

Intel·ligència artificial aplicada a la construcció

Bloc 2: Introducció a les xarxes neuronals artificials

Toni Guillamon

Dept. Matemàtiques - UPC

Setembre 2021

Preàmbul

Aquests apunts són una introducció a la intel·ligència artificial adreçada a estudiants del Grau d'Arquitectura Tècnica i Edificació de l'EPSEB (UPC). L'objectiu és:

- Mostrar exemples d'aplicació de la intel·ligència artificial a la construcció.
- Mostrar els fonaments matemàtics d'alguns mètodes d'IA (centrats bàsicament en les xarxes neuronals artificials) a un nivell molt elemental.
- Obrir la porta a programari que permet l'aprenentatge automàtic de xarxes i la predicció.
- Fer que l'estudiant sigui capaç d'aplicar-ho a bases de dades existents i analitzar-ne els resultats.

Evidentment, doncs, és només una *pinzellada* per estimular l'entrada a la IA. No hi trobareu avenços nous en IA (el mèrit del contingut és de les diverses fonts que se citen); la nostra feina ha consistit en recollir la informació, ordenar-la, dissenyar les pràctiques/exercicis i presentar-ho tot plegat al nivell adequat.

Guió

- 1 La intel·ligència artificial en la construcció
- 2 Les xarxes neuronals dins la intel·ligència artificial
- 3 Introducció a les xarxes neuronals artificials a través del perceptró
- 4 Inspiració biològica
- 5 Xarxes neuronals *feedforward*
- 6 Entrenament, validació i prova d'una ANN
 - Fase d'entrenament; algorisme de retropropagació
 - Fase de validació; sub/sobre-aprenentatge
 - Fase de prova
- 7 Exemples en Matlab
 - Regressió
 - Classificació
- 8 Altres mètodes d'aprenentatge automàtic
- 9 Referències

La intel·ligència artificial en la construcció

IA en construcció

Aplicacions: desenvolupament d'edificis eficients i bonics



Vídeo: *Building the architectural future with new technologies*, <https://youtu.be/35DjNZFCCI0>. Nous materials, seguiment 3D del seu comportament, avantatges mediambientals.

- Els *enginyers de dades* analitzen **dades acumulades** sobre com es creen nous edificis i sobre **plans de construcció** (factors que afecten la qualitat general, costos, ...) per formar models d'aprenentatge automàtic que puguin detectar punts febles dels edificis futurs.
- Creació **de nous materials**.



Diagrids en lloc d'estructures d'acer per reduir els costos de desenvolupament, protegir l'edifici de càrregues sísmiques i construir de manera sostenible.

- **Col·locació òptima** plaques solars i aerogeneradors.

IA en construcció

Aplicacions: seguretat i riscos laborals

- Reducció de la probabilitat d'**accidents** (humans i relacionats amb l'equipament). La indústria de la construcció és una de les més traumàtiques.
- Hi ha llibreries de codi obert analitzen **imatges i vídeos de les càmeres de seguretat** en temps real, que poden detectar comportaments anormals i alertar l'empresa constructora.
- Models d'aprenentatge automàtic per detectar anomalies a les fotos i evitar despeses addicionals. Bases de dades d'accidents.



Vídeo: *Optimizando procesos en la construcción con Machine Learning*,

<https://www.youtube.com/watch?v=VvbTcPx5QP4>.

Ús d'imatges per millorar seguretat i producció.
Sostenibilitat (e.g., aprofitament del formigó).

IA en construcció

Gestió d'usos

- Com augmentar l'ús reduint l'espai? Controls d'ocupació i moviments mitjançant sensors (*embedded deep learning*), reassignació d'espais, protocols de neteja (e.g., ús de WCs),...



Extretes de <https://www.youtube.com/watch?v=TyogAAPI2oA> (*PointGrab's CogniPoint*).

- Gestió de calefacció, ventilació i aire condicionat (**HVAC**). Aplicacions a **cases intel·ligents** o determinades construccions (e.g., centres de dades, hotels) que, per poder ser competitives, depenen de sistemes HVAC complexos amb alta **eficiència energètica**.

IA en construcció

Aplicacions diverses I

- Orquestració de processos de construcció: planificació de tasques en empreses de certa dimensió. Per exemple, **calendari intel·ligent** (càrrega de treball, seguiment, evitar retards, ...)
- Impressió 3D.



Construcció d'un pont a Amsterdam, *Joris Laarman lab*,
vegeu *TEDtalk* de M. Conti,
<https://www.youtube.com/watch?v=aR5N2Jl8k14>
(10'10-10'41").

- Robotització de processos

IA en construcció

Aplicacions diverses II

- Optimització del reciclatge de materials.
- Digitalització de **plans industrials**. Extracció de dades d'elements complexos (disseny de canonades, esquemes de fabricació,...).
Classificació de documents.
- Evitar **sobrecostos**. Basat en diversos factors (complexitat de la feina, mida del projecte, materials usats o tipus de contracte), la IA pot estimar els costos aproximats comparant amb projectes anteriors.
- Processament massiu de dades.

IA en construcció

Problemes potencials... potencialitats problemàtiques

- Gestió de dades i emmagatzematge. **Cal molta informació** (dades) perquè l'aprenentatge automàtic sigui útil. Cal crear una **plataforma unificada** per emmagatzemar, accedir i manipular les dades. [**esforç col·laboratiu**]
- La IA requereix **ajuda/aprovació humana**, especialment a la indústria de la construcció (és difícil controlar tots els factors externs *a priori*). **Interacció persona-robot**, com en el projecte *Bishop*, vegeu *TEDtalk* de M. Conti (**creativitat augmentada**),
<https://www.youtube.com/watch?v=aR5N2JI8k14> (8'45"-9'20"). .
- Calen **interlocutors dels enginyers de dades** en cada disciplina. «*Neix un nou tipus de treballador de la construcció, un científic de dades, que empra la IA per a nombroses tasques: predir la probabilitat de lesions personals, planificar accions, etc*»

<https://www.autodeskjournal.com/inteligencia-artificial-aplicada-a-la-construccion/>

IA en construcció I

Un exemple paradigmàtic: tractament de fusta

The Living va dur a terme un projecte per crear un edifici sostenible utilitzant fusta recuperada d'estructures de bastides i ampliar els usos de la fusta com a material de construcció; vegeu vídeo *Embodied Computation Lab.*



- Identificació humana de nusos en fusta. Un algoritme d'aprenentatge va analitzar les dades per a poder detectar i identificar nusos en imatges futures.
- Desenvolupament d'una màquina especial per netejar amb sorra per tal d'exposar els microcontorns dels nusos.
- Ordenació de 900 taulons recuperats, segons to de color i nusos.
- Anàlisi solar de la façana.

IA en construcció II

Un exemple paradigmàtic: tractament de fusta

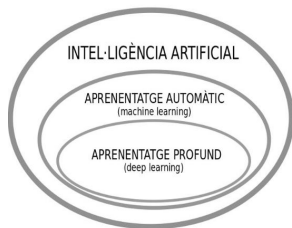
- Disposició de taulons per tal de fer **coincidir l'anàlisi solar amb l'aïllament tèrmic específic** basat en els microcontorns (addicionalment: sensors prop de la fusta per recopilar dades sobre vida útil de l'edifici).
- L'estètica gira al voltant dels nusos, amb ombres canviants durant el dia i el "desgast" al llarg del temps crea una superfície dinàmica.

Ref: <https://www.wtin.com/article/2018/july/090718/artificial-intelligence-through-the-lens-of-biomimicry/>

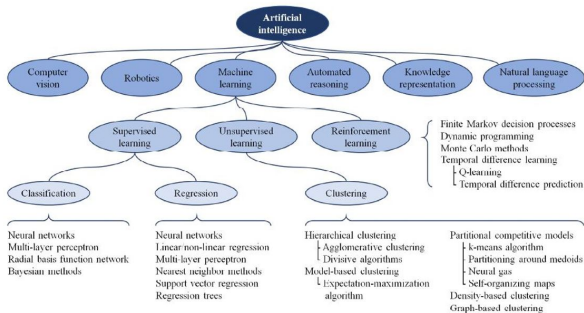
Però... com aprenen els ordinadors?

Bloc 2: Una incursió tècnica a la IA a través de les xarxes neuronals artificials.

Ubicació dels xarxes neuronals (ANN) dins de la IA

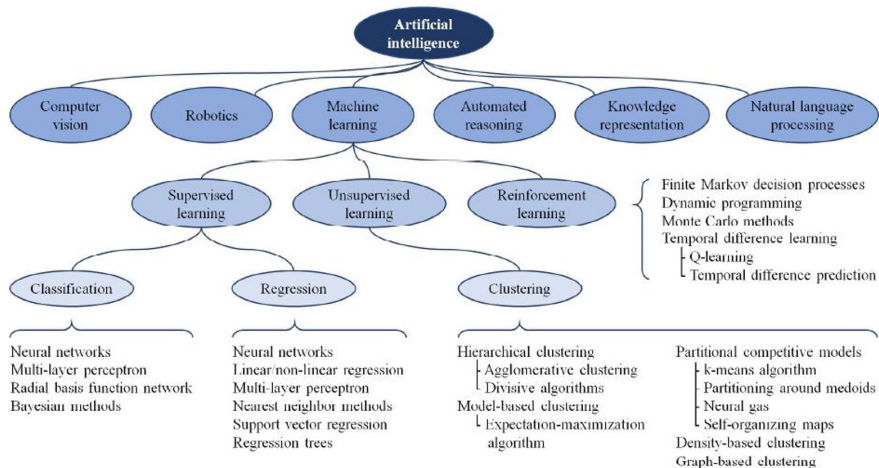


Una visió simplificada

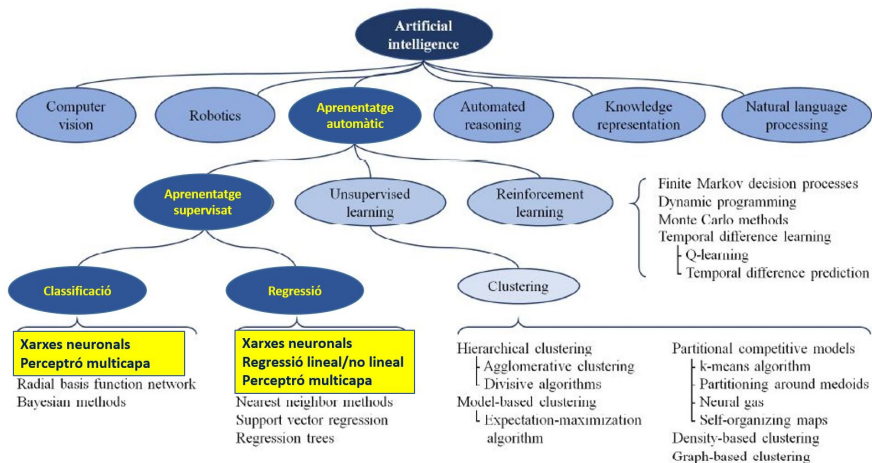


Una visió completa
[Luckey et al, 2020]

Ubicació dels xarxes neuronals (ANN) dins de la IA

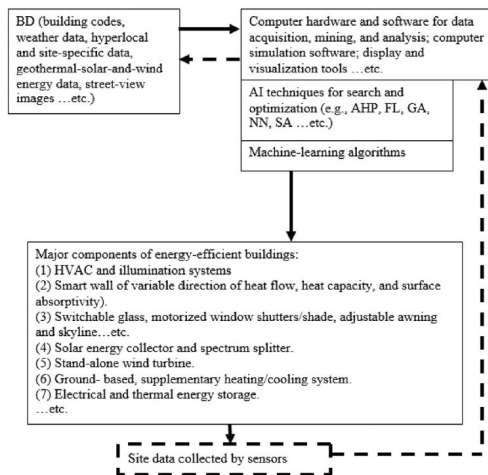


Ubicació dels xarxes neuronals (ANN) dins de la IA



Aprentatge automàtic (ML) i bases de dades

Exemple: eficiència energètica en edificis



[Mahmood et al, 2019]: **lectura MOLT recomanada**

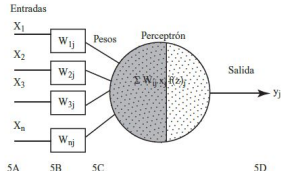
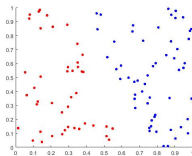
Introducció a les xarxes neuronals artificials a través del perceptró

Estructura d'una xarxa neuronal: el perceptró

- El **perceptró** és l'ANN més simple que hi ha. És un discriminador binari lineal.
- Tenim un conjunt de dades $\{(x_{1k}, \dots, x_{nk}, y_k)\}_{k=1}^M$, on $x_1, \dots, x_n$ són les **entrades** (*input*), i ξ_k és el **valor de classificació** (*output*).
- Per exemple, punts en un pla que pertanyen a dos subconjunts, que representem per $y \in \{-1, 1\}$.
- Volem trobar $\{w_0, w_1, \dots, w_n\}$ tals que

$$z = w_0 + \sum_{j=1}^n w_j x_j$$

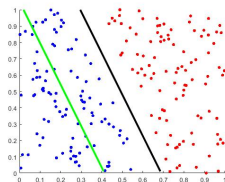
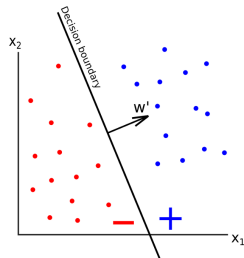
sigui un bon **classificador** de les dades.



Perceptró. Entrades (5A), pesos (5B), sumatori pesos per entrades (5C) i la sortida (5D). [Ponce, 2011]

Perceptró: exemple bàsic al pla

- **Objectiu.** Trobar $\{w_0, w_1, w_2\}$ tals que $z = w_0 + w_1 x_1 + w_2 x_2$ classifiqui.
- **Algorisme d'entrenament.**
 - Prenem $w_0 := (w_0, w_1, w_2)$ aleatori i avaluem
 $z_1 = w_0 + w_1 x_{1,1} + w_2 x_{2,1} = w_0 \cdot x_1$, on $x_k = (1, x_{1,k}, x_{2,k})$.
 - Si $z_1 y_1 < 0$, aleshores $w_1 = w_0 + \eta y_1 x_1$.
 - Anem repetint per a cada x_k , per a $k = 2, \dots, M$, tot i que es pot tenir un resultat satisfactori amb antelació.



$$w' = (w_1, w_2).$$

- La funció $z(x) = w_M \cdot x$ serà el classificador que cercàvem i ens servirà per a fer **prediccions**.

Perceptró: exemple bàsic al pla

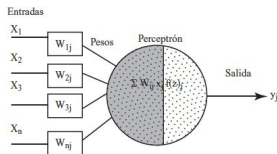
Per què funciona bé?

<https://towardsdatascience.com/perceptron-explanation-implementation-and-a-visual-example-3c8e76b4e2d1>

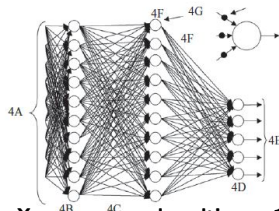
Estructura d'una xarxa neuronal

Perceptró, neurona i xarxes multicapa

- El **perceptró** és l'ANN més simple que hi ha. És un discriminador binari lineal i es pot entrenar.
- Veurem que una **neurona** serà un element (no necessàriament binari) similar al perceptró.
- Veurem que les ANN augmenten en complexitat a l'afegir més **capes** i més **nodes** a cada capa.



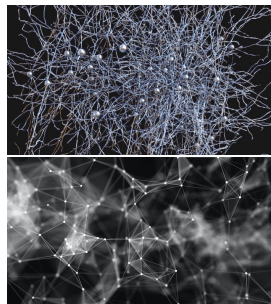
Perceptró. Entrades (5A), pesos aleatoris (5B), sumatori de la multiplicació dels pesos per les seves entrades resp. (5C) i la sortida, que és el càlcul de tots els pesos i les seves entrades (5D).



Xarxa neuronal multicapa 10-10-5. Entrada (4A). Capa d'entrades (4B), capa oculta (4C) i capa de sortides (4D). Sortides (4E). Els pesos (4F) estan representats per un punt negre (4G).

Idea general sobre les ANN

- Les **xarxes neuronals artificials** (ANN, acrònim en anglès) s'inspiren en el funcionament de les neurones i les seves sinapsis.
- Les ANN **aprenen** de problemes ja resolts per **entrenar/millorar** les seves connexions o **pesos** per tal de construir un sistema que realitzi **classificacions** i prengui **decisions**, **aproximi** funcions. . .
- L'**entrenament** es basa en l'algorisme de **retropropagació de l'error** (*back-propagation error*) (gradient descent) o altres mètodes d'optimització (per exemple, Levenberg-Marquardt). Els pesos es van actualitzant amb l'ús d'entrades i sortides desitjades.



Avantatges i inconvenients de les ANN

● Avantatges:

- Poden sintetitzar algorismes a través d'un **procés d'aprenentatge**.
- Per aplicar-les, no calen coneixements matemàtics avançats; bàsicament, cal conèixer les dades.
- Resolen **problemes no lineals**.
- Són **robustes**: poden fallar alguns elements de processament però la xarxa continua treballant.

● Inconvenients:

- Cada ANN és **específica d'un problema**. L'**entrenament és llarg** (determinació de l'estructura òptima de la xarxa) i **costós computacionalment**.
- Es necessiten **moltes dades**.

Una mica d'història de les ANN

W. McCulloch i W. Pitts proposen un model binari de neurona (1943)

B. Widrow i M. E. Hoof implementen el model de McCulloch i Pitts en una màquina que anomenen Adaline (Adaptive Linear Element)

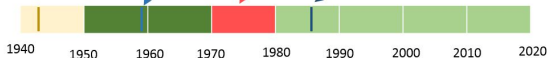
Frank Rosenblatt (1959): el **perceptró**.

Tasques de categorització, podia reconèixer les lletres de l'alfabet.

Dècada dels 70: Minsky i Papert demostren que els perceptrons no poden fer tasques simples (e.g., XOR)

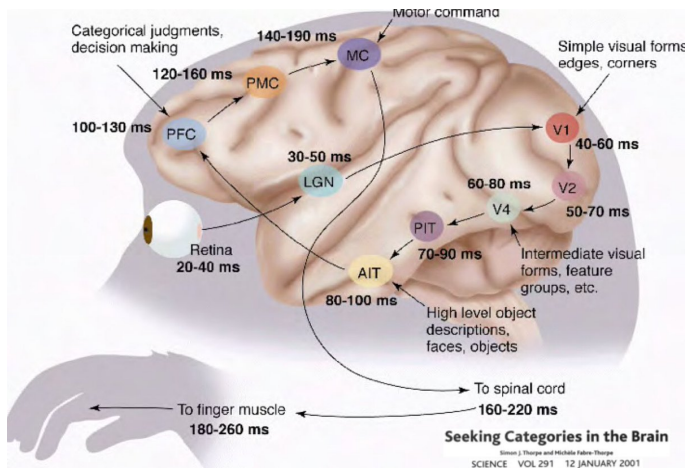
David Rumelhart i James McClelland (1986) publiquen *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, un clàssic en ANN, que va impulsar de nou el seu estudi (pros i contres).

Els avenços no han parat en les darreres dècades, on els termes *machine learning* i, en particular, *deep learning* han anat guanyant protagonisme per les múltiples aplicacions que s'han trobat (cercadors d'internet, xarxes socials, ciutats intel·ligents, aplicacions en assistència mèdica,...).



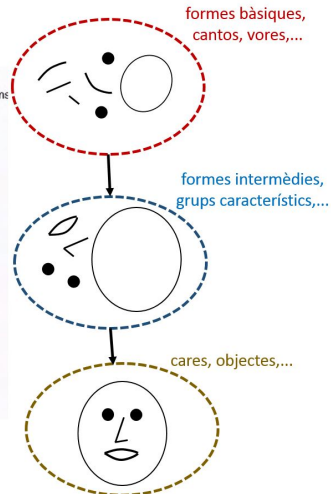
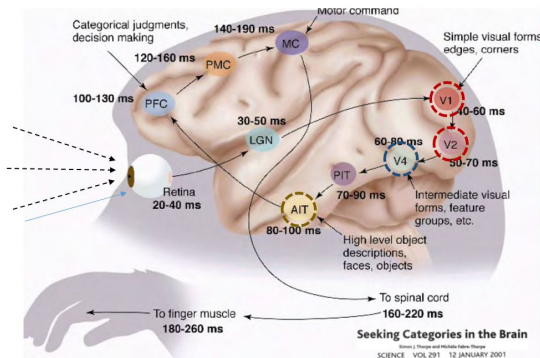
Inspiració biològica

Inspiració biològica: el còrtex visual



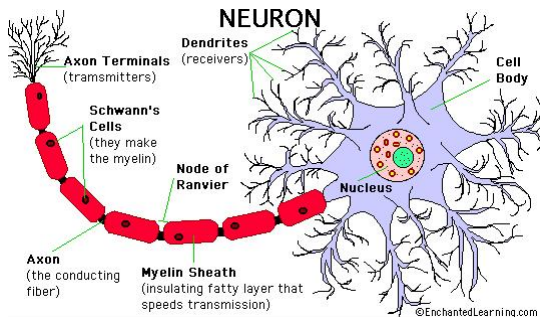
[Simon Thorpe, Science 2001]

Inspiració biològica: el còrtex visual



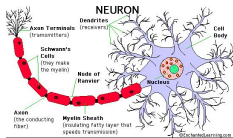
Inspiració biològica: neurones i sinapsis

- Parts d'una neurona: dendrites, soma, axó, terminal axònic.
- Tenim al voltant de 10^{12} neurones, que formen al voltant de 10^{15} sinapsis.

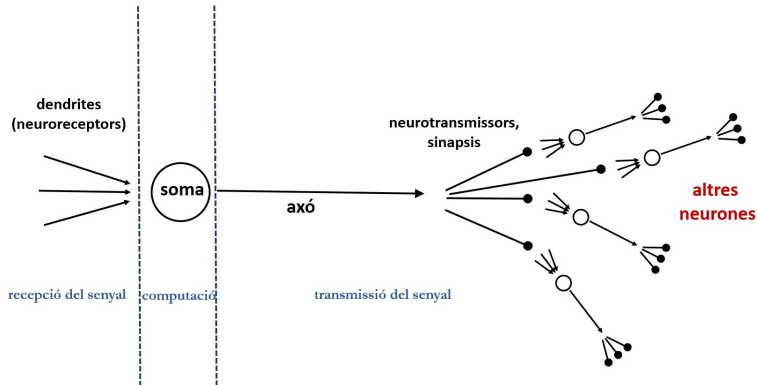


[Szu-Moon, 2018]

Inspiració biològica: neurones i sinapsis



- Potencial d'acció, **llindar de dispar**.
- Freqüència de dispar (*firing rate*), r .



Xarxes neuronals *feedforward*

Neurona artificial

Pesos de les connexions, biaix, funció d'activació

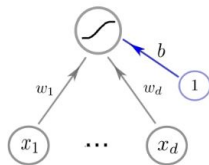
- **Preactivació** de la neurona (o activació de l'entrada):

$$a(x) = b + \sum_{i=1}^d w_i x_i = b + \mathbf{w}^T \mathbf{x}, \quad \mathbf{x} = (x_1, \dots, x_d), \quad \mathbf{w} = (w_1, \dots, w_d).$$

- **Activació** de la neurona (o activació de la sortida):

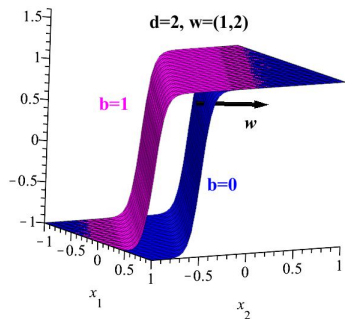
$$h(x) = g(a(x)) = g\left(b + \sum_{i=1}^d w_i x_i\right).$$

- w són els **pesos** de les connexions.
- b és el **biaix** de la neurona.
- g és la **funció d'activació**. Quan g és una funció escaló obtenim el perceptró.



Neurona artificial

Representació de l'activació



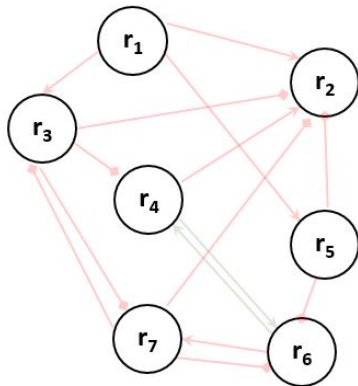
$$h(x) = g\left(b + \sum_{i=1}^d w_i x_i\right)$$

- $h(x) = g(b + w_1 x_1 + w_2 x_2)$ per a $d = 2$.
- g controla l'**altura**; per exemple, $g(x) = \arctan x$ o $g(x) = \frac{1 - e^{-x}}{1 + e^{-x}}$ prenen valors a $(-1, 1)$.
- b desplaça el “cingle” o “talús”.
- Els pesos w_i controlen l'**orientació del cingle**.
- El vector w és **perpendicular** a l'hiperplà que defineix el talús.

(Inspiració biològica)

El *firing rate* representa l'activitat de la xarxa

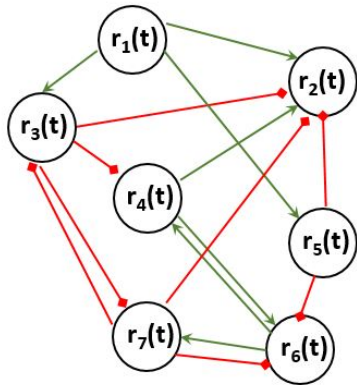
- Cada neurona té la seva freqüència natural, que s'altera per la interacció amb les altres.
- En una **xarxa artificial**:
 - L'**activació** de cada neurona juga el paper de *firing rate*.
 - Els **pesos** entre neurones reflecteixen les **sinapsis** (excitadores, inhibidores, força, ...).
 - Les **funcions d'activació** i els **biaixos** van associats al **llindar de dispar**.



(Inspiració biològica)

El *firing rate* representa l'activitat de la xarxa

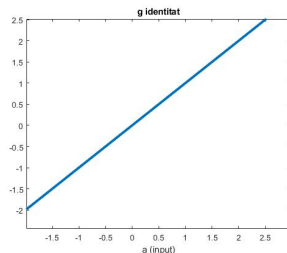
- Cada neurona té la seva freqüència natural, que s'altera per la interacció amb les altres.
- En una **xarxa artificial**:
 - L'**activació** de cada neurona juga el paper de *firing rate*.
 - Els **pesos** entre neurones reflecteixen les **sinapsis** (excitadores, inhibidores, força, ...).
 - Les **funcions d'activació** i els **biaixos** van associats al **llindar de dispar**.



Neurona artificial: funcions d'activació

La funció identitat

- $g(a) = a$:
 - No és gaire interessant: tot el que entra surt igual.
 - Sempre positiva.
 - No introdueix no linealitats (útils per a càlculs complicats).

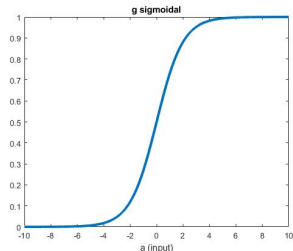


Neurona artificial: funcions d'activació

La funció sigmoïdal

- $g(a) = \frac{1}{1+\exp(-a)}$ (funció **sigmoïdal**):
 - Comprimeix tota la sortida en $(0, 1)$ (satura a $a = 0$ i $a = 1$).
 - Sempre **positiva** i **creixent**.
 - Fitada.
 - Si volem que l'interval sigui $(-1, 1)$ (**canvia de signe!**), obtenim

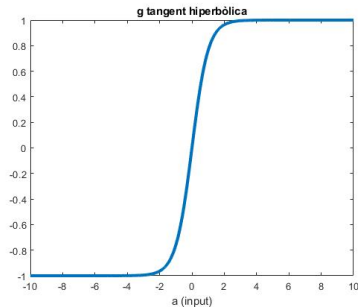
$$\tilde{g}(a) = 2g(a) - 1 = \frac{1 - \exp(-a)}{1 + \exp(-a)}.$$



Neurona artificial: funcions d'activació

La funció tangent hiperbòlica

- $g(a) = \tanh a$ (tangent hiperbòlica).
 - Comprimeix tota la sortida en $(-1, 1)$ (satura a $a = -1$ i $a = 1$).
 - Canvia de signe i és creixent.
 - Fitada.

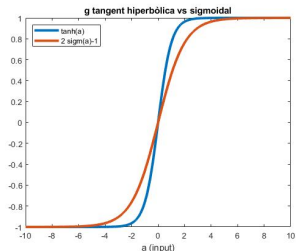


$$\tanh a = \frac{1 - \exp(-2a)}{1 + \exp(-2a)}.$$

Neurona artificial: funcions d'activació

Tangent hiperbòlica vs sigmoïdal

- $g(a) = \tanh a$ (**tangent hiperbòlica**).
 - Comprimeix tota la sortida en $(-1, 1)$ (satura a $a = -1$ i $a = 1$).
 - **Canvia de signe** i és **creixent**.
 - Fitada.

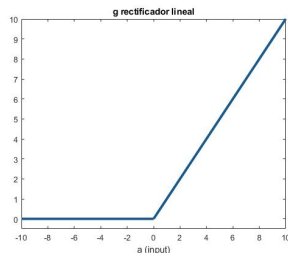


Comparativa tangent hiperbòlica versus sigmoïdal $(-1, 1)$.

Neurona artificial: funcions d'activació

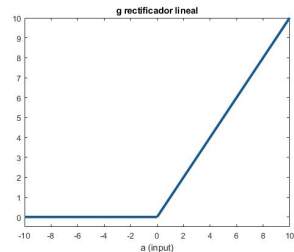
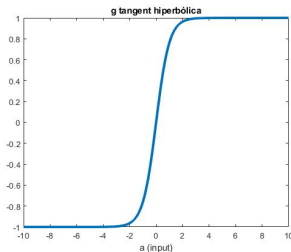
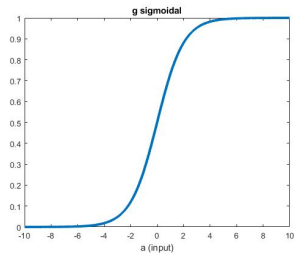
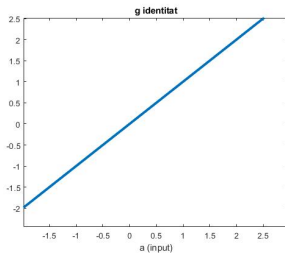
El rectificador lineal

- $g(a) = \max(0, a) = \begin{cases} 0 & \text{si } a < 0, \\ a & \text{si } a \geq 0. \end{cases} :$
 - Fitada només per sota $a = 0$, però no fitada superiorment.
 - Tendeix a provocar una activació esparsa de les neurones.



Neurona artificial: funcions d'activació

Resum



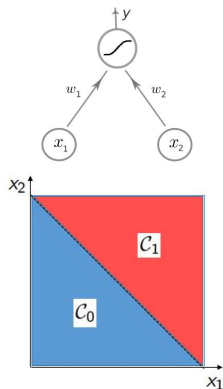
Neurona artificial

Capacitat d'una neurona (complexitat de càlcul)

- **Classificació binària**: volem distingir entre 2 categories, $y \in \{0, 1\}$. De fet, volem calcular $p(y = 1|x)$.
- El problema més senzill de **categorització**, **presa de decisions**,...
- Usarem la **funció sigmoïdal** perquè discrimina bé entre 0 i 1, però podríem usar també la tanh.

$$h(x) = g\left(\sum_{i=1}^d w_i x_i\right) > 0.5 \ (\leq 0.5) \Rightarrow y = 1 \ (y = 0).$$

- S'anomena **classificador de regressió logístic**.

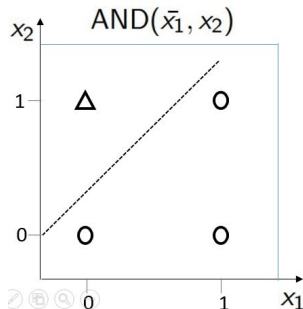
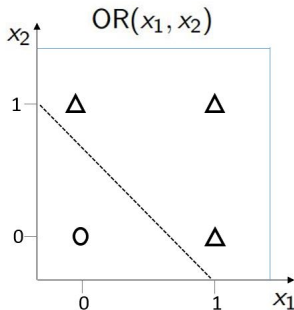


Si dues categories es poden separar per un hiperplà, aleshores es podrà fer la tasca de classificació.

Neurona artificial

Capacitat d'una neurona (complexitat de càlcul)

- Una neurona pot resoldre **problemes separables linealment**.

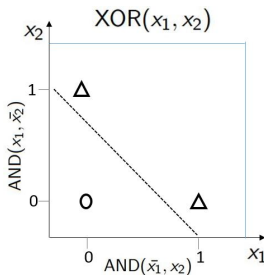
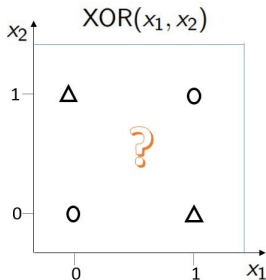


Exemple d'operadors lògics separables linealment. Per tant, es poden modelar amb una sola neurona artificial.

Neurona artificial

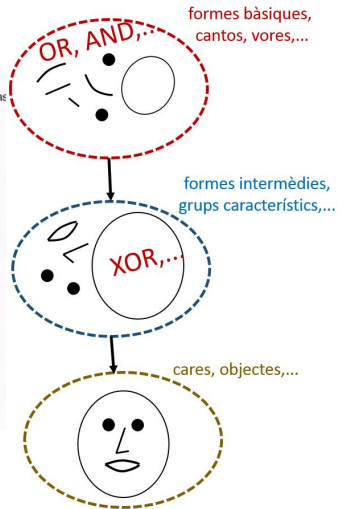
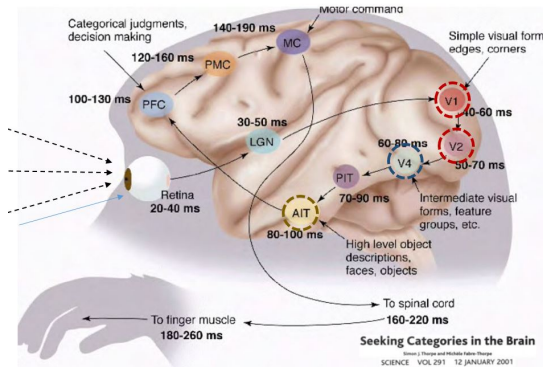
Capacitat d'una neurona (complexitat de càlcul)

- Però hi ha problemes senzills que ja **no es poden separar linealment**.



Exemple d'operador lògic NO separable linealment, però transformable **afegint una etapa**. Per tant, sembla que amb **més d'una neurona** podem aconseguir classificar tasques més complexes.

Inspiració biològica: el còrtex visual



Xarxes multicapa

Xarxes somes (*shallow*)

- Introduïm una **capa oculta**.
 - Preactivació de la capa oculta:

$$a(x) = b^{(1)} + w^{(1)}x$$

$$\left(a(x)_i = b_i^{(1)} + \sum_j w_{i,j}^{(1)} x_j \right).$$

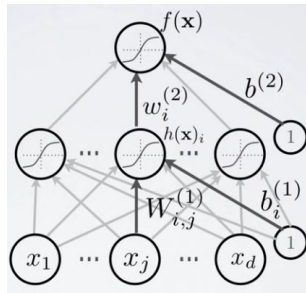
- Activació de la capa oculta:

$$h^{(1)}(x) = g_1(a(x)).$$

- Activació de la capa de sortida:

$$f(x) = g_2 \left(b^{(2)} + w^{(2)T} h^{(1)}(x) \right),$$

on g_2 és la **funció d'activació de sortida** (*output activation function*). Pot ser diferent de l'altra funció d'activació g_1 .



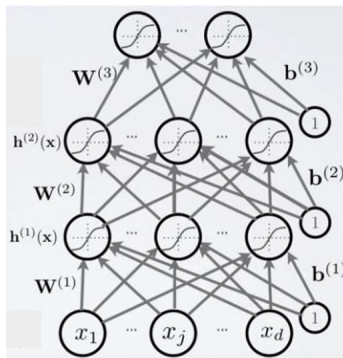
[Hugo Larochelle, 2013]

Xarxes multicapa: exemples en Matlab

- Xarxa soma (`BackPropagation.mlx`).
- Building dataset (App Neural Net Fitting).

Xarxes multicapa

Xarxes amb múltiples capes ocultes o profundes (*deep*).



[Hugo Larochelle, 2013]

No detallem les fórmules. Les capes es diuen ocultes perquè no coneixem el seu valor: només coneixem les dades d'entrada i els valors de sortida que voldríem obtenir.

Xarxes multicapa

Capacitat: aproximadors universals

Quines funcions $f(x) = g_2 \left(b^{(2)} + w^{(2)T} h^{(1)}(x) \right)$ podem modelar amb una xarxa neuronal i quines no?

Teorema

Tota funció continua de l'espai $[0, 1]^n$ es pot aproximar mitjançant una xarxa neuronal soma.

- Podem **confiar** en trobar sempre una aproximació.
- A la pràctica, **no** és un resultat **constructiu**, i el problema és saber quantes neurones calen donat un conjunt de dades.
- Per explicar com s'avalua si una xarxa neuronal s'ajusta bé a unes dades, ens cal entrar més a fons en:
 - Els **mètodes d'ajustament**, és a dir, de minimització de l'error.
 - Les principals **etapes** del procediment: l'**entrenament**, la **validació** i la **prova**.

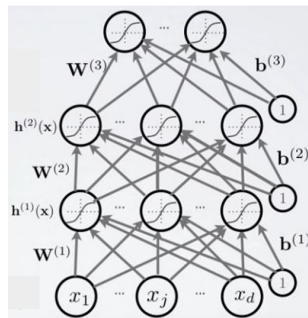
Entrenament, validació i prova d'una ANN

Elements i objectius d'una ANN (repàs)

- Disposem d'un **conjunt de dades** $\mathcal{D} = \{(x_j, t_j)\}_{j=1}^M$, on x és l'entrada o *input*, t és l'entrada o *output* i M és el nombre de dades disponible.
- Hem vist exemples d'*aprenentatge supervisat* aplicats a problemes de **classificació** (t només pren valors en un conjunt finit) i de **regressió** (t només pren valors en un continu), en xarxes *feedforward* (que no formen cicles).
- Construïm una xarxa de neurones artificials, amb una determinada **arquitectura** i volem estimar **els pesos** de les connexions que siguin més **compatibles amb la informació del conjunt de dades**.

Funcionament general d'una ANN

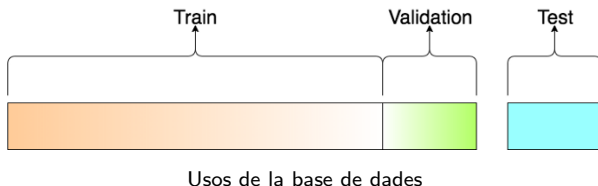
- Apliquem una entrada (x) començant amb uns pesos aleatoris ($\theta := (w, b)$), produïm una sortida ($f_{NN}(x; \theta)$), com hem fet a l'Exercici B2.1.
- Llavors, comparem $f_{NN}(x; \theta)$ amb la sortida esperada ($t(x)$, de *target*) i **quantifiquem l'error**.
- Retornem aquesta informació a la xarxa per tal de **reduir aquest error**.



Xarxes multicapa: aproximadors universals.

Entrenament, validació i prova

- Dividim el conjunt de dades $\mathcal{D}$ en tres subconjunts: d'**entrenament** (esquema iteratiu de millora dels pesos), de **validació** (ajustar els hiperparàmetres, és a dir, nombre de capes, nombre de nodes a cada capa, entre un conjunt d'arquitectures "candidates") i de **prova / test** (valoració de l'error comès).



Fase d'entrenament; algorisme de retropropagació

Repàs breu d'eines matemàtiques

Com anem de derivades en una i dues variables?

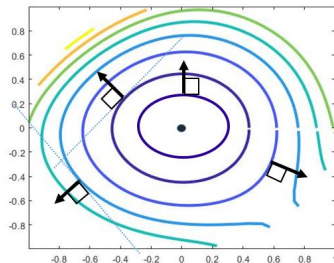
- **Regla de la cadena:** si $h(x) = f(g(x))$, aleshores,

$$\frac{dh(x)}{dx} = \frac{df(y)}{dy} \bigg|_{y=g(x)} \frac{dg(x)}{dx}; \quad h'(x) = f'(g(x)) \cdot g'(x).$$

Exemple: $h(x) = \sin(\ln x) \Rightarrow h'(x) = \cos(\ln x) \cdot \frac{1}{x}$.

- **Derivada parcial** d'una funció F de n variables. P.ex., si $n = 2$, $\frac{\partial F(x_0, y_0)}{\partial x}$ és la derivada de $F(x, y)$ respecte la variable x en un punt (x_0, y_0) .
- El **gradient** d'una funció de n variables i el seu significat.

$$\nabla F = \left(\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y} \right).$$



Corbes de nivell i alguns vectors
gradient.

Entrenament

Estratègia de reducció de l'error

- El conjunt de **dades d'entrenament** és el que s'utilitza per **ajustar els paràmetres** del model (determinació de pesos i biaixos).
- **Època**: Cada cop que passem les M dades per la xarxa i ajustem els pesos per tal de millorar l'**error**

$$E(\theta; \mathbf{x}) = \frac{1}{2} \sum_{i=1}^M ||\mathbf{t}(\mathbf{x}_i) - f(\mathbf{x}_i; \theta)||^2.$$

- Usarem el mètode de **retropropagació** o *backpropagation*, que es basa en la idea de **gradient descent**, vegeu també els apunts del Bloc 1 del curs.

$$\theta_{i+1} = \theta_i - \eta \nabla E(\theta_i; \mathbf{x}_i),$$

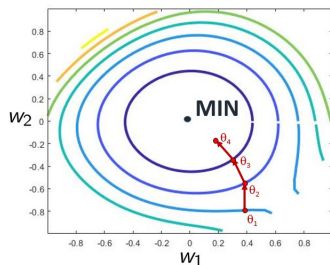
on η s'anomena **paràmetre d'aprenentatge**.

Entrenament

Estratègia de reducció de l'error

$$E(\theta; \mathbf{x}) = \frac{1}{2} \sum_{i=1}^M \|\mathbf{t}(\mathbf{x}_i) - f(\mathbf{x}_i; \theta)\|^2.$$

$$\theta_{i+1} = \theta_i - \eta \nabla E(\theta_i; \mathbf{x}_i) \quad (\text{gradient descent})$$



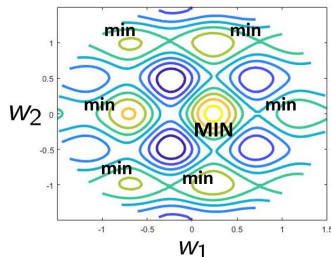
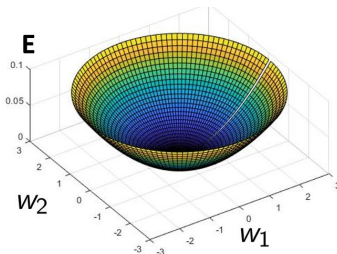
Corbes de nivell de la funció E . Idea del descens pel gradient, on $\theta = (w_1, w_2)$.

Entrenament

Estratègia de reducció de l'error

$$\theta_{i+1} = \theta_i - \eta \nabla E(\theta_i; x_i)$$

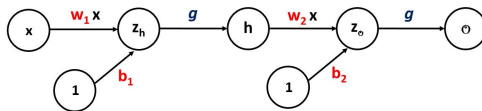
(gradient descent)



Gràfiques i corbes de nivell de la funció d'error (exemple simbòlic): escenari de mínims global i locals.

- Existeixen algorismes modificats que no estudiarem aquí (per exemple, de **Levenberg-Marquardt**).

Algorisme de *backpropagation*: idea general



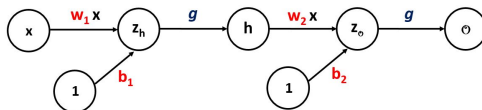
Un exemple de xarxa soma per il·lustrar *backpropagation*

- És ràpid, senzill i fàcil de programar.
- No requereix coneixements previs sobre la xarxa i, per tant, molt **adaptable**.
- No necessita conèixer les característiques de la funció que es vol aprendre.

Pàgina recomanada:

<https://www.anotsorandomwalk.com/backpropagation-example-with-numbers-step-by-step/>

Algorisme de *backpropagation*: exemple



El recorregut sencer ja no es pot expressar tan fàcilment en una sola expressió. Desglossem primer els diferents passos:

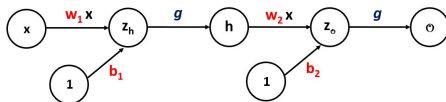
- $z_h = w_1 x + b_1 := L_1(w_1, b_1; x)$
- $h = g(z_h)$
- $z_o = w_2 h + b_2 := L_2(w_2, b_2; h)$
- $O = g(z_o) \quad (O = f(x))$

Si componem tots els passos, podem escriure:

$$O = O(w, b; x) = g(L_2(w_2, b_2; g(L_1(w_1, b_1; x)))),$$

on $w = (w_1, w_2)$, $b = (b_1, b_2)$, i x anirà prenent valors entre les dades d'entrada (*inputs*).

Algorisme de *backpropagation*: exemple



- $z_h = w_1 x + b_1 := L_1(w_1, b_1; x)$
- $h = g(z_h)$
- $z_o = w_2 h + b_2 := L_2(w_2, b_2; h)$
- $\mathcal{O} = g(z_o) \quad (\mathcal{O} = f(x))$

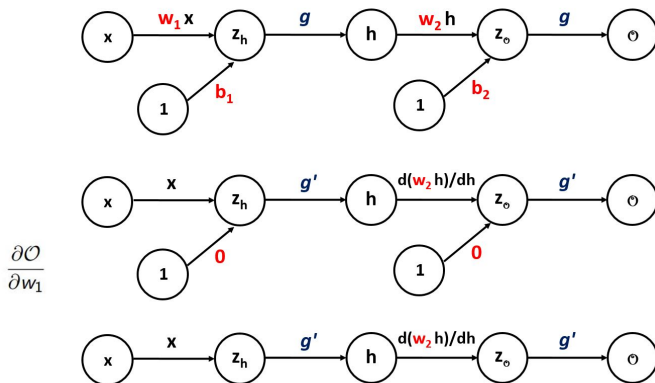
La funció d'error és $E(w, b; x_i) = \frac{1}{2} (\mathcal{O}(w, b; x_i) - t_i)^2$. Prenem una dada qualsevol x_i i, utilitzant la **regla de la cadena**, calculem les derivades respecte els paràmetres que ens serviran per construir el gradient:

$$\frac{\partial E}{\partial w_1} = \frac{d\mathcal{O}}{dz_o} \frac{dz_o}{dh} \frac{dh}{dz_h} \frac{dz_h}{dw_1} = g'(z_o) w_2 g'(z_h) x$$

$$\frac{\partial E}{\partial b_1} = \frac{d\mathcal{O}}{dz_o} \frac{dz_o}{dh} \frac{dh}{dz_h} \frac{dz_h}{db_1} = g'(z_o) w_2 g'(z_h) 1$$

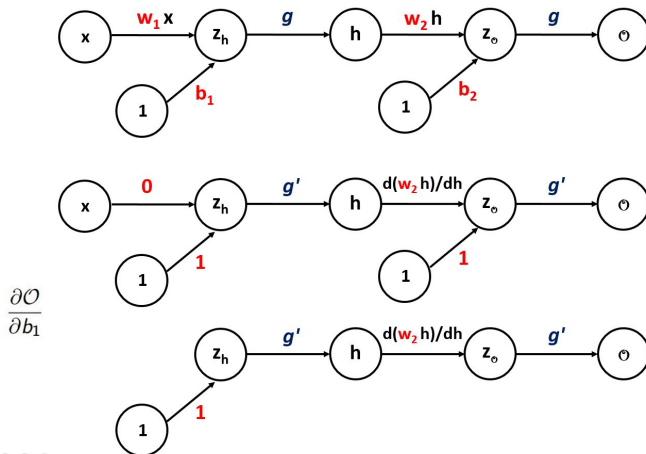
$$\frac{\partial E}{\partial w_2} = g'(z_o) h = g'(z_o) g(w_1 x + b_1)$$

Algorisme de *backpropagation*: exemple



Retropropagació en termes de w_1 .

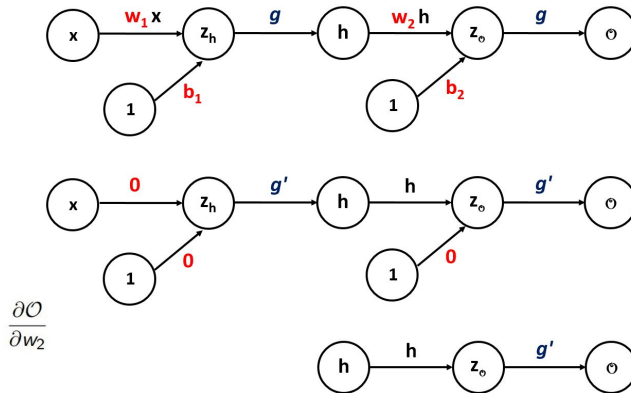
Algorisme de *backpropagation*: exemple



Retropropagació en termes de b_1 .

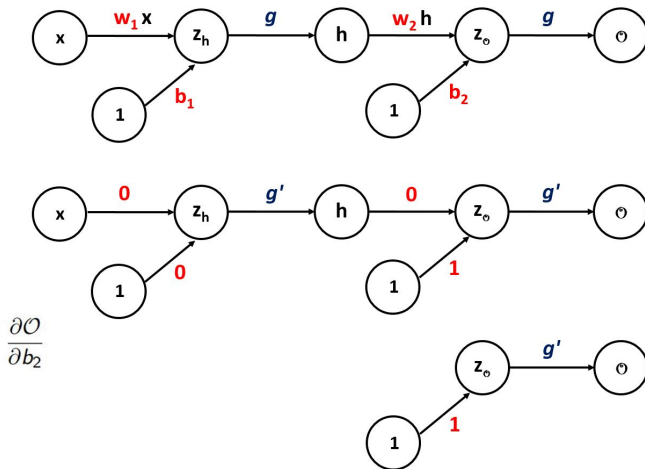


Algorisme de *backpropagation*: exemple



Retropropagació en termes de w_2 .

Algorisme de *backpropagation*: exemple

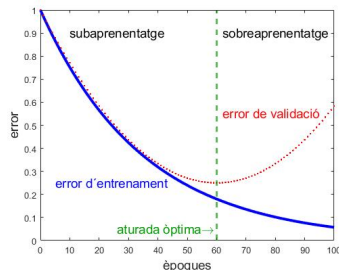


Retropropagació en termes de b_2 .

Fase de validació; sub/sobre-aprenentatge

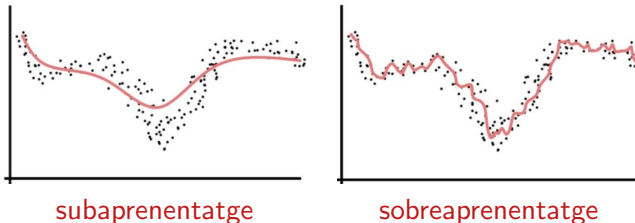
Validació

- El conjunt de **dades de validació**, **independent** del conjunt d'entrenament, s'usa per proporcionar una avaluació del model mentre s'afinen els seus hiperparàmetres (disseny/arquitectura de la xarxa). El model no aprèn d'aquestes dades.
- El **disseny/arquitectura de la xarxa** es manifesta, per exemple, en el nombre de capes o el nombre de neurones a cada capa oculta.
- Per exemple, les dades d'entrenament s'utilitzen per entrenar diverses arquitectures o per diferents algorismes, i les dades de validació s'utilitzen per decidir quin és millor, basant-se en característiques com ara **precisió**, **sensibilitat**, **especificitat**,...



Si introduïm massa dades, pot ser que la xarxa identifiqui relacions aparents en les dades d'entrenament que no tenen en general.

Importància de la validació: sub/sobre-aprenentatge



◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡

Fase de prova

Prova/test

- El conjunt de **dades de prova** és independent del conjunt de dades d'entrenament, i s'utilitza **per avaluar el rendiment** un cop el model s'ha entrenat.
- És desitjable que el conjunt de dades de prova tingui un **error similar al del conjunt d'entrenament**. Un error massa gran comparat amb el del conjunt d'entrenament pot indicar un sobreaprenentatge.
- Sol contenir dades mostrejades acuradament que abasten les diverses classes amb què s'enfrontarà el model quan s'utilitzi en les aplicacions reals.

Exemples en Matlab: regressió

Matlab

App Neural Network Fitting: conjunt de dades

Conjunt de dades sobre **ús d'energia en edificis** a partir dels temps (de rellotge) i les condicions climàtiques.

Al carregar `building_dataset.MAT`, es carreguen dues matrius que s'emmagatzemen en sengles variables:

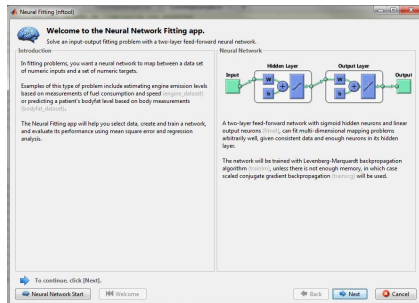
- `buildingInputs`: una matriu 14×4208 que conté 14 atributs de 4208 edificis diferents:
 - 1-10. Dia de la setmana, instant durant el dia; 11. Temperatura; 12. Humitat; 13. Força de la radiació solar; 14. Vent.
- `buildingTargets`: una matriu 3×4208 que conté informació sobre l'ús d'energia a cada edifici.

Matlab: App Neural Network Fitting

Activació



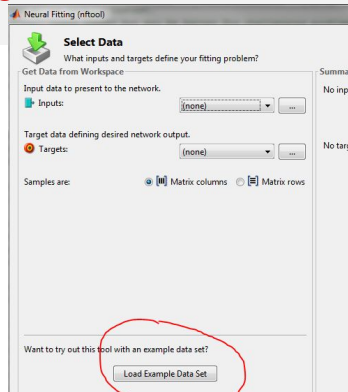
- Seleccioneu l'app **Neural Network Fitting**.
- Fixem-nos detingudament en els comentaris, especialment en la informació sobre l'**estructura de la xarxa** i l'**algorisme d'entrenament**



Matlab: *App* Neural Network Fitting

Càrrega de dades

- Carregar la base de dades des de **Load Example Data Set**.
- També podem carregar la nostra pròpia base de dades, important des d'un full de càlcul (per exemple, excel) i gravant el contingut de la base de dades en un fitxer `.mat`.
- Al fitxer `PreparacioBaseDades.mlx` trobareu el codi per **convertir una taula** importada d'una base de dades **a una matriu** i emmagatzemar-la a un fitxer `.mat`.
- Cal tenir en compte que la **matriu d'entrades** ha de ser de m files i N columnes, on m és el nombre d'*inputs* diferents, és a dir, la dimensió del vector de dades d'entrada, i N és el nombre de vectors de dades (mida de la població d'entrenament).



Matlab: App Neural Network Fitting

Elecció d'hiperparàmetres

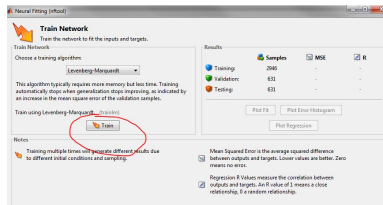
- A **Validation and Test Data** es poden deixar, de moment, les opcions per defecte (70%-15%-15%).
- A **Network Architecture** es poden deixar, de moment, les opcions per defecte (10 neurones a la capa oculta).

The screenshot displays the MATLAB Neural Network Fitting App interface. The top panel, 'Validation and Test Data', shows the default settings for dividing 4208 samples: 70% for Training (2946 samples), 15% for Validation (631 samples), and 15% for Testing (631 samples). The 'Explanation' section on the right describes the roles of Training, Validation, and Testing data. The bottom panel, 'Network Architecture', shows the 'Hidden Layer' settings with 'Number of Hidden Neurons' set to 10. The 'Recommendation' section on the right suggests returning to this panel if the network does not perform well. At the bottom, the 'Neural Network' diagram shows an Input layer with 14 neurons, a Hidden Layer with 10 neurons, and an Output Layer with 3 neurons.

Matlab: *App* Neural Network Fitting

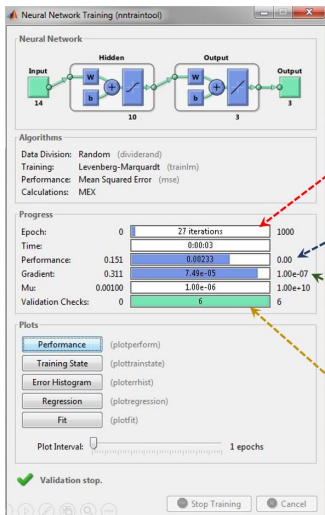
Entrenament de la xarxa

- A **Train Network** es poden deixar les opcions per defecte (mètode L-M) i prémer **Train**.



Matlab: *App* Neural Network Fitting

Resultats: rendiment



Èpoques

Nombre de vegades que s'usa cada vector de dades en l'entrenament.

Rendiment

Ens informa del MSE (*mean squared error*, error quadràtic mitjà):

$$MSE = \frac{1}{N} \sum_{i=1}^N ||\mathbf{t}_i - f(\mathbf{x}_i)||^2$$

($\mathbf{x}_i, \mathbf{t}_i$) dades; $f(\mathbf{x}_i)$ sortida de la xarxa;

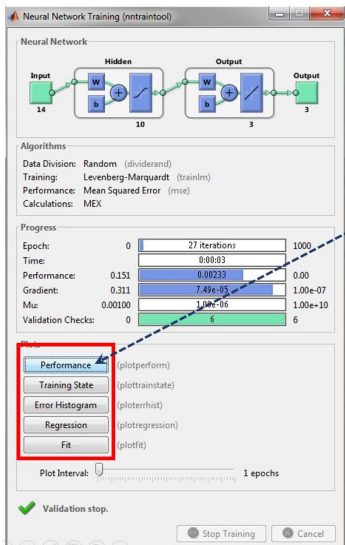
Gradient: Valor del gradient de l'error.

Mu: paràmetre d'adaptació del mètode de LM (tecnicisme).

Nombre de **validacions** exitoses consecutives.

Matlab: *App* Neural Network Fitting

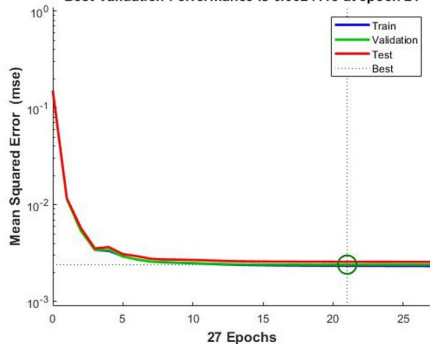
Resultats: gràfiques



Rendiment

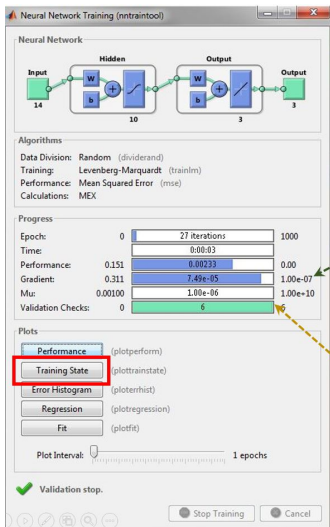
$$MSE = \frac{1}{N} \sum_{i=1}^N ||\mathbf{t}_i - f(\mathbf{x}_i)||^2$$

Best Validation Performance is 0.0024115 at epoch 21



Matlab: *App* Neural Network Fitting

Resultats: gràfiques



Training state

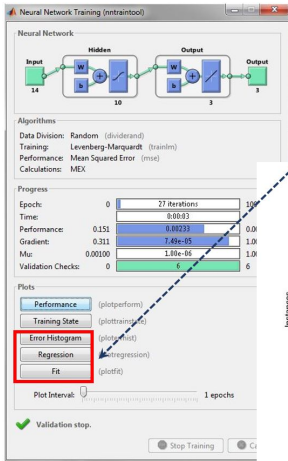
Gradient: Valor del gradient de l'error.

Mu: paràmetre d'adaptació del mètode de LM (tecnicisme).

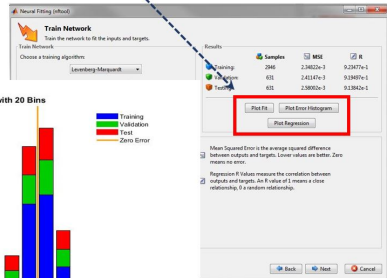
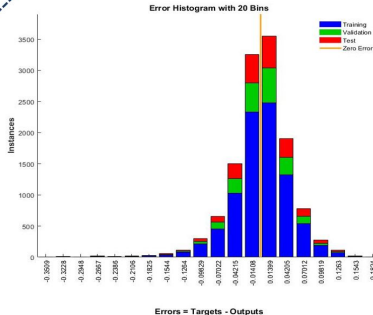
Nombre de **validacions** exitoses consecutives.

Matlab: App Neural Network Fitting

Resultats: gràfiques

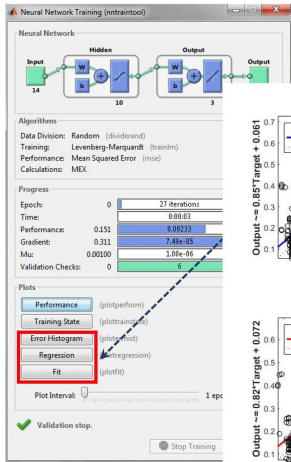


Gràfiques Histograma

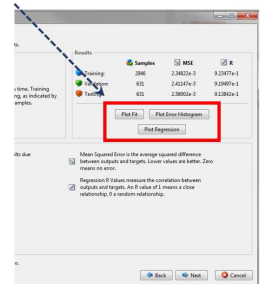
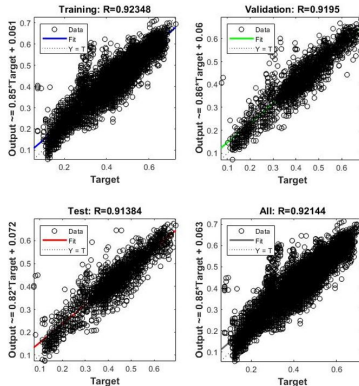


Matlab: *App* Neural Network Fitting

Resultats: gràfiques

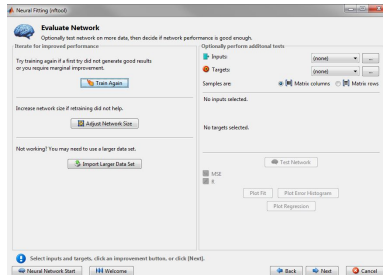


Gràfiques Regressió



Matlab: *App* Neural Network Fitting

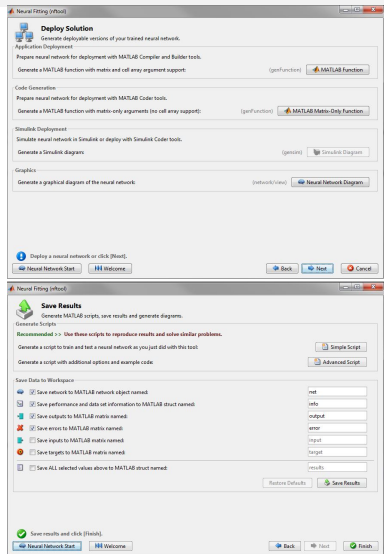
- Possibilitat de **reentrenament**,
d'ampliar la base de dades...
- Opcional.



Matlab: App Neural Network Fitting

Obtenció de codi, desar dades,...

- A **Code Generation** es pot exportar un fitxer `myNeuralNetworkFunction.m` que serà fonamental per a fer **prediccions**.
- També es pot generar el **diagrama de la xarxa**.
- Per **desar** dades i resultats.



Exemples en Matlab: classificació

Matlab

App Neural Network Pattern Recognition: conjunt de dades

Usarem el conjunt de dades `glass_dataset` per classificar vidres que aptes o no per a fabricar finestres, en funció de les seves característiques químiques. La base de dades s'estructura així:

- `glassInputs`: una matriu 9×214 que conté 9 atributs de 214 mostres de vidre diferents:
 1. Índex refractiu; 2. Sodi¹; 3. Magnesi; 4. Alumini; 5. Silicona; 6. Potassi; 7. Calci; 8. Bari; 9. Ferro
- `glassTargets`: una matriu 2×214 que ens indica a quina categoria (1. apte per a finestres; 2. no apte) pertany cadascuna de les 214 mostres.

¹unitat de mesura: percentatge de pes de l'òxid corresponent 

Activitats puntuables

- ❶ Exercici B2.1: Càlcul del valor de sortida d'una xarxa monocapa (20% del bloc) [8NOV]. Individual.
- ❷ Exercici B2.2: Aplicació del mètode de retropropagació a una xarxa simple (20% del bloc) [15NOV]. Individual.
- ❸ Exercici B2.3: <https://archive.ics.uci.edu/ml/datasets.php> (60% del bloc) [22NOV]. Presentacions de 10-15 minuts. Grups de 2 persones.
 - ❶ *Apartment for rent classified Data Set*
 - ❷ *Appliances energy prediction Data Set*
 - ❸ *Concrete Compressive Strength Data Set*
 - ❹ *Concrete Slump Test Data Set*
 - ❺ *Energy efficiency Data Set*
 - ❻ *Human Activity Recognition Using Smartphones Data Set*
 - ❼ *Residential Building Data Set Data Set*
 - ❽ ...

Altres mètodes d'aprenentatge automàtic

Altres mètodes d'aprenentatge automàtic

Alguns noms i algunes idees

- Processos de jerarquia analítica: problemes de presa de decisions.
- Lògica difusa.
- Algorismes genètics. Problemes d'optimització.
- *Simulated annealing*. Problemes d'optimització. Mètode de cerca.
- Xarxes de Hopfield. Recurrents.
- . . .

[Mehmood *et al*, Energy& Buildings, 2019].

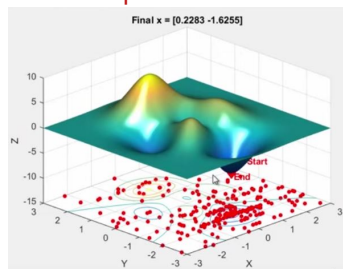
Algorismes genètics

- Escollim un conjunt de valors inicials (p.ex., a l'atzar). Els expressem en sistema binari.
- Definim la **funció d'aptitud**: avalua la capacitat de resoldre el problema.
- Fem els següents passos fins a assolir la condició d'aturada.
 - Avaluem l'**aptitud** (*fitness*) dels *individus* de la població.
 - Seleccionem parells d'*individus* entre els que tenen millor aptitud i eliminem la resta.
 - Generem nous *individus* per **encreuament** (*crossover*) i **mutació**.

https://ca.wikipedia.org/wiki/Algorisme_gen%C3%A8tic,

<https://www.youtube.com/watch?v=1i8muvzZkPw>

Optimització



Encreuament

```
parent1 = [ 1 0 1 0 0 1 1 0 0 0 ]
parent2 = [ 1 0 0 1 0 0 1 0 1 0 ]
child    = [ 1 0 0 0 0 1 1 0 1 0 ]
```

Mutació

```
parent = [ 1 0 1 0 0 1 1 0 0 0 ]
child   = [ 0 1 0 1 0 1 0 0 0 1 ]
```

Referències utilitzades

Introducció a la IA

- Muhammad Uzair Mehmood *et al.* *A review of the applications of artificial intelligence and big data to buildings for energy-efficiency and a comfortable indoor living environment*, Energy & Buildings, 2019.
- Daniel Luckey *et al.*, *Artificial intelligence techniques for smart city applications*, preprint, 2020.
- *Artificial Intelligence In Building And Construction*,
<https://azati.ai/artificial-intelligence-in-building-and-construction/>
- Marie O'Mahony. *Artificial intelligence through the lens of biomimicry*, WTIN, 2018,
<https://www.wtin.com/article/2018/july/090718/artificial-intelligence-through-the-lens-of-biomimicry/>.
- *Che cos'è l'intelligenza artificiale (IA)?*,
<https://it.mathworks.com/discovery/artificial-intelligence.html>
- Altres enllaços citats a les presentacions.

Referències utilitzades

Xarxes neuronals

- Joan Bruna, Sebastià Xambó. *Aprenentatge algorímic i xarxes neuronals profundes*, apareixerà al Butlletí de la Societat Catalana de Matemàtiques, vol. 36, no. 1, 2021.
- Hugo Larochelle. Vídeos sobre *Neural Networks*, <https://www.youtube.com/watch?v=SGZ6BttHMPw> ([1.1]).
- Pedro Ponce Cruz. *Inteligencia Artificial Con Aplicaciones a la Ingeniería*, Alfaomega, 2011.
- Tarang Shah, *About Train, Validation and Test Sets in Machine Learning*, <https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7>
- *Backpropagation Example With Numbers Step by Step*, <https://www.anotsorandomwalk.com/backpropagation-example-with-numbers-step-by-step/>

Agraïments

Per converses molt útils per preparar el curs i orientació en les referències:

- Inma Rodríguez Cantalapiedra (Dept. Física Aplicada, UPC).
- Enrique Romero (Dept. Ciències de la Computació, UPC).
- Néstor Roqueiro (Dept. of Automação e Sistemas, Univ. Fed. Santa Catarina, Brasil).
- Sebastià Xambó (Dept. Matemàtiques, UPC).