="red2")
dev.off()

#Grafic QQ plot de les mitjes d'expressio genètica predit vs observat
png(filename = "QQ_plot_mitjes_expressio.png",
    width = 1000, height = 1000, pointsize = 25)
qqplot(MEAN.gens, MEAN.gens.predit, asp = 1, main="Mitjanes d'expressió observades vs predites",
       sub="Gràfic Quantil-Quantil de les expressió mitjanes dels gens",
       ylab="Predit",
       xlab="Observat",
       pch=19,
       cex=0.2)
abline(a=0,b=1,col="red")
dev.off()


#Distrivucció de les mitjanes predites vs observades.
png(filename = "comparaccio_distribucio_expressions.png",
    width = 10, height = 10,units = "cm", pointsize = 25, res = 300)
MEANS <- data.frame(Observat = MEAN.gens, Predit = MEAN.gens.predit)
MEANS <- melt(MEANS)
grf <- ggplot(data = MEANS, aes(value)) +
   geom_density(data=subset(x = MEANS,subset = variable=="Observat"),fill = "red", alpha=0.2) +
   geom_density(data=subset(x = MEANS,subset = variable=="Predit"),fill = "green", alpha=0.2) +
  theme_light()+ theme(plot.title = element_text(hjust = 0.5))  +
  ylab("") + xlab("Mitja")+ggtitle("Distribució de les mitjes d'expressió")
grf
dev.off()



#Calcular MSE amb les matrius escalades amb sistema minmax01 tal com es defineix
#en el procés de preprocessament de l'article
    escalat01 <- function(x, rang.dades = NULL){
      if (is.null(rang.dades)) {rang.dades  <- c(min(x),max(x))}
      x <- (x- rang.dades[1]) / (rang.dades[2] - rang.dades[1])
      return(x)}

matriu.predita.01 <- apply(matriu.predita, 2, escalat01)
m_HTSeq.01 <- apply(m_HTSeq, 2, escalat01)
mse01 <-  mean((matriu.predita.01 - m_HTSeq.01)^2)

mse.tots  <-  sapply(seq(ncol(m_HTSeq)),  function  (i){mean((matriu.predita[,i]  - m_HTSeq[,i])^2)})
MSE <- mean(mse.tots)

save(
  matriu.predita,
  m_HTSeq,
  tests.correlacions,
  correlacions,
  p.valors,
  R2.tots,
  R2,
  mse.mlpsae.parcials,
  acc.dae.parcials,
  acc.sae.parcials,
  mse01,
  mse.tots,
```

```
  MSE,
  file = "Resultat.model.MLPSAE.RData"
)
```

6.1.3 crear_entrena_avalua.R

Carrega les dades d'un arxiu temporal i realitza l'entrenament i avaluació del model. Retorna a través d'un fitxer temporal la matriu de prediccions de les lectures del grup de validació, i el paràmetres de rendiment de les diferents parts de l'entrenament.

```
# Jordi Masip Masip - Treball de final de màster.
# Màster universitari de Bioinformàtica i Bioestadística (UOC-UB)
# Llicència Creative Common. Reconeixement-CompartirIgual CC BY-SA
# 3.0 España de Creative Commons
# https://creativecommons.org/licenses/by-sa/3.0/es/


# Crea el MLPSAE
# Entrena el MLPSAE
# Retorna:
#   Matriu de prediccions sobre el grup VALIDATE
#   Val_MSE del MLPSAE
#   Val_MSE del DAE


arg <- commandArgs(trailingOnly = TRUE)

library(keras)
library(caret)
source("directoris_i_funcions.R")

#Carrega els tensors d'entrenament.
load(f.temporal)
file.remove(f.temporal)


#Crea els tensors d'entrada al model, incoent els generadors de soroll.
snp_shape <- ncol(T_snp_TRAIN)
htseq_shape <- ncol(T_htseq_TRAIN)
crear.capes.entrada.mlpsae(input_shape =snp_shape,
                           n.TRAIN = nrow(T_snp_TRAIN),
                           n.VALIDATE = nrow(T_snp_VALIDATE),
                           n.TEST = NULL)

#Parametres:
#   input_shape: shape del tensor de SNP, sense el batch
#   n.TRAIN, n.VALIDATE, n.TEST. El número de observacions de cada grup.
#   n.TEST és NULL per defecte, no generant el tensor de soroll corresponent.
#           Si n.TEST és no NULL, es creen els tensors per poder validar
#           amb dades noves.  No utilitzat en el Cross-Validation.
#Genera:
#   entrada.SNP, entrada.soroll: "place-holders" per a les entrades.
#   T_soroll_TRAIN, T_soroll_VALIDATE. També T_sorroll_TEST si n.TEST no NULL.


#Crear el model
source(file = file.path(dir.source, "crear_model_mlpsae.R"))

#Entrenar el model
source(file = file.path(dir.source, "entrenar_MLPSAE.R"))

#avalua el model
source(file = file.path(dir.source, "avaluar_MLPSAE.R"))
```

```


#Desa els valors de rendiment i els valors predits del grup validate
  mlpsae.val.mse <- rendiment.mlpsae[[2]]
  sae.val.acc <-
    tail(historia.SAE.p2$metrics$val_acc, n = 1)
  dae.val.acc <- tail(historia.DAE1$metrics$val_acc, n = 1)

  save(valors.predits, mlpsae.val.mse, sae.val.acc, dae.val.acc,
       file = f.temporal)
```

6.1.4 *crear_model_mlpsae.R*

Donada l'existència en l'entorn de les variables adequades que defineixen el model, aquest codi crea el model Keras per a posterior ús.

```
# Jordi Masip Masip - Treball de final de màster.
# Màster universitari de Bioinformàtica i Bioestadística (UOC-UB)
# Llicència Creative Common. Reconeixement-CompartirIgual CC BY-SA
# 3.0 España de Creative Commons
# https://creativecommons.org/licenses/by-sa/3.0/es/


  #parametres dels models
  fun.optimizer <<- optimizer_rmsprop(lr = 1e-4)
  fun.regularize.dae1 <<- regularizer_l2(l=FLAGS$dae1_encoder1_regularizer_l2)
  fun.regularize.ae2 <<- regularizer_l2(l=FLAGS$ae2_encoder1_regularizer_l2)
  fun.regularize.mlpsae <<- regularizer_l2(l = FLAGS$mlpsae_regularizer_l2)


  #callbacks:
  #Ajust learning rate segons epoca
  train.DAE1.model.callbacks <<-
    list(
      callback_early_stopping(monitor = "val_loss",
                              patience = FLAGS$early_stop_patience,
                              restore_best_weights = TRUE),
      callback_reduce_lr_on_plateau(
        monitor = "val_loss",
        factor = 0.5,
        patience = 100,
        cooldown = 100,
        min_lr = 1e-5
      )
    )
  train.SAE.model.callbacks <<-
    list(
      callback_early_stopping(monitor = "val_loss",
                              patience = FLAGS$early_stop_patience,
                              restore_best_weights = TRUE),
      callback_reduce_lr_on_plateau(
        monitor = "val_loss",
        factor = 0.5,
        patience = 100,
        cooldown = 100,
        min_lr = 1e-5
      )
    )

  train.model.callbacks.mlpsae.p1 <-
    list(
      callback_early_stopping(monitor = "val_loss",
                              patience = FLAGS$early_stop_patience,
                              restore_best_weights = TRUE),
```

```
49        callback_reduce_lr_on_plateau(
50          monitor = "val_loss",
51          factor = 0.5,
52          patience = 100,
53          cooldown = 100,
54          min_lr = 1e-5
55        )
56      )
57
58
59    train.model.callbacks.mlpsae.p2 <-
60      list(
61        callback_early_stopping(monitor = "val_loss",
62                                patience = FLAGS$early_stop_patience,
63                                restore_best_weights = TRUE),
64        callback_reduce_lr_on_plateau(
65          monitor = "val_loss",
66          factor = 0.5,
67          patience = 100,
68          cooldown = 100,
69          min_lr = 1e-5
70        )
71      )
72
73 # DAE1 encoder ------------------------------------------------------------
74 dae1.encoder <-
75   layer_maximum(inputs = c(entrada.SNP, entrada.soroll),
76                 name = "agregador_de_soroll") %>%
77   layer_dense(
78     name = "dae1_encoder1",
79     units = FLAGS$dae1_encoder1_units,
80     activation = "relu",
81     kernel_regularizer = fun.regularize.dae1
82   ) %>%
83   layer_dense(
84     name = "dae1_encoder2",
85     units = FLAGS$dae1_encoder2_units,
86     activation = "relu",
87     kernel_regularizer = fun.regularize.dae1
88   ) %>%
89   layer_dropout(rate = FLAGS$droprate, name = "dae1_dropout") %>%
90   layer_dense(
91     name = "dae1_model_comprimit",
92     units = FLAGS$dae1_model_comprimit,
93     activation = "relu",
94     kernel_regularizer = fun.regularize.dae1
95   )
96
97
98 # model DAE1 ---------------------------------------------------------------
99
100 dae1.decoder <-
101   dae1.encoder %>%
102   layer_dense(
103     name = "dae1_sortida",
104     units = snp_shape,
105     activation = "sigmoid",
106     kernel_regularizer = fun.regularize.dae1
107   )
108
109 DAE1 <-
110   keras_model(inputs = c(entrada.SNP, entrada.soroll),
111               outputs = dae1.decoder)
112
113
114 # autoencoder 2 ------------------------------------------------------------------------
115 ae2.encoder <-
116   dae1.encoder %>%
117   layer_dense(name="ae2_encoder1",
118               units = floor(mean(FLAGS$ae2_model_comprimit,
```

```
                                FLAGS$dae1_model_comprimit)),
              activation = "relu",
              kernel_regularizer = fun.regularize.ae2) %>%
  layer_dropout(rate=FLAGS$droprate) %>%
  layer_dense(name="ae2_model_comprimit",
              units = FLAGS$ae2_model_comprimit,
              activation = "relu",
              kernel_regularizer = fun.regularize.ae2)


# SAE -------------------------------------------------------------------



ae2.decoder <-
  ae2.encoder %>%
  layer_dense(name="sortida_SAE",
              units = snp_shape,
              activation = "sigmoid",
              kernel_regularizer = fun.regularize.ae2)

SAE <- keras_model(inputs=c(entrada.SNP, entrada.soroll), outputs = ae2.decoder)


# MLPSAE ----------------------------------------------------------------

mlpsae <- ae2.encoder %>%
  layer_dense(units = FLAGS$interna_mlpsae1_units,
              activation = "relu",
              name = "interna_mlpsae1",
              kernel_regularizer = fun.regularize.mlpsae) %>%
  layer_dense(units = FLAGS$interna_mlpsae2_units,
              activation = "tanh",
              name = "interna_mlpsae2",
              kernel_regularizer = fun.regularize.mlpsae
              ) %>%
  layer_dense(units = htseq_shape,
              activation = "linear",
              name = "sortida_mlpsae")
model.mlpsae <-
  keras_model(inputs = list(entrada.SNP, entrada.soroll),
              outputs = mlpsae)
```

6.1.5 entrenar_MLPSAE.R

Donada l'existència del model Keras, i les variables adequades amb les dades per l'entrenament del model, aquest codi realitza l'entrenament de les diferents parts del model i l'entrenament final del mateix, així com realitza les prediccions dels valors de la matriu de validació.

```
# Jordi Masip Masip - Treball de final de màster.
# Màster universitari de Bioinformàtica i Bioestadística (UOC-UB)
# Llicència Creative Common. Reconeixement-CompartirIgual CC BY-SA
# 3.0 España de Creative Commons
# https://creativecommons.org/licenses/by-sa/3.0/es/


# Pre-entrenament DAE1 --------------------------------------------------

  #Entrenar DAE1
DAE1 %>% compile(optimizer = fun.optimizer,
                 loss = "binary_crossentropy",
```

```
13                        metrics = "accuracy")
14
15 historia.DAE1 <- DAE1 %>% fit(
16   x = list(T_snp_TRAIN, T_soroll_TRAIN),
17   y = list(T_snp_TRAIN),
18   validation_data = list(list(T_snp_VALIDATE, T_soroll_VALIDATE),
19                          T_snp_VALIDATE),
20   validation_steps =1,
21   steps_per_epoch = 1,
22   callbacks = train.DAE1.model.callbacks,
23   verbose = FLAGS$train_verbose,
24   epochs = FLAGS$epoques_dae1
25 )
26
27
28
29 # Pre-.enrenament SAE -----------------------------------------------------
30
31
32   #Entrenar SAE
33     #Congelar DAE1
34 DAE1 %>% freeze_weights()
35     #Entrenament 1 SAE
36 DAE1 %>%  compile(optimizer = fun.optimizer,
37                   loss = "binary_crossentropy",
38                   metrics = "accuracy")
39 SAE %>% compile(optimizer = fun.optimizer,
40                 loss = "binary_crossentropy",
41                 metrics = "accuracy")
42 historia.SAE.p1 <- SAE %>% fit(
43   x = list(T_snp_TRAIN, T_soroll_TRAIN),
44   y = list(T_snp_TRAIN),
45   validation_data = list(list(T_snp_VALIDATE, T_soroll_VALIDATE),
46                          T_snp_VALIDATE),
47   validation_steps =1,
48   steps_per_epoch = 1,
49   callbacks = train.SAE.model.callbacks,
50   verbose = FLAGS$train_verbose,
51   epochs = FLAGS$epoques_ae2_p1
52 )
53     #Descongelar part DAE1
54 DAE1 %>% unfreeze_weights(from = "dae1_encoder2" )
55 DAE1 %>%  compile(optimizer = fun.optimizer,
56                   loss = "binary_crossentropy",
57                   metrics = "accuracy")
58 SAE %>% compile(optimizer = fun.optimizer,
59                 loss = "binary_crossentropy",
60                 metrics = "accuracy")
61     #Fine-tunning SAE
62 historia.SAE.p2 <- SAE %>% fit(
63   x = list(T_snp_TRAIN, T_soroll_TRAIN),
64   y = list(T_snp_TRAIN),
65   validation_data = list(list(T_snp_VALIDATE, T_soroll_VALIDATE),
66                          T_snp_VALIDATE),
67   validation_steps =1,
68   steps_per_epoch = 1,
69   callbacks = train.SAE.model.callbacks,
70   verbose = FLAGS$train_verbose,
71   epochs = FLAGS$epoques_ae2_p2
72 )
73
74 # Entrenament MLPSAE -----------------------------------------------------
75
76
77 #Entrenar MLPSAE
78   #Congelar SAE
79 SAE %>% freeze_weights()
80 model.mlpsae %>% compile(optimizer = fun.optimizer,
81                        loss = "mse",
82                        metrics = "mse")
```

```

  #Entrenament 1 MLPSAE

historia.mlpsae.p1 <- model.mlpsae %>% fit(
  x = list(T_snp_TRAIN, T_soroll_TRAIN),
  y = T_htseq_TRAIN,
  validation_data = list(list(T_snp_VALIDATE,T_soroll_VALIDATE), T_htseq_VALIDATE),
  steps = 1,
  validation_steps = 1,
  verbose = FLAGS$train_verbose,
  callbacks = train.model.callbacks.mlpsae.p1,
  epochs = FLAGS$epoques_mlpsae_p1
)
  #Descongelar part SAE
unfreeze_weights(SAE, from = "ae2_encoder1")
model.mlpsae %>% compile(optimizer = fun.optimizer, loss = "mse", metrics = "mse")
SAE %>% compile(optimizer = fun.optimizer, loss = "mse", metrics = "mse")
  #Fine-tunning SAE

historia.mlpsae.p2 <- model.mlpsae %>% fit(
  x = list(T_snp_TRAIN, T_soroll_TRAIN),
  y = T_htseq_TRAIN,
  validation_data = list(list(T_snp_VALIDATE, T_soroll_VALIDATE), T_htseq_VALIDATE),
  steps = 1,
  validation_steps = 1,
  verbose = FLAGS$train_verbose,
  callbacks = train.model.callbacks.mlpsae.p2,
  epochs = FLAGS$epoques_mlpsae_p2
)
```

6.1.6 *avaluar_MLPSAE.R*

Donada l'existència del model Keras entrenat, i de les variables creades durant el seu entrenament, aquest codi realitza uns pocs càlculs del rendiment del model no realitzats en les fases prèvies.

```
# Jordi Masip Masip - Treball de final de màster.
# Màster universitari de Bioinformàtica i Bioestadística (UOC-UB)
# Llicència Creative Common. Reconeixement-CompartirIgual CC BY-SA
# 3.0 España de Creative Commons
# https://creativecommons.org/licenses/by-sa/3.0/es/


#Calcular valors fit
#Amb TEST si definit, amb Validate si no.

#Si hi ha conjunt de dades de TEST, fer-les servir.
if(is.null(T_soroll_TEST)){
  snp_test <- T_snp_VALIDATE
  htseq_test <- T_htseq_VALIDATE
  soroll_test <- T_soroll_VALIDATE
} else {
  snp_test <- T_snp_TEST
  htseq_test <- T_htseq_TEST
  soroll_test <- T_soroll_TEST
    }

#Calcular MSE
valors.predits <- model.mlpsae %>% predict(x=list(snp_test, soroll_test),
                                           steps=1)
rendiment.mlpsae <- model.mlpsae %>% evaluate(x=list(snp_test, soroll_test),
                                              y=htseq_test,
                                              steps=1)
```

6.1.7 funcions_suport.R

Recopilació de funcions que s'utilitzen en diferents punts del codi. Donada la execució separada de les diferents fases de l'entrenament del codi, aquest fitxer facilita el recuperar funcions que s'utilitzen en més d'un indret i evita tenir-les duplicades per facilitar el manteniment i correcció de les mateixes.

```
# Jordi Masip Masip - Treball de final de màster.
# Màster universitari de Bioinformàtica i Bioestadística (UOC-UB)
# Llicència Creative Common. Reconeixement-CompartirIgual CC BY-SA
# 3.0 España de Creative Commons
# https://creativecommons.org/licenses/by-sa/3.0/es/


#Conversior de SNPs numeric o caracter a FACTOR.
converteix.a.factor  <- function(data.frame.entrada){
  #Converteix data.frame a columnes format FACTOR, assegurant-se de que totes
  #les columnes tinguin els mateixos LEVELS

  #determina els nivells existents en tot el data.frame
  nivells.globals <-
    levels(factor(unname(unlist(data.frame.entrada))))
  #converteix les columnes a factor, amb els nivells de tot el data.frame
  data.frame.sortida <-
    data.frame(apply(data.frame.entrada, 1,
                     factor, levels = nivells.globals))
  return(data.frame.sortida)
}


# Functions 5-fold Cross-Validation ---------------------------------------


#FLAGS del model.
recupera.FLAGS <- function (){
  FLAGS <<- flags(
    flag_integer(name = "dae1_encoder1_units",
                 default =  6158, # original 6158
                 description =  "Unitats primera capa DAE1"),
    flag_integer(name = "dae1_encoder2_units",
                 default = 3448, # original 3448
                 description = "Unitats segona capa DAE1"),
    flag_integer(name = "dae1_model_comprimit",
                 default = 739, # original 739
                 description = "Unitats representació reduida DAE1"),
    flag_integer(name = "ae2_model_comprimit",
                 default = 184,  # original 184
                 description = "Unitats representacio reduida SAE"),
    flag_integer(name = "interna_mlpsae1_units",
                 default = 2000, # original 2000
                 description = "Unitats primera capa sortida MLPSAE"),
    flag_integer(name = "interna_mlpsae2_units",
                 default = 2000, # original 2000
                 description = "Unitats segona capa sortida MLPSAE"),
    flag_numeric(name = "dae1_encoder1_regularizer_l2",
                 default = 0.0002,
                 description = "Valor regularizer L2 entrenament SAE1"),
    flag_numeric(name = "ae2_encoder1_regularizer_l2",
                 default = 0.00005,
                 description = "Valor regularizer L2 entrenament AE2"),
    flag_numeric(name = "mlpsae_regularizer_l2",
                 default = 0.00005, #Original 0.0001
                 description = "Valor regularizer L2 entrenament MLPSAE"),
```

```
 57     flag_numeric(name = "prop_soroll",
 58                  default = 0.03333333,
 59                  description = "Proporció soroll afegit a les dades SNPs"),
 60     flag_numeric(name = "droprate",
 61                  default =  0.2,
 62                  description = "Proporció de 'drop out'"),
 63     flag_integer(name = "epoques_dae1",
 64                  default = 1000, # original 600
 65                  "Número de epoques de pre-entrenament DAE1"),
 66     flag_integer(name = "epoques_ae2_p1",
 67                  default = 1000, # original 350
 68                  description = "Epoques de la primera part entrenament SAE"),
 69     flag_integer(name = "epoques_ae2_p2",
 70                  default = 2500, # original 850
 71                  description = "Epoques de la segona part entrenament SAE"),
 72     flag_integer(name="epoques_mlpsae_p1",
 73                  default = 1000,
 74                  description = "Epoques de la primera part entrenament MLPSAE"),
 75     flag_integer(name = "epoques_mlpsae_p2",
 76                  default = 2500), # original 300,
 77     flag_integer(name = "train_verbose",
 78                  default = 0,
 79                  description = "Valor de 'verbose' a passar a la funció fit()"),
 80     flag_integer((name = "early_stop_patience"),
 81                  default = 250,
 82                                  description  =  "Valor  paràmetres  'patience'  de
    callback_early_stopping().")
 83   )
 84 }
 85
 86 # Funcions CREA-ENTRENA-AVALUA -------------------------------------------
 87
 88 carrega.dades <- function()
 89   #Carrega els tensors de l'expressió SNP i HTseq preparats, a l'entorn global.
 90 {
 91   load(file = f.htseq.c, envir = .GlobalEnv)
 92   load(file = f.SNP, envir = .GlobalEnv)
 93 }
 94
 95
 96 segrega.dades <- function(T_SNP,
 97                          htseq.c,
 98                          i.TRAIN,
 99                          i.VALIDATE,
100                          i.TEST){
101   #fun auxiliar local
102   segrega <- function(M, i){as.matrix(M[i,])}
103
104   T_htseq_TRAIN <<- segrega(htseq.c,i.TRAIN)
105   T_htseq_VALIDATE <<-segrega(htseq.c,i.VALIDATE)
106   T_htseq_TEST <<- segrega(htseq.c,i.TEST)
107   T_snp_TRAIN <<- segrega(T_SNP,i.TRAIN)
108   T_snp_VALIDATE <<- segrega(T_SNP,i.VALIDATE)
109   T_snp_TEST <<- segrega(T_SNP,i.TEST)
110
111 }
112
113 #Definir els Tensors d'entrada, per dades i per soroll al MLP-SAE
114
115 crear.capes.entrada.mlpsae <- function(input_shape,
116                                        n.TRAIN,
117                                        n.VALIDATE,
118                                        n.TEST = NULL)
119 {
120   entrada.SNP <<-
121     layer_input(shape = c(NULL, input_shape),
122                 name = "entrada_SNP",
123                 dtype = 'float32')
124   entrada.soroll <<-
125     layer_input(shape = c(NULL, input_shape),
```

```
                name = "font_de_soroll",
                dtype = 'float32')

  #Tensors per a l'enrada de soroll.
  T_soroll_TRAIN <<-
    k_random_binomial(shape = c(n.TRAIN, input_shape),
                      p = FLAGS$prop_soroll,
                      dtype = "float32")
  T_soroll_VALIDATE <<-
    k_zeros(shape = c(n.VALIDATE, input_shape), dtype = "float32")
  if (!is.null(n.TEST)) {
    T_sorroll_TEST <<-
      k_zeros(shape = c(n.TEST, input_shape), dtype = "float32")} else {
        T_soroll_TEST <<- NULL
      }

}
```

6.1.8 directoris_i_funcions.R

Aquest codi es crida al inici de qualsevol procés per definir les funcions i els noms dels directoris i fitxers que s'utilitzen durant les diferents fases del procés.

```
# Jordi Masip Masip - Treball de final de màster.
# Màster universitari de Bioinformàtica i Bioestadística (UOC-UB)
# Llicència Creative Common. Reconeixement-CompartirIgual CC BY-SA
# 3.0 España de Creative Commons
# https://creativecommons.org/licenses/by-sa/3.0/es/


#Funcions.

  # -executar els fitxers de funcions de suport
  # -crear les variables per apuntar a fitxers i funcions.

#carrega les llibreries
library(keras)
library(caret)


# Definir directoris -----------------------------------------------------

f.funcions.suport <- "funcions_suport.R" #nom fitxers de funcions a reutilitzar

#directori general, fa escapament de espais.
dir.treball <- paste(unlist(strsplit(getwd(), " ")), collapse =  "\ ")
dir.source <- dir.treball

#dades
dir.dades.originals <- file.path(dir.treball,"dades_originals")


#Dades finals HTseq.c (netes i imputades)
f.htseq.c <- file.path(dir.dades.originals, "htseq.counts.RData")

#Objecte final de missForest (fitxer final d'imputació)
f.htseq.imp <- file.path(dir.dades.originals, "htseq.imp.RData")

#Dades finals SNP (to_categorical -> resape)
f.SNP <- file.path(dir.dades.originals, "Tensor_SNP.RData")

#Fitxers Crea-Entrena-Avalua
f.temporal <- file.path(dir.treball, "fitxer_temporal_intercanvi.RData")
```

```


# Executar codi funcions suport ----------------------------------------

source(file = f.funcions.suport)


# Finalitzacio -------------------------------------------------------
```