Universitat
de les Illes Balears

TREBALL DE FI DE GRAU

GRAPH NEURAL NETWORKS EN L'ANÀLISI DE XARXES METABÒLIQUES: PREDICCIÓ DE GENS ESSENCIALS

Miquel Àngel Llauger Suau

Grau de Matemàtiques

Escola Politècnica Superior

Any acadèmic 2024-25

GRAPH NEURAL NETWORKS EN L'ANÀLISI DE XARXES METABÒLIQUES: PREDICCIÓ DE GENS ESSENCIALS

Miquel Àngel Llauger Suau

Treball de Fi de Grau

Escola Politècnica Superior

Universitat de les Illes Balears

Any acadèmic 2024-25

Paraules clau del treball: TFG, memòria, LaTeX

Tutor: Maria de la Mercè Llabrés Segura

Autoritz la Universitat a incloure aquest treball en el repositori institucional per consultar-lo en accés obert i difondre'l en línia, amb finalitats exclusivament acadèmiques i d'investigació	Autor/a Sí	Autor/a No	Tutor/a Sí	Tutor/a No
	☑	☐	☑	☐

Als meus pares,

SUMARI

ACRÒNIMS

ANN Artificial Neural Network

FBA Flux Balance Analysis

GAT Graph Attention

GNN Graph Neural Network

GPR Gene-To-Protein Reaction

IA Intel·ligència Artificial

MFG Mass Flow Graph

NFG Normalised Flow Graph

NPV Negative predictive value

PRAUC Precision-Recall Area Under Curve

SGD Stochastic Gradient Descent

TFG Treball de Fi de Grau

Resum

Aquest treball aborda un problema fonamental de la biologia computacional: identificar quins gens són essencials per a la supervivència cel·lular. Els gens essencials són aquells que, quan s'eliminen, comprometen la viabilitat d'una cèl·lula. Aquesta identificació pot ser crucial per trobar objectius terapèutics per al càncer, optimitzar la producció de compostos d'alt valor en la biotecnologia o, simplement, per comprendre els mòduls funcionals mínims necessaris per la vida cel·lular.

Es definiran una sèrie d'eines matemàtiques bàsiques com la Teoria de Grafs i l'Optimització, i s'introduirà el mètode computacional més utilitzat per a la determinació d'essencialitat gènica: el *Flux Balance Analysis*. A més, es repassaran les matemàtiques que s'amaguen darrere les Artificial Neural Network (ANN) amb una sèrie d'exemples, fent especial esment a les Graph Neural Network (GNN).

El cos del treball consisteix en la construcció d'un mètode híbrid que millora les tècniques computacionals tradicionals mitjançant la incorporació dels Mass Flow Graphs (MFG) per representar xarxes metabòliques i la implementació d'una Graph Neural Network entrenada amb dades extretes d'assajos. Tot plegat ens conduirà a un procediment general per a la determinació d'essencialitat gènica de qualsevol bacteri, amb un rendiment que supera lleugerament el dels mètodes computacionals tradicionals en el cas de l'*Escherichia Coli*. Concretament, arribem a un model d'Intel·ligència Artificial (IA) entrenat amb tres espècies completament distintes i capaç de capturar l'essencialitat de qualsevol gen dins d'una xarxa metabòlica.

CAPÍTOL 1

INTRODUCCIÓ

La identificació de gens essencials constitueix un dels reptes més fonamentals de la biologia computacional moderna. Un gen essencial es defineix com aquell que, quan s'elimina o s'inactiva, compromet la viabilitat cel·lular o impedeix la supervivència de l'organisme en condicions específiques. La determinació experimental d'essencialitat gènica ha evolucionat significativament amb l'arribada de tecnologies d'alt rendiment. Malgrat aquests avenços tecnològics, els assajos experimentals continuen sent costosos, laboriosos i requereixen temps considerable per obtenir resultats complets [1].

El Flux Balance Analysis (FBA) s'ha establert com el mètode computacional de referència per a la predicció d'essencialitat gènica en gens metabòlics. Aquest mètode es basa en principis d'optimització per calcular distribucions de flux metabòlic a escala genòmica que maximitzen un objectiu cel·lular específic, típicament la taxa de creixement. L'eficiència computacional del FBA constitueix una de les seves principals avantatges. Imposant una sèrie de restriccions sobre cada flux metabòlic, els problemes de FBA els podem resoldre amb algorismes de programació lineal eficients, que permeten simular de manera immediata l'eliminació de gens [2]. Malgrat els èxits del FBA en organismes model com l'*Escherichia Coli*, les seves prediccions per organismes eucariotes i de major complexitat han donat resultats dubtosos. La limitació més crítica del FBA rau en **l'assumpció d'optimalitat universal**. És a dir, la suposició que, una soca salvatge i una amb restriccions de flux, busquen optimitzar la mateixa funció objectiu. Si bé aquesta assumpció es pot considerar certa en el cas d'una soca salvatge, no ho és en el cas de soques d'eliminació mutants, que redirigeixen el seu metabolisme cap a altres objectius de supervivència.

Les limitacions del FBA tradicional han motivat l'exploració d'aproximacions alternatives, com la integració de mètodes d'Aprenentatge Automàtic amb modelització metabòlica. Els Mass Flow Graph (MFG) representen una innovació metodològica significativa en la representació de xarxes metabòliques. Tot incorporant la distribució de flux obtinguda en resoldre un problema de FBA, els MFG codifiquen la direccionalitat del flux dels metabòlits dins d'una xarxa. Aquesta representació ofereix avantatges substancials sobre construccions tradicionals de grafs metabòlics. Les *Graph Neural Network* (GNN) han demostrat un èxit notable en una àmplia varietat de tasques que involucren dades estructurades com a grafs.La capacitat de les GNNs per capturar dependències locals i globals en estructures de graf les converteix en candidates ideals per a l'anàlisi de xarxes metabòliques.

L'objectiu principal d'aquest treball és desenvolupar un mètode computacional innovador

que superi les limitacions dels enfocaments tradicionals per a la predicció d'essencialitat gènica. Els objectius específics d'aquest treball inclouen:

1. Contribució metodològica: Donar un mètode que no necessiti de l'assumpció d'optimalitat universal per a la predicció d'essencialitat gènica.
2. Comprensió del model matemàtic: L'elaboració d'un text matemàtic que proporcioni una anàlisi rigorosa dels procediments i assumpcions tot incloent demostracions i teoremes que donen suport a la teoria.
3. Validació empírica: Demostrar l'eficàcia del mètode proposat usant dades experimentals d'alta qualitat d'*Escherichia Coli*, l'organisme model més ben caracteritzat metabòlicament.
4. Anàlisi de generalització: Esbrinar si es pot estendre el mètode a altres tipus de bacteris, com el *Bacillus Subtillis* i el *Mycobacterium tuberculosis*.

Partint del model *FlowGAT* [1], en aquest Treball de Fi de Grau (TFG) s'implementaran de bell nou tres models híbrids similars capaços de combinar la modelització del FBA amb la potència de les Xarxes Neuronals per a capturar patrons complexes en les dades. La memòria d'aquest treball s'estructura de la següent manera:

- **Capítol 2:** proporciona els fonaments teòrics necessaris, incloent-hi Teoria de Grafs i Optimització matemàtica. Així mateix, es defineixen conceptes com *estoiquiometria* i *xarxa metabòlica*.
- **Capítol 3:** defineix rigorosament els models matemàtics presents en la reconstrucció de xarxes metabòliques. Donam eines per a la resolució de problemes de *Flux Balance Analysis* i seguim el procediment per a la construcció dels Mass Flow Graphs.
- **Capítol 4:** introdueix les Artificial Neural Networks (ANN) tot partint de la idea bàsica darrere del Perceptró. Amb demostracions i exemples s'avança cap a la definició de les Graph Neural Networks (GNN).
- **Capítol 5:** es presenten els detalls en l'arquitectura general del model entrenat amb dades de l'*Escherichia Coli*. Es fa un repàs sobre les diferents mètriques de rendiment en classificació binària. Es presenten els resultats obtinguts amb els tres models, incloent un model entrenat amb dades del bacteri *Bacillus Subtilis*. Finalment es discuteix la possiblitat d'un model generalitzat.

Per avaluar el rendiment dels models, s'han utilitzat dades d'assajos disponibles en línia [3]. Trobareu l'enllaç a tots els repositoris als Annexos.

CAPÍTOL 2

PRELIMINARS

En aquest capítol s'introduiran les definicions dels objectes matemàtics que es tractaran al llarg del treball, així com la notació que s'hi farà servir. Amb l'objectiu que aquest treball sigui el màxim autocontingut possible, donarem les definicions bàsiques de la Teoria de Grafs, Optimització i de Xarxes Metabòliques.

2.1 Teoria de Grafs

Un *graf* G és un parell ordenat de conjunts (V, E), amb V finit i no buit i E un subconjunt de parells (ordenats o desordenats) d'elements de V [4]. V és el conjunt de *vèrtexos* o *nodes* de G, que sovint denotarem com V(G), i $E = E(G)$ n'és el conjunt d'arestes. Definim l'*ordre* d'un graf com el seu nombre de vèrtexos $|\mathrm{V}(G)|$ i la seva *mida* $|\mathrm{E}(G)|$, com el nombre d'arestes que conté.

Distingirem entre *grafs dirigits* i *no dirigits* quan donem, o no, una orientació a les seves arestes, respectivament. Així, en un graf dirigit, una aresta $e = (u, v) \in E$ és un element del producte cartesià $V \times V$ que és distint de l'aresta $e' = (v, u)$. Intuitivament, hom pot pensar que no és el mateix anar del node u al v, que anar del node v al u. D'altra banda, en grafs no dirigits, no tindrem en compte la direcció de la connexió entre nodes: $(u, v) = (v, u) = \{u, v\}$. Per a facilitar-ne l'escriptura, és comú indicar les arestes del graf per uv en lloc de posar (u, v) indistintament si treballam en grafs dirigits o no dirigits, amb el benentés que, si el graf és dirigit $uv \neq vu$ i, si és no dirigit, $uv = vu$.

Una aresta que uneix dos vèrtexos iguals s'anomena un *llaç*. Direm que dos vèrtexos són *adjacents* si hi ha una aresta que els uneix; altrament direm que són *independents*.

Definim el *subgraf induït* (o generat) de G per $S \subset V(G)$ com $G[S] = (S, E')$ tal que $E' = \{uv \in E | u, v \in S\}$.

Els grafs s'acostumen a representar per dibuixos al pla, amb els vèrtexos representats per punts i les arestes per línies que uneixen els seus extrems. És comú representar les arestes per línies rectes, però això no és necesari. La representació de les arestes només ha d'indicar la connexió entre els corresponents vèrtexos. Ara bé, més endavant veurem que, en grafs de gran mida, aquesta representació ja no és tan clarificadora.

Donat un node $v \in V$ d'un graf no dirigit, podem definir el seu *grau* com el nombre de nodes

als quals és adjacent:

$$d(v) = |\{v \in V : uv \in E, u \neq v\}|.$$

En el cas dirigit, definim el grau d'entrada d'un node com el nombre d'arestes que tenen el node com a node final. Formalment,

$$d_e(v) = |\{u \in V | (u,v) \in E, v \neq u\}|,$$

i definim el grau de sortida d'un node com el nombre d'arestes que tenen el node com a node inicial. Formalment,

$$d_s(v) = |\{u \in V | (v,u) \in E, v \neq u\}|.$$

Definirem un *graf amb etiquetes* com un graf $G = (V,E)$ que té una sola etiqueta o un sol vector associat a cada node mitjançant la funció $l : V \to L$, on L és el conjunt d'etiquetes, o $f : V \to \mathbb{R}$, respectivament. Aquest últim cas el farem servir més endavant en la construcció d'un vector de característiques associat a cada node.

Un *graf ponderat* o amb pesos és un graf $G = (V,E)$ que té una funció $w : E \to \mathbb{R}$ que assigna pesos reals a les arestes.

Sigui G un graf i siguin $u, v \in V$. Un *recorregut* dins G de u a v (o que porta u a v, o que connecta u i v) és una seqüència de nodes $(v_0, v_1, \ldots, v_k)$ tal que:

- $u = v_0$ i $v = v_k$
- $v_i v_{i+1} \in E$ per a tot $i = 0, \ldots, k-1$.

Els nodes u i v s'anomenen els *extrems* del recorregut. La resta de nodes, $v_1, \ldots, v_{k-1}$ s'anomenen els *nodes intermedis*. Notem que en la definició de recorregut no s'exclou la possibilitat de tenir repetició de vèrtexos i arestes. El nombre d'arestes que es recorren, és a dir k, s'anomena la *longitud* del recorregut. Un *camí* dins G de v_0 a v_k és un recorregut $(v_0, v_1, ..., v_k)$ tal que:

- $v_i \neq v_j \ \forall i, j \in \{0, ..., k-1\}$.

Definim un *cicle* com un camí que conclou amb el primer node.

Definim la *matriu d'adjacència* d'un graf G com una matriu quadrada $A = (a_{u,v})$ de dimensió $n \times n$, on $n = |V|$ com:

$$a_{u,v} = \begin{cases} 1 & \text{si } (u,v) \in E \\ 0 & \text{altrament.} \end{cases}$$

Si, a més, G és ponderat i $w : E \to \mathbb{R}$ és la funció que assigna pesos a les arestes, aleshores la seva matriu d'adjacència és

$$a_{u,v} = \begin{cases} w((u,v)) & \text{si } (u,v) \in E \\ 0 & \text{altrament.} \end{cases}$$

Exemple 1. *Considerem el següent graf dirigit i ponderat (veure Figura 2.1):*

$$G = (\{1,2,3,4,5\}, \{(1,2),(1,3),(2,4),(2,3),(3,5),(4,5),(5,1)\}).$$

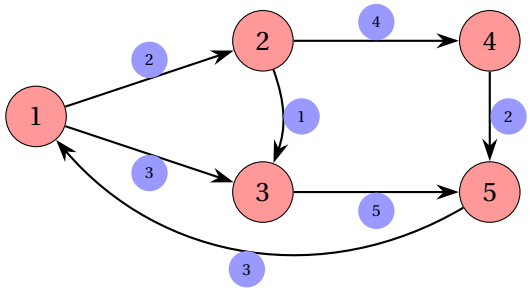


Figura 2.1: Representació del graf ponderat de l'exemple 1.

És fàcil comprovar que, amb la definició que hem donat anteriorment, la seva matriu d'adjacència seria:

$$A_G = \begin{pmatrix} 0 & 2 & 3 & 0 & 0 \\ 0 & 0 & 1 & 4 & 0 \\ 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 & 2 \\ 3 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Proposició 2. *Sigui A^k la potència k-èssima de la matriu d'adjacència d'un graf no dirigit G i sigui $a_{i,j}^{(k)}$ l'entrada en la fila i, columna j de la matriu A^k. Llavors $a_{i,j}^{(k)}$ compta el nombre de recorreguts entre v_i i v_j de longitud k.*

Demostració. Vegem-ho per inducció sobre k:

Pel cas base, està clar que $a_{i,j} \neq 0$ sii existeix una aresta que connecta v_i amb v_j. Per fer el pas inductiu, suposem que és cert per un cert $k \in \mathbb{N}$ i escrivim $A^{k+1} = A^k A$:

$$a_{i,j}^{(k+1)} = \sum_{l=1}^{n} a_{i,l}^{(k)} a_{l,j}^{(1)}.$$

Una condició necessària per què no s'annul·li el sumatori és que, per algun $l \in \{1 \dots n\}$:

$$\begin{cases} a_{i,l}^{(k)} \neq 0 & \text{i.e } \exists R_k = (u_i \dots . u_l) \text{ recorregut de longitud k (hipòtesi d'inducció)} \\ a_{l,j}^{(1)} \neq 0 & \text{i.e } \exists (u_l, u_j) \in E(G). \end{cases}$$

Ara, concatenant R_k amb l'aresta (u_l, u_j) obtenim un nou recorregut R_{k+1} de longitud $k+1$. En efecte, cada terme del sumatori és, o bé 0, o bé 1, en funció de si existeix un recorregut de longitud $k+1$ entre cada parell de nodes. Per tant, s'està comptant el nombre de recorreguts, com volíem veure.

□

Definim el *quadrat d'un graf* [5] no dirigit $G = (V, E)$, ho denotarem com $G^2 = (V, E')$, com el graf que té els mateixos nodes que G i que té per arestes aquelles que uneixen dos nodes a distància menor o igual a 2 en el graf original. És a dir, en G^2 dos nodes seran adjacents si i només si, existeix un camí de longitud 2 o són adjacents en G.

Un graf *bipartit* és un graf no dirigit on el conjunt de nodes es pot dividir en dos conjunts disjunts tal no hi hagi arestes entre nodes del mateix conjunt. Formalment, $G = (V, E)$ és bipartit si existeixen dos conjunts A i B tals que $V = A \cup B$, $A \cap B = \emptyset$ i $uv \in E \rightarrow (u \in A \wedge v \in B) \oplus (u \in B \wedge v \in A)$. En aquest cas, sovint escriurem $G = (U, V, E)$ de tal manera que, la matriu d'adjacència ens queda:

$$A = \begin{pmatrix} 0_r & B \\ B^\top & 0_s \end{pmatrix},$$

on B és una matriu $r \times s$, i 0_r i 0_s representen les matrius zero de dimensions r i s respectivament, on $|U| = r$ i $|V| = s$. En aquest cas, la matriu més petita B representa de manera única el graf, i la resta de parts d'A es poden descartar com a redundants. B s'anomena de vegades la *matriu de biadjacència*.

La *meitat bipartida* o *meitat quadrada* d'un graf bipartit $G = (U, V, E)$ és el graf $G^2[U]$ on el superíndex 2 denota el quadrat d'un graf i els claudàtors denoten el seu subgraf induït. El resultat és un graf tal que el seu conjunt de vèrtexs és un dels dos costats de la bipartició (sense pèrdua de generalitat, U) i tal que hi ha una aresta $u_i u_j$ per a cada parell de vèrtexs u_i, u_j en U si i només si, estan a distància menor o igual a 2 en G.

Lema 3. *La matriu d'adjacència de $G^2[U]$ és $B^T B$, on B és la matriu de biadjacència del graf bipartit $G = (U, V, E)$.*

Demostració. En efecte, sigui A la matriu d'adjacència de G. Aleshores, per ser bipartit, la podem escriure com:

$$A = \begin{pmatrix} 0_{r,r} & B \\ B^\top & 0_{s,s} \end{pmatrix},$$

on B té dimensions $|U| = r \times s = |V|$. Aleshores la matriu que ens dóna els recorreguts de longitud 2 (els de longitud mínima que uneixen membres de U i V amb membres de la mateixa classe) és:

$$A^2 = \begin{pmatrix} BB^T & 0 \\ 0 & B^T B \end{pmatrix},$$

Resulta obvi doncs, que BB^T coincideix amb el subgraf generat per U del graf quadrat G^2.

□

Un *multigraf* $\mathscr{M}$ és un graf que pot tenir arestes múltiples (també anomenades arestes paral·leles), és a dir, arestes que tenen els mateixos nodes extrems. Així, dos nodes de $\mathscr{M}$ poden estar connectats per més d'una aresta.

Hi ha 2 nocions diferents d'arestes múltiples:

- *Arestes sense identitat pròpia*: La identitat d'una aresta es defineix únicament pels dos nodes que connecta. En aquest cas, el terme "arestes múltiples" significa que la mateixa aresta pot aparèixer diverses vegades entre aquests dos nodes.
- *Arestes amb identitat pròpia*: Les arestes són entitats primitives com els nodes. Quan múltiples arestes connecten dos nodes, aquestes són arestes diferents.

En aquest treball, usarem la noció d'arestes amb identitat pròpia. Definim un *multidígraf* (o multigraf dirigit) com una 4-tupla $G = (V, A, s, t)$ amb

- V un conjunt de nodes.
- A un conjunt d'arestes.
- $s : A \to V$ que assigna a cada aresta el seu node de partida.
- $t : A \to V$ que assigna a cada aresta el seu node d'arribada.

Aquesta noció de multidígraf ens permet capturar la idea que, donats dos nodes u i v, es pugui anar d'un a l'altre indistintament. A la definició anterior podem afegir-hi etiquetes i pesos de la següent manera:

- Σ_V, Σ_A alfabets finits dels vèrtexos i arestes disponibles.
- $l_V : V \to \Sigma_V$ i $l_A : A \to \Sigma_A$ funcions que assignen l'etiquetatge de nodes i els pesos a les arestes, respectivament.

2.2 Xarxes metabòliques

Una *xarxa metabòlica* [6] es refereix a l'entramat de metabòlits i les seves interconversions (reaccions bioquímiques) en un organisme. Els *metabòlits* són normalment molècules petites com la glucosa i els aminoàcids, però també poden ser macromolècules com polisacàrids i glicans. La interconversió normalment és catalitzada per *enzims* (proteïnes). Només algunes reaccions a la cèl·lula són espontànies i, per tant, no enzimàtiques.

Una *via metabòlica* pot considerar-se com una petita àrea local d'una xarxa metabòlica, sovint associada a una funció metabòlica específica, mentre que una xarxa metabòlica ofereix una visió millor i més completa del metabolisme cel·lular. Una xarxa metabòlica completa hauria de mostrar tots els possibles modes de flux de materials a la cèl·lula, indicant així tot el potencial i la capacitat metabòlica de la cèl·lula. En altres paraules, la xarxa metabòlica és el centre de processament de materials per a una cèl·lula funcional. La cèl·lula depèn d'aquesta xarxa per absorbir i digerir substrats del medi ambient, per generar energia i per sintetitzar components que són necessaris per al seu creixement i supervivència.
Una reacció es diu que és *reversible* quan pot ocórrer en els dos sentits, és a dir, els productes es comporten com a reactius de la reacció inversa, que dona com a producte els reactius originals. Diferents factors, com són el tipus de reacció, la concentració dels metabòlits, la temperatura o la pressió poden condicionar que una reacció sigui reversible o no [7].

Una *equació estoiquiomètrica* [8] és una representació d'una reacció química amb les proporcions exactes entre els reactius i productes segons la llei de conservació de la massa. La combustió del metà n'és un exemple:

$$CH_4 + 2O_2 \rightarrow CO_2 + 2H_2O.$$

La *matriu estoiquiomètrica* d'una reacció és una representació matricial de l'equilibri estoiquiomètric. Les files de la matriu es corresponen amb els diferents composts i les columnes amb les reaccions. D'aquesta manera, cada element indica el coeficient estoiquiomètric de cada espècie en cada reacció, i el signe de cada coeficient depèn de l'orientació escollida en la reacció. Vegi's el següent exemple de jugueta.

Exemple 4. *Suposem que tenim 4 metabòlits A,B,C i D que intervenen en dues reaccions R_1 i R_2 de la següent manera:*

$$R_1 : A + 2B \rightarrow C + D$$

$$R_2 : 2A + B \rightarrow C.$$

*Aleshores la matriu estoiquiomètrica **S** del conjunt seria:*

$$S = \begin{bmatrix} -1 & -2 \\ -2 & -1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}.$$

Una xarxa metabòlica consisteix en metabòlits que es converteixen en altres mitjançant reaccions bioquímiques catalitzades per enzims. Així, en un enfocament general es pot representar com un graf amb dos tipus diferents de vèrtexs (metabòlit i reacció). Per simplicitat, les xarxes metabòliques es poden representar com un graf (o multigraf) dirigit, on els vèrtexs representen metabòlits i les arestes corresponen a reaccions que converteixen un metabòlit en un altre. Una manera de modelitzar-ho és amb els *reaction graphs*, que definirem més endavant.

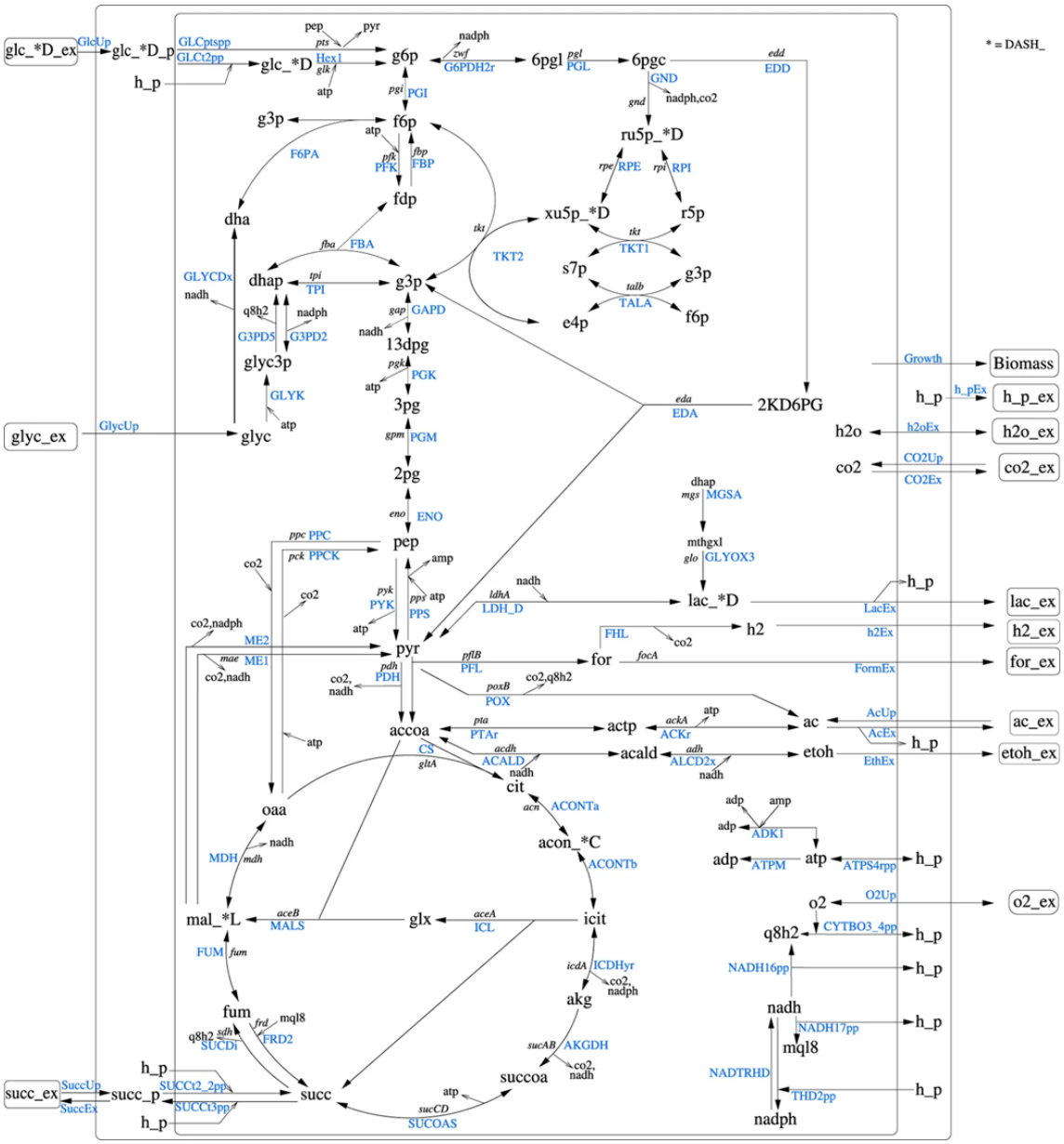


Figura 2.2: Xarxa metabòlica de la bacteria *Escherichia Coli*

2.3 Optimització

L'optimització és una eina fonamental alhora de resoldre problemes que involucren la presa de decisions. La idea de decidir entre un conjunt d'opcions donades es correspon sovint amb la noció d'escollir la "millor". Aquesta mesura de com de bona és una opció ve donada per

una funció *objectiu*. L'optimització busca doncs, trobar la millor alternativa atenent el criteri de la funció objectiu [9]. En aquest treball farem servir mètodes de cerca per a funcions reals $f : \mathbb{R}^n \to \mathbb{R}$ que usen el seu *gradient*.

Recordem que un *conjunt de nivell* d'una funció f és el conjunt de punts $\boldsymbol{x} \in \mathbb{R}^n$ que satisfan $f(\boldsymbol{x}) = k$ per a qualque constant k prefixada. El vector *gradient* d'una funció f en un punt $\boldsymbol{x}_0$, que escrivim com $\nabla f(\boldsymbol{x}_0)$, sempre que sigui no nul, és ortogonal al conjunt de nivell de f en aquest punt. En altres paraules, el vector gradient apunta en la direcció de major creixement de f. Per veure-ho, recordem que $\langle \nabla f(\boldsymbol{x}), \boldsymbol{d} \rangle$, $||\boldsymbol{d}|| = 1$ és la derivada direccional de f en la direcció $\boldsymbol{d}$ en el punt $\boldsymbol{x}$. Aleshores, per la desigualtat de Cauchy-Schwarz,

$$\langle \nabla f(\boldsymbol{x}), \boldsymbol{d} \rangle \leq ||\nabla f(\boldsymbol{x})||$$

ja que $||d|| = 1$. Llavors agafant $\boldsymbol{d} = \nabla f(\boldsymbol{x})/||\nabla f(\boldsymbol{x})||$, tenim

$$\left\langle \nabla f(\boldsymbol{x}), \frac{\nabla f(\boldsymbol{x})}{||\nabla f(\boldsymbol{x})||} \right\rangle = ||\nabla f(\boldsymbol{x})||.$$

És a dir, la direcció del vector gradient $\nabla f(\boldsymbol{x})$ és la de major increment de f en $\boldsymbol{x}$. Anàlogament, la direcció del vector $-\nabla f(\boldsymbol{x})$ és la de màxim decreixement de f en $\boldsymbol{x}$. Suposem ara, que estem buscant minimitzar una funció donada, llavors la direcció del gradient negatiu pot ser útil en aquest sentit. Procedim de la següent manera. Sigui $\boldsymbol{x^{(0)}}$ un punt inicial, i considerem el punt $\boldsymbol{x^{(0)}} - \alpha \nabla f(\boldsymbol{x^{(0)}})$. Aleshores, pel teorema de Taylor obtenim

$$f(\boldsymbol{x^{(0)}} - \alpha \nabla f(\boldsymbol{x^{(0)}}) = f(\boldsymbol{x^{(0)}}) - \alpha ||\nabla f(\boldsymbol{x^{(0)}})||^2 + o(\alpha).$$

Així, si $\nabla f(\boldsymbol{x^{(0)}}) \neq 0$, aleshores per a $\alpha > 0$ prou petit, tenim

$$f(\boldsymbol{x^{(0)}} - \alpha \nabla f(\boldsymbol{x^{(0)}})) < f(\boldsymbol{x^{(0)}}).$$

Això vol dir que, el punt $\boldsymbol{x^{(0)}} - \alpha \nabla f(\boldsymbol{x^{(0)}})$ és una millora respecte al punt $\boldsymbol{x^{(0)}}$ pel que fa a trobar un minimitzador.

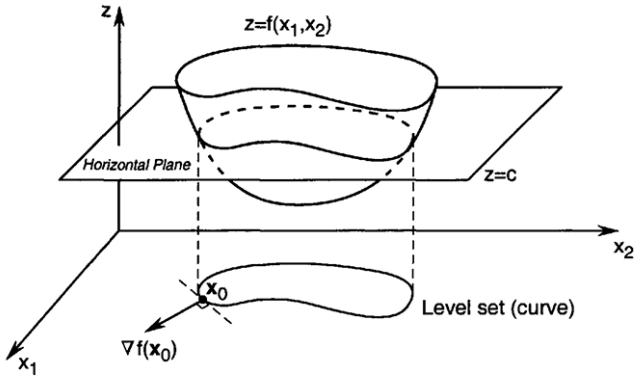


Figura 2.3: El vector gradient és ortogonal a la corba de nivell de f en $\boldsymbol{x}_0$

Per formular un algorisme que implementi la idea anterior, suposem que ens donen un punt $\boldsymbol{x}^{(k)}$. Per trobar el següent punt $\boldsymbol{x}^{(k+1)}$, començem a $\boldsymbol{x}^{(k)}$ i ens movem una quantitat $-\alpha_k \nabla f(\boldsymbol{x}^{(k)})$, on α_k és un escalar positiu anomenat el *pas*. El procediment anterior porta al següent algorisme iteratiu:

$$x^{(k+1)} = x^{(k)} - \alpha_k \nabla f(x^{(k)}).$$

Ens referim a n'aquest tipus d'algorismes com algorismes de descens del gradient (o algorismes de gradient).

Exemple 5. *Considerem el següent problema d'optimització dins el marc de l'Aprenentatge Automàtic. Una de les tasques més bàsiques dins el món de la IA que, a més, tractarem en profunditat en aquest treball és la classificació binària. Suposem que tenim n mostres d'una taula de dades numèrica, on cada observació $x^{(i)}$ té una etiqueta $y_i \in \{0,1\}$. El nostre objectiu és predir les etiquetes de cada x_i amb una funció $h(x)$ que depèn d'un cert paràmetre θ en el sentit que, volem minimitzar una funció d'error. La funció de cost* Binary Cross-Entropy *es pot implementar com:*

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$

on:

- $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$ *és la funció sigmoide*
- $y^{(i)} \in \{0,1\}$ *és l'etiqueta real de la mostra i*
- $x^{(i)}$ *és el vector de característiques de la mostra i*
- θ *és el vector de paràmetres del model*

El gradient de la funció Cross-Entropy respecte als paràmetres θ és:

$$\nabla_\theta J(\theta) = \frac{1}{n}\sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})x^{(i)}$$

En forma vectorial:

$$\nabla_\theta J(\theta) = \frac{1}{n}X^T(h_\theta(X) - y)$$

on X és la matriu de característiques i y és el vector d'etiquetes. Considerem un exemple simple amb dues característiques. Sigui:

$$\theta^{(0)} = \begin{bmatrix} 0.1 \\ -0.2 \\ 0.3 \end{bmatrix}, \quad \alpha = 0.01$$

$$X = \begin{bmatrix} 1 & 2.1 & 1.5 \\ 1 & -1.2 & 0.8 \\ 1 & 3.0 & -0.5 \\ 1 & -2.1 & -1.2 \end{bmatrix}, \quad y = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Actualitzarem els paràmetres segons com hem vist anteriorment:

$$\theta^{(k+1)} = \theta^{(k)} - \alpha\nabla_\theta J(\theta^{(k)})$$

Iteració 1:

$$h_{\theta^{(0)}}(X) = \begin{bmatrix} \sigma(0.1+2.1(-0.2)+1.5(0.3)) \\ \sigma(0.1+(-1.2)(-0.2)+0.8(0.3)) \\ \sigma(0.1+3.0(-0.2)+(-0.5)(0.3)) \\ \sigma(0.1+(-2.1)(-0.2)+(-1.2)(0.3)) \end{bmatrix} = \begin{bmatrix} 0.62 \\ 0.71 \\ 0.37 \\ 0.52 \end{bmatrix}$$

$$\nabla_\theta J(\theta^{(0)}) = \frac{1}{4} X^T (h_{\theta^{(0)}}(X) - y) = \frac{1}{4} X^T \begin{bmatrix} -0.38 \\ 0.71 \\ -0.63 \\ 0.52 \end{bmatrix}$$

$$= \frac{1}{4} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2.1 & -1.2 & 3.0 & -2.1 \\ 1.5 & 0.8 & -0.5 & -1.2 \end{bmatrix} \begin{bmatrix} -0.38 \\ 0.71 \\ -0.63 \\ 0.52 \end{bmatrix} = \begin{bmatrix} 0.055 \\ -0.125 \\ -0.085 \end{bmatrix}$$

$$\theta^{(1)} = \theta^{(0)} - \alpha \nabla_\theta J(\theta^{(0)}) = \begin{bmatrix} 0.1 \\ -0.2 \\ 0.3 \end{bmatrix} - 0.01 \begin{bmatrix} 0.055 \\ -0.125 \\ -0.085 \end{bmatrix} = \begin{bmatrix} 0.0994 \\ -0.1987 \\ 0.3009 \end{bmatrix}$$

Aquest procés es repeteix fins que el gradient sigui prou petit o fins que el canvi en el cost sigui negligible.

En el cas de l'Aprenentatge Automàtic, sovint es considera la tasca de minimitzar funcions dels tipus:

$$E(w) = \frac{1}{n} \sum_{i=1}^{n} E_i(w)$$

on es vol estimar el paràmetre w que minimitzi la funció d'error E. Cadascuna de les funcions E_i sol estar associada a una observació (la i-èssima en el conjunt de dades), ben bé com hem fet en l'exemple anterior. En alguns casos, calcular la suma anterior és prou simple i es pot fer amb un cost computacional raonable. Ara bé, en molts casos pot ser molt costós el càlcul del gradient de E, ja que requereix avaluar el gradient de cadascuna de les funcions summand. Especialment en problemes d'Aprenentatge Automàtic a gran escala.

El *Descens de Gradient Estocàstic* o Stochastic Gradient Descent (SGD) en anglès, consisteix en una família de mètodes iteratius que optimitza la funció objectiu corresponent tot considerant una aproximació estocàstica del descens del gradient. El que fa és substituir el gradient real de la funció (que molt sovint fa servir el *dataset* sencer) per una aproximació (calculada a partir d'un subconjunt de dades seleccionades aleatòriament). La idea consisteix en considerar subconjunts de dades anomenats 'mini-lots' (*mini-batch* en anglès) amb els què calcular el gradient a cada pas. Aquest algorisme combinat amb l'algorisme de *backpropagation*, que veurem més endavant, és el que fan servir bona part de les xarxes neuronals en el seu entrenament.

CAPÍTOL 3

RECONSTRUCCIÓ DE XARXES METABÒLIQUES

El conjunt de reaccions metabòliques que tenen lloc dins d'una cèl·lula juguen un paper fonamental en l'obtenció d'energia i en el funcionament general de l'organisme. L'existència de models metabòlics a escala genòmica de diferents organismes, com l'iML1515 de Monk et al. [10] ha permès la reconstrucció i anàlisi d'aquestes xarxes des d'un punt de vista matemàtic o, si més no, informàtic.

Podem veure el metabolisme d'una cèl·lula com una col·lecció de reaccions enzimàtiques associades a un conjunt de funcions cel·lulars. Tanmateix, les reaccions metabòliques estan altament interconnectades: els enzims converteixen múltiples reactius en productes i viceversa; els enzims poden catalitzar diverses reaccions, i algunes reaccions són catalitzades per múltiples enzims, i així successivament. L'anàlisi de xarxes [11] és, per tant, naturalment adequada per l'anàlisi d'aquesta xarxa entrellaçada de reaccions, un enfocament que s'ha aplicat amb èxit a diferents aspectes de la biologia cel·lular i molecular, com ara les interaccions proteïna-proteïna, la regulació transcripcional o l'estructura de les proteïnes.

En aquest capítol, il·lustrarem breument alguns dels mètodes i conceptes més usats en l'estudi de les xarxes metabòliques a escala genòmica i en donarem exemples.

3.1 *Flux Balance Analysis* (FBA)

El FBA és una de les eines més usades en la bioquímica per a la reconstrucció de xarxes metabòliques d'escala genòmica [12]. Aquesta reconstrucció pretén incorporar la totalitat de les reaccions metabòliques conegudes dins un organisme i els gens que codifiquen cada enzim. L'objectiu principal del FBA és doncs, calcular la quantitat de metabòlits que viatgen dins la xarxa, és a dir, el seu flux. Tot fent possible de predir la quantitat de metabòlits produïts per una reacció de la xarxa o la producció de biomassa d'un organisme.

Suposem que tenim una xarxa metabòlica composta per n metabòlits $X_i (i = 1, ..., n)$ que participen en m reaccions:

$$R_j : \sum_{i=1}^{n} \alpha_{ij} X_i \Longleftrightarrow \sum_{i=1}^{n} \beta_{ij} X_i$$

on $\alpha_{i,j}$ i $\beta_{i,j}$ denoten els coeficients estoiquimètrics de X_i en la reacció R_j. Definim el vector n-dimensional de concentracions dels metabòlits $x(t) = (x_1(t), ..., x_n(t))^T$ mesurats en unitats

de concentració per temps. Expressarem la velocitat a què es produeix una reacció bioquímica dins una xarxa, en termes del vector de flux v:

$$v(t) = (v_1(t), ..., v_n(t))^T$$

típicament en unitats de concentració per temps $mmol/gDW/h$ [13].

Quan la única causa en la variació de concentracions dels metabòlits són les pròpies reaccions bioquímiques, l'evolució temporal de les concentracions ve donada per les equacions d'equilibri [13]:

$$\frac{d}{dt}x_i^{'}(t) = \sum_{j=1}^{m} s_{ij}v_j(t)$$

on s_{ij} i v_j denoten el coeficient estoiquiomètric del metabòlit X_i i el flux en la reacció R_j a temps t, respectivament. Aquesta equació la podem escriure en forma matricial com l'equació diferencial lineal:

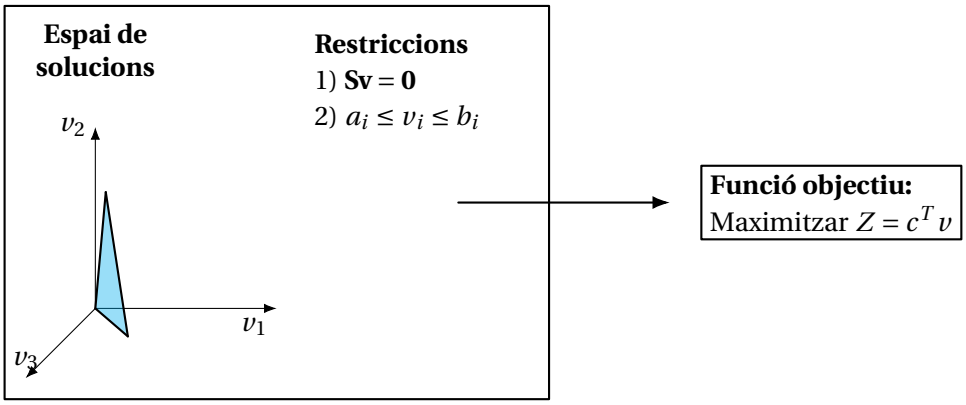
$$\dot{x} = Sv. \tag{3.1}$$

Suposant que ens trobam en un sistema bioquímic en estat d'equilibri, l'equació d'equilibri ens queda:

$$Sv = 0. \tag{3.2}$$

Aquesta equació estoiquiomètrica ens defineix un conjunt de restriccions sobre el flux de metabòlits, una suposició fonamental en l'enfocament del FBA. En un estadi d'equilibri de flux, l'equació anterior ens assegura una igualtat entre la quantitat total de producció de qualsevol compost i la seva consumició. A més, a cada reacció R_i li podem assignar unes restriccions $\alpha_i \leq v_i \leq \beta_i$ en el seu flux que, juntament amb l'assumpció d'equilibri, defineixen l'espai total de distribucions possibles de flux. Alhora d'assignar les fites inferiors i superiors pel flux es té en compte la reversibilitat de cada reacció. Una reacció R_i reversible, tendrà potencialment, una fita inferior $\alpha_i < 0$ i una superior $\beta_i > 0$, indicant que el flux pot anar en una direcció o una altra. Al contrari, per aquelles reaccions no reversibles, necessàriament imposarem $\alpha_i \geq 0$.

La següent passa en el FBA és definir un objectiu biològic de rellevància en el context del problema a estudiar. La funció objectiu més àmpliament usada és la maximització de la formació de biomassa [6]. Típicament, si ens centrem en la taxa de creixement d'un organisme, aleshores cal incidir en la seva producció de biomassa. La representació matemàtica d'una reacció de biomassa forma part de cada model metabòlic i ve determinada a través de mètodes experimentals. Molt sovint, cada model incorpora una reacció artificial extra en la seva estoiquiometria que simula la producció de biomassa. Una vegada hem ubicat l'objectiu dins el model, és possible predir la velocitat màxima de creixement de l'organisme, tot calculant les condicions que faciliten un flux màxim de metabòlits a través d'aquesta reacció. Tot plegat, el que tenim és una funció objectiu i un conjunt de restriccions que plantejarem com un sistema d'equacions lineals per a un problema de Programació Lineal:



on c és un vector constant de pesos i v el vector de flux. Per tant, la nostra funció objectiu Z no és més que una combinació lineal de fluxes [12]. En un model metabòlic realista, venen representades un major nombre de reaccions que de metabòlits ($m < n$) i, per tant, la solució $v*$ és tal que $v* \in Ker(S)$, on $Ker(S) \neq 0$. D'aquesta manera, el FBA pretén donar com a resultat, una distribució òptima del flux donat un conjunt de restriccions amb un sentit biològic.

Una eina àmpliament emprada en la bioinformàtica per a computar els resultats del FBA és la *Constraint-Based Reconstruction and Analysis* (COBRA), disponible en el llenguatge de programació MATLAB, que incorpora el tractament de models metabòlics [2].

3.2 Grafs metabòlics

Amb tota la Teoria de Grafs a la nostra disposició, cal cercar una manera de modelitzar tot allò que coneixem sobre la xarxa metabòlica. Ara bé, hi ha moltes maneres diferents de construir un graf G que modelitzi una xarxa donada. Per exemple, un pot construir un graf amb els metabòlits com a nodes units mitjançant una aresta si, i només si, comparteixen una reacció. Una altra opció és considerar un graf bipartit amb ambdós metàbolits i reaccions en forma de conjunts disjunts de nodes. Tanmateix, els resultats obtinguts depenen molt de la representació escollida.

Una característica clau present en les reaccions metabòliques és la **direccionalitat del flux**: la totalitat de la xarxa metabòlica està formada per reaccions irreversibles i d'altres reversibles, que poden canviar la seva direcció en funció de les condicions ambientals. Per aquest motiu, veurem diferents models que incorporen la direccionalitat del flux en la seva definició així com la probabilitat que un metabòlit d'una reacció sigui produït o consumit per una altra.

En tots els models que veurem a continuació, les reaccions metabòliques venen representades en forma de nodes, i les arestes denoten relacions de producció/consumició de metabòlits. D'aquesta manera, passam a definir el *Reaction Adjacency Graph* (RAG) d'una xarxa metabòlica. Un primer model simplista pel que fa a la informació de flux.

Donada la matriu estoiquiomètrica **S** d'una xarxa metabòlica, considerem la seva versió binària:

$$\hat{S}_{ij} = \begin{cases} 0, & \text{si } S_{ij} = 0, \\ 1, & \text{si } S_{ij} \neq 0. \end{cases}$$

definim el *Reaction Adjacency Graph* de la xarxa com el graf que té per matriu d'adjacència

$$\boldsymbol{A} = \hat{\boldsymbol{S}}^T \hat{\boldsymbol{S}}$$

on $\hat{\boldsymbol{S}}$ és la versió binària de $\boldsymbol{S}$.

Observació 6. *Aquesta definició ve motivada pel fet que, podem veure $\hat{\mathbf{S}}$ com la matriu de biadjacència del graf bipartit (definit als Preliminars) que té per nodes la unió disjunta del conjunt de reaccions i de metabòlits. De tal manera que, el RAG obtingut és el resultat de calcular-ne la meitat quadrada.*

Exemple 7. *Considerem un model metabòlic amb la següent estoiquiometria:*

$$\mathbf{S} = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & -1 & -1 & -2 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & -1 \end{pmatrix}.$$

Podem veure la matriu **S** *com la matriu de biadjacència d'un graf bipartit i construir-ne el RAG:*

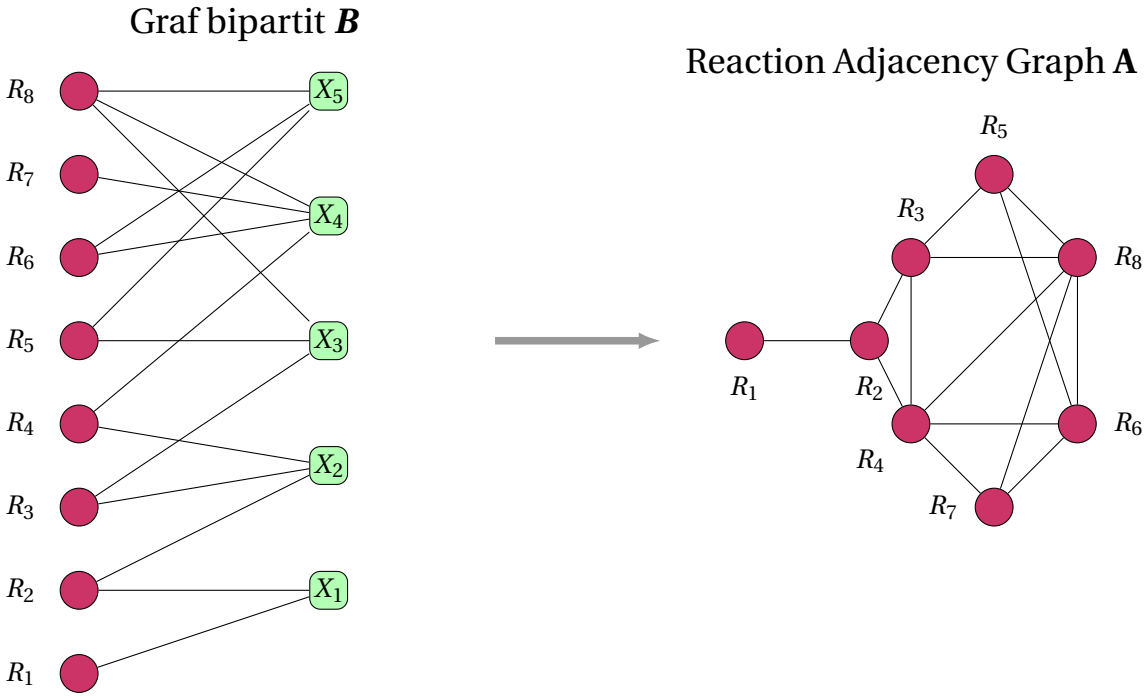


Figura 3.1: Construcció d'un RAG a partir d'un graf bipartit.

Per abordar les limitacions que presenten els *reaction adjacency graphs*, anem a donar una formulació de graf que tengui en compte el flux de metabòlits entre reaccions. Per construir el nostre graf, despleguem cada reacció en dues direccions separades (endavant i enrere) i redefinim els enllaços entre els nodes (reaccions) per reflectir les relacions productor-consumidor. Específicament, dues reaccions estan connectades mitjançant una aresta si i només si, una produeix un metabòlit que és consumit per l'altra.

Suposem que tenim coneixement sobre els fluxos del sistema en un moment donat per exemple, suposem que hem resolt un problema de FBA que ens dóna una configuració $\mathbf{v}$ de fluxos. Així, si tenim coneixement sobre la reversibilitat de les reaccions, definim el *vector de reversibilitat* $\mathbf{r}$ de la següent manera: $r_i = 1$ si la reacció R_i és reversible i $r_i = 0$ si és irreversible. Aleshores, podem reescriure el vector de fluxos com:

$$\mathbf{v} := \mathbf{v}^+ - \mathbf{v}^- = \mathbf{v}^+ - \text{diag}(\mathbf{r})\mathbf{v}^-, \tag{3.3}$$

on $\mathbf{v}^+$ i $\mathbf{v}^-$ són vectors no negatius que contenen els fluxos de reacció en les dues direccions, endavant i enrere respectivament. Aquí, la matriu diag($\mathbf{r}$) de dimensions $m \times m$ conté $\mathbf{r}$ a la seva diagonal principal. Amb aquestes definicions, podem reescriure el model metabòlic a l'Eq. (3.1) com:

$$\dot{\mathbf{x}} = \mathbf{S}\mathbf{v} = \begin{bmatrix}\mathbf{S} & -\mathbf{S}\end{bmatrix} \begin{bmatrix}\mathbf{I}_m & 0 \\ 0 & \text{diag}(\mathbf{r})\end{bmatrix} \begin{bmatrix}\mathbf{v}^+ \\ \mathbf{v}^-\end{bmatrix} := \mathbf{S}_{2m}\mathbf{v}_{2m}, \tag{3.4}$$

on $\mathbf{v}_{2m} := [\mathbf{v}^+ \ \mathbf{v}^-]^T$ és el vector desplegat de fluxos, $\mathbf{I}_m$ és la matriu identitat $m \times m$, i hem definit $\mathbf{S}_{2m}$, com la versió desplegada de la matriu estequiomètrica.

3.2.1 *Normalised Flow Graph* (NFG)

El desplegament del vector de fluxos en les direccions endavant i enrere ens porta a la definició de les matrius estequiomètriques de *producció* i *consumició*:

$$\text{Producció}: \quad \mathbf{S}^+_{2m} = \frac{1}{2}(\text{abs}(\mathbf{S}_{2m}) + \mathbf{S}_{2m}) \tag{5}$$

$$\text{Consumició}: \quad \mathbf{S}^-_{2m} = \frac{1}{2}(\text{abs}(\mathbf{S}_{2m}) - \mathbf{S}_{2m}), \tag{3.5}$$

on abs($\mathbf{S}_{2m}$) és la matriu dels valors absoluts dels elements corresponents de $\mathbf{S}_{2m}$. Cal destacar que cada element de la matriu $\mathbf{S}^+_{2m}$, denotat s^+_{ij}, dóna el nombre de molècules del metabòlit X_i produïdes per la reacció R_j. En canvi, els elements de $\mathbf{S}^-_{2m}$, denotats s^-_{ij}, corresponen al nombre de molècules del metabòlit X_i consumides per la reacció R_j. Noti's que, estem considerant el doble de reaccions que en el model original i que, ambdues matrius tenen entrades no-negatives.

Passem ara a donar una noció *probabilística* sobre les relacions productor-consumidor que es puguin donar entre dues reaccions. Suposem que ens donen una matriu estequiomètrica $\mathbf{S}$ sense cap informació biològica addicional, hom pot pensar la probabilitat que una molècula de metabòlit X_k triada uniformement a l'atzar de $\mathbf{S}$ sigui produïda per la reacció R_i i consumida per la reacció R_j com:

$$P(\textit{Una molècula de } X_k \textit{ és produïda per } R_i \textit{ i consumida per } R_j) = \frac{s^+_{ki}\, s^-_{kj}}{w^+_k\, w^-_k}, \tag{3.6}$$

on $w^+_k = \sum_{h=1}^{2m} s^+_{kh}$ i $w^-_k = \sum_{h=1}^{2m} s^-_{kh}$ són el nombre total de molècules de X_k produïdes i consumides per totes les reaccions que s'han tingut en compte a $\mathbf{S}_{2m}$.

Observació 8. *En el cas què existeixi un metabòlit X_k tal que $w^+_k w^-_k = 0$ podem assignar-li una probabilitat de 0. Això és degut a què les matrius $\mathbf{S}^+_{2m}$ i $\mathbf{S}^-_{2m}$ tenen entrades no-negatives. Llavors $w^+_k = 0$ implica necessàriament $s^+_{ki} = 0, \forall i = 1, ..., 2m$.*

Així, definim el pes de la relació entre els nodes de reacció R_i i R_j com la probabilitat que qualsevol metabòlit triat a l'atzar sigui produït per R_i i consumit per R_j. Això és, sumar el valor anterior per tots els metabòlits X_k amb $k = 1, ..., n$ i normalitzant:

$$d_{ij} = \frac{1}{n} \sum_{k=1}^{n} \frac{s^+_{ki}\, s^-_{kj}}{w^+_k\, w^-_k}, \tag{3.7}$$

Lema 9. *Es compleixen les següents condicions:*

1. $d_{i,j} \geq 0, \forall i = 1, ..., 2m, j = 1, ..., 2m$

2. $\sum_{i,j} d_{ij} = 1$

Demostració. La desigualtat de la primera condició és inmediata a partir de la definició de les matrius de producció i consumició. Vegem 2:

$$\sum_{i,j} d_{ij} = \sum_{i,j} \frac{1}{n} (\sum_{k=1}^{n} \frac{s_{ki}^{+} s_{kj}^{-}}{w_k^{+} w_k^{-}}) = \frac{1}{n} \sum_{i,j} (\sum_{k=1}^{n} \frac{s_{ki}^{+} s_{kj}^{-}}{w_k^{+} w_k^{-}}) = \frac{1}{n} \sum_{k=1}^{n} (\sum_{i,j} \frac{s_{ki}^{+} s_{kj}^{-}}{w_k^{+} w_k^{-}}) =$$
$$\frac{1}{n} \sum_{k=1}^{n} \frac{1}{w_k^{+} w_k^{-}} (\sum_{i,j} s_{ki}^{+} s_{kj}^{-}) = \frac{1}{n} \sum_{k=1}^{n} \frac{1}{w_k^{+} w_k^{-}} (\sum_{i} s_{ki}^{+} \sum_{j} s_{kj}^{-}) =$$
$$\frac{1}{n} \sum_{k=1}^{n} \frac{1}{w_k^{+} w_k^{-}} (w_k^{+} w_k^{-}) = 1.$$

□

Definició 10. *Donada* **S** *una estoiquiometria per a un model metabòlic, definim el Normalised Flow Graph (NFG) del model com el graf que té* $D = (d_{ij})$ *per matriu d'adjacència.*

Exemple 11. *Considerem el model estoiquiomètric de l'Exemple 6 i suposem que la reacció* R_4 *n'és l'única reversible. Calculem els coeficients* d_{ij} *per* $i = 1,...,n$ *i* $j = 1,...,2m$ *i obtenim la següent representació del NFG tot suprimint els nodes desconnectats:*

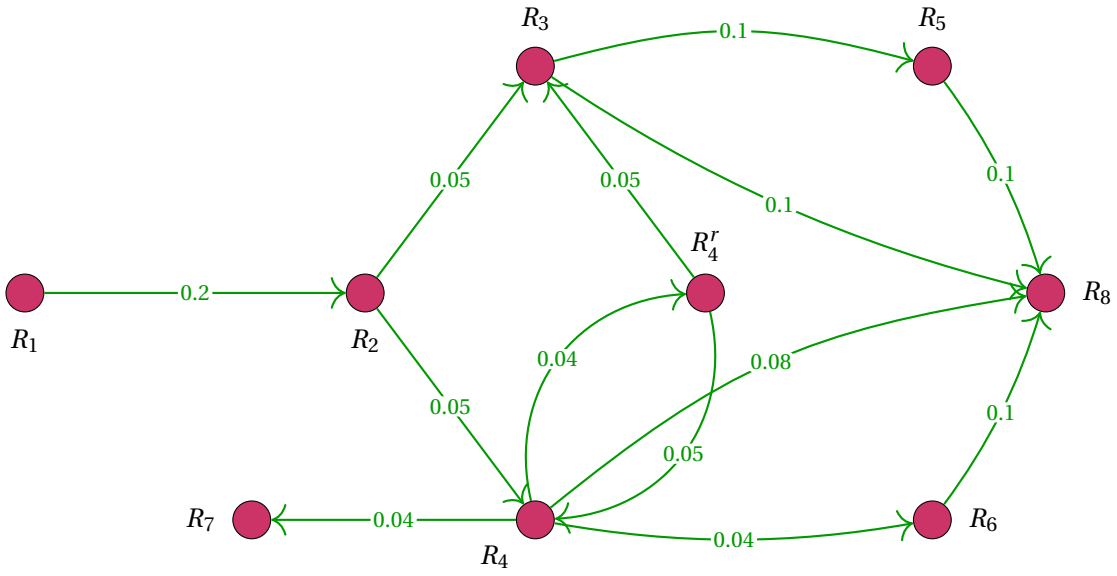


Figura 3.2: El Normalised Flow Graph (NFG) del model de l'Exemple 6 és un multigraf dirigit.

Definició 12. *Sigui* $A \in \mathbb{R}^{m \times n}$*, definim la pseudoinversa d'A com la matriu* $A^{\dagger} \in \mathbb{R}^{n \times m}$ *que satisfà les següents condicions:*

1. $AA^{\dagger}A = A$
2. $A^{\dagger}AA^{\dagger} = A^{\dagger}$
3. $(AA^{\dagger})^{T} = AA^{\dagger}$

4. $(A^\dagger A)^T = A^\dagger A$

Aquestes condicions es coneixen com les condicions de Moore-Penrose i sovint ens referirem a A^+ com la pseudoinversa de Moore-Penrose. Es pot demostrar l'existència i la unicitat de la pseudoinversa per a qualsevol matriu amb entrades reals o complexes [14].

Un corol·lari immediat del Teorema d'existència i unicitat per a la pseudoinversa, és el fet que, en el cas de matrius invertibles, la inversa coincideix amb la pseudoinversa. Això ens permet donar una versió generalitzada del concepte d'inversa d'una matriu. Anem a veure ara un altre cas particular.

Corol·lari 13. *Sigui* $A = diag(v) \in \mathbb{R}^{n\times m}$ *una matriu diagonal, aleshores* $A^\dagger = diag(v^\dagger)$*, on:*

$$v_i^+ = \begin{cases} \frac{1}{v_i} & si\ v_i \neq 0 \\ 0 & si\ v_i = 0 \end{cases}$$

Lema 14. *La matriu d'adjacència del Normalised Flow Graph definit anteriorment es correspon amb la següent forma matricial:*

$$\mathscr{D} = \frac{1}{n}\left(\mathbf{W}_+^\dagger \mathbf{S}_{2m}^+\right)^T \left(\mathbf{W}_-^\dagger \mathbf{S}_{2m}^-\right) \tag{3.8}$$

on $\mathbf{W}_\perp^\dagger = diag(\mathbf{S}_{2m}^+ \mathbf{1}_{2m})^\dagger$, $\mathbf{W}_\perp^\dagger = diag(\mathbf{S}_{2m}^- \mathbf{1}_{2m})^\dagger$, $\mathbf{1}_{2m}$ *és un vector d'uns, i* † *denota la pseudoinversa de Moore-Penrose.*

3.2.2 *Mass Flow Graph* (MFG)

Hem vist anteriorment com les cèl·lules ajusten els seus fluxos metabòlics en resposta a condicions ambientals. El *Flux Balance Analysis* és un dels mètodes més usats per a predir un vector v^* de flux que maximitza un objectiu metabòlic.

Amb els NFG hem pogut incorporar la noció de quantitat i direccionalitat de flux de metabòlits dins d'una xarxa. Ara, el que volem és afegir la informació que ens dóna el FBA en l'estuctura del nostre graf metabòlic.

Sigui v^* la solució a un problema de *Flux Balance Analysis*. De manera similar a l'equació 3.4, desplegam el vector de fluxes en dues direccions: endavant i enrere:

$$\mathbf{v}_{2m}^* = \begin{bmatrix} \mathbf{v}^{*+} \\ \mathbf{v}^{*-} \end{bmatrix} = \frac{1}{2}\begin{bmatrix} \text{abs}(\mathbf{v}^*) + \mathbf{v}^* \\ \text{abs}(\mathbf{v}^*) - \mathbf{v}^* \end{bmatrix}$$

Lema 15. *En un estat d'equilibri de flux, tots els metabòlits tenen la mateixa taxa de producció que de consumició:*

$$\mathbf{S}_{2m}^+ \mathbf{v}_{2m}^* = \mathbf{S}_{2m}^- \mathbf{v}_{2m}^*$$

on $\mathbf{S}_{2m}^+$ *i* $\mathbf{S}_{2m}^-$ *són, respectivament, les matrius de producció i consumició definides anteriorment.*

Demostració. En efecte, es té la següent cadena d'igualtats:

$$\begin{aligned}
\mathbf{S}_{2m}^{+}\mathbf{v}_{2m}^{*} - \mathbf{S}_{2m}^{-}\mathbf{v}_{2m}^{*} &= \\
(\mathbf{S}_{2m}^{+} - \mathbf{S}_{2m}^{-})\mathbf{v}_{2m}^{*} &= \\
\frac{1}{2}(\mathbf{S}_{2m} + \mathrm{abs}(\mathbf{S}_{2m}) + \mathbf{S}_{2m} - \mathrm{abs}(\mathbf{S}_{2m}))\mathbf{v}_{2m}^{*} &= \\
\mathbf{S}_{2m}\mathbf{v}_{2m}^{*} &= \\
[\mathbf{S}, -\mathrm{diag}(r)\mathbf{S}]\mathbf{v}_{2m}^{*} &= \\
\mathbf{S}\mathbf{v}^{*+} - \mathrm{diag}(r)\mathbf{S}\mathbf{v}^{*-} &= \\
\mathbf{S}\mathbf{v} &= \\
0.
\end{aligned}$$

D'on podem concloure el fet que $\mathbf{S}_{2m}^{+}\mathbf{v}_{2m}^{*} = \mathbf{S}_{2m}^{-}\mathbf{v}_{2m}^{*}$.

□

El lema anterior ens permet definir el vector $\mathbf{j}(\mathbf{v}^*)$ de fluxos de producció i consumició com:

$$\mathbf{j}(\mathbf{v}^*) = \mathbf{S}_{2m}^{+}\mathbf{v}_{2m}^{*} = \mathbf{S}_{2m}^{-}\mathbf{v}_{2m}^{*}. \tag{3.9}$$

La k-èsima entrada de $\mathbf{j}(\mathbf{v}^*)$ (té dimensions $2m \times 1$) és el flux al qual el metabòlit X_k és produït i consumit alhora.

Per construir el nostre *Mass Flow Graph*, definim el pes de la relació entre les reaccions R_i i R_j com el flux de massa total de metabòlits produïts per R_i que són consumits per R_j. Assumint que la quantitat de metabòlit produïda per una reacció es distribueix entre les reaccions que el consumeixen en proporció al seu flux (i respectant l'estequiometria), el flux del metabòlit X_k de la reacció R_i a R_j ve donat per:

$$\text{Flux de } X_k \text{ de } R_i \text{ a } R_j = (\text{flux de } X_k \text{ produït per } R_i) \times \left(\frac{\text{flux de } X_k \text{ consumit per } R_j}{\text{flux total de consum de } X_k}\right).$$

Sumant el valor anterior per tots els metabòlits, com hem fet amb els NFG, obtenim el pes de la relació entre les reaccions R_i i R_j:

$$M_{ij}(\mathbf{v}^*) = \sum_{k=1}^{n} s_{ki}^{+} v_{2mi}^{*} \times \left(\frac{s_{kj}^{-} v_{2mj}^{*}}{\sum_{h=1}^{2m} s_{kh}^{-} v_{2mh}^{*}}\right). \tag{3.10}$$

Definició 16. *Donada* $\mathbf{S}$ *una estoiquiometria i* v^* *un vector de fluxos per a un model metabòlic, definim el MFG del model com el graf que té* $\mathscr{M} = (M_{ij})$ *per matriu d'adjacència.*

Com hem fet anteriorment, podem donar una forma matricial per a la matriu d'adjacència del nostre MFG:

$$\mathbf{M}(\mathbf{v}^*) = (\mathbf{S}_{2m}^{+}\mathbf{V}^*)^T \mathbf{J}_v^{\dagger} (\mathbf{S}_{2m}^{-}\mathbf{V}^*) \tag{3.11}$$

on $\mathbf{V}^* = \mathrm{diag}(\mathbf{v}_{2m}^*)$, $\mathbf{J}_v = \mathrm{diag}(\mathbf{j}(\mathbf{v}^*))$ i $\dagger$ denota la matriu pseudoinversa.

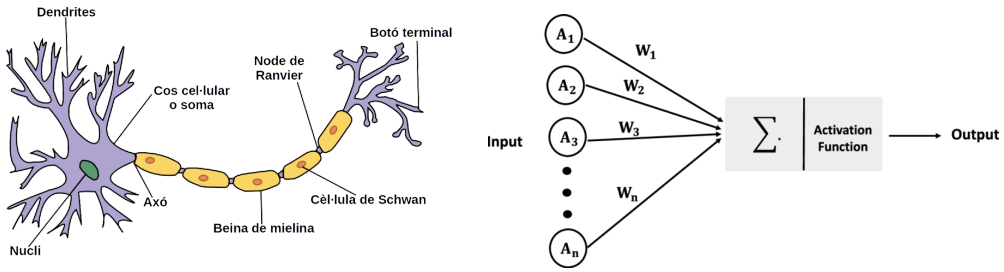
CAPÍTOL 4

Graph Neural Networks (GNN)

4.1 Xarxes neuronals

Les Xarxes Nuronals Artificials (ANN sigles en anglès) [15, 16, 17] són models computacionals que neixen amb l'objectiu de simular una xarxa de cèl·lules nervioses (neurones) d'un Sistema Nerviós Central biològic (humà o animal). Tot i que poguem pensar les ANN com una abstracció del seu homòleg en la biologia, l'objectiu de les xarxes neuronals no és ben bé replicar exactament allò que succeeix en un sistema biològic, sinó fer servir tot el coneixement sobre la funcionalitat d'una xarxa biològica per a resoldre problemes complexos.

La major contribució que han portat les ANN és la seva capacitat per a resoldre problemes complexos, especialment, aquells que són no lineals i/o no analítics tot fent servir operacions computacionalment molt senzilles (sumes, productes i operadors lògics). A més, han de tenir la capacitat per generalitzar en la solució de problemes del mateix tipus amb una tècnica algorísmica i computacional simple.



(a) Neurona (b) Representació d'una Neurona Artificial

La primera proposta d'implementació d'una Xarxa Neuronal Artificial apareix l'any 1943 de la mà de McCulloch i Pitts [18]. Un poc més tard, l'any 1958, el psicòleg Frank Rosenblatt desenvolupa el *Perceptró* [19], un tipus de neurona artificial que acaba essent clau en la investigació en els anys seixanta [20]. Tanmateix, avui en dia és més comú fer feina amb altres tipus de neurones com, per exemple la neurona *sigmoide*, com veurem més endavant. Ara bé, per entendre-la cal primer endisar-nos en el model del *Perceptró*.

4.1.1 La Neurona Artificial

Definim el *Perceptró* com una funció $f : \{0,1\}^n \to \{0,1\}$ que rep un vector binari $(x_1, x_2, \dots x_n)$ i retorna una *decisió* en forma de valor binari $\hat{y}$. Per a determinar aquesta decisió, Rosenblatt [19] proposa fer servir un conjunt de pesos $w_1, \dots w_n$, nombres reals que representen la 'importància' de cadascun dels inputs o *característiques*. Aleshores, la decisió $\hat{y}$ que prèn el Perceptró queda determinat per segons si la suma $\sum_{i=1}^{n} w_i x_i$ assoleix, o no, un *llindar* fixat:

$$\hat{y} = \begin{cases} 0 & \text{si } \sum_j w_j x_j \leq \theta \\ 1 & \text{si } \sum_j w_j x_j > \theta \end{cases}$$

on θ és un valor fixat. Aquest és el model matemàtic més bàsic de com podem veure una neurona. Una unitat independent que prèn decisions com una *funció d'activació* que té en compte la importància d'unes característiques d'entrada.

Més en general, podem estendre el Perceptró a un conjunt d'entrades no binàries. Suposem que volem reconstruir una funció $F : \mathbb{R}^n \to \{0,1\}$ a partir d'un conjunt de mostres $(\boldsymbol{x_1}, y_1), ..., (\boldsymbol{x_p}, y_p)$ aleshores la nostra tasca es redueix a trobar l'hiperplà que **millor separa o classifica** les antimatges de $\boldsymbol{F}$, on $y_i = \boldsymbol{F(x_i)}$. A partir d'ara, ens referirem a aquesta tasca com *l'aprenentatge* i al conjunt $\{(\boldsymbol{x_i}, y_i)\}_{i=1}^{p}$ com el *conjunt d'entrenament* del model. Així doncs, podem pensar les Xarxes Neuronals i, en particular, el Perceptró com aproximadors de funcions.

Matemàticament, podem veure la classificació amb una sola neurona com un problema de regions segons un hiperplà. En efecte, afegint una observació $x_0 = 1$ i prenent $w_0 = \theta$ ens queda:

$$\hat{y}(\boldsymbol{x}) = \begin{cases} 0 & \text{si } \boldsymbol{x^T w} \leq 0 \\ 1 & \text{si } \boldsymbol{x^T w} > 0 \end{cases}$$

on $\boldsymbol{x} = [1, x_1, ..., x_n]^T$ és el vector d'imputs o *característiques* i $\boldsymbol{w} = [\theta, w_1, ... w_n]^T$ els pesos associats a cadascuna d'elles.

Podem formular l'*aprenentatge* d'una neurona com el següent problema d'optimització en $\boldsymbol{w}$:

$$\min \frac{1}{2} \sum_{i=1}^{p} (y_i - x_i^T w_i)^2$$

.

Aquest problema es pot atacar amb algorismes iteratius de descens del gradient sense restriccions [9]. Concretament, en aquest model i molts d'altres farem servir algorismes de **descens del gradient estocàstic** (SGD són les sigles en anglès). Frank Rosenblatt [19] va demostrar la convergència, sota certes hipòtesis del que anomenà *regla d'aprenentatge del Perceptró*. Hi ha moltes regles d'aprenentatge pel Perceptró, nosaltres en donarem una i en provarem la convergència.

Regla d'aprenentatge per al Perceptró: convergència

Es defineix:

- r: la taxa d'aprenentatge, típicament menor o igual a 1.
- $D = \{(\boldsymbol{x_i}, y_i)\}_{i=1}^{p}$ el conjunt d'entrenament, on $\boldsymbol{x_i} \in \mathbb{R}^n$, $y_i \in \{0, +1\}$ i $\boldsymbol{x_{i,0}} = 1$.
- $\boldsymbol{w}(k)$ el vector de pesos en la iteració k-èssima.

Considerem el següent algorisme iteratiu:

1. Inicialitzem els pesos $\boldsymbol{w}$ (típicament a $\mathbf{0}$).
2. Per a cada mostra $\boldsymbol{x_i}$ de D, calculem la predicció $\hat{y}_{w_k}(\boldsymbol{x_i})$ amb els pesos actuals i actualitzem $\boldsymbol{w}$ segons: $\boldsymbol{w}_{(k+1)} := \boldsymbol{w}_{(k)} + r(y_i - \hat{y}_{w_k}(\boldsymbol{x_i}))\boldsymbol{x_i}$.

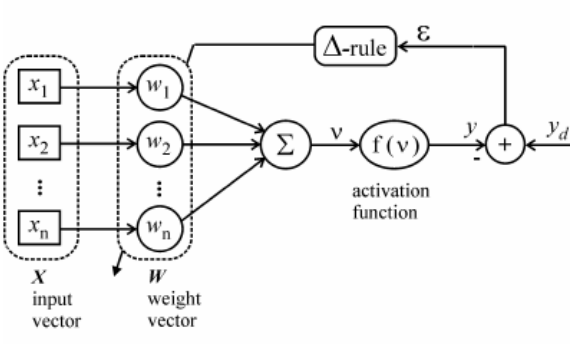


Figura 4.2: Esquema de l'aprenentatge

El valor de $\boldsymbol{w}$ s'actualitza amb cada mostra. Es considera que s'assoleix l'aprenentatge quan es completa un cicle sencer al dataset on totes les prediccions han estat correctes, això és, un cicle sense actualitzar el valor dels pesos. Sovint s'anomena el terme $\hat{y}_{w_k}(\boldsymbol{x_i}) - y_i(\boldsymbol{x_i})$ com l'error en la passa k-èssima, que prèn valors en el conjunt $\{-1, 0, 1\}$.

Exemple 17. *Volem entrenar un perceptró perquè aprengui la funció lògica AND, amb quatre mostres d'entrada:*

x_0	x_1	x_2	$y = x_1 \wedge x_2$
1	*0*	*0*	*0*
1	*0*	*1*	*0*
1	*1*	*0*	*0*
1	*1*	*1*	*1*

Regla d'actualització:
Per a cada exemple $(\boldsymbol{x}, y)$*, on* $\boldsymbol{x} = (x_0, x_1, x_2)$*:*

$$\hat{y} = \begin{cases} 1 & si\ \boldsymbol{w} \cdot \boldsymbol{x} > 0 \\ 0 & altrament \end{cases}$$

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + r(y - \hat{y})\boldsymbol{x}$$

Amb $r = 1$*, i inicialment* $\boldsymbol{w} = (w_0, w_1, w_2) = (0, 0, 0)$.
Entrenament:

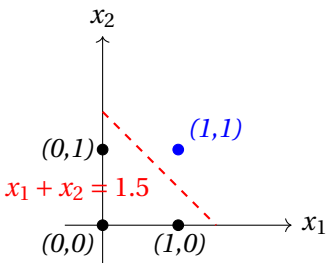
- $(1,0,0) \rightarrow y = 0,\ \hat{y} = 0 \Rightarrow$ *No canvi*
- $(1,0,1) \rightarrow y = 0,\ \hat{y} = 0 \Rightarrow$ *No canvi*
- $(1,1,0) \rightarrow y = 0,\ \hat{y} = 0 \Rightarrow$ *No canvi*
- $(1,1,1) \rightarrow y = 1,\ \hat{y} = 0 \Rightarrow w = (1,1,1)$

Següent cicle:

- $(1,0,0)\cdot(1,1,1) = 1 \Rightarrow \hat{y} = 1 \rightarrow$ *Error!*
- $w \leftarrow (1,1,1) + (-1)(1,0,0) = (0,1,1)$
- $(1,1,1) \rightarrow y = 1$, $\hat{y} = 1 \rightarrow$ *No canvi*
- $(1,0,0) \rightarrow y = 0$, $\hat{y} = 0 \rightarrow$ *No canvi*
- $(1,0,1) \rightarrow y = 0$, $\hat{y} = 1 \rightarrow$ *Error* $\Rightarrow \boldsymbol{w} = (-1,1,0)$
- *etc...*

Si continuem aquest procés arribarem a que el Perceptró acaba trobant un vector $\boldsymbol{w}$ *amb què es completa un cicle sense errors. Aleshores podem calcular el que anomenarem la frontera de decisió, tot igualant* $\boldsymbol{w}^T \cdot \boldsymbol{x} = 0$*:*

Representació gràfica:



La recta $x_1 + x_2 = 1.5$ *és una possible frontera de decisió.*

Intuitivament, hom pot pensar que no tots els problemes de classificació són resolubles amb el Perceptró en tant que, es poden donar situacions en les què no poguem separar mitjançant un hiperplà dues famílies de punts. En aquest cas, direm que el conjunt d'entrenament D no és *linealment separable.*

Anem ara a donar condicions suficients per a la convergència de l'algorisme anterior.

Teorema 18. ***(Convergència del Perceptró):*** *Donat un conjunt de dades* D*, tal que*

$$\max_{(\boldsymbol{x},y)\in D} \|\boldsymbol{x}\|_2 = R,$$

i és linealment separable per algun vector unitari $\boldsymbol{w'}$*, aleshores l'algorisme anterior és convergent per qualsevol taxa d'aprenentatge* r*.*

Demostració. Podem separar el conjunt D en dues classes:

$$D = D^+ \sqcup D^-$$

on $D^+ = \{(\boldsymbol{x}, y) \in D : y = 1\}$ i $D^- = \{(\boldsymbol{x}, y) \in D : y = 0\}$. El que farem serà traslladar el nostre problema a un nou conjunt de punts i demostrarem la seva equivalència pel que fa a la separabilitat lineal. Per fer-ho, considerem un nou conjunt d'entrenament:

$$D^* = D^+ \sqcup D'$$

on $D' = \{(-\boldsymbol{x}, y) : (\boldsymbol{x}, y) \in D^-\}$.

Suposar que D és linealment separable per un vector unitari $\boldsymbol{w'} = (w_0, w_1, ..., w_n)$ equival a suposar que $\boldsymbol{w'} \cdot \boldsymbol{x} \neq 0, \forall \boldsymbol{x} : (x, y) \in D$ i que,

$$\begin{cases} \boldsymbol{w'} \cdot \boldsymbol{x} > 0 & \rightarrow \boldsymbol{x} \in D^+ \\ \boldsymbol{w'} \cdot \boldsymbol{x} < 0 & \rightarrow \boldsymbol{x} \in D^- \end{cases}$$

o equivalentment, $\boldsymbol{w'} \cdot \boldsymbol{x} > 0, \forall\ \boldsymbol{x} : (\boldsymbol{x}, y) \in D^*$. És a dir, en el nou domini D^*, una predicció és incorrecta si, i només si, la funció de predicció ens dona el valor 0 (i.e $\boldsymbol{w}_{(k)} \cdot \boldsymbol{x}_i \leq 0$ en qualque passa). Per tant, l'error en aquest cas seria $y(\boldsymbol{x}_i) - \hat{y}_{w_k}(\boldsymbol{x}_i) = 1 - 0 = 1$.

Definim la constant γ com:

$$\gamma := \min_{(\boldsymbol{x}, y) \in D} |\boldsymbol{w'} \cdot \boldsymbol{x}| = \min_{(x, y) \in D^*} \boldsymbol{w'} \cdot \boldsymbol{x}$$

Per la hipòtesi d'existència de $\boldsymbol{w'}$ tenim que $\gamma > 0$. Per demostrar la convergència del Perceptró en D^*, afitarem el nombre total d'errors que pot cometre per una constant positiva M.

Suposem que després d'un cert nombre d'iteracions sobre els valors $\boldsymbol{w}_{(k)}$, s'han produït N errors de classificació en les dades $\boldsymbol{x}_{(e1)}, ..., \boldsymbol{x}_{(eN)}$. Aleshores, per una certa $k \in \mathbb{N}$ i $S \in \mathbb{N}$ amb $1 \leq S \leq N$ s'ha donat la següent actualització:

$$\boldsymbol{w}_{(k+1)} := \boldsymbol{w}_{(k)} + r(y_{eS} - \hat{y}_{w_k}(\boldsymbol{x}_{(eS)}))\boldsymbol{x}_{(eS)} = \boldsymbol{w}_{(k)} + r(1-0)\boldsymbol{x}_{(eS)} = \boldsymbol{w}_{(k)} + r\boldsymbol{x}_{(eS)}$$

on $\hat{y}$ és la funció de predicció del Perceptró. Començant l'algorisme amb $w_0 = 0$, com que el nombre d'errors és N (i, per tant, només s'han produït N actualitzacions en els $\boldsymbol{w}_{(k)}$) és fàcil veure que $\exists k_0 \in \mathbb{N}$:

$$\boldsymbol{w}_{(k_0)} = \boldsymbol{w}_{(0)} + r(\boldsymbol{x}_{(eN)} + \cdots + \boldsymbol{x}_{(e1)})$$

Multiplicant per $\boldsymbol{w'}$ i prenent valors absoluts,

$$|\boldsymbol{w'} \cdot \boldsymbol{w}_{(k_0)}| = |\boldsymbol{w'} \cdot (\boldsymbol{w}_0 + r(\boldsymbol{x}_{(eN)} + \cdots + \boldsymbol{x}_{(e1)})| = r|\boldsymbol{w'} \cdot \boldsymbol{x}_{(eN)} + \cdots + \boldsymbol{w'} \cdot \boldsymbol{x}_{(e1)}| \geq rN\gamma$$

Ara, aplicant la desigualtat de Cauchy-Schwarz:

$$|\boldsymbol{w'} \cdot \boldsymbol{w}_{(k_0)}| \leq \|\boldsymbol{w'}\|_2 \|\boldsymbol{w}_{(k_0)}\|_2 = \|\boldsymbol{w}_{(k_0)}\|_2$$

Arribam així a la següent desigualtat:

$$\|\boldsymbol{w}_{(k_0)}\|_2 \geq rN\gamma \tag{4.1}$$

D'altra banda, en l'actualització k_0 es té la següent igualtat:

$$\begin{aligned} \|\boldsymbol{w}_{(k_0)}\|_2^2 - \|\boldsymbol{w}_{(k_0-1)}\|_2^2 &= \|\boldsymbol{w}_{(k_0-1)} + r\boldsymbol{x}_{(eN)}\|_2^2 - \|\boldsymbol{w}_{(k_0-1)}\|_2^2 \\ &= 2r(\boldsymbol{w}_{(k_0-1)} \cdot \boldsymbol{x}_{(eN)}) + r^2\|\boldsymbol{x}_{(eN)}\|_2^2 \end{aligned}$$

com que $\boldsymbol{w}_{(k_0-1)} \cdot \boldsymbol{x}_{(eN)} \leq 0$, (i.e la predicció ha estat errònia amb els coeficients de la iteració anterior), aleshores:

$$\|\boldsymbol{w}_{(k_0)}\|_2^2 - \|\boldsymbol{w}_{(k_0-1)}\|_2^2 \leq r^2\|\boldsymbol{x}_{(eN)}\|_2^2 \leq r^2R^2 \tag{4.2}$$

Més en general, donat un $k \in \mathbb{N}$ qualsevol, la diferència $\|\boldsymbol{w}_{(k)}\|_2^2 - \|\boldsymbol{w}_{(k-1)}\|_2^2$ és, o bé 0 si la predicció en aquella iteració ha estat acertada, o bé es pot afitar de manera anàloga a 4.2.

Ara, començant amb $w_{(0)} = 0$ i després de cometre N errors, es pot veure fàcilment fent servir les desigualtats anteriors:

$$\| \boldsymbol{w}_{(k_0)} \|_2 \leq \sqrt{N r^2 R^2} = r R \sqrt{N} \tag{4.3}$$

Ajuntant les desigualtats 4.1 i 4.3 veiem que:

$$NR\gamma \leq \| \boldsymbol{w} \|_2 \leq r R \sqrt{N}$$

$$\sqrt{N} \leq \frac{r}{\gamma}$$

Prenent $M = \frac{r^2}{\gamma^2}$ hem trobat la fita que cercàvem.

□

Observació 19. *En el cas de dades linealment separables, tenim garantida la convergència del Perceptró a qualque solució. Ara bé, res ens garanteix que la recta solució sigui la 'millor' d'entre totes les possibles alhora de generalitzar a un conjunt de dades més gran. Els Support Vector Machines (SVM) són models que adrecen aquest problema (Krauth i Mezard [21]).*

La neurona sigmoide

Una de les mancances que trobam en el model simplista del Perceptró és la falta de regularitat. Com hem vist en la secció anterior, la presa de decisions del Perceptró es basa en l'activació (o no) d'una funció escaló. Una pregunta que ens podem fer és, què passa doncs, si hi ha una petita variació en les dades? En efecte, una petita variació en els pesos w_i o en les observacions x_i, pot resultar en un canvi important en la classificació. Que passi, per exemple, d'un 0 a un 1.

Per evitar aquesta intestabilitat i recórrer a ajustar cadascun dels w_i per obtenir el funcionament desitjat, podem fer servir una funció d'activació més suau: la funció sigmoide. D'aquesta manera, les *neurones sigmoide* són aquelles que, ben igual que els models anteriors, reben un conjunt d'inputs reals, però retornen un valor real comprès entre 0 i 1.

Definim la funció *sigmoide* o *logística* com la següent funció de variable real:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

És inmediat veure que, $0 < \sigma(z) < 1, \forall z \in \mathbb{R}$ i que $\sigma(0) = \frac{1}{2}$. Ben igual que abans, una vegada hem trobat els pesos $w_i \in \mathbb{R}$ que fan minima una certa funció desitjada (p.ex. l'error quadràtic mitjà), la resposta de la neurona sigmoide serà:

$$\hat{y}(\boldsymbol{x}) = \sigma(\boldsymbol{x}^T \boldsymbol{w}) = \frac{1}{1 + e^{-\boldsymbol{x}^T \boldsymbol{w}}}$$

Com interpretam el valor de sortida $\hat{y}(\boldsymbol{x})$ de la neurona sigmoide? Típicament es fixa un llindar del 0.5 per a la classifació binària. Més endavant discutirem com influeix la variació del llindar en l'obtenció de prediccions errònies.

4.1.2 L'algorisme de *backpropagation*

Definim una *Xarxa Neuronal Artificial* [9] com un circuit de neurones artificials on els *inputs* per a cada neurona són una combinació dels *outputs* d'altres. Aquesta interconnexió és la que permet l'intercanvi d'informació entre neurones.

En una Xarxa Neuronal *directa* (o *feed-forward nural network* en anglès) les neurones estan connectades per capes de tal manera que la informació només viatja en una direcció. Aquest

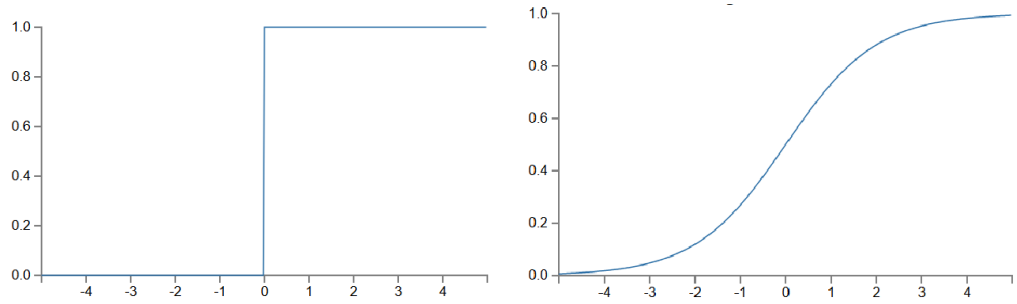


Figura 4.3: Dues funcions d'activació

model evita cicles recurrents i només permet l'arribada d'informació de les neurones en la capa anterior. Anomenarem *capa d'entrada* o *input layer* a la primera capa de neurones, que reben la informació 'original' de les dades. Les *capes amagades* o *hidden layers* són aquelles que es troben entre la d'entrada i la de sortida o *output layer*.

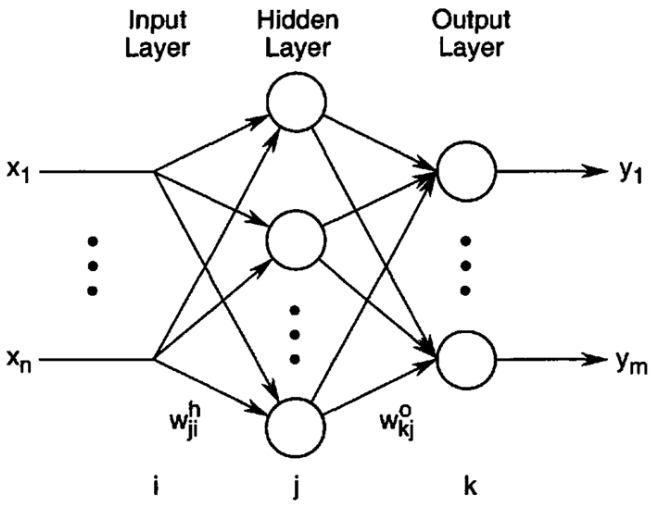


Figura 4.4: Esquema d'una Xarxa Neuronal Directa de 3 capes

En la secció anterior, hem considerat el problema d'entrenar una sola unitat neuronal. Anem ara a veure l'entrenament d'una xarxa *directa* fent servir l'algorisme de *backpropagation* [9]. Per la simplicitat en els càlculs, suposem que treballem amb 3 capes: entrada, amagada i sortida.

Suposem que tenim n inputs x_i, on $i = 1,...,n$ i m outputs y_s, $s = 1,...,m$. Suposem a més, que tenim un nombre l de neurones en la capa amagada. L'output de les neurones en aquesta capa el denotarem com z_j, on $j = 1,...,l$. En aquest problema, podem veure les neurones de la capa d'entrada com distribuidores de la informació d'entrada, on la seva funció d'activació és la identitat. Denotam les funcions d'activació de les neurones en les capes amagada i de sortida, respectivament, com f_j^h i f_s^o. Es farà la derivació de l'algorisme per a funcions derivables f_j^h i f_s^o arbitràries (podrien ser, per exemple una sigmoide).

La correspondència general que implementa la xarxa neuronal ve donada per

$$\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} F_1(x_1,\ldots,x_n) \\ \vdots \\ F_m(x_1,\ldots,x_n) \end{bmatrix}.$$

Considerem el cas on el conjunt d'entrenament consisteix en un sol parell (x_d, y_d), on $x_d \in \mathbb{R}^n$ i $y_d \in \mathbb{R}^m$. Això no és gaire restrictiu ja que, com discutirem més endavant, entrenar amb més d'una mostra consisteix en aplicar l'algorisme que veurem a continuació a mostres diferents. Aquesta derivació, és per tant, rellevant pel cas general.

Denotam els pesos de la capa amagada com $w^h_{j,i}$, $i = 1, ..., n$, $j = 1, ..., l$. Els pesos pels inputs de la capa amagada a la capa de sortida $w^o_{s,j}$. Sigui v_j l'input per a la j-èssima neurona en la capa amagada i z_j el seu output:

$$v_j = \sum_{i=1}^{n} w^h_{ji} x_i,$$

,

$$z_j = f^h_j \left(\sum_{i=1}^{n} w^h_{ji} x_i \right).$$

I la sortida de la neurona s-èssima de la capa de sortida

$$y_s = f^o_s \left(\sum_{j=1}^{l} w^o_{sj} z_j \right).$$

L'entrenament de la xarxa neuronal es basa en ajustar un conjunt de pesos de manera que la sortida generada per la xarxa per a l'entrada donada $x_d = [x_{d1}, \ldots, x_{dn}]^T$ sigui tan propera a y_d com sigui possible. Així com hem fet en la secció anterior, una possible formulació matemàtica és el següent problema d'optimització:

$$\text{minimitzar } \frac{1}{2} \sum_{s=1}^{m} (y_{ds} - y_s)^2,$$

on y_s, $s = 1, \ldots, m$, són les sortides reals de la xarxa neuronal en resposta a les entrades $x_{d1}, \ldots, x_{dn}$, donades per

$$y_s = f^o_s \left(\sum_{j=1}^{l} w^o_{sj} f^h_j \left(\sum_{i=1}^{n} w^h_{ji} x_i \right) \right).$$

La minimització anterior es realitza sobre tots els w^h_{ji}, w^o_{sj}, $i = 1, \ldots, n$, $j = 1, \ldots, l$, $s = 1, \ldots, m$. Per simplicitat de notació, utilitzem el símbol w per al vector

$$w = (w^h_{ji}, w^o_{sj}) \text{ amb } i = 1, \ldots, n, j = 1, \ldots, l, s = 1, \ldots, m,$$

i el símbol E per a la funció objectiu a minimitzar; és a dir,

$$E(w) = \frac{1}{2} \sum_{s=1}^{m} (y_{ds} - y_s)^2$$

$$= \frac{1}{2} \sum_{s=1}^{m} \left(y_{ds} - f^o_s \left(\sum_{j=1}^{l} w^o_{sj} f^h_j \left(\sum_{i=1}^{n} w^h_{ji} x_i \right) \right) \right)^2.$$

La funció objectiu $E(w)$, també anomenada *funció de cost* (o *loss function* en anglès) també pot variar en funció del model. Altres funcions de *loss* que farem servir més endavant inclouen la funció d'entropia creuada (o *Cross-Entropy* en anglès).

Ben bé com abans, per a resoldre aquest problema farem servir l'algorisme de descens del gradient a pas fix. Per formular l'algorisme, necessitem calcular les derivades parcials de

E respecte a cada component de w. Per a això, primer calculem les derivades parcials de E respecte als índexs i, j i s. Primer calculem la derivada parcial de E respecte a w^o_{sj}. Per a això, escrivim

$$E(w) = \frac{1}{2}\sum_{s=1}^{m}\left(y_{ds} - f_s^o\left(\sum_{j=1}^{l} w^o_{sj} z_j\right)\right)^2,$$

on, per a cada $j = 1,\ldots,l$,

$$z_j = f_j^h\left(\sum_{i=1}^{n} w^h_{ji} x_i\right).$$

Utilitzant la regla de la cadena, obtenim

$$\frac{\partial E}{\partial w^o_{sj}}(w) = -(y_{ds} - y_s) f_s^{o'}\left(\sum_{j=1}^{l} w^o_{sj} z_j\right) z_j,$$

on $f_s^{o'} : \mathbb{R} \to \mathbb{R}$ és la derivada de f_s^o.
Per calcular $\frac{\partial E}{\partial w^o_{sj}}$, apliquem la regla de la cadena de la següent manera:

$$\frac{\partial E}{\partial w^o_{sj}} = \frac{\partial E}{\partial y_s} \cdot \frac{\partial y_s}{\partial\left(\sum_{j=1}^{l} w^o_{sj} z_j\right)} \cdot \frac{\partial\left(\sum_{j=1}^{l} w^o_{sj} z_j\right)}{\partial w^o_{sj}}$$

On:

$$\frac{\partial E}{\partial y_s} = -(y_{ds} - y_s) \quad \text{(derivada de l'error quadràtic)}$$

$$\frac{\partial y_s}{\partial\left(\sum_{j=1}^{l} w^o_{sj} z_j\right)} = f_s^{o'}\left(\sum_{j=1}^{l} w^o_{sj} z_j\right) \quad \text{(derivada de la funció d'activació)}$$

$$\frac{\partial\left(\sum_{j=1}^{l} w^o_{sj} z_j\right)}{\partial w^o_{sj}} = z_j \quad \text{(derivada de la suma ponderada)}$$

Multiplicant aquestes tres expressions, obtenim:

$$\frac{\partial E}{\partial w^o_{sj}} = -(y_{ds} - y_s) \cdot f_s^{o'}\left(\sum_{j=1}^{l} w^o_{sj} z_j\right) \cdot z_j$$

Per simplicitat de notació, escrivim

$$\delta_s = (y_{ds} - y_s) f_s^{o'}\left(\sum_{j=1}^{l} w^o_{sj} z_j\right).$$

Podem pensar en cada δ_s com un error de sortida escalat, ja que és la diferència entre la sortida real y_s de la xarxa neuronal i la sortida desitjada y_{ds}, escalada per $f_s^{o'}\left(\sum_{j=1}^{l} w^o_{sj} z_j\right)$. Utilitzant la notació δ_s, tenim

$$\frac{\partial E}{\partial w^o_{sj}}(w) = -\delta_s z_j.$$

A continuació calculem la derivada parcial de E respecte a w^h_{ji}. Comencem amb l'equació

$$E(w) = \frac{1}{2}\sum_{s=1}^{m}\left(y_{ds} - f^o_s\left(\sum_{j=1}^{l} w^o_{sj} f^h_j\left(\sum_{i=1}^{n} w^h_{ji} x_i\right)\right)\right)^2.$$

Utilitzant la regla de la cadena una vegada més, obtenim

$$\frac{\partial E}{\partial w^h_{ji}}(w) = -\sum_{s=1}^{m}(y_{ds} - y_s) f^{o'}_s\left(\sum_{j=1}^{l} w^o_{sj} z_j\right) w^o_{sj} f^{h'}_j\left(\sum_{i=1}^{n} w^h_{ji} x_i\right) x_i,$$

on $f^{h'}_j : \mathbb{R} \to \mathbb{R}$ és la derivada de f^h_j. Simplificant l'anterior, obtenim

$$\frac{\partial E}{\partial w^h_{ji}}(w) = -\left(\sum_{s=1}^{m} \delta_s w^o_{sj}\right) f^{h'}_j(v_j) x_i.$$

Ara estem preparats per formular l'algorisme de gradient per actualitzar els pesos de la xarxa neuronal. Escrivim les equacions d'actualització per als dos conjunts de pesos w^o_{sj} i w^h_{ji} separadament. Tenim

$$w^{o(k+1)}_{sj} = w^{o(k)}_{sj} + r\delta^{(k)}_s z^{(k)}_j,$$

$$w^{h(k+1)}_{ji} = w^{h(k)}_{ji} + r\left(\sum_{s=1}^{m} \delta^{(k)}_s w^{o(k)}_{sj}\right) f^{h'}_j(v^{(k)}_j) x_i,$$

on r és el pas de l'algorisme (fixat).

Aquesta regla d'actualització dels pesos rep el nom de regla de propagació cap enrere (*back-propagation* en anglès). Això es deu a que, els errors $\delta^{(k)}_1, ..., \delta^{(k)}_m$ es propaguen des del final de la xarxa cap a la capa amagada per a actualitzar les equacions anteriors. En el cas general de tenir diverses capes amagades, aquest error es va propagant cap enrere de capa en capa per actualitzar els pesos. Qualitativament, podem dividir l'algorisme en dues passes, la passa cap endavant (o *forward pass*) on es calculen els valors $v^{(k)}_j, z^{(k)}_j, y^{(k)}_s, \delta^{(k)}_j$ avançant el flux d'informació cap endavant, i la passa cap endarrere (o *reverse pass*) on aquesta informació viatja a les capes anteriors per a la corresponent actualització dels pesos.

Exemple 20. *Vegem un exemple d'aplicació de la propagació cap enrere amb la funció sigmoide com a funció d'activació* $f(v) = \frac{1}{1+e^{-v}}$.
En primer lloc, calculam la derivada i veim que compleix $f'(v) = f(v)(1 - f(v))$.

Suposem que hem elegit la següent configuració inicial per al pesos de la xarxa: $w^{h(0)}_{11} = 0.15$, $w^{h(0)}_{12} = 0.25$, $w^{h(0)}_{21} = 0.35$, $w^{h(0)}_{22} = 0.45$, $w^{o(0)}_{11} = 0.55$, *i* $w^{o(0)}_{12} = 0.65$. *Sigui* $x_d = [0.3, 0.7]^T$ *i* $y_d = 0.8$ *la nostra única observació. Recordem que, el nostre objectiu és ajustar la funció* $F : \mathbb{R}^2 \to \mathbb{R}$ *tal que* $y_d = F(x_d)$

Farem servir un pas $r = 5$ *per a la backpropagation.*

Començam amb la passa cap endavant (forward pass) tot calculant l'entrada als dos nodes de la capa amagada:

$$v^{(0)}_1 = w^{h(0)}_{11} x_{d1} + w^{h(0)}_{21} x_{d2} = 0.15 \cdot 0.3 + 0.35 \cdot 0.7 = 0.29$$
$$v^{(0)}_2 = w^{h(0)}_{12} x_{d1} + w^{h(0)}_{22} x_{d2} = 0.25 \cdot 0.3 + 0.45 \cdot 0.7 = 0.39$$

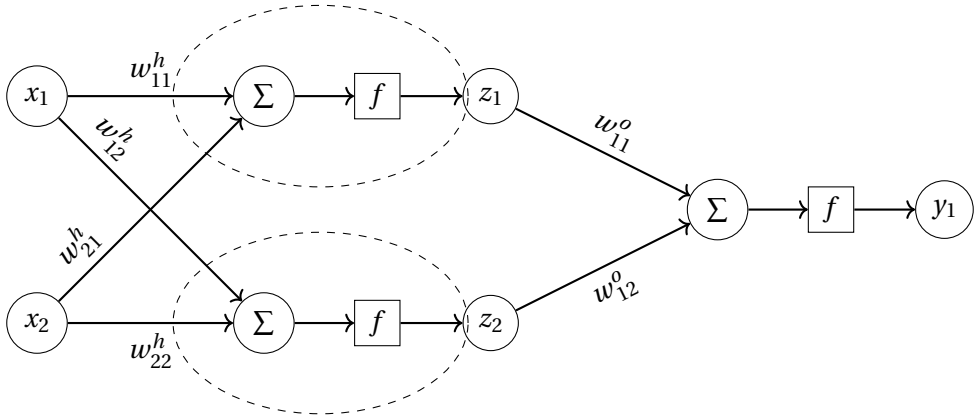


Figura 4.5: Exemple de ANN de 3 capes amb neurones sigmoide per a l'Exemple 13.2

A continuació, aplicam l'activació sigmoide

$$z_1^{(0)} = f(v_1^{(0)}) = \frac{1}{1+e^{-0.29}} = 0.5720$$
$$z_2^{(0)} = f(v_2^{(0)}) = \frac{1}{1+e^{-0.39}} = 0.5963$$

I finalment la sortida

$$\begin{aligned} y_1^{(0)} &= f\left(w_{11}^{o(0)} z_1^{(0)} + w_{12}^{o(0)} z_2^{(0)}\right) \\ &= f(0.55 \cdot 0.5720 + 0.65 \cdot 0.5963) \\ &= f(0.7005) \\ &= 0.6685 \end{aligned}$$

que ens permet calcular l'error:

$$\begin{aligned} \delta_1^{(0)} &= (y_d - y_1^{(0)}) y_1^{(0)} (1 - y_1^{(0)}) \\ &= (0.8 - 0.6685) \cdot 0.6685 \cdot (1 - 0.6685) \\ &= 0.0293 \end{aligned}$$

Aquí acaba la forward pass.
Anem amb la backpropagation actualitzant els pesos

$$\begin{aligned} w_{11}^{o(1)} &= w_{11}^{o(0)} + \eta \delta_1^{(0)} z_1^{(0)} \\ &= 0.55 + 5.0 \cdot 0.0293 \cdot 0.5720 \\ &= 0.55 + 0.0837 \\ &= 0.6337 \\ w_{12}^{o(1)} &= w_{12}^{o(0)} + \eta \delta_1^{(0)} z_2^{(0)} \\ &= 0.65 + 5.0 \cdot 0.0293 \cdot 0.5963 \\ &= 0.65 + 0.0873 \\ &= 0.7373 \end{aligned}$$

i fent servir que $f'(v_j^{(0)}) = f(v_j^{(0)})(1 - f(v_j^{(0)})) = z_j^{(0)}(1 - z_j^{(0)})$*, obtenim*

$$\begin{aligned}
\delta_1^h &= \delta_1^{(0)} w_{11}^{o(0)} z_1^{(0)}(1 - z_1^{(0)}) \\
&= 0.0293 \cdot 0.55 \cdot 0.5720 \cdot (1 - 0.5720) \\
&= 0.0038 \\
\delta_2^h &= \delta_1^{(0)} w_{12}^{o(0)} z_2^{(0)}(1 - z_2^{(0)}) \\
&= 0.0293 \cdot 0.65 \cdot 0.5963 \cdot (1 - 0.5963) \\
&= 0.0046
\end{aligned}$$

Actualitzam els pesos de la capa amagada:

$$\begin{aligned}
w_{11}^{h(1)} &= w_{11}^{h(0)} + \eta \delta_1^h x_{d1} \\
&= 0.15 + 5.0 \cdot 0.0038 \cdot 0.3 \\
&= 0.15 + 0.0057 \\
&= 0.1557 \\
w_{21}^{h(1)} &= w_{21}^{h(0)} + \eta \delta_1^h x_{d2} \\
&= 0.35 + 5.0 \cdot 0.0038 \cdot 0.7 \\
&= 0.35 + 0.0133 \\
&= 0.3633 \\
w_{12}^{h(1)} &= w_{12}^{h(0)} + \eta \delta_2^h x_{d1} \\
&= 0.25 + 5.0 \cdot 0.0046 \cdot 0.3 \\
&= 0.25 + 0.0069 \\
&= 0.2569 \\
w_{22}^{h(1)} &= w_{22}^{h(0)} + \eta \delta_2^h x_{d2} \\
&= 0.45 + 5.0 \cdot 0.0046 \cdot 0.7 \\
&= 0.45 + 0.0161 \\
&= 0.4661
\end{aligned}$$

Així es completa una iteració de l'algorisme de backpropagation. Efectivament ens hem apropat a un bon resultat en el sentit que $|y_d - y_1^{(0)}| = |0.8 - 0.6685| = 0.1315$*; ens hem apropat al valor de sortida òptim.*

Després de 25 iteracions, hom pot veure que s'arriba a la següent configuració de la Xarxa:

$$\begin{aligned}
w_{11}^{h(25)} &= 0.2453 \\
w_{21}^{h(25)} &= 0.4982 \\
w_{12}^{h(25)} &= 0.3621 \\
w_{22}^{h(25)} &= 0.5740 \\
w_{11}^{o(25)} &= 0.8192 \\
w_{12}^{o(25)} &= 0.9381
\end{aligned}$$

I l'output resultant per a l'input $x_d = [0.3, 0.7]^T$ *és* $y_1 = 0.7926$*, molt aprop de* $y_d = 0.8$.

4.2 Introducció a les *GNN*

Moltes de les dades amb què ens podem trobar venen representades per un graf. Són exemples del món real l'estructura de les molècules en la química així com les connexions d'una xarxa social. Com també ho són les imatges de píxels i el processament de text. Tots aquests exemples tenen en comú una manera de ser representats: arestes que vinculen diferents conjunts de nodes. Però com de diferent arriba a ser cada graf?

L'estructura dels grafs que trobam en la naturalesa és extremadament variada, desde xarxes amb molts nodes però poques arestes fins a tot el contrari.

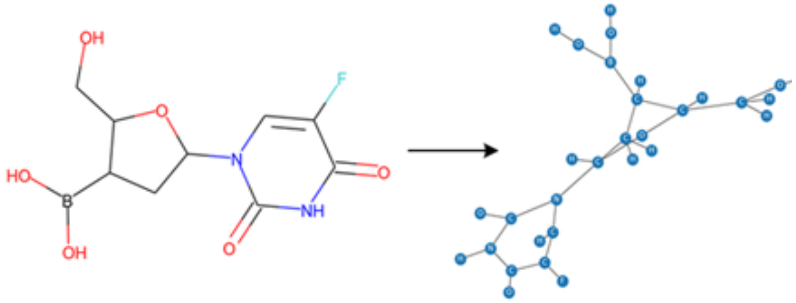


Figura 4.6: Exemple de la representació d'una molècula

Ara bé, una vegada tenim les nostres dades en forma de graf, ens podem plantejar resoldre problemes de predicció, com per exemple, la classificació. Distingim tres tipus de tasques de classificació en grafs:

- Classificació de **nodes**: predir el rol de cada node dins d'un graf.
- Classificació d'**arestes**: predir quins nodes comparteixen aresta i el seu valor.
- Classificació de **grafs**: predir la propietat d'un graf sencer.

Aquests tres problemes poden ser resolts amb una mateixa classe de model, una *GNN* (*Graph Neural Network* en anglès) [22].

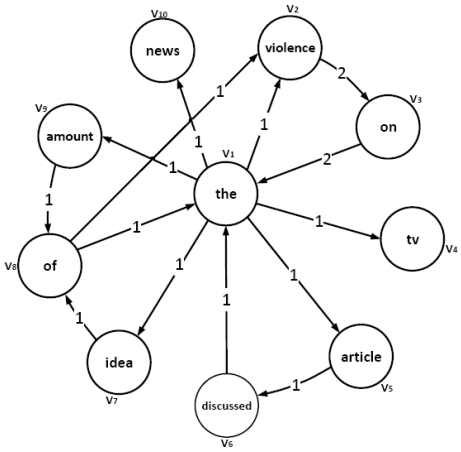


Figura 4.7: Els grafs també poden descriure text.

En aquest treball ens centrarem en un problema concret de **classificació de nodes**, com veurem més endavant.

4.2.1 Representació del model

Potencialment, la informació que volem fer servir d'un graf inclou les arestes (poden incloure informació addicional com un pes), els nodes (poden incloure un vector de característiques) i la connectivitat. Molts dels models d'Aprenentatge Automàtic fan servir *arrays* rectangulars com a input. Tanmateix, en el cas de grafs, representar la connectivitat amb una matriu d'adjacència pot ser excessivament costós ja que el nombre de nodes pot ser de l'ordre de milions, especialment quan la matriu conté un gran nombre de zeros (*sparse matrix* en anglès).

Una idea per a representar el nostre graf d'una manera alternativa a la matriu d'adjacència tradicional que, a més, és més eficient en termes de memòria és la de les llistes d'adjacència [22]. Per a descriure la connectivitat d'una aresta e_k entre nodes v_i i v_j usarem una tupla (i, j) en la k-èssima entrada d'una llista d'arestes. D'aquesta manera estem estalviant memòria ja que, normalment, el nombre d'arestes d'un graf G és menor a n^2, on $n = |V(G)|$.

Sigui $G = (V, E, l_v, l_e)$ un graf on l_v i l_e són vectors contenint les etiquetes de, respectivament els vèrtexos i les arestes de G. Donat un vèrtex v, definim el conjunt de veïns de v com $N(v) = \{u \in V(G) : uv \in E(G) | vu \in E(G)\}$

Suposem, en un marc molt general, que volem realitzar la següent tasca. Donat un graf G, ens interessa aproximar una funció objectiu $\tau(G, v) \in \mathbb{R}^m$, que envia cada node v d'un graf a un vector. Si volguéssim classificar grafs, τ no dependria de v, però no és el nostre cas. Es defineix l'*estat* (*state vector* en anglès) d'un node v com $\boldsymbol{x}_v \in \mathbb{R}^s$, amb s fixat. Una elecció raonable, en el marc d'aprenentatge automàtic és considerar l'estat de cada node com la imatge d'una funció paramètrica a ajustar. En aquest sentit, definirem la funció de transició d'estat (*state transition function* en anglès) f_w, que ha de recollir informació de cada node, la seva etiqueta, la seva connectivitat i dels estats dels seus nodes veïns [23, 24]:

$$\boldsymbol{x}_v = f_w(l_v, \boldsymbol{x}_{N(v)}, l_{N(v)})$$

on w denota un conjunt de paràmetres a determinar i l_v, $\boldsymbol{x}_{N(v)}$ i $l_{N(v)}$ són respectivament el vector d'etiquetes de v, l'estat dels seus nodes veïns i els seus vectors d'etiquetes.

Equivalentment, podem escriure $\boldsymbol{x_v}$ com una suma d'informació dels seus veïns, que més endavant anomenarem missatge:

$$\sum_{u \in N(v)} h_w(l_u, \boldsymbol{x_u}, l_v)$$

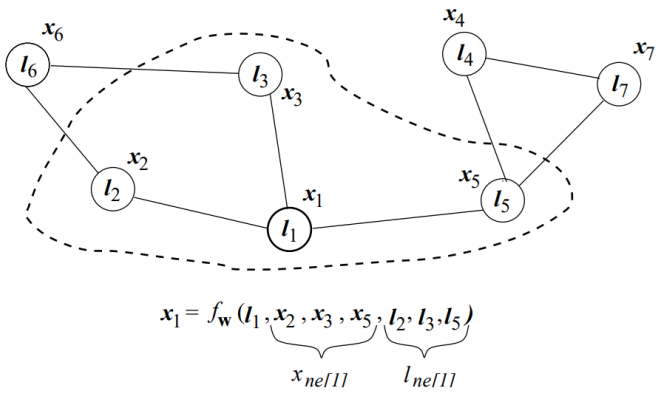


Figura 4.8: L'estat del node depèn dels seus nodes veïns

Per cada node v, també definirem el seu vector d'output $\boldsymbol{o}_v \in \mathbb{R}^m$ com la imatge per una funció paramètrica g_w del seu estat $\boldsymbol{x}_v$ i de la seva etiqueta l_v:

$$\boldsymbol{o}_v = g_w(\boldsymbol{x_v}, l_v)$$

Siguin $\boldsymbol{x}$ i $\boldsymbol{l}$ el resultat de concatenar tots els estats i etiquetes dels vèrtexos del graf. Les equacions anteriors es poden reescriure com:

$$\boldsymbol{x} = F_w(\boldsymbol{x}, \boldsymbol{l}) \tag{4.4}$$
$$\boldsymbol{o} = G_w(\boldsymbol{x}, \boldsymbol{l}) \tag{4.5}$$

on F_w i G_w denoten $|V(G)|$ instàncies de f_w i g_w, respectivament. Noti's que, $\boldsymbol{x}$ estarà ben definit si, i només si, l'equació 4.1 té solució i aquesta és única.

Definició 21. *Direm que una funció $f : X \to X$ té un punt fix $\boldsymbol{x*} \in X$ si es satisfà l'equació $f(\boldsymbol{x*}) = \boldsymbol{x*}$.*

Definició 22. *Sigui (X,d) un espai mètric. Un operador $T : X \to X$ és k-contractiu si existeix un nombre real k, $0 \leq k < 1$ tal que, per a tot $y_1 \in X$, $y_2 \in X$, $d(T(y_1), T(y_2)) \leq kd(y_1, y_2)$.*

Teorema 23 (Teorema del Punt Fix de Banach)**.** *Sigui (X, d) un espai mètric complet, si suposam que:*

i) Tenim un operador $T : M \to M \subseteq X$

ii) M és un conjunt tancat no buit en un espai mètric complet (X, d)

iii) T és un operador k-contractiu, és a dir:

$$d(T(x), T(y)) \leq k \cdot d(x, y) \quad \forall x, y \in M \text{ amb } k \in [0, 1) \text{ fixada}$$

Llavors, tenim el següent:

a) T té un únic punt fix a M.

b) La successió $\{x_n\}$ d'iteracions donada per $x_{n+1} = T(x_n)$ convergeix al punt fix x^* per un punt $x_0 \in M$ inicial arbitrari.

Demostració. Com ens trobam en un espai mètric complet (X, d), sabem que tota successió de Cauchy és convergent a un punt de l'espai. Aleshores, el què farem serà veure que els termes de la successió d'iterats de T és, en efecte, de Cauchy.

És a dir, volem veure que:

$$\forall \epsilon > 0 \; \exists n_0 \in \mathbb{N} \text{ tal que } \forall m, n \geq n_0 : d(x_m, x_n) < \epsilon$$

Sigui $x_0 \in M$ arbitrari. Si $T(x_0) = x_0$, ja hem acabat, en cas contrari, procedirem a calcular:

$$\{x_0, x_1 = T(x_0), x_2 = T(x_1), ..., x_n = T(x_{n-1}), ...\}$$

Com l'operador T és k-contractiu, es compleix $d(T(x),T(y)) \leq k \cdot d(x,y)$ i, fent servir la recursivitat en la definició dels x_n es té la següent cadena de desigualtats:

$$\begin{aligned} d(x_{n+1},x_n) = d(T(x_n),T(x_{n-1})) &\leq k \cdot d(x_n,x_{n-1}) && (4.6)\\ &\leq k^2 \cdot d(x_{n-1},x_{n-2}) && (4.7)\\ &\vdots && (4.8)\\ &\leq k^n \cdot d(x_{n-n+1},x_{n-n}) = k^n \cdot d(x_1,x_0) && (4.9) \end{aligned}$$

Sense pèrdua de generalitat, suposarem $m > n \geq 0$ i, fent servir la desigualtat triangular i la fita derivada anteriorment, veiem:

$$\begin{aligned} d(x_m,x_n) &\leq d(x_m,x_{n+1}) + d(x_{n+1},x_n) && (4.10)\\ &\leq d(x_m,x_{n+2}) + d(x_{n+2},x_{n+1}) + d(x_{n+1},x_n) && (4.11)\\ &\vdots && (4.12)\\ &\leq d(x_m,x_{m-1}) + d(x_{m-1},x_{m-2}) + \ldots + d(x_{n+1},x_n) && (4.13)\\ &\leq k^{m-1}d(x_1,x_0) + k^{m-2}d(x_1,x_0) + \ldots + k^n d(x_1,x_0) && (4.14)\\ &= (k^{m-1} + k^{m-2} + \ldots + k^n)d(x_1,x_0) && (4.15) \end{aligned}$$

Notem que els termes $k^{m-1} + k^{m-2} + \ldots + k^n$ són suma d'una progressió geomètrica i, en conseqüència, podem calcular el valor de la seva suma:

$$\sum_{i=n}^{m-1} k^i = \frac{k^n - k^m}{1-k} = \frac{k^n(1-k^{m-n})}{1-k}$$

Per tant:

$$d(x_m,x_n) \leq (k^{m-1} + k^{m-2} + \ldots + k^n)d(x_1,x_0) = \frac{k^n - k^m}{1-k}d(x_1,x_0) \underset{m>0}{\leq} \frac{k^n}{1-k}d(x_1,x_0)$$

Com el valor $k \in [0,1)$ és un valor fixat, la successió $\{k^n\}_{n=1}^{\infty}$ convergeix cap a 0, és a dir:

$$\forall \epsilon_1 > 0, \exists \tilde{n} \geq 0 : \forall n \geq \tilde{n}, k^n < \frac{\epsilon_1(1-k)}{d(x_0,x_1)}$$

Notem que la distància entre el punt x_0 i x_1 està fixada, no depèn de n, llavors podem considerar $\epsilon_1 = \frac{\epsilon(1-k)}{d(x_0,x_1)}$, en conseqüència tenim:

$$d(x_m,x_n) \leq \frac{k^n}{1-k}d(x_0,x_1) < \frac{\epsilon_1(1-k)}{d(x_0,x_1)} \cdot \frac{1}{1-k} \cdot d(x_0,x_1) = \epsilon$$

Per tant, acabam de veure que $\{x_n\}_{n=1}^{\infty}$ és successió de Cauchy. Com ens trobam en un espai mètric complet, la successió de Cauchy és convergent i per tant:

$$\exists x^* \in X : \lim_{n\to\infty} x_n = x^*$$

Vegem que, per ser T un operador contractiu, aleshores és continu.

Volem veure que:

$$\forall \epsilon > 0, \exists \delta > 0 \text{ tal que si } d(x,y) < \delta \Rightarrow d(T(x),T(y)) < \epsilon$$

Siguin $x, y \in M$, aplicant la contractivitat de l'operador i prenent $\delta = \frac{\epsilon}{k}$ (si $k \neq 0$), obtenim el que volíem veure:

$$d(T(x), T(y)) \leq k \cdot d(x, y) < k \cdot \delta = k \cdot \frac{\epsilon}{k} = \epsilon$$

Si $k = 0$ aleshores és trivialment contínua, ja que la distància és una funció no-negativa.

Podem aplicar ara el resultat que ens diu que, donada una funció contínua i una successió, la successió d'imatges convergeix a la imatge del límit. En efecte, per ser T continu la successió $\{T(x_n)\}_{n=1}^{\infty}$ convergeix a $T(x^*)$. Ara bé, per la forma en la qual hem construït la successió $\{x_n\}_{n=1}^{\infty}$ tenim que:

$$\{T(x_n)\}_{n=1}^{\infty} = \{x_n\}_{n=2}^{\infty}$$

La primera successió tendeix a $T(x^*)$ com hem acabat de veure i la segona tendeix a x^*. Per tant, $T(x^*) = x^*$, és a dir, x^* és un punt fix de l'operador T. Per tenir $T(M) \subseteq M \subseteq X$ i $x_0 \in M$ es té que $x_n \in M \ \forall n$. Llavors per ser $M \subseteq X$ un tancat, el límit de la successió també hi pertany. Per tant, queda demostrada l'existència d'un punt fix $x^* \in M$.

Per veure la unicitat, suposem que $\exists x^*, y^*$ tals que $x^* = T(x^*)$ i $y^* = T(y^*)$, llavors:

$$d(x^*, y^*) = d(T(x^*), T(y^*)) \leq k \cdot d(x^*, y^*)$$

.

Per ser $d : X \times X \to \mathbb{R}$ una mètrica, si $k = 0$ aleshores

$$d(x^*, y^*) = 0 \Rightarrow x^* = y^*$$

i, altrament, si $k \in (0, 1)$ és contradicció amb el fet que $d(x^*, y^*) \leq k \cdot d(x^*, y^*)$. Deduim per tant el que volíem veure: T té un únic punt fix en M.

□

Una estratègia àmpliament usada per a resoldre l'equació 4.1 és precisament, escollir F_w que sigui una funció contractiva i usar un mètode iteratiu que s'aproximi el suficient al punt fix (que sabem que existeix pel Teorema anterior).

Tornem als nostres objectius. Similarment a capítols anteriors, plantejam matemàticament el nostre problema com el de trobar els paràmetres $\boldsymbol{w}$ que minimitzen una certa funció de pèrdua:

$$e_w = \sum_{i=1}^{p} (y_i - \varphi_w(G, v_i))^2.$$

Ara, per implementar el model descrit anteriorment, necessitem:

1. Un mètode per resoldre les equacions 4.1 i 4.2 (Banach).
2. Un algoritme d'aprenentatge per optimitzar w: mètode de descens del gradient.
3. Una implementació de f_w i g_w.

Càlcul dels estats

L'algorisme iteratiu que proposa Banach en la demostració ens fa plantejar-nos resoldre:

$$x(k+1) = F_w(x(k), l) \tag{4.16}$$

on $\boldsymbol{x}(k)$ denota l'iterat k-èssim de $\boldsymbol{x}$.

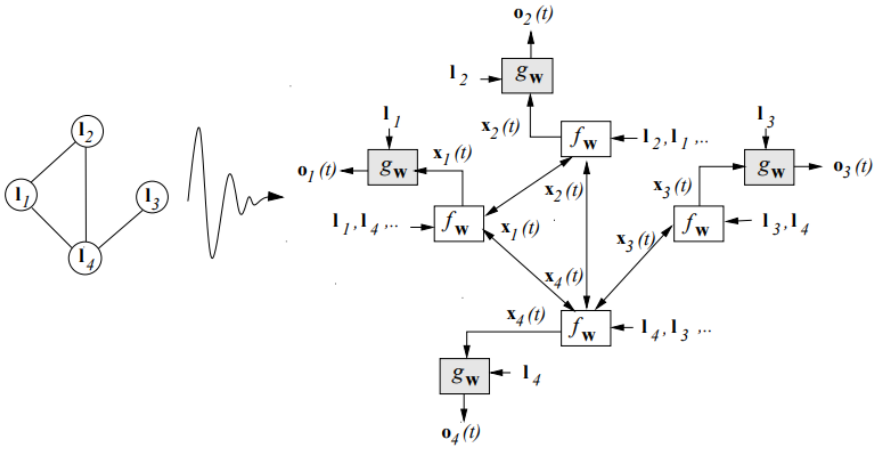


Figura 4.9: L'estat d'un node depèn dels estats anteriors dels seus veïns

És a dir, hem de calcular, per cada node:

$$x_v(k+1) = f_w(l_v, \boldsymbol{x}_{N(v)}(k), l_{N(v)}) \tag{4.17}$$

$$o_v(k+1) = g_w(x_v(k+1), l_v) \tag{4.18}$$

Algorisme d'aprenentatge:

(a) Els estats $x_v(k)$ s'actualitzen iterativament segons 4.4 fins que la successió 4.3 convergeix a un punt fix.

(b) Es calcula el gradient $\frac{\partial e\mathbf{w}^{(T)}}{\partial \mathbf{w}}$ i els pesos $\mathbf{w}$ s'actualitzen tot seguint una estratègia de descens del gradient.

D'aquesta manera, l'apartat (a) ens duu el sistema a una solució estable mentre que el (b) adapta els pesos per obtenir les respostes desitjades. Per fer-ho es poden fer servir algorismes similars a la *backpropagation* vista anteriorment.

4.2.2 La idea del *message-aggregation*

La característica principal d'una *GNN* és la transmissió d'informació entre nodes [25], un marc de treball que anomenarem *message-passing*. Com hem vist, a cada iteració del *message-passing*, un vector $\boldsymbol{h}_u^{(k)}$ corresponent a cada node $u \in V(G)$ s'actualitza agregant informació dels seus veïnats en $N(u)$ i aplicant una xarxa neuronal.

Podem descomposar una capa *GNN* en dues parts més simples. En general, primer es crea el missatge en cada node i després s'agrega la informació de tots els veïns (*message passing*). D'aquesta manera, cada node crea un missatge a partir dels seu estat:

$$\boldsymbol{m}_v^{(l)} = MSG^{(l)}(\boldsymbol{h}_v^{(l-1)})$$

on MSG és una funció *missatge* qualsevol. I després s'ajunta la informació dels veïns de v:

$$\boldsymbol{h}_v^{(l)} = AGG^{(l)}(\{\boldsymbol{m}_u^{(l)} : u \in N(v)\})$$

on L'exemple més bàsic de la implementació d'aquesta idea del *message-passing* és:

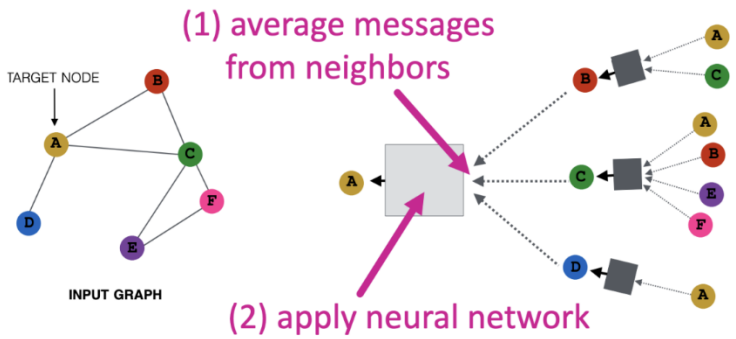


Figura 4.10: Esquema de l'estructura *message-passsing*

$$\begin{aligned} \boldsymbol{h}_v^{(0)} &= \boldsymbol{x}_v \\ \boldsymbol{h}_v^{(k)} &= \sigma\left(\mathbf{W}_{\text{self}}^{(k)} \boldsymbol{h}_v^{(k-1)} + \mathbf{W}_{\text{neigh}}^{(k)} \sum_{u \in N(v)} \boldsymbol{h}_u^{(k-1)}\right) \\ \boldsymbol{z}_u &= \boldsymbol{h}_v^{(K)} \end{aligned}$$

on $\mathbf{W}_{\text{self}}^{(k)}, \mathbf{W}_{\text{neigh}}^{(k)} \in \mathbb{R}^{d^{(k)} \times d^{(k-1)}}$ són matrius de pesos a optimitzar i σ una funció no lineal (p.ex sigmoide o ReLu).

Aquesta manera de recoplilar informació és capaç de transmetre informació estructural del graf a cada node. Si $\boldsymbol{z}_v = \boldsymbol{h}_v^{(K)}$ és el resultat d'aplicar K capes d'agregació al vector de característiques de v, aleshores emmagatzema informació de nodes a distància K. És a dir, a mesura que es van succeïnt les iteracions sobre l'agregació de missatges, anam obtenint informació de nodes que es troben més 'allunyats'.

4.2.3 *Graph attention network (GAT)*

Una altra manera de tractar l'agregació de missatges és la *neighbourhood attention*. La idea bàsica que hi ha al darrere és assignar un pes a cada node veïnat per ponderar la seva 'importància' en el missatge [26, 27]. El primer model de *GNN* que va incorporar aquesta visió va ser la *Graph Attention Network* (Graph Attention (GAT)) de Velickovic et al. [26] que fa servir *coeficients d'atenció*.

Es vol 'aprendre' quines són les parts importants de la xarxa. Volem quelcom de la forma:

$$\boldsymbol{h}_u^{(l)} = \sigma\Big(\sum_{v \in N(u)} \alpha_{u,v} \boldsymbol{W}^{(l)} \mathbf{h}_v^{(l-1)}\Big). \tag{4.19}$$

Per obtenir els coeficients d'atenció, es fa servir una funció $a(.)$ anomenada *mecanisme d'atenció*. Aquesta calcula, a partir dels missatges:

$$e_{vu} = \boldsymbol{a}(\boldsymbol{m}_u^{(k)}, \boldsymbol{m}_v^{(k)})$$

on $e_{u,v}$ denota la importància del missatge del node u per v. Ara, cada e_{vu} és normalitzat per obtenir els *coeficients d'atenció* finals:

$$\alpha_{u,v} = \frac{\exp(e_{vu})}{\sum_{v' \in N(u)} \exp(e_{v'u})}.$$

Ara bé, quin és el mecanisme d'atenció? Existeixen diferents opcions, típicament una combinació lineal o bilineal dels missatges o una capa d'una xarxa. La idea és que sigui entrenable juntament amb la resta de paràmetres. En el model original del *Graph Attention Network* [26] es fa servir el següent mecanisme d'atenció:

$$\alpha_{u,v} = \frac{\exp\left(\mathbf{a}^\top \left[\mathbf{W}\mathbf{h}_u \,\|\, \mathbf{W}\mathbf{h}_v\right]\right)}{\sum_{v' \in N(u)} \exp\left(\mathbf{a}^\top \left[\mathbf{W}\mathbf{h}_u \,\|\, \mathbf{W}\mathbf{h}_{v'}\right]\right)},$$

on $\boldsymbol{a}$ és un vector a entrenar i || denota concatenació.

Així mateix, també és possible construir diversos *attention heads*. Com implementarem més endavant, es calculen K coeficients d'atenció $\alpha_{u,v,k}$ fent servir capes d'atenció independents tot reproduint el procés de l'equació 4.19, i es fa la seva mitjana:

$$\tilde{h}_u = \|_{k=1}^{K} \sigma\left(\sum_{v \in N(u)} \alpha_{u,v}^k \mathbf{W}^k \boldsymbol{h}_v^{(l-1)}\right) \tag{4.20}$$

$$\boldsymbol{h}_u = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{v \in N(u)} \alpha_{ij}^k \mathbf{W}^k \boldsymbol{h}_v^{(l-1)}\right). \tag{4.21}$$

CAPÍTOL 5

PREDICCIONS PER A L'*Eschericia Coli K12*

En el Capítol 3 hem vist com el FBA, una eina basada en l'optimització, ens permet predir distribucions en el flux d'una cèl·lula. En el nostre objectiu d'entendre la fisiologia cel·lular, aquest mètode ha donat molt bons resultats en el cas de l'*Escherichia coli*. L'*Escherichia coli* o *colibacil* és un dels principals tipus d'eubacteris que viuen en la part baixa dels intestins dels animals de sang calenta. Resulta imprescindible per a la bona digestió dels aliments i el seu nom prové del seu descobridor, Theodor Escherich [28].

Els models metabòlics a escala genòmica resulten imprescindibles per a reconstruir la fisiologia cel·lular. El més complet per a l'estudi de l'*Escherichia* és el model iML1515 [10]. Conté 2719 reaccions metabòliques tot involucrant 1192 metabòlits que defineixen el seu graf metabòlic. Així mateix, inclou les relacions enzim-reacció que discutirem més endavant. Podem trobar aquest conjunt de dades en el repositori 'BiGG models': http://bigg.ucsd.edu/models/iML1515.

5.1 Què és un gen essencial?

La identificació de gens essencials d'una cèl·lula no només ens dona informació sobre la mateixa, també té moltes aplicacions en la biotecnologia i la biomedicina. En una teràpia de càncer, pot ser clau identificar un gen essencial d'una cèl·lula cancerosa per millorar l'eficàcia del tractament. En la biotecnologia industrial, els gens no essencials poden ser eliminats per a optimitzar el flux metabòlic cap a la producció de quelcom d'interès.

El Flux Balance Analysis (FBA) ens permet predir amb un cert error, gens essencials. És a dir, aquells que alteren la probabilitat de supervivència de la cèl·lula. Per a determinar si un gen és essencial o no de manera empírica, cal fer un estudi de laboratori on s'elimini el gen en qüestió. Només això ens dona l'etiqueta vertadera (o *ground truth* en anglès) del gen. Ara bé, l'eficàcia del FBA rau, precisament en el fet que és capaç de fer prediccions amb una *accuracy* del 0.934 sobre 1 pels gens de l'*Eschericia coli K12* [10].

En el Capítol 3 definíem el FBA com la capacitat que té una cèl·lula per optimitzar una funció objectiu (típicament, la de biomassa). Una cèl·lula pot desviar els seus objectius metabòlics quan detecta la supressió d'un gen [1], en el sentit que, deixa d'optimitzar la mateixa funció objectiu. Per tant, és complicat parlar de taxes de creixement en soques mutants. Definirem la *taxa de creixement* d'una cèl·lula en un estat d'equilibri de flux com el valor que prèn la funció objectiu resultant d'aplicar FBA, i.e el valor *emph* del problema de maximització amb restriccions.

Definim el Gene-To-Protein Reaction (GPR) com una funció entre el conjunt de gens $\mathscr{G}$ i el conjunt de parts del conjunt de les reaccions $\mathscr{P}(\mathscr{R})$. El GPR ens dóna una connexió formal explícita entre genotip i fenotip en una reconstrucció a escala genòmica [10], concretament, relaciona el gen (G) amb la proteïna (P) que *catalitza* la reacció (R) en la xarxa. D'aquesta manera, podem establir una relació de dependència entre gens i distribució de flux d'una xarxa metabòlica. Quan eliminem un gen $G \in \mathscr{G}$ aquest provoca la desactivació d'un conjunt de reaccions $GPR(G) \subseteq \mathscr{R}$ que, lògicament, alteren la distribució de flux de la xarxa. En efecte, eliminar un gen és afegir un conjunt nou de restriccions al problema de FBA, ja que cal imposar $\boldsymbol{v}^*(R) = 0, \forall R \in GPR(G)$, fet que condicionarà la nova *taxa de creixement* de la cèl·lula.

Definició 24. *Sigui T^* el valor òptim per a un problema de FBA sense restriccions del tipus $\boldsymbol{v}^*(R) = 0$ per qualque $R \in \mathscr{R}$. Aleshores, direm que un gen $G \in \mathscr{G}$ és FBA-essencial, si i només si:*

$$\frac{T_G}{T^*} < 0.5 \tag{5.1}$$

on T_G és el valor òptim del problema de FBA amb les restriccions resultants d'eliminar el gen G.

Noti's la diferència entre la definició d'un gen essencial en el sentit de *ground truth* (informació extreta d'un assaig de laboratori) i la de FBA-essencialitat. En aquest treball s'han fet servir les etiquetes *ground truth* de l'article [10]. Llavors no s'ha assumit l'optimalitat universal. Pel que fa a la implementació del GPR, en aquest treball ens hem reduït a estudiar només les relacions 1 a 1.

5.2 Arquitectura del model

El nostre model, inspirat en *FlowGAT* [1], és una implementació en Python d'una estratègia híbrida FBA-GNN per a la predicció de gens essencials. La clau del model és la representació en forma de graf de la xarxa metabòlica del flux predit per l'FBA a través dels MFG. El nostre objectiu és complementar o millorar els assajos de laboratori que ens donen les etiquetes *ground truth*, certament molt costosos, en la cerca de mètodes computacionals eficients.

En la nostra GNN, cada node del graf es correspon amb una reacció (en tant que MFG) que té associat un índex d'essencialitat a predir en la tasca de classificació binària de nodes. Ara bé, com passam de gens essencials a reaccions/nodes essencials? Un gen pot desactivar múltiples reaccions, però una reacció pot ser desactivada també per múltiples gens. En aquest entramat de relacions lògiques, l'aproximació que dona millors resultats és la més simple. Per aquells parells gen-reacció (G-R) amb una relació d'un a un (i.e, $|GPR(G)| = |GPR^{-1}(GPR(G))| = 1$), hem transferit l'etiqueta (o bé 0 o bé 1) directament del gen G al node $R = GPR(G)$. Per qualsevol altre cas, hem assignat un -1 al node, tot indicant que no participarà de les mètriques d'avaluació del model.

El MFG resultant per a la xarxa metabòlica de l'*Escherichia coli K12* en condicions de creixement aeròbic en glucosa com a única font de carboni, després d'eliminar aquells nodes desconnectats (i.e, aquells amb un flux de 0) compta amb 434 nodes, 185 dels quals són essencials (1), 66 no essencials (0) i 183 no classificats (-1).

5.2.1 Processament de les dades

Per obtenir les dades amb què hem entrenat el model, hem seguit les següents passes:

1. Descarregar i llegir el model metabòlic en format *json* fent servir la llibreria COBRA [2].
2. Aplicar FBA amb les restriccions corresponents.

3. Llegir l'etiquetatge *ground truth* per a l'*Escherichia coli*de gens essencials segons l'article [10] i aplicam el GPR amb la informació del model.

4. Construim el MFG segons la distribució de flux obtinguda en el punt 2. Afegim les etiquetes d'essencialitat obtigudes en el punt anterior.

5.2.2 Entrenament del model

Una vegada hem processat les dades, farem servir un conjunt de tècniques que milloren l'entrenament de la GNN.

El *Grid-Search* és un mètode de cerca de millors *hiperparàmetres* d'un model d'Aprenentatge Automàtic. Definim un hiperparàmetre com un valor prefixat NO entrenable que es fa servir en qualque passa de l'entrenament. En el nostre cas, un exemple pot ser la taxa d'aprenentatge per a la *backpropagation*. Aleshores el que fa *Grid-Search* és entrenar el model moltes vegades amb totes les possibles combinacions d'hiperparàmetres (la *hyperparameter grid*). De tots aquests aprenentatges, ens quedem amb el que doni millors mètriques. Per xarxes de paràmetres molt grans, pot ser computacionalment costós ja que requereix un nombre molt alt d'entrenaments distints. Nosaltres hem fet feina amb la següent *hyperparameter grid* bastant reduïda:

- ***hidden_features*** = $\{8, 16, 32\}$: Prefixa la dimensionalitat de la representació de cada node en la capa de la GAT. Quan més gran, més informació recollim de cada node.
- ***num heads*** = $\{4, 8\}$: Controla el nombre de *attention heads* en el mecanisme de *multi-head* attention.
- ***dropout*** = $\{0.3, 0.5\}$: És la proporció de característiques que s'ajusten a 0 en l'entrenament. Amb l'objectiu d'aconseguir un entrenament més robust (similar a la regularització).
- ***lr*** = $\{0.01, 0.005, 0.001, 0.0005\}$: És la *learning rate*, o la mida del pas en un algorisme de descens.
- ***weight_decay*** = $\{5e-4, 1e-4\}$: És el terme que s'afegeix en la funció de *loss* proporcional a la suma dels quadrats dels coeficients a entrenar. L'objectiu és regularitzar els coeficients en el sentit de la norma L^2.
- ***use_class_weights*** = $\{True, False\}$: Com el seu nom indica, determina si s'ha d'ajustar la funció de loss per donar-li més importància a la classe minoritària o no.

Amb les nostres dades, el resultat ha estat el següent:

$$\begin{aligned} \textbf{\textit{hidden\_features}}&: 32,\\ \textbf{\textit{num\_heads}}&: 8,\\ \textbf{\textit{dropout}}&: 0.5,\\ \textbf{\textit{lr}}&: 0.01,\\ \textbf{\textit{weight\_decay}}&: 0.0005,\\ \textbf{\textit{use\_class\_weights}}&: False \end{aligned}$$

L'*overfitting* o sobreajustament d'un model ocorre quan és capaç d'ajustar bé les dades d'entrenament però falla de manera consistent en dades que no ha vist mai (o dades tipus *test*). Per evitar o detectar l'*overfitting* existeixen un conjunt de mètodes que milloren l'avaluació de

l'aprenentatge. La *k-fold Cross Validation* és una tècnica que consisteix en dividir el conjunt d'entrenament en *k* parts iguals (o no), anomenades *folds*. Aleshores, entrenam el model amb la unió d'un conjunt de *folds* i en deixem un per a la *validació*. Aquest darrer *fold*, que no ha vist mai, és amb el que farem una predicció sencera i avaluarem la qualitat del model entrenat. Aquest procediment es repeteix per a cada un dels k-folds i al final es fa la mitjana de les mètriques obtingudes en cada cas. El *Stratified k-fold* n'és una variant que s'assegura que en cada *fold* hi hagi la mateixa proporció d'observacions de la mateixa classe. És especialment apropiat en el nostre cas, on el nombre d'observacions de cada classe està esbiaixat cap a la classe positiva (1).

5.3 Avaluació del model

5.3.1 Mètriques en classificació binària

Donat el nostre graf metabòlic G i una predicció feta amb el nostre model. Definim:

- TP (vertaders positius) com el nombre total de nodes essencials que s'han classificat com a tal.
- FN (falsos negatius) com el nombre total de nodes que es corresponen amb reaccions essencials però que han estat classificats com a no essencials.
- FP (falsos positius) com el nombre total de nodes no essencials que s'han classificat com essencials.
- TN (vertaders negatius) com el nombre total de reaccions no essencials classificades com a tal.

Per avaluar la qualitat del nostre model, cal introduir una sèrie de mètriques sobre les prediccions realitzades que ens permetran determinar els punts més forts i més dèbils de la xarxa:

- La *precisió* mesura la proporció de vertaders positius (TP) entre totes les instàncies classificades com a positives (siguin falses o no). És a dir, estableix una mesura sobre les prediccions positives.
$$precisió = \frac{TP}{TP + FP}$$
- El *recall* (o sensibilitat) es centra en la capacitat de discriminació d'instàncies postives. És la qualitat de la qualitat de la predicció només tinguent en compte la classe positiva. Calcula la proporció entre TP i tots els nodes que són en realitat positius:
$$recall = \frac{TP}{TP + FN}$$
- El *F1 score* es defineix com la mitjana harmònica entre la precisió i el *recall*. És per tant una mètrica que penalitza valors baixos de sensibilitat i precisió (recordem que la mitjana harmònica de dos nombres és sempre menor o igual a la seva mitjana aritmètica).
$$F1 = \frac{2TP}{2TP + FP + FN}$$

- L'*accuracy* és una manera global de mesurar la ràtio d'encerts. Calcula la proporció de prediccions correctes sobre el nombre total de prediccions:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Quina de les anteriors mètriques podríem pensar que és la "millor"? Si bé, ens interessa que tots aquests valors estiguin a prop de l'1 per poder demostrar l'eficàcia del nostre model, no hi ha una resposta universal per tots els casos.

Suposem, per exemple, que tenim un model que no distingeix entre una observació positiva d'una negativa. La seva predicció és consistentment defectuosa i classifica com essencial totes les observacions. Aleshores el *recall* serà exactament igual a 1 (no hi ha falsos negatius!). Suposem, a més, que el nostre *dataset* presenta un desequilibri important en el nombre d'observacions de cada classe, a favor de la classe positiva. Llavors la precisió del model s'acostarà a 1 en proporció al desequilibri de les dades. En efecte, amb un *dataset* amb 90 observacions positives i 10 negatives obtindrem una precisió del 0.9! En conseqüència, el *F1 score* també tindria un valor entre el 0.9 i l'1 amb un model incapaç de capturar cap patró en les dades.

Necessitem, per tant, trobar una mètrica **sensible a l'equilibri entre classes**, Per això, definirem el F1-score *classe a classe*.

- L'*especificitat* mesura la capacitat per identificar observacions de la classe negativa. De manera similar a com ho hem fet anteriorment:

$$especificitat = \frac{TN}{TN + FP}$$

- El Negative predictive value (NPV) es defineix de la següent manera:

$$NPV = \frac{TN}{TN + FN}$$

Fixem-nos que aquestes dues definicions són ben homòlogues a la *sensibilitat* i la *precisió* però per a la classe negativa. Llavors la seva mitjana harmònica es correspon amb el F1-score de les classes invertides. Això ens porta a definir el F1-score mitjà (*Average* F1-score en anglès) com la mitjana de les diferents F1-score classe a classe. Aquesta és una mètrica invariant a la inversió de classes. És a dir, una mesura que dona la mateixa importància a les dues classes.

Amb la llibreria *Scikit-learn* de Python [29], es pot implementar el F1-score mitjà amb el paràmetre "average=macro". A la Figura 5.1 es pot veure un exemple d'una predicció feta amb el nostre model.

5.3.2 Altres mètriques

Hem vist una manera relativament simple d'obtenir mesures de rendiment d'un model una vegada assignades etiquetes binàries a les nostres observacions. Ara bé, què passa quan treballem amb funcions de predicció contínues? Com assignam una etiqueta binària a cada node? Aquest és el cas de les funcions de predicció sigmoides (Secció 4.1), que són les que hem fet servir en aquest treball.

Com hem vist anteriorment, cal escollir un llindar $l \in (0,1)$ que divideixi la imatge de la funció sigmoide en dues classes:

$$P(z) = \begin{cases} 1 & \text{si } \sigma(z) \geq l \\ 0 & \text{si } \sigma(z) < l \end{cases}$$

Classification Report:				
	precision	recall	f1-score	support
0	0.94	0.72	0.82	43
1	0.94	0.99	0.97	208
accuracy			0.94	251
macro avg	0.94	0.86	0.89	251
weighted avg	0.94	0.94	0.94	251

Figura 5.1: Mètriques per a la predicció de gens essencials en l'*Escherichia Coli* K12

on P és l'etiqueta final en la predicció i $0 < \sigma(z) < 1$ és la imatge de la funció sigmoide, també anomenada la *prediction score*.

L'elecció d'aquest llindar es pot assignar per defecte al 0.5, o bé es pot optimitzar amb l'objectiu de millorar alguna de les mètriques vistes en la subsecció anterior. Per exemple, suposem que la nostra xarxa neuronal ens acaba donant una sèrie de *prediction scores* compresos entre el 0 i l'1 i coneixem l'etiqueta real de cada node:

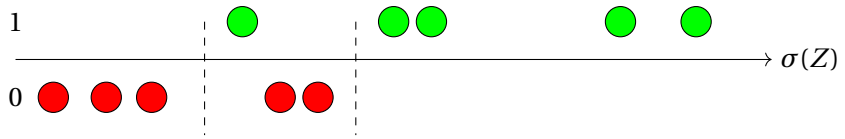


Figura 5.2: Dos llindars diferents en un mateix problema de classificació

Com que **no hi ha cap llindar que separi perfectament les observacions** positives de les negatives, proposam dos llindars diferents (vegi's la Figura 5.2). Calculem les corresponents mètriques:

Llindar	*precisió*	*sensibiltat*	*F1*
l_1	$\frac{5}{7}$	1	0.83
l_2	1	$\frac{4}{5}$	0.89

Podem veure que ambdós llindars obtenen "bons"resultats alhora de discriminar entre classes. Per una banda, un captura totes les instàncies realment positives i l'altre les negatives. Hom podria considerar doncs, un nou llindar en un punt intermig, per obtenir així un rendiment menys esbiaixat.

Tanmateix, aquesta és una elecció que **depèn del problema**. En l'àmbit sanitari, la detecció de casos positius pot arribar a ser prioritària, llavors es busca optimitzar la sensibilitat. En canvi, la detecció semiautomàtica d'un possible *offside* en un partit de futbol pot decantar de manera significativa el resultat d'una final. És per això que es busca una precisió d'allò més alta possible.

Tornem a l'exemple de la Figura 5.2. Què passaria si movem el llindar a posicions més properes al 0? El nombre de falsos negatius decreix fins que la sensibilitat queda ben igual a 1. Anàlogament, si movem el llindar a posicions més altes, la precisió esdevé immillorable a

partir d'un cert punt. Això és un símptoma que el nostre classificador ha capturat un patró en les dades, és a dir, que separa força bé les dues classes.

Suposem ara que tenim un classificador perfecte, en el sentit que fa la següent predicció (vegi's Figura 5.3). Vegem com són de diferents les relacions entre precisió i sensibilitat a mesura que fem avançar el nostre llindar:

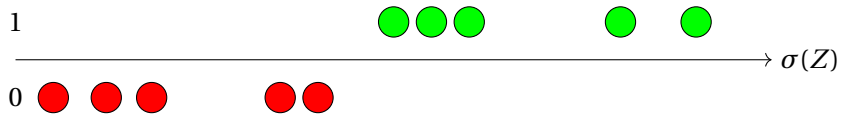


Figura 5.3: El classificador ideal

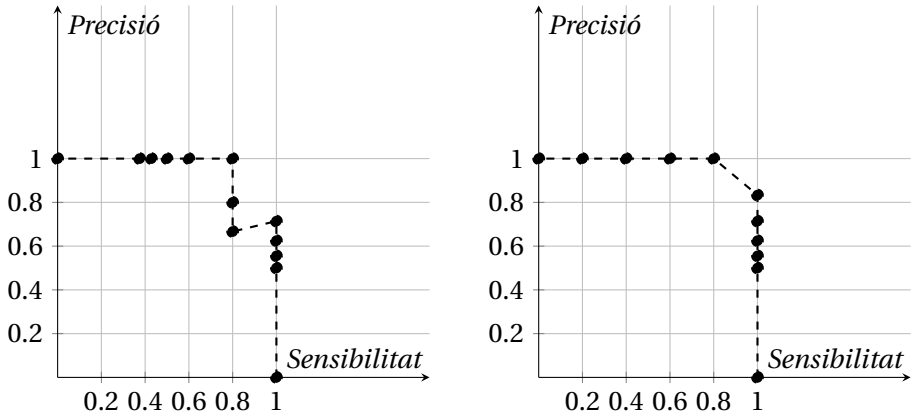


Figura 5.4: El *trade-off* entre precisió i sensibilitat. A la dreta hi ha el classificador perfecte. A l'esquerra el classificador de la Figura 5.2.

La Precision-Recall Area Under Curve (PRAUC) és una de les mètriques més usades en problemes de classificació i es basa en la compensació precisió-sensibilitat per avaluar la qualitat del nostre model. Com el seu nom indica, calcula l'àrea davall de la corba de *precision-recall*. Idealment, la corba que maximitza aquesta àrea és el quadrat, que té àrea 1. Llavors es considera un model prou bo aquell que hi tengui una PRAUC prou semblant.

5.4 Resultats i Generalització

A la figura 5.6 veiem els resultats obtinguts pel nostre model entrenat amb el MFG del metabolisme de l'*Escherichia coli K12*.

El *Bacillus Subtilis* és un bacteri habitualment trobat en el sòl gastrointestinal d'humans i d'animals remugants. N'hem construït el MFG a partir del model 'iYO844'. El que farem doncs, serà plantejar-nos dues qüestions. En primer lloc, podem entrenar el nostre model amb només la mostra del *Bacillus*? En tal cas, podem realitzar un entrenament conjunt amb ambdues espècies o amb una tercera?

Certament, el MFG del *Bacillus* conté 1250 reaccions i 990 metabòlits i el model estàndard de FBA ens dóna una taxa de creixement del 0.62418. Després de trobar l'etiqueta *ground truth* pels seus gens, aplicar el GPR i eliminar nodes de flux zero, arribem a un graf amb 44 nodes essencials (1), 135 no essencials (0) i 157 amb valor desconegut (-1). Es tracta doncs d'un conjunt de dades desbalancejat a favor de la classe negativa, al contrari què ens passava amb l'*Escherichia*. Quins resultats podríem esperar doncs? El que veiem és, una altra vegada, una **forta capacitat predictiva** del model, que es fa palesa en les diferents matrius de confusió (veure Figura 5.7).

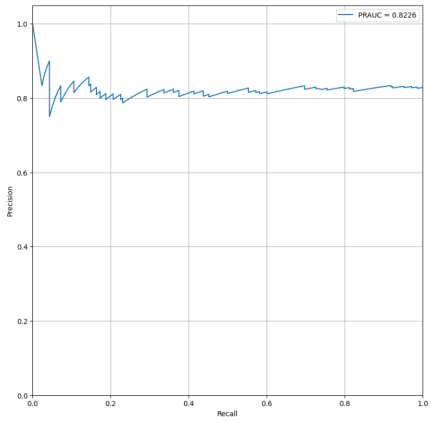
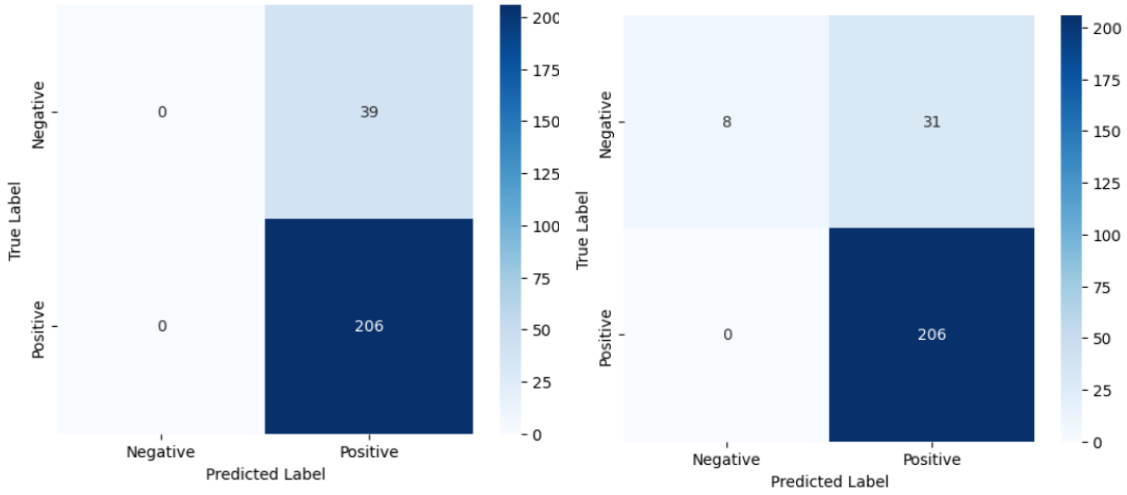
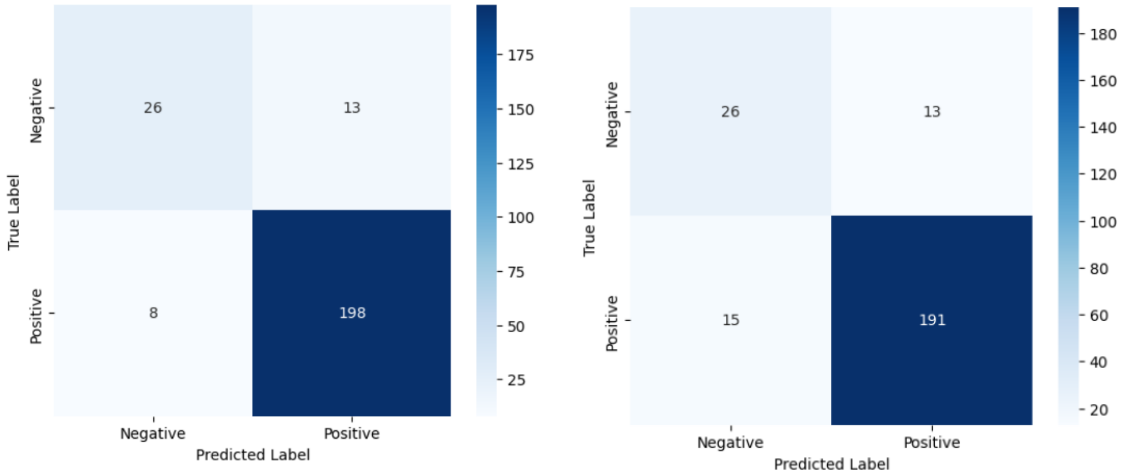


Figura 5.5: L'àrea PRAUC davall de la corba és del 0.8226.



(a) 0.1

(b) 0.25

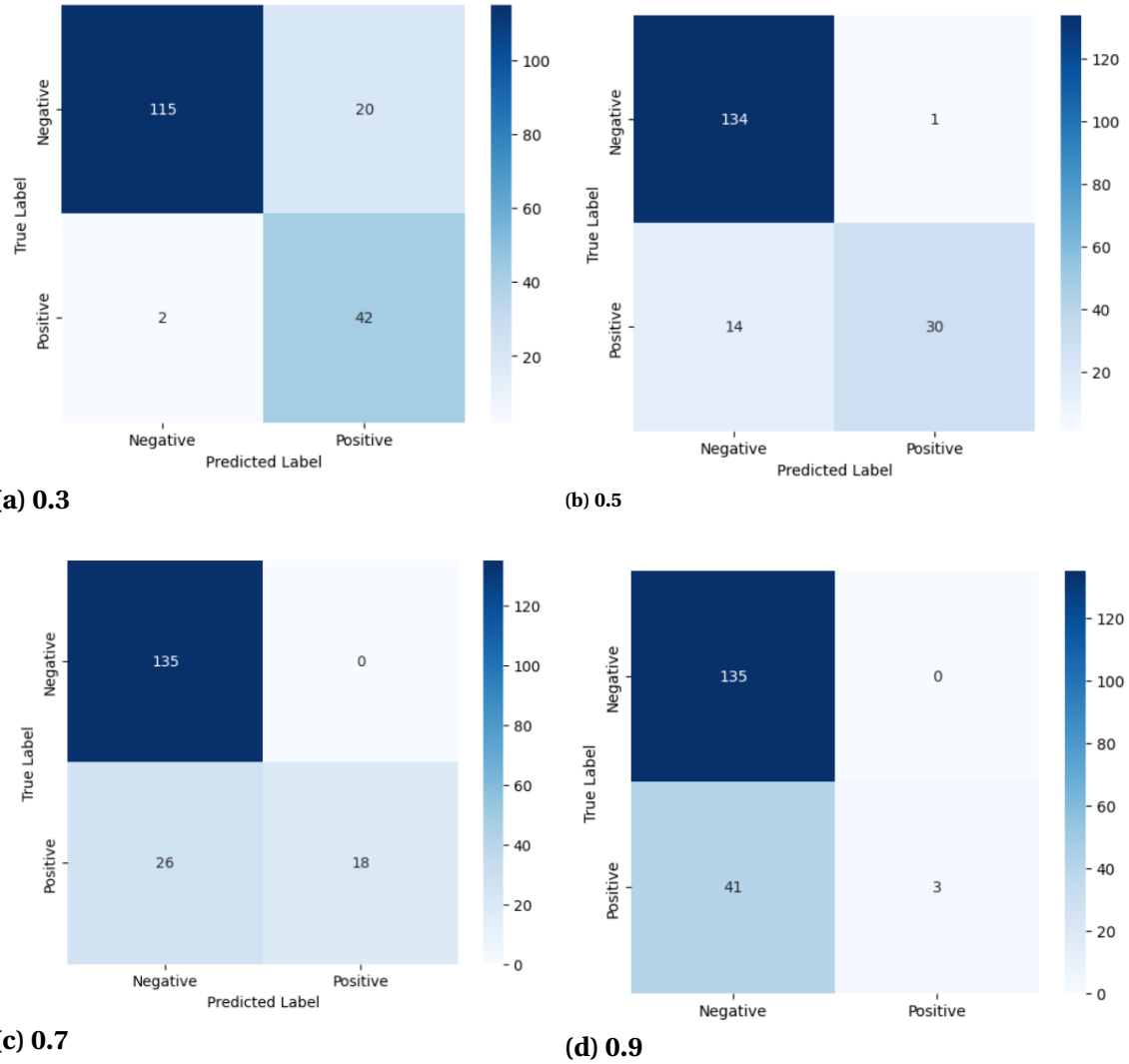


(c) 0.5

(d) 0.75

Figura 5.6: Matrius de confusió per a l'*Escherichia Coli* amb llindars del 0.1,0.25,0.5 i 0.75, respectivament.

Segueix havent-hi una forta dependència del llindar escollit pel que fa a la classificació, que en aquest cas, amaga quelcom d'interessant. Si bé, el llindar que dóna una major *accuracy* és el del 0.5 (així com amb l'*Escherichia*), aquest penalitza la taxa de falsos negatius. És a dir, amb el llindar del 0.5 podem identificar gairebé tots els gens no essencials però a costa de falsos negatius. Aquest és un resultat invers a l'obtingut en l'anterior espècie, amb un llindar del 0.5, cada model és capaç de capturar bé gairebé **totes les instàncies de la classe majoritària** en cada cas. I en el cas del *Bacillus*, el llindar del 0.3 dóna resultats prou acceptables pel que fa a la predicció dels gens essencials. Podríem dir que és el llindar que captura millor la identificació d'essencials.



(a) 0.3

(b) 0.5

(c) 0.7

(d) 0.9

Figura 5.7: Matrius de confusió per al *Bacillus Subtilis* amb llindars del 0.3,0.5,0.7 i 0.9, respectivament

El *Mycobacterium tuberculosis* és l'eubacteri responsable de la major part dels casos de tuberculosi en humans. També podem trobar la seva reconstrucció genòmica a BiGG models. Hem construït el seu MFG ben igual que amb les espècies anteriors i ens ha donat 205 gens essencials (1), 56 gens no essencials (0) i 215 sense etiquetar (-1). Què pot passar doncs, si

ajuntem les dades de les tres espècies en una sola matriu de la forma $G = \begin{pmatrix} G_e & 0 & 0 \\ 0 & G_b & 0 \\ 0 & 0 & G_m \end{pmatrix}$, és a dir, en forma d'un graf format amb tres (o més) components connexes? El resultat serà un conjunt de dades amb gens de diferents espècies i més equilibrat respecte dels anteriors individuals. Implementant la funció *stack_graphs_diagonally*(G_1, G_2,G_3, *'label'*) s'arriba a un MFG amb tres components connexes i 457 nodes essencials (1) i 228 nodes no essencials. Aplicant el mateix algorisme d'entrenament que en els casos anteriors (*k-fold* i *GridSearch* inclosos), arribem als següents resultats (vegi's Figura 5.8).

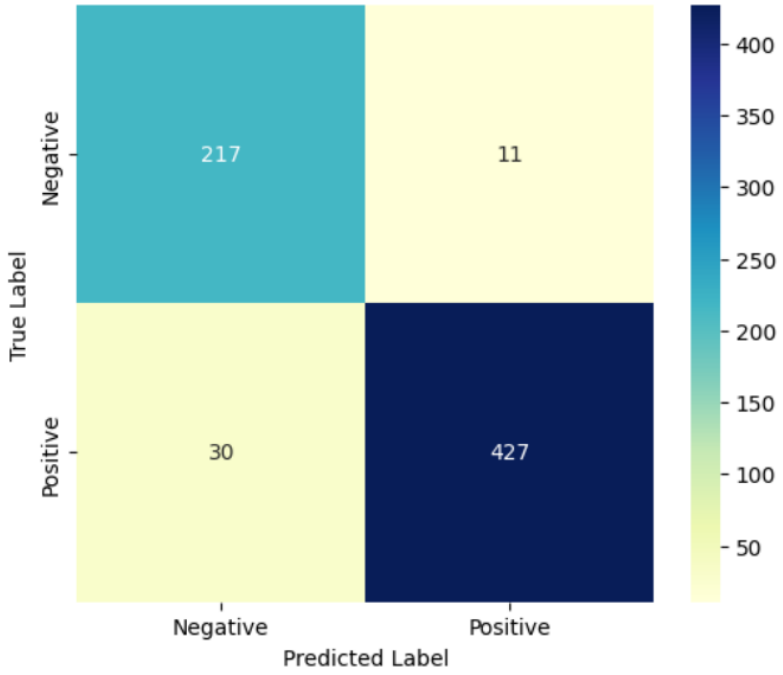


Figura 5.8: Model híbrid generalitzat

L'*accuracy* global és del 0.94 i les dues classes estan força ben separades, concloem per tant, que el model és capaç de capturar patrons i és un bon separador. A més, l'error en aquest cas està bastant repartit entre les dues classes. Gràficament, això es pot veure com una forta simili·litud entre els colors de l'anti-diagonal de la matriu de confusió.

CAPÍTOL 6

CONCLUSIONS

En aquest treball hem obtingut un seguit de resultats que ens permeten validar els Mass Flow Graphs com a eines capaces de capturar patrons en la distribució de fluxos metabòlics. La bona capacitat predictiva de les diferents Graph Neural Networks indica que les característiques basades en la distribució de flux són biològicament rellevants i que els gens essencials tendeixen a ocupar posicions topològiques similars.

En primer lloc, hem resolt, sota condicions de creixement aeròbic en glucosa i oxigen, el problema FBA per a tres espècies de bactèries diferents, l'*Escherichia Coli*, el *Bacillus Subtillis* i el *Mycobacterium tuberculosis*. Hem trobat els respectius Mass Flow Graphs, i hem entrenat tres models de GNN amb les dades obtingudes. Tot i que ho fa lleugerament, el model que obté els millors valors d'*accuracy* és l'entrenat únicament amb l'*Escherichia*. Això, possiblement es deu a que, és l'espècie més estudiada, que compta amb un model metabòlic més precís.

L'èxit en la capacitat predictiva del model entrenat amb dades de les tres espècies distintes, demostra que el plantejament de la màquina *FlowGAT* [1] és generalitzable a diverses espècies de bacteris. És a dir, ha après un patró en organismes filogenèticament diferents. Les modificacions fetes al model citat anteriorment inclouen la normalització a variància 1 de les característiques dels nodes i la simplificació del GPR. Aquests són dos fets diferencials que cal esmentar i és possible que hagin contribuït a capturar millor els patrons en les dades.

Si bé és cert que el rendiment dels tres models és menor pel que fa als gens no essencials, aquest és un fet documentat. A més, tot i que els tres organismes estudiats són diversos, tots són bacteris; la generalització a eucariotes o arqueus resta per ser investigada. En conclusió, la capacitat del nostre model per generalitzar entre espècies bacterianes diverses representa un avenç significatiu en la predicció computacional d'essencialitat gènica. Aquests resultats suggereixen que els principis subjacents de l'essencialitat metabòlica són més universals del que es podria esperar, obrint noves vies per al desenvolupament de predictors a escala bacteriana.

APÈNDIX A

Enllaç al repositori del Treball (GitHub):

`https://github.com/MiquelAngelLlauger-Cdm/TFG2.git`

Enllaç a la base de dades dels **models metabòlics** (BiGG Models):

Escherichia Coli K12:

`http://bigg.ucsd.edu/models/iML1515`

Bacillus Subtillis str.168:

`http://bigg.ucsd.edu/models/iYO844`

Mycobacterium tuberculosis

`http://bigg.ucsd.edu/models/iEK1008`

Enllaç a la base de dades de **gens essencials**:

Escherichia Coli K12:

`https://v3.ogee.info/?#/browse/83333/Escherichia%20coli%20K12`

Bacillus Subtilis str.168:

`https://v3.ogee.info/?#/browse/224308/Bacillus%20subtilis%20subsp.%20subtilis%20str.%20168`

Mycobacterium tuberculosis

`https://v3.ogee.info/#/browse/83332/Mycobacterium%20tuberculosis%20H37Rv`

BIBLIOGRAFIA

[1] R. Hasibi, T. Michoel, and D. A. Oyarzún, "Integration of graph neural networks and genome-scale metabolic models for predicting gene essentiality," *npj Systems Biology and Applications*, vol. 10, no. 1, p. 24, March 2024. [Online]. Available: https://doi.org/10.1038/s41540-024-00348-2 1, 1, 5.1, 5.2, 6

[2] S. A. Becker, A. M. Feist, M. L. Mo, G. Hannum, B. Palsson, and M. J. Herrgård, "Quantitative prediction of cellular metabolism with constraint-based models: the cobra toolbox," *Nature Protocols*, vol. 2, no. 3, pp. 727–738, 2007. 1, 3.1, 1

[3] S. Gurumayum, P. Jiang, X. Hao, T. L. Campos, N. D. Young, P. K. Korhonen, R. B. Gasser, P. Bork, X.-M. Zhao, L. jie He, and W.-H. Chen, "OGEE v3: Online GEne Essentiality database with increased coverage of organisms and human cell lines," *Nucleic Acids Research*, vol. 49, no. D1, pp. D998–D1003, January 2021. [Online]. Available: https://doi.org/10.1093/nar/gkaa884 1

[4] J. C. P. Mayol, *Matemàtica Discreta*, 2021. 2.1

[5] J. A. Bondy and U. S. R. Murty, *Graph Theory*, ser. Graduate Texts in Mathematics. London: Springer, 2008, vol. 244. 2.1

[6] S. Klamt and J. Stelling, "Metabolic networks," in *Analysis of Biological Networks*, B. H. Junker and F. Schreiber, Eds. Hoboken, NJ: Wiley-Interscience, 2008, ch. 7, pp. 163–184. 2.2, 3.1

[7] A. L. Ferre, "Tècniques de comparació de grafs: Aplicació a les xarxes metabòliques," 2024. 2.2

[8] M. Feinberg, "Chemical reaction network structure and the stability of complex isothermal reactors—i. the deficiency zero and deficiency one theorems," *Chemical Engineering Science*, vol. 42, no. 10, pp. 2229–2268, 1987. 2.2

[9] E. K. P. Chong and S. H. Żak, *An Introduction to Optimization*, 4th ed. John Wiley & Sons, 2013. 2.3, 4.1.1, 4.1.2, 4.1.2

[10] J. M. Monk, A. Koza, M. A. Campodonico, D. Machado, J. M. Seoane, and B. Palsson, "iml1515, a knowledgebase that computes escherichia coli traits," *Nature Biotechnology*, vol. 35, no. 10, pp. 904–908, 2017. [Online]. Available: https://www.nature.com/articles/nbt.3956 3, 5, 5.1, 5.1, 3

[11] M. Beguerisse-Díaz, G. Bosque, D. Oyarzún, J. Picó, and M. Barahona, "Flux-dependent graphs for metabolic networks," *npj Systems Biology and Applications*, vol. 4, no. 1, p. 32, 2018. [Online]. Available: https://www.nature.com/articles/s41540-018-0067-y 3

[12] J. D. Orth, I. Thiele, and B. Ø. Palsson, "What is flux balance analysis?" *Nature Biotechnology*, vol. 28, no. 3, pp. 245–248, 2010. 3.1, 3.1

[13] R. Heinrich and S. Schuster, *The Regulation of Cellular Systems*. Springer US, 1996. 3.1

[14] R. Penrose, "A generalized inverse for matrices," *Proceedings of the Cambridge Philosophical Society*, vol. 51, pp. 406–413, 1955, communicated by J. A. Todd. Received 26 July 1954. 3.2.1

[15] D. Graupe, *Principles of Artificial Neural Networks*, 2nd ed., ser. Advanced Series on Circuits and Systems. Singapore: World Scientific, 2007, vol. 6. 4.1

[16] P. L. Treceño, "Xarxes neuronals per a la generació i representació d'ones," Barcelona, gener 2017, realitzat al Departament de Matemàtiques i Informàtica. 4.1

[17] I. Basheer and M. Hajmeer, "Artificial neural networks: Fundamentals, computing, design, and application," *Journal of Microbiological Methods*, vol. 43, no. 1, pp. 3–31, 2000. 4.1

[18] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943. 4.1

[19] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958. 4.1, 4.1.1

[20] M. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015, http://neuralnetworksanddeeplearning.com/. 4.1

[21] W. Krauth and M. Mézard, "Learning algorithms with optimal stability in neural networks," *Journal of Physics A: Mathematical and General*, vol. 20, no. 11, pp. L745–L752, 1987. 19

[22] B. Sanchez-Lengeling, E. Reif, A. Pearce, and A. Wiltschko, "A gentle introduction to graph neural networks," *Distill*, vol. 6, no. 8, 2021. [Online]. Available: https://distill.pub/2021/gnn-intro/ 4.2, 4.2.1

[23] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 2. IEEE, 2005, pp. 729–734. 4.2.1

[24] V. D. Massa, G. Monfardini, L. Sarti, F. Scarselli, M. Maggini, and M. Gori, "A comparison between recursive neural networks and graph neural networks," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 2. IEEE, 2005, pp. 1753–1758. 4.2.1

[25] W. L. Hamilton, *Graph Representation Learning*, ser. Synthesis Lectures on Artificial Intelligence and Machine Learning. San Rafael, CA: Morgan & Claypool Publishers, 2020, vol. 14, no. 3. 4.2.2

[26] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations (ICLR)*, 2018. [Online]. Available: https://arxiv.org/abs/1710.10903 4.2.3, 4.2.3

[27] L. J. V. Jordi, "Introduction to graph neural networks," 2025, predoctoral researcher presentation, February 21, 2025. 4.2.3

[28] T. Escherich, *Die Darmbakterien des Neugeborenen und Säuglings: Pathogenität und Ätiologie der Darmbakterien*. München: M. Rieger'sche Universitäts-Buchhandlung, 1885. 5

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011. 5.3.1