UNIVERSITAT DE BARCELONA
Facultat de Matemàtiques i Informàtica

GRAU DE MATEMÀTIQUES

Treball final de grau

AUTOENCODERS

Autor: Eduard Planasdemunt Cobo

Director: Dr. Josep Fortiana
Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 24 de gener de 2023

Abstract

In this project we study antoencoders, a machine learning tecnique used for dimensionality reduction of databases, analizing images or generating new data. We compare them with tradicional dimensionality reduction method, the principal component analysis (PCA). Even though in some fields (specially with small databases) PCA is useful we show that autoencoders can accomplish the same tasks with better results and even accomplish new ones unattainable with PCA. We prepared programs in Python implementing several versions of autoencoders, applied frequently used databases, comparing results with those obtained with PCA, when applicable.

Resum

En aquest treball estudiem els autoencoders, una tècnica d'aprenentatge automàtic utilitzat per a la reducció de dimensionalitat de les dades i, a més, per a l'anàlisi d'imatges o la generació de noves dades . El comparem amb el mètode tradicional de reducció de la dimensionalitat, l'anàlisi de components principals (o PCA). Mostrem que, tot i que en alguns àmbits (sobretot amb bases de dades petites) el PCA és útil, els autoencoders poden dur a terme les mateixes tasques obtenint millors resultats i fins i tot fer-ne de noves que eren impossibles amb PCA. Hem preparat programes en llenguatge Python implementant diverses versions d'autoencoders, que hem aplicat a algunes bases de dades habituals, comparant els resultats amb els de PCA, quan aquest mètode és aplicable.

Índex

1 Introducció

En aquest treball es parlarà dels autoencoders, una tècnica d'aprenentatge automàtic, i la comparació d'aquests amb el mètode tradicional PCA. Per fer-ho començarem explicant què és PCA, com funciona i com interpretar els seus resultats. Seguirem parlant dels tipus d'autoencoders, el seu funcionament intern i com utilitzar-los. Per últim parlarem d'alguns exemples comparant un mètode amb l'altre en un mateix problema i, en els casos on no sigui pràctic aplicar PCA parlarem d'exemples fets únicament amb autoencoders.

El reconeixement de patrons ha estat una de les principals raons per la qual l'ésser humà ha arribat on és avui dia. És més, el nostre cervell està preparat per trobar-los. En un inici es van anar descobrint aquests patrons per supervivència i de manera involuntària: les primeres generacions van adonar-se de que alguns aliments tendien a créixer a certs llocs específics sota certs condicions així que sabrien on buscar-los, o que els animals quan s'espanten fugen cap a llocs on s'hi podria emboscar-los per conduir-los a una mort segura i així alimentar a tota la tribu.

Més endevant van sorgir els primers científics: somiadors que mirant al cel podien veure com aquest canviava nit rere nit, humans que per ajudar als seus propers estudiaven les propietats de les plantes, curiosos que volien comprendre el per què de l'engranatge perfecte que és la natura. Després d'aquests van venir els grans erudits als que podem posar noms: Newton descobrint la gravetat, Pitagoras descobrint el seu famós teorema, Copernic amb l'heliocentrisme, i una infinitat d'altres grandíssims pilars de la nostra era: l'era de la informació.

Tot va començar amb la màquina de Turing que va sentar un precedent dins del món de la computació: aquesta màquina era capaç de desencriptar missatges automàticament. Des d'aquell moment els ordinadors s'han fet servir com a eines que milloraven el rendimen de la persona utilitzant-la en un munt de tasques, o així va ser fins el 1959 amb el naixement del Machine Learning. El Machine Learning va ser una revolució al món computacional: es va començar a treballar amb la idea de que els ordinadors poguessin aprendre de manera automàtica de situacions quotidianes a partir de bases de dades gegants.

A partir d'ara imaginarem que podem representar els esdeveniments que volem analitzar amb un nombre de variables finit i ens interessa conservar l'essència d'aquests esdeveniments en un espai mínim, és a dir, trobar un nombre menor de variables amb les que poguem aproximar un resultat similar. La primera idea que es ve a la ment és descartar les variables que menys influèixen al resultat per així conservar la màxima informació possible. Veiem un exemple. Imaginem que tenim una base de jugadors de bàsquet amb els seus percentatges d'encert i un seguit de variables amb dades sobre els jugadors, com poden ser l'altura, l'embergadura, la llargada d'una passa i el color de cabell. Intuitivament podríem pensar que el color del cabell no hauria d'afectar el resultat, així que procedim a descartar-la. Aquest mètode és efectiu per a aquest cas, però no tots els casos són així de directes. Imaginem ara que dels mateixos esportistes les variables que es prenen són altura, embergadura, llargada de passa, pes, capacitat toràcica i edat. Amb quin criteri podríem descartar unes variables respecte unes altres? Antigament al béisbol es triaven variables subjectives com a criteri de tria dels millors jugadors, però les matemàtiques van demostrar que era un mètode ineficient, convertint-lo en uns dels esports més basats en l'estudi estadístic que existeix.

Després va aparèixer el mètode basat en l'anàlisis de components principals (en parlarem en amb més extensió al capítol 2). Aquest el que busca és trobar noves variables

combinació lineal de les inicials de manera que afegir cada una maximitzi la variabilitat de les dades. És prou útil en casos on les variables estiguin fortament correlacionades. Imaginem que volem evaluar les propietats en una base de dades amb únicament tres variables: l'altura, la embergadura i la llargada de passa. Prenent dues variables que siguin combinació lineal de les altres tres que representin resultats similars és un bon mètode (només cal veure l'home de Vitruvi de Da Vinci), però inclús prenent una única variable podem representar prou fidelment els resultats (dividint per 3 la dimensió de la base de dades). Novament aquesta tria no és intuitiva i es basa en un mètode de maximització de variabilitat que veurem més endevant.

Un cop acabada aquesta explicació del PCA arribarem al tema central del treball. Parlarem dels autoencoders. Aquests defineixen mitjançant un procés cíclic una manera de trobar un nombre menor de variàbles anomenades variables latents de manera no lineal amb funcions més complexes (en parlarem per aquesta finalitat al capítol 4). Veurem que els autoencoders poden arribar a reduir la dimensió de la base de dades en una proporció enorme (veurem un exemple que aconsegueix reduir-la en 240 vegades). Seguirem amb els autoencoders variacionals, fets servir noves dades, fictícies però factibles, similars a les de la base de dades estudiada basats en la inferència bayessiana (capítol 5).

Per últim parlarem d'exemples de programes amb algunes de les moltes aplicacions dels autoencoders, i els compararem en els casos que sigui possible a PCA.

2 Anàlisi de Components Principals (PCA)

2.1 Conceptes bàsics

En aquest apartat introduirem un seguit de conceptes estadístics i matemàtics que farem servir més endevant.

Definició 2.1. *Sigui $M \in \mathbb{R}^{n\times n}$ una matriu quadrada i sigui $v \in \mathbb{R}^n$ un vector no nul, si existeix $\lambda \in \mathbb{R}$ tal que es compleix*

$$Mv = \lambda v$$

diem que v és un vector propi (VEP) de M amb valor propi (VAP) associat λ.

Definició 2.2. *Sigui X una variable aleatòria absolutament contínua amb funció de densitat $f_X(x)$, definim l'esperança de X com*

$$E[X] = \int_{-\infty}^{\infty} x f_X(x)\, dx$$

Definició 2.3. *Sigui X una variable aleatòria absolutament contínua amb funció de densitat $f_X(x)$ definim el moment central d'ordre k μ_k com*

$$\mu_k = E[(X - E[X])^k] = \int_{-\infty}^{\infty} (x-\mu)^k f_X(x) dx$$

on μ és l'esperança matemàtica de X. El moment central d'ordre 2 és la variància.

Definició 2.4. *Donats n elements $x_1, ..., x_n \in \mathbb{R}$ definim la mitjana aritmètica com*

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

i definim la variància com

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} (\bar{x} - x_i)^2.$$

Definició 2.5. *Siguin X i Y dues variables aleatòries definim la covariància entre X i Y com*

$$Cov(X,Y) = E[(X - E[X])(Y - E[Y])]$$

quan E[X], E[Y] i E[XY] és finita. A més, sigui $Z = (Z_1, ..., Z_n)$ un vector aleatòri definim la matriu de covariància de Z com

$$\Sigma = \begin{pmatrix} Cov(Z_1,Z_1) & Cov(Z_1,Z_2) & ... & Cov(Z_1,Z_n) \\ Cov(Z_2,Z_1) & Cov(Z_2,Z_2) & ... & Cov(Z_2,Z_n) \\ \vdots & \vdots & \ddots & \vdots \\ Cov(Z_n,Z_1) & Cov(Z_n,Z_2) & ... & Cov(Z_n,Z_n) \end{pmatrix}$$

Definició 2.6. *Sigui $Q = (q_1, ..., q_m)$ una matriu real amb $q_1, ..., q_n$ vectors columna, diem que Q és ortogonal si*

$$\langle q_i, q_j \rangle = 0 \ \forall i, j \in \{1, ..., m\} | i \neq j$$
$$\langle q_i, q_i \rangle = 1 \ \forall i \in \{1, ..., m\}.$$

Les matrius ortogonals compleixen que

$$Q^{-1} = Q^T.$$

Definició 2.7. *Sigui M una matriu simètrica real de dimensió $n \times n$, diem que M és semidefinida positiva si i només si*

$$x^T M x \geq 0 \text{ per qualsevol } x \in \mathbb{R}^n \setminus \{0\}$$

Lema 2.8. *Els valors propis d'una matriu real semidefinida positiva són no negatius.*

Demostració. Considerem $S \in \mathbb{R}^{n \times n}$ una matriu simètrica semidefinida positiva. Com que és simètrica existeix una descomposició $S = Q^T D Q$ amb Q matriu ortogonal i D matriu diagonal. Per ser Q ortogonal, $S = \sum_{i=1}^{n} q_i d_i q_i^T$ siguent q_i vectors columna de Q.

Per altra banda, $x^T S x = x^T (\sum_{i=1}^{n} q_i d_i q_i^T) x = \sum_{i=1}^{n} x^T q_i d_i q_i^T x = \sum_{i=1}^{n} (q_i^T x)^T d_i q_i^T x = \sum_{i=1}^{n} d_i (q_i^T x)^2 \geq 0$ per definició de semidefinida positiva, i per tant d_i són tots no negatius.

Proposició 2.9. ***Descomposició espectral***
Per qualsevol matriu simètrica real S existeix una descomposició

$$S = V \Lambda V^T$$

on V es la matriu ortogonal formada pels VEPS de S per columnes, i Λ la matriu diagonal formada pels VAPS corresponents.

Observació 2.10. La matriu de covariància és sempre semidefinida positiva. Sigui $c = (c_1, ..., c_n)^T$ un vector de n components tals que $c_i = Z_i - E[Z_i]$ observem que podem reescriure qualsevol element de la matriu com

$$\Sigma_{ij} = E[c_i c_j]$$

i per tant podem reescriure la matriu com

$$\Sigma = E[cc^T].$$

Veiem doncs que donat un vector real $x = (x_1, ..., x_n)^T$ qualsevol aleshores

$$x^T \Sigma x = x^T E[cc^T] x = E[x^T cc^T x] = E[(x^T c)^2] \geq 0$$

degut a que $(x^T c)^2 \geq 0$.

Definició 2.11. *Sigui X i Y dues variables aleatòries que prenen n valors x_i, y_i $\forall i \in \{1, ..., n\}$ respectivament, definim la covariància mostral S_{XY} i la covariància corretgida $\overline{S}_{XY}$ com*

$$S_{XY} = \frac{1}{n}\sum_{i=1}^{n}((x_i - \bar{x})(y_i - \bar{y}))$$

$$\overline{S}_{XY} = \frac{1}{n-1}\sum_{i=1}^{n}((x_i - \bar{x})(y_i - \bar{y}))$$

on $\bar{x}$ i $\bar{y}$ són les mitjanes aritmètiques de X i Y respectivament.

D'ara en endevant quan parlem de covariància entre mostres de variables aleatòries estarem parlant de la covariància corretgida.

A més usarem el terme informal base de dades per referir-nos a una matriu de n files i m columnes, on cada fila preserva els valors de les m variables de cada element que forma la població estadística.

2.2 Components Principals

La idea principal de l'Anàlisi de Component Principals (PCA) és la de reduir la dimensionalitat d'una base de dades formada per elements que depenen nombre de variables correlacionades, perdent la mínima informació possible. Això s'aconsegueix transformant les variables en unes de noves anomenades Components Principals (PC) mitjançant una projecció. Les PCs són un conjunt ordenat de variables no correlacionades tal que les primeres contenen la major part de la variabilitat de la base de dades. Definim el concepte de variabilitat.

Definició 2.12. *Sigui $x = (x_1, ..., x_m)$ un vector aleatori amb matriu de covariància associada Σ, definim la variabilitat de x com*

$$Vari(x) = Tr(\Sigma)$$

on $Tr(\Sigma)$ és la suma dels elements de la diagonal de Σ.

En el cas de matrius quadrades de dimensió $m \times m$

$$Tr(\Sigma) = \sum_{i=1}^{m}\lambda_i$$

on $(\lambda_1, ..., \lambda_m)$ són els VAPS de Σ.

Definirem la primera PC (PC1) com una combinació lineal de les variables inicials amb variabilitat màxima. Sigui $x = (x_1, ..., x_m)$ un vector aleatori, definirem la PC1 com:

$$PC1 = xv_1 = \sum_{i=1}^{m} x^i v_1^i,$$

on $v_1 = (v_1^1, ..., v_1^m)^T$ és un vector unitari de nombres reals i

$$Vari(xv_1)$$

és màxima. Tot seguit definim PC2 com

$$PC2 = xv_2 = \sum_{i=1}^{m} x^i v_2^i,$$

on $v_2 = (v_2^1, ..., v_2^m)^T$ és un vector unitari de nombres reals tal que PC1 i PC2 no estan correlacionades i

$$Vari(xv_2)$$

és màxima.

De forma iterativa definim la PCk com

$$PCk = xv_k = \sum_{i=1}^{m} x^i v_k^i,$$

on $v_k = (v_k^1, ..., v_k^m)^T$ és un vector unitari de nombres reals, PCk no està correlacionada amb PCi $\forall i < k$ i a més

$$Vari(xv_k)$$

és màxima.

Mitjançant aquest mètode podem obtenir fins a m components principals. Per la reducció de dimensionalitat en prendrem p, amb $p < m$.

Definició 2.13. *Sigui $n \in \mathbb{N}$ un nombre natural, $S \subseteq \mathbb{R}^n$ un subconjunt no buit de $\mathbb{R}^n$, $C \subseteq S$ un subconjunt no buit de S i $x_0 \in C$ un punt de C i $f : S \longrightarrow \mathbb{R}$, aleshores diem que f té un màxim global condicionat per C a x_0 si $f(x) \leq f(x_0)$ $\forall x \in C$.*

Teorema 2.14. ***Teorema dels multiplicadors de Lagrange*** *Siguin $n, m \in \mathbb{N}$ dos naturals tals que $m < n$, $S \subseteq \mathbb{R}^n$ un subconjunt obert no buit de $\mathbb{R}^n$, $f : S \longrightarrow \mathbb{R}$ i $g = (g_1, ..., g_m)$ dues funcions de classe $\mathcal{C}^1$ a S, X_0 el subconjunt de S donat per $X_0 = \{x \in S | g(x) = 0\}$ i x_0 un punt de X_0. Suposem que f té un màxim global condicionat per X_0 a x_0 i que la matriu jacobiana de g té a x_0 rang màxim, aleshores existeixen m nombres reals $\lambda_1, ..., \lambda_m \in \mathbb{R}$ verificant*

$$D_i f(x_0) + \sum_{j=1}^{m} \lambda_j D_i g_j(x_0) = 0 \quad i = 1, ..., n$$

Un cop vista la forma de les PCs cal identificar els respectius $v_1, ..., v_p$. Suposem que coneixem la matriu de covariància Σ de les variables aleatòries que formen la base de dades que volem estudiar. Aquesta matriu de covariància no té VAPS repetits. El cas on aquesta matriu té VAPS repetits succeeix amb una probabilitat 0, i inclús en el cas on pogues ocorrer només caldria sumar-li un nombre infinitèssimament petit a qualsevol de les variables d'un dels seus elements i desfer-nos així d'aquesta casuística sense alterar la base de dades. Degut a que els VAPS de Σ i Λ són els mateixos amb la descomposició espectral farem servir Λ per simplificar la notació. Podem calcular la variabilitat aportada per la i-èsima PC com

$$Vari(xv_1) = v_1^T \Lambda v_1$$

i del fet de que v_1 sigui unitari comporta la restricció següent:

$$v_1^T v_1 = 1 \iff v_1^T v_1 - 1 = 0.$$

Aplicant el mètode dels multiplicadors de Lagrange amb $g = v_1^T v_1 - 1$ busquem un extrem de

$$v_1^T \Lambda v_1 - \lambda(v_1^T v_1 - 1)$$

i derivant respecte v_1 i s'obté

$$\frac{\partial}{\partial v_1}(v_1^T \Lambda v_1 - \lambda(v_1^T v_1 - 1)) = \Lambda v_1 - \lambda v_1 = 0 \iff (\Lambda - \lambda Id)v_1 = 0$$

i per definició de VEP, v_1 és un VEP de Λ amb VAP associat λ. A més l'elecció de v_1 unitari comporta que

$$Vari(xv_1) = v_1^T \Lambda v_1 = v^T \lambda v_1 = \lambda v_1^T v_1 = \lambda$$

i com volem que aquesta sigui màxima (i la variància és sempre positiva o 0, i per tant també tots els elements de la diagonal de la matriu de covariància) aleshores

$$Vari(xv_1^T) = \lambda_1,$$

on λ_1 és el VAP més gran de Λ, amb VEP associat v_1.

No obstant, amb la resta de PCs tenim una altra restricció a l'hora d'aplicar el mètode dels multiplicadors de Lagrange, i és que no estiguin correlacionades amb les PCs anteriors, és a dir

$$Cov[xv_i, xv_k] = 0 \ \forall i < k \iff v_i^T \Lambda v_k = 0 \ \forall i < k$$

Comencem caracteritzant v_2.

$$v_1^T \Lambda v_2 = 0 \iff v_2^T \Lambda v_1 = 0 \iff v_2^T \lambda_1 v_1 = 0 \iff \lambda_1 v_2^T v_1 = 0 \iff v_2^T v_1 = 0$$

i podem deduir que la segona restricció equival a $v_2^T v_1 = 0$. Tornant a aplicar el mètode de Lagrange la funció a la que s'ha de trobar els extrems és

$$v_2^T \Lambda v_2 - \lambda_1(v_2^T v_2 - 1) - \lambda_2(v_2^T v_1)$$

i derivant respecte la variable

$$\frac{\partial}{\partial v_2}(v_2^T \Lambda v_2 - \lambda_2(v_2^T v_2 - 1) - \lambda_1(v_2^T v_1)) = \Lambda v_2 - \lambda_2 v_2 - \lambda_1 v_1 = 0 \iff$$
$$v_1^T(\Lambda v_2 - \lambda_2 v_2 - \lambda_1 v_1) = 0 \iff \lambda_1 = 0.$$

Per tant la expresió que volíem trobar-li els extrems és

$$v_2^T \Lambda v_2 - \lambda_2(v_2^T v_2 - 1).$$

Observem que aquesta expresió coincideix tret dels index amb la de PC1 i per tant podem afirmar que v_2^T és el VEP unitari de Λ amb VAP associat λ_2, on λ_2 és el segon VAP més gran. Podem generalitzar aquesta expresió per la resta de PC, obtenint

$$PCk = xv_k$$
$$Vari(xv_k) = \lambda_k$$

on λ_k és el k-èssim VAP més gran de Λ, amb VEP unitari associat v_k, transformant així el nostre problema d'optimització en un problema de resolució de matrius.

Així doncs hem trobat una forma de calcular les PCs, però en un cas amb hipòtesis molt restretes: que la matriu de covariància és coneguda i que no té VAPS repetits ni el 0 com a VAP.

D'ara en endevant considerem que l'unica informació que tinguem serà una mostra X amb n elements que considerarem vectors aleatoris de mida p i més endevant parlarem dels casos que necessiten un tractament concret.

Definirem S la matriu de covariància mostral de X. Seguirem el mateix procés efectuat per aconseguir les PCs amb la matriu de covariància coneguda, fent servir la matriu de covariància mostral en lloc de la matriu de covariància i restringirem les PCs de manera que maximitzin la variabilitat.

Sigui $\tilde{X}$ una base de dades de mida $n \times m$ i sigui $\overline{x} = (\overline{x}_1, ..., \overline{x}_m)$ el vector de mitjanes de $\tilde{X}$ i $u \in \mathbb{R}^{n\times 1}$, $u = (1, ..., 1)^T$ definim

$$X = \tilde{X} - u\overline{x}$$

Direm que X és la representació centrada de $\tilde{X}$. A més aquestes bases de dades comparteixen matriu de correlacions.

D'ara en endevant quan parlem de base de dades farem referencia a la seva representació centrada.

Definició 2.15. *Sigui X una base de dades, $S \in \mathbb{R}^{m\times m}$ la matriu de covariància mostral associada a X i λ_k el k-èssim VAP més gran de S aleshores*

- $z_k = xv_k$ *és la PCk*
- $Vari(z_k) = \lambda_k$
- $v_k = (v_1^k, ..., v_m^k)^T$ *és el VEP unitari de S associat al VAP* λ_k

Observació 2.16. Les matrius de S i de $X^T X$ són proporcionals amb factor de proporcionalitat $\dfrac{1}{n-1}$.

Recuperant el fet de que la matriu de covariància és simètrica i per tant té una descomposició tal que

$$S = V\Lambda V^T$$

amb Λ matriu diagonal conenint els VAPS de S i V la matriu amb VEPS per columnes que podem reescriure el primer producte com el següent sumatori matricial:

$$V\Lambda = \sum_{i=1}^{m} \lambda_i v_i$$

i degut a l'associativitat del producte i que cada producte $\lambda_i v_i$ és una matriu amb valors únicament a la i-èsima columna

$$V\Lambda V^T = \sum_{i=1}^{m} \lambda_i v_i V^T = \sum_{i=1}^{m} \lambda_i v_i v_i^T.$$

Proposició 2.17. *Sigui S la matriu de covaiància associada a una base de dades X amb descomposició espectral $S = V\Lambda V^T$, sigui V_p la matriu amb vectors columna els primers p VEPS de S, aleshores $Y = XV_p$ és la projecció ortogonal de la base de dades que minimitza la pèrdua de variància. A més a més, podem caracteritzar aquesta pèrdua amb*

$$\Delta Vari = Vari(X) - Vari(Y) = \sum_{i=p+1}^{m} \lambda_i$$

on λ_i és l'i-èsim VAP de S.

Demostració: La variabilitat de X ve donada per

$$Vari(X) = Tr(\Lambda) = \sum_{i=1}^{m} \lambda_i$$

Per una banda, la matriu de covariància de Y és proporcional a Y^TY

$$Tr(\tilde{S}) = Tr(\tfrac{1}{n-1} Y^TY = V_p^T X^T X V_p) = Tr(V_p^T S V_p) = Tr(V_p^T V \Lambda V^T V_p),$$

i com que V i V^T són ortogonals, $V^{-1} = V^T$ i per tant

$$V_p^T V = Id_p,$$

on Id_p és una matriu amb 1 als primers p elements de la diagonal i 0 a la resta de la matriu. Com $V^TV_p = (V_p^TV)^T$ i Id_p és diagonal

$$Tr(\tilde{S}) = Tr(Id_p \Lambda Id_p) = \sum_{i=1}^{p} \lambda_i$$

i com volíem veure

$$\Delta Vari = \sum_{i=1}^{m} \lambda_i - \sum_{i=1}^{p} \lambda_i = \sum_{i=p+1}^{m} \lambda_i$$

Un cop vist tot això cal parlar del cas especial on la matriu de covariància té el 0 com a VAP amb multiplicitat t aleshores vol dir que existeix un espai vectorial de dimensió $p-t$, i per tant que t variables són combinació lineal de les altres, esdevenint redundant i innecessàries. Per solucionar aquest problema és necessari reduir la dimensió de la matriu de covariància mostral desfent-nos de les combinacions trobades abans d'aplicar PCA.

Ara ja hem vist com podem dur a terme el PCA. Es poden calcular els VAPS de les matrius directament, però pot ser lent i poc estable degut als errors acumulats. L'alternativa que es duu a terme és la de calcular els VAPS i VEPS de X^TX. Per fer-ho es duu a terme la descomposició en valors singulars.

2.3 Descomposició en valors singulars

La descomposició en valors singulars (SVD) és una descomposició de matrius tant reals com complexes. És més ràpid i estable que els mètodes tradicionals. En el nostre cas ens centrarem en el cas real.

Definició 2.18. *Sigui M una matriu real, diem que σ és un valor singular de M si existeixen vectors unitaris u i v tals que*

$$Mv = \sigma u \quad M^Tu = \sigma v$$

Aquests valors singulars tenen una relació important amb els VAPS de les matrius.

Lema 2.19. *Sigui A una matriu, aleshores $rang(A) = rang(A^TA)$.*

Demostració. Es suficient probar que $Ax = 0$ té les mateixes solucions que $A^TAx = 0$. Veiem que $Ax = 0 \iff A^TAx = 0$
$\Rightarrow$] Aquest cas és trivial. Si $Ax = 0$, aleshores $A^TAx = A^T0 = 0$
$\Leftarrow$] $A^TAx = 0 \Rightarrow x^TA^TAx = 0 \Rightarrow (Ax)^TAx = \langle Ax, Ax\rangle = 0$, i això succeeix únicament quan $Ax=0$ per propietats del producte escalar.

Proposició 2.20. ***SVD***
Tota matriu $X \in \mathbb{R}^{n\times m}$ amb $n \geq m$ i $rang(X) = k, k \leq n$ pot escriure's com

$$X = U\Sigma V^T$$

on:

- *$U \in \mathbb{R}^{n\times n}$ és una matriu ortogonal amb u_i vectors columna*
- *$V \in \mathbb{R}^{m\times m}$ és una matriu ortogonal amb v_i vectors columna*
- *$\Lambda \in \mathbb{R}^{n\times m}$ és una matriu on els valors de la diagonal λ_i són els valors singulars de X i 0 la resta*

Proposició 2.21. *σ és un valor singular de $X \iff \sigma^2$ és un VAP de X^TX.*

Donada una base de dades X, una vegada aplicat el SVD a aquesta matriu obtindrem una descomposició de X. Només cal prendre Σ^2 per obtenir els VAPS de X^TX, que com hem vist a una observació anterior són proporcionals als VAPS de la matriu de covariància

mostral S de X. A més la matriu U està formada pels VEPS de XX^T i la matriu V pels VEPS de X^TX. Aquest procés és computacionalment més eficient que el de calcular la matriu de covariància mostral. Cal observar que V coincideix amb la matriu V dels apartats anteriors (la V utilitzada a la descomposició espectral) degut a que ambdues estan formades pels VEPS de X^TX unitaris.

Recuperant l'expresió de la projecció ortogonal de la base de dades

$$Y = XV$$

i descomposant X amb SVD

$$Y = XV = U\Sigma V^T V = U\Sigma$$

2.4 Reducció de la dimensionalitat mitjançant PCA

Fer servir p PCs en lloc de les m components del problema redueix de manera important la dimensionalitat de la base de dades quan m és molt més petita que p. No obstant això per calcular aquestes PCs a vegades utilitzem totes les components de la base de dades, així que pot ser preferible prendre p o més components inicials per dur a terme la reducció de dimensionalitat. Dividirem aquesta secció en dues parts: una per parlar de criteris per a la tria de les PCs i una segona sobre com triar un subconjunt de les components inicials adient per la reducció de dimensionalitat.

2.4.1 Criteris per a la tria de PCs

Com hem vist anteriorment quan efectuem PCA busquem els valors singulars de X^TX, per així trobar els VAPS de la matriu de covariància, ordenant-los de major a menor per maximitzar així la variabilitat aconseguida a cada passa. La qüestió no és trobar quines prendre, sinó quantes. A continuació descriurem alguns dels mètodes utilitzats.

Variabilitat total explicada

Correspon a fixar un llindar de variabilitat explicada i es prenen tantes PCs com siguin necessàries per sobrepassar aquest llindar. Per calcular la variabilitat acumulada es pren la variabilitat aportada per les PCs triades i es divideix per la variabilitat total de la base de dades:

$$VTE(r) = 100 \sum_{i=1}^{p} \lambda_i / \sum_{i=1}^{m} \lambda_i$$

on λ_i són els VAPS de S, o el que és el mateix els elements de la matriu diagonal de la descomposició SVD. És a dir, fixat un llindar t, es pren el nombre de components p que compleix

$$\begin{aligned} VTE(r) &\geq t \\ VTE(r-1) &< t. \end{aligned}$$

No hi ha una elecció de valor pel llindar bona a cada cas, sinó que s'ha d'adaptar a cada base de dades. Usualment es prenen valors entre 70 i 90.

Variabilitat de la PC

Un altre criteri és prendre un llindar mínim de variabilitat aportada per les PC triades. Prendrem tota PC que compleixi

$\lambda_i > \lambda^*$, on $\lambda^* = \tilde{\lambda}c$, λ_i és el VAP associat a la PCi i $\tilde{\lambda}$ és la mitjana dels VAPS.

Veurem en els programes que és recomanable prendre $c < 1$, ja que sinó el nombre de PCs triades és massa petit, i per tant la pèrdua de variabilitat gran.

Diagrama dels VAPS i del logaritme dels VAPS

Tot i ser útils aquests dos criteris tenen una forta component subjectiva a l'hora de triar aquests llindars: en el primer exemple escollim el percentatge de variabilitat que volem que mantingui el PCA, mentre que al segon escollim una constant que determina quanta variabilitat aporta cada PC.

Aquest tercer és un criteri que implica enfrontar a una gràfica els VAPS amb el nombre de PC que representen, és a dir unir els punts

$$(\lambda_i, i)\ \forall i \in \{1, ..., p\}$$

amb una linia. Un cop dibuixada aquesta gràfica busquem un colze a la corba, i prenem les m PCs anteriors a aquest colze.

Una variant d'aquest criteri consisteix en enfrontar

$$(\log \lambda_i, i)\ \forall i \in \{1, ..., p\}.$$

El problema d'aquest criteri és que la corba definida no té per què tenir un colze, així que no sempre és aplicable.

2.4.2 Selecció d'un subconjunt de variables

En aquest apartat parlarem breument sobre com trobar un subconjunt de les variables inicials que mantingui una bona part de la variabilitat de la base de dades utilitzant PCA. És possible efectuar PCA després de dur a terme aquesta selecció amb els criteris de tria de PCs de l'apartat anterior.

Associació directa

Una vegada fet el PCA i fixat un nombre p de PCs volem associar a cada una d'aquestes una variable inicial. Com hem vist en apartats anteriors

$PCk = xv_k$, on $v_k = (v_1^k, ..., v_m^k)$ és un vector real i $x = (x_1, ..., x_m)$ les variables inicials.

Relacionarem la PCi amb la variable x_j si v_j és el coeficient en valor absolut més gran tal que no és el coeficient en valor absolut més gran de cap altra PC anterior a i. Una vegada triades les p variables inicials eliminarem la resta, i aquesta serà la nova base de dades utilitzada al problema.

Associació inversa

Una vegada fixat un nombre p de PCs volem associar a les $p^* = m - p$ PCs descartades per PCA unes variables inicials així desestimar aquestes últimes. Aquesta selecció es duu a terme igual que l'associació directa, però començant per la última PC. Un cop feta aquesta relació es procedeix a eliminar les variables inicials seleccionades, obtenint així la nova base de dades.

Per últim, cal destacar que també es pot fer servir el PCA pel tractament d'imatges. El que es fa és imaginar cada imatge com un seguit de capes apilades i mitjanÃ§ant el teorema espectral s'aconsegueix la separació de les capes. Una vegada separades amb la suma matricial s'apilen i així s'obté la nova imatge.

(a) 1 principal component (b) 5 principal component (c) 9 principal component

(d) 13 principal component (e) 17 principal component (f) 21 principal component

(g) 25 principal component (h) 29 principal component

3 Xarxes Neuronals Artificials

3.1 Introducció

Una Xarxa Neuronal Artificial (Artificial Neural Network o ANN) és un model computacional creat per simular el procés d'anàlisi d'informació del cervell humà. La unitat més petita de formació d'aquestes ANN són les neurones artificials.

Definició 3.1. *Sigui $x = (x_1, ..., x_n)$ un vector d'entrada $w = (w_1, ..., w_n)$ un vector de pesos o paràmetres, b un biaix i σ una funció*

$$\sigma \colon R^{n+1} \longrightarrow R$$

definim la neurona artificial com la funció

$$y = \sigma(\sum_{i=i}^{n} x_i w_i + b)$$

que envia els valors d'entrada a un valor de sortida y. La funció σ rep el nom de funció d'activació.

Per facilitar la notació d'ara en endevant considerarem $x = (b, x_1, x_2, ..., x_n)^T$ i $w = (1, w_1, w_2, ..., w_n)$. D'aquesta manera podem reescriure la funció de la neurona artificial com

$$y = \sigma(wx)$$

Veiem un exemple hipotètic per veure on una neurona artificial pot ser útil. Imaginem que estem realitzant un estudi sobre els millors llançadors de bàsquet de tota Catalunya. Podriem considerar el percentatge d'encert sense tenir en compte la posició de llançament, però això no seria just ja que al bàsquet no tots els llançaments valen la mateixa puntuació (els triples valen 3 i els tirs lliures 1). Tampoc seria just considerar únicament l'encert en llançaments de triple, ja que aquests solen suposar menys d'un 30% dels punts d'un partit: el que busquem doncs són jugadors amb un bon encert de totes tres posicions així que guardem en una matriu $X \in \mathbb{R}^{n\times 3}$ el percentatge d'encert de cada jugador(cada fila representa un jugador). Com podem imaginar, el nombre de jugadors de bàsquet a Catalunya és elevat per realitzar un estudi individualitzat (l'any 2019 hi havia aproximadament 73000 jugadors federats a Catalunya) i no tots aquests poden ser considerats bons llançadors. Per tant molts no ens interessen per el nostre estudi. Podriem fixar un llindar per cada tipus de llançament i desfer-nos dels jugadors que no superin aquests tres llindars, però així podriem descartar jugadors que ens interessen: els bons llançadors de tir lliure solen estar per sobre del 80% d'encert, no obstant això hi ha molt bons llançadors que no arriben a aquest percentatge degut a la tipologia de tir totalment diferent a la resta (amb el joc parat). Ens interessaria així trobar un llindar col·lectiu entre les tres categories. Podriem pensar-ho com una neurona artificial que: a partir d'un input (una fila de la matriu X), uns pesos prefixats per nosaltres que donin una importància a cada tipus de llançament, una funció d'activació (en aquest cas la Heaviside, que explicarem més endevant) i un llindar ens digui si el jugador el podem considerar un bon llançador o no, així passant a tenir un estudi d'una mostra de 73000 a una més petita i selecta.

Si no tenim grans coneixements de bàsquet podem veure com aquest exemple necessita l'ajuda i supervisió d'algú amb coneixements del camp, però en cas de saber-ne es poden aconseguir tasques com la de per exemple l'extracció d'informació (en aquest cas classificar llançadors en bons i dolents. No obstant això, i el que busquem, és que aquestes neurones siguin capaces d'analitzar dades sense necessitat de supervisió humana per evitar així el factor subjectiu.

Per simplificar la notació faré referència a les neurones artificials simplement com a neurones.

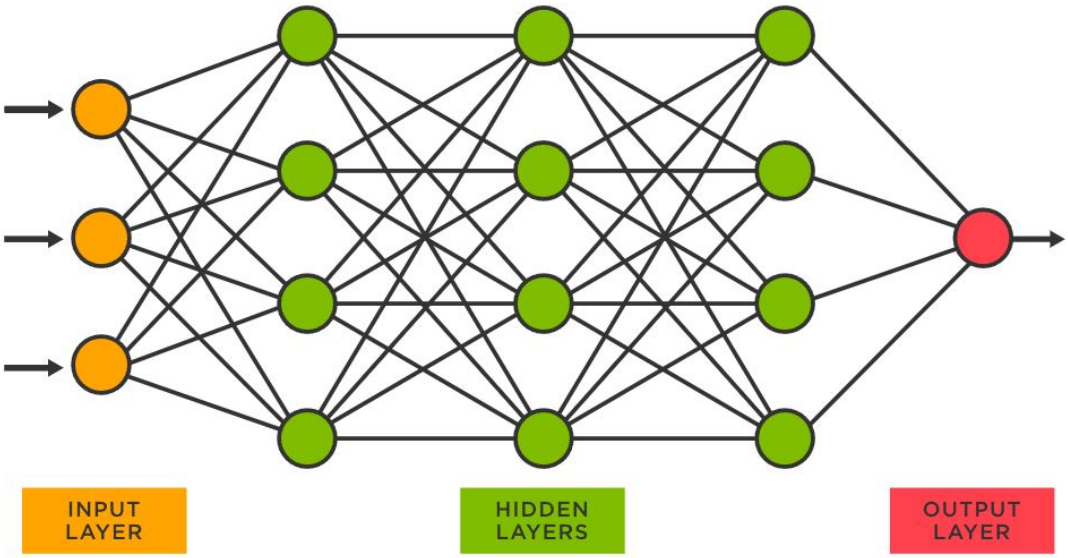
3.2 Estructura de les ANN

Una ANN és un conjunt de neurones agrupades de manera que la entrada d'un subconjunt de neurones és la sortida d'un altre subconjunt. Aquests subconjunts de neurones s'anomenen capes (layer).

Definició 3.2. *Anomenem capa a una familia de neurones que comparteixen entrada, una mateixa funció d'activació i operen a un cert nivell de profunditat dins de l'ANN. S'anomenen capes ocultes (hidden layers) si a més estan connectades únicament a altres capes de l'ANN tant a l'entrada coma la sortida.*

A partir d'ara direm que una ANN té profunditat N si té N-1 capes ocultes. Anomenarem l-capa a la capa amb profunditat l, essent la 0-capa la entrada i l'N-capa la sortida de l'ANN. Dins d'una ANN les operacions es duen a terme per capes, estalviant així en cost computacional.

Exemple: el dense layer és una estructura de layer on totes les neurones estan connectades amb les neurones del layer anterior i del layer següent.



Sigui X una base de dades, una l-capa amb N neurones, aleshores a nivell computacional es pren la matriu $W = (w_1, ..., w_N)$ (on w_i és el vector de pesos de la i-èsima neurona de la capa) com a matriu de pesos. Aleshores la sortida de la capa és

$$Y = \sigma(W^T X)$$

Per últim falta definir la funció d'activació.

Definició 3.3. *Una funció d'activació és una aplicació no lineal*

$$\sigma \colon R^n \longrightarrow A \subset R^n$$

3.3 Aprenentatge d'una ANN

La nostra ANN queda determinada tant per l'estructura de les capes com pels pesos assignats a cada neurona. Aquests pesos poden donar fixats per el programador com hem vist a la introducció d'aquest capítol, però ens centrarem en el cas on no ho són. En aquest segon cas el que farem és donar-li a l'ANN la qualitat de model estadístic predictiu, amb paràmetres els pesos i biaixos.

Sigui X la base de dades, definim els valors desitjats com

$$y = f(x) \quad \forall x \in X.$$

Sigui l'espai de paràmetres $\Theta \subset \mathbb{R}^k$ on k és el nombre de pesos de la ANN i $\theta \in \Theta$ un vector de paràmetres. Definim la sortida (o predicció) de la ANN com

$$\widehat{y} = f_\theta(x) \quad \forall x \in X.$$

D'ara en endevant farem referència a Y com a la matriu amb valors desitjats per columna (respectivament a $\widehat{Y}_\theta$ amb els valors predits per columna).

Cal trobar una manera de calcular com de diferents són els valors predits dels desitjats o com és de bona la predicció. Introduim així les funcions de pèrdua. Una funció de pèrdua (loss function) és una funció que associa a una entrada, una sortida desitjada i uns pesos un valor numèric. Com més petit és aquest valor millor és l'aproximació.

Exemple: MSE (Mean Squared Error) Sigui Θ l'espai vectorial dels paràmetres i X la base de dades d'entrada, definim el MSE com

$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n}(f_\theta(x_i) - f(x_i))^2 = \frac{1}{n}\sum_{i=1}^{n}(\widehat{y}_i - y_i)^2.$$

És especialment útil quan el valor desitjat és la mateixa entrada (més endevant veurem que és especialment útil en el cas dels autoencoders).

Ens interessa per tant minimitzar aquesta funció de pèrdua. El procés d'optimització d'aquests paràmetres és el procés d'aprenentatge (també entrenament o training). Dur a terme aquest procés farem servir diferents algoritmes d'optimització.

3.3.1 Algoritmes d'optimització

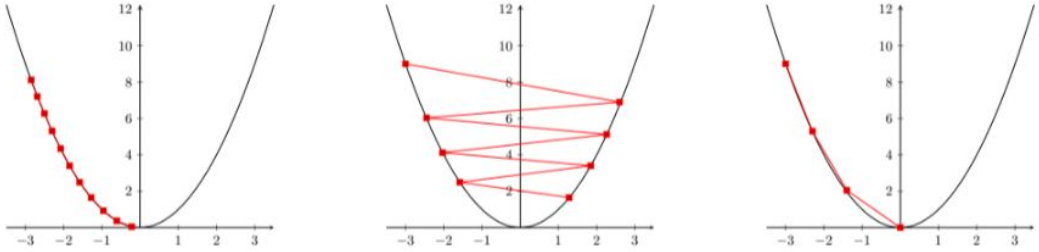
Descens seguint el gradient

El descens seguint el gradient (Gradient Descent o GD) és un mètode iteratiu on a cada passa es minimitza la funció de pèrdua avançant en la direcció que apunta el gradient d'aquesta funció de pèrdua. Es basa en el fet que el gradient d'un camp escalar (en el nostre cas l'espai de paràmetres) evaluat en un punt dóna la direcció sobre la qual varia més (dona la direcció sobre la qual la derivada direccional és superior).

Definició 3.4. *Sigui $\mathcal{L}(\theta)$ la funció de pèrdua, Θ l'espai de paràmetres i $\theta_0 \in \Theta$ un valor inicial triat aleatòriament. El mètode Gradient Descent consisteix en crear una succeció*

$$\theta_{n+1} = \theta_n - \eta \nabla \mathcal{L}(\theta_n).$$

η és l'anomenat factor d'aprenentarge (learning rate) i és un hiperparàmetre, és a dir, un paràmetre fixat prèviament. Aquest factor marca la llargada de les passes que succeeixen amb cada iteració i no pot ser qualsevol valor, ja que si el valor és massa gran no es pot garantir la convergència del mètode i si és massa petit es pot tenir convergència en un nombre massa elevat de passes, el qual convertiria la ANN en lenta.



(a) η too small (b) too large η (c) reasonable η

També tenim l'opció de triar valors per η diferents a cada passa. Aquest procés garanteix la convergència cap a un mínim local, però té un cost computacional superior.

Informalment direm que un subconjunt d'un espai vectorial real és convex si el segment que uneix dos punts qualsevols està contingut en la seva totalitat en el conjunt.

Definició 3.5. *Sigui U un subconjunt convex d'un espai vectorial real V. Una funció és convexa si*

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y) \quad \forall x, y \in U, t \in [0,1]$$

Teorema 3.6. *Sigui U un conjunt convex d'un espai normat i sigui*

$$f : U \longrightarrow R$$

una funció definida sobre aquest conjunt, aleshores els mínims locals de f són també mínims absoluts de la funció.

Pel teorema anterior si prenem una funció de pèrdua convexa aleshores el mètode garanteix una convergència cap a un mínim absolut de la funció.

Observació 3.7. La funció MSE és una funció convexa. Cal prendre

$$\mathcal{L}(\theta) = \frac{1}{n}||\theta^T X - Y||^2$$

amb la norma euclidiana i com és una norma és convexa (directament de la desigualtat triangular).

Per altra banda per calcular el gradient cal fer

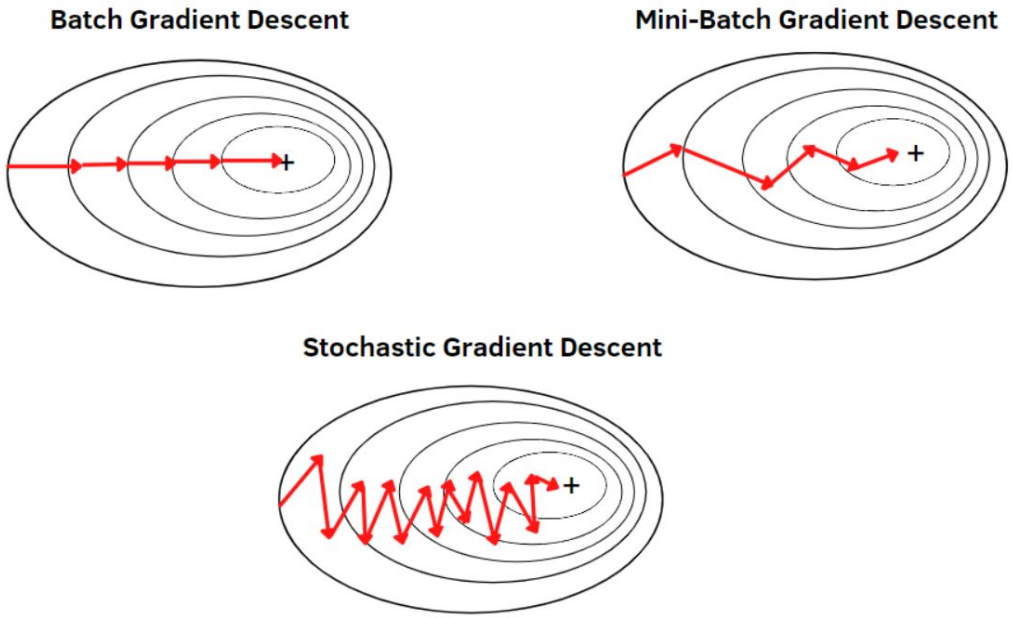
$$\nabla \mathcal{L}(\theta) = \nabla(\frac{1}{n}||\theta^T X - Y||^2 = \frac{2}{n} X^T(\theta^T X - Y).$$

Aquest procés pot ser lent per bases de dades grans degut a la quantitat de càlculs duts a terme. Aquest fet és el que motiva el Descens del Gradient Estocàstic.

Descens seguint el gradient estocàstic

El descens seguint el gradient estocàstic (Stochastic Gradient Descent o SGD) utilitza el propi GD fent servir un subconjunt aleatori de mida fixa de la base de dades pel càlcul del gradient a cada iteració. Anomenarem lot al subconjunt de la base de dades que es pren a cada iteració del mètode. Direm que el mètode ha conclòs una època quan s'ha fet servir una vegada tots els elements de la base de dades.

El SGD mostra respecte al GD un augment en la velocitat de computació degut a que les matrius amb les que s'opera són de dimensió inferior i a més permet la possibilitat (no sempre succeeix) d'escapar mínims locals per trobar el mínim global. Per altra banda, mentre que amb el GD es pot garantir que cada iteració redueix la funció de pèrdua això no passa amb el SGD: aquesta optimització es pot garantir amb cada pas d'una època, però no amb cada iteració individualment. A més a més el mètode convergeix a un entorn del mínim, però no a un valor mínim exacte.



Hem vist una expressió per calcular el gradient de forma matricial de la funció de pèrdua. No obstant això existeix un algoritme automàtic per calcular el gradient de la funció de pèrdua.

3.3.2 Computació del gradient: retropagació

Com hem vist en els mètodes basats en la idea de GD i derivats d'aquests s'ha de calcular el gradient a cada iteració. Amb el mètode de la retropagació computarem aquest gradient començant per la última capa i anant fins a la primer de l'ANN.

Durant aquest apartat ens referirem amb z_l com a l'entrada rebuda per la l-èssima capa

de l'ANN ($z_l = \theta_l^T x_{l-1}$), on ($x_{l-1} = \sigma_{l-1}(z_{l-1})$) és la sortida de la (l-1)-capa amb θ_l amb els pesos d'aquesta. Considerarem que tenim un ANN amb L capes. Utilitzant la regla de la cadena derivant la funció de pèrdua a la l-capa

$$\frac{\partial \mathcal{L}}{\partial \theta_l} = \frac{\partial \mathcal{L}}{\partial z_l}\frac{\partial z_l}{\partial \theta_l}$$

Denotarem $\delta_l = \dfrac{\partial \mathcal{L}}{\partial z_l}$. A més

$$z_l = \theta_l^T x_{l-1} \Rightarrow \frac{\partial z_l}{\partial \theta_l} = x_{l-1} \Rightarrow$$
$$\Rightarrow \frac{\partial \mathcal{L}}{\partial \theta_l} = \delta_l x_{l-1},$$

i per tant per computar aquest gradient és suficient computar δ_l. Podem observar que aquesta retropropagació és l'aplicació de la regla de la cadena als ANN. Com hem definit la retropagació començarem a computar els δ_i començant per la última capa. En el cas que prenem el MSE com a funció de pèrdua

$$\delta_L = \frac{\partial \mathcal{L}}{\partial z_L} = \frac{\partial(\sigma(z_L) - \widehat{y})^2}{\partial z_L} = 2(\sigma_L(z_L - \widehat{y}) \odot (\sigma'_L(z_L) = 2(y - \widehat{y}) \odot (\sigma'_L(z_L),$$

amb $\odot$ el producte per elements de les matrius. A més es pot probar que la resta de deltes segueixen la fórmula

$$\delta_l = (\theta_{l+1}\delta_{l+1}) \odot \sigma'_l(z_l),$$

de manera que hem trobat una manera de computar el valor del gradient a cada iteració, i així una manera d'actualitzar també els pesos.

3.4 Funcions d'activació

Hem vist que durant l'entrenament es fan servir algoritmes basats en el GD, i més en concret es computa aquest gradient mitjançant la retropagació. Vam arribar a demostrar que si la funció de pèrdua era MSE, aleshores aquest gradient depenia de la derivada de la funció d'activació, però es pot demostrar que aquest és el cas sigui quina sigui la funció de pèrdua. Per tant no només ens interessa que la funció d'activació aporti informació, sinó que també la seva derivada sigui fàcilment computable. Separarem les funcions d'activació en lineals, pseudolineals i no lineals.

Funcions d'activació lineals

Aquests com el seu nom indica venen donades per les expresions

$$f(x) = kx$$
$$f(x) = k$$

on k és una constant. D'aquest fet deriva que els pesos de cada funció calculats a cada nova iteració seran menors que l'anterior, de manera que l'aleatorietat dels pesos inicials presos pot fer que l'algoritme convergeixi a una solució gens adient per al problema. La composició de diverses capes amb aquesta mateixa funció d'activació no aporta una informació adicional degut a que la composició de funcions lineals segueix sent lineal.

Funcions d'activació pseudolineals

Les funcions d'activació pseudolineals reben aquest nom degut a són funcions a troços contínues on una part és una funció lineal.

Funció ReLU

La funció ReLU (Rectified Linear Unit) és la funció d'activació pseudolineal més senzilla de totes. La funció ReLU i la seva derivada venen donades per

$$f(x) = max\{0, x\}$$

$$f'(x) = \begin{cases} 0 & si \quad x \leq 0 \\ 1 & si \quad x > 0 \end{cases}$$

respectivament. És actualment la funció d'activació més utilitzada degut a la seva alta velocitat d'execució i és especialment prevalent en els autoencoders convolutius. Com les funcions d'activació lineals mostra el problema de que els valors segueixen una successió decreixent, i per tant si una neurona rep el valor de pes 0 aleshores aquesta quedarà inutilitzada.

Funció ReLU paramètrica

La funció ReLU paramètrica neix de la limitació de la funció ReLU per evaluarels nombres negatius. És una petita alteració de ReLU que dona als nombres negatius una recta amb poc pendent. Es pot escriure com

$$f(x) = max\{ax, x\}$$

$$f'(x) = \begin{cases} a & si \quad x \leq 0 \\ 1 & si \quad x > 0 \end{cases}$$

amb $a < 1$, usualment $a = 0.1$. Aquesta permet l'ús de la retropagació per actualitzar els pesos inclus negatius, però el fet que tingui un pendent tant petit en aquests valors fa que sigui lenta. A més el fet de dependre d'un hiperparàmetre fixat anteriorment fa que pugui variar els resultats obtinguts depenent d'aquest.

Funció ELU

La funció exponencial d'unitats lineals (ELU) és una funció similar a la ReLU que aporta un decreixement suau de la corba pels termes negatius, però amb un cost superior degut al càlcul de la exponencial. Ve representada com

$$f(x) = \begin{cases} x & si \quad x \geq 0 \\ \alpha(e^x - 1) & si \quad x < 0 \end{cases}$$

$$f'(x) = \begin{cases} 1 & si \quad x \geq 0 \\ \alpha e^x & si \quad x > 0 \end{cases}$$

Funcions d'activació no lineals

Funció sigmoidal logística

Aquesta funció envia qualsevol valor a l'interval (0,1). Matemàticament pot ser expresada com

$$f(x) = \frac{1}{1 + e^{-x}}$$
$$f'(x) = \frac{-e^{-x}}{(1 + e^{-x})^2}.$$

Cal observar que $f'(x) = f(x)(1 - f(x))$. Aquesta funció és especialment utilitzada en una última capa quan la funció de l'autoencoder és classificar els elements en diferents grups. És així degut a que la seuva sortida està acotada entre 0 i 1, i per tant pot representar una probabilitat. Només és útil per valors de x entre -3 i 3, degut a que fora d'aquest interval el valor de la derivada és proper a 0, i per tant un punt d'esvaiment del gradient. També si és utilitzada en capes intermitges pot portar el problema de que totes les neurones tinguin com a sortida un valor del mateix signe, i pot convertir amb això a les següents capes en inestables.

Funció tangent hiperbòlica

La funció tangent hiperbòlica (Tanh) és similar a la sigmoidal i té com a recorregut $[-1, 1]$. Matemàticament pot escriure's com

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

$$f'(x) = f(x)^2.$$

Pateix el mateix esvaiment del gradient per valors fora de l'interval $[-3, 3]$, però millora l'estabilitat de la funció logística degut a que permet classificar els seus valors de sortida en propers a 0, molt positius (propers a 1) i molt negatius (propers a -1) i ajuda així a facilitar l'aprenentatge dels següents layers de l'autoencoder.

Funció softmax

La funció softmax és una generalització de la logística per diverses dimensions. Com hem vist anteriorment la funció logística envia elements a un número de l'interval (0,1). Quan aquesta funció és aplicada a tota una capa aleshores aquesta tindrà com a sortida un vector d'elements de l'interval (0,1), sense res que ens garanteixi que la seva suma sigui 1 i per tant pugui ser considerada una probabilitat. El que fa softmax és essencialment

normalitzar aquesta sortida de manera que el vector de sortida sigui un vector que conservi la probabilitat de que un element pertanyi a cada conjunt diferent. L'ièssima sortida d'aquesta funció per a una entrada de $\widehat{n}$ variables ve donada per

$$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{\widehat{n}} e^{x_j}}$$

.

Funció swish

La funció d'activació swish a dia d'avui és la funció d'activació que millora més el rendiment de la fase d'aprenentatge degut a que evita el problema d'esvaiment del gradient dins de la retropagació. Matemàticament és la funció

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

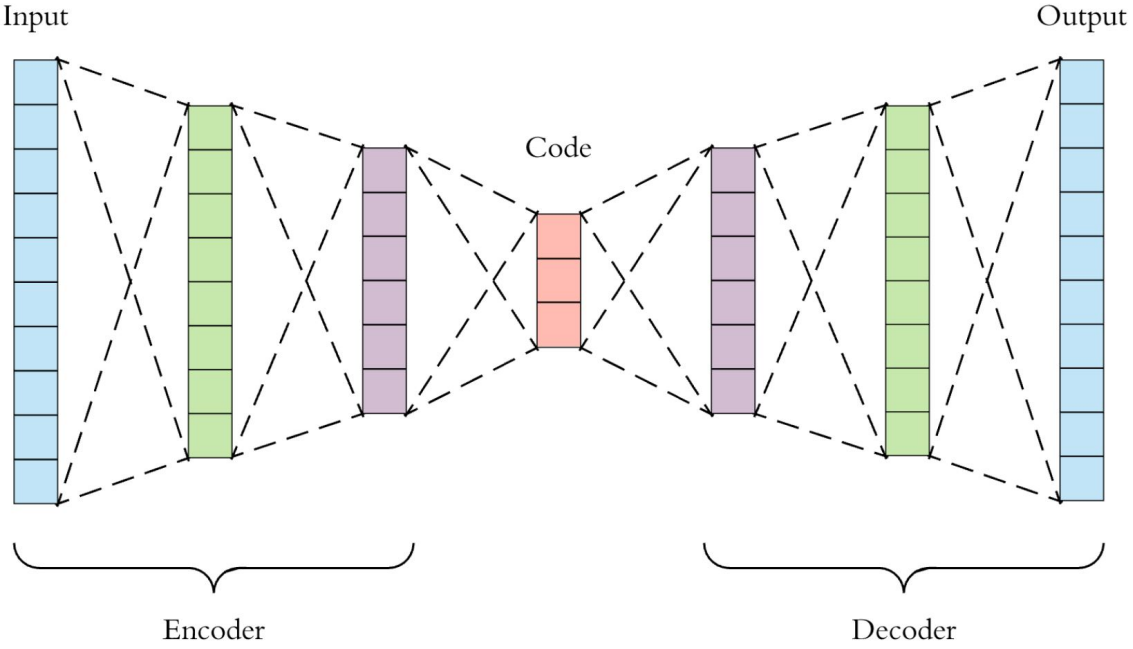
.

Un cop vistes les estructures dels ANN i el seu tractament passem a veure què són els autoencoders.

4 Autoencoders

4.1 Autoencoders regulars

Un autoencoder és un tipus d'ANN capaç d'extreure una representació de la entrada sense supervisió. Aquesta representació s'anomena representació latent. Una particularitat dels autoencoders respecte els altres tipus d'ANN és que per dur a terme el seu procés d'aprenentatge es pren com a valor desitjat la mateixa entrada, és a dir, Y=X. Consten de dues parts: l'encoder, format per un conjunt de capes apilades (l'entrada d'una capa és la sortida de l'anterior) que transformen l'entrada en una representació latent de la mateixa, i el decoder, format també per un conjunt de capes apilades, que intenta regenerar l'entrada a partir de la representació latent. L'estructura queda reflectida a la següent imatge.



Definició 4.1. *Sigui X una base de dades d'entrada, un autoencoder es una tríada $(\phi, \psi, \mathcal{L})$ tal que*

$$\phi \colon X \longrightarrow \mathcal{F}$$

$$\psi \colon \mathcal{F} \longrightarrow X$$

$$\phi, \psi = argmin_{\phi,\psi} \mathcal{L}(X, \psi(\phi(X))$$

on $\mathcal{L}$ és una funció de pèrdua. ϕ és l'encoder, ψ el decoder i $\mathcal{F}$ l'espai d'entitats (Feature Space).

Observació 4.2. Podem considerar tant l'encoder com el decoder com ANNs amb la mateixa funció de pèrdua.

Definició 4.3. *Sigui X l'entrada i $\mathcal{F}$ l'espai d'entitats:*

- *si $dim(\mathcal{F}) < dim(X)$ tractem d'autoencoder sub-complet(UAE)*

- *si $dim(\mathcal{F}) > dim(X)$ tractem d'autoencoder sobre-complet(OAE)*

Els autoencoders serveixen per tot tipus de tasques, i els diferenciarem en dos grans grups: els autoencoders (AE), que seran destinats a tasques principalment de reducció de dimensionalitat i extracció d'entitats i els autoencoders variacionals (VAE) que compliran motius generadors. D'aquests segons en parlarem més al capítol 5.

4.2 Reducció de dimensionalitat i extracció d'entitats

Per aquest apartat farem servir el símbol # per referir-nos a la quantitat de valors numèrics diferents que hem de guardar a la memòria de l'ordinador. Al món computacional és important l'espai que s'ocupa per l'emmagatzament de dades, degut a que aquest espai que s'ocupa a la màquina no pot ser ocupat per altre material i pot arribar inclús a disminuir la velocitat.

Sigui X la base de dades d'entrada, podem dir que aquesta entrada ocupa un espai

$$\#X = m * n,$$

on m és el nombre de variables de cada element i n el nombre d'elements diferents. Aquesta m bé fixada d'inici per la base de dades que estem evaluant, però en canvi la n no és fixa: només cal afegir nous elements a la base de dades per tal de que aquesta augmenti. Per altra banda, suposem que passem la nostra base de dades per un autoencoder ja entrenat adequadament. Primer de tot aconseguim amb l'encoder una reducció de dimensionalitat de la base de dades d'entrada. Aquesta transformació de la base de dades l'anomenarem a partir d'ara $\widehat{X}$. L'espai que ocupa aquesta nova base de dades és de

$$\#\widehat{X} = \widehat{m} * n$$

on $\widehat{m}$ és el nombre de neurones de l'últim layer de l'encoder (la dimensió latent) i per tractar-se d'un UAE $\widehat{m} < m$. A partir d'aquesta representació latent i el decoder podem crear una reconstrucció de X (anomenada X') amb una pèrdua d'informació calculable mitjançant la funció de pèrdua. L'espai que ocupa l'autoencoder a la memòria de la nostra màquina és constant (K, on K és el nombre de paràmetres de l'ANN), i per tant l'espai que ocupa l'autoencoder amb la nova base de dades és

$$\#A = \#\widehat{X} + K = \widehat{m} * n + K < m * n$$

per n prou gran. Un cop reduida aquesta dimensionalitat de la base de dades podem podem observar que tot i ocupar menys espai a la memòria que la base de dades inicial, si l'autoencoder ha après correctament la mèrdua d'informació és petita. De fet, si l'autoencoder està pròpiament entrenat podrem regenerar la base de dades original a partir de menys variables, és a dir, tindrem un nombre menor de variables que retindran la informació de tota la base de dades. Aquest és el conegut com a extracció d'entitats, i analitzant aquestes noves variables podem obtenir nova informació a estudiar.

A part dels autoencoders senzills en aquest grup s'hi troben unes variants d'aquests que per casos concrets són especialment útils. Solen tenir una arquitectura similar als autoencoders, però amb la diferència que durant l'entrenament no només es pren una funció de pèrdua per comparar l'output i l'input, sinó també una altra funció que calcula un

altre tipus de diferència. Aquests són els coneguts com a autoencoders regularitzats, i presentarem a continuació algun d'ells. No serà una llista exhaustiva ja que el món del machine learning és un món incipient, i cada any s'innoven amb noves propostes de models d'autoencoders.

Autoencoders convolutius

Els autoencoders convolutius (Convolutional Autoencoder) són un tipus d'autoencoders que reben com a entrada una imatge i com a sortida donen una aproximació de la imatge inicial. Serveixen per tota una gran varietat d'exercicis, com per exemple per netejar de soroll les imatges, detectar objectes, classificar objectes o inclús afegir color a imatges en blanc i negre, tot això a l'hora que reduir la dimesió de la base de dades. L'encoder s'ocuparà de transformar la imatge inicial en una versió de la mateixa amb pitjor qualitat (menor nombre de pixels i pitjor gamma d'intensitat d'imatge) anomenada downsample. La funció d'aquest tipus d'autoencoder queda fortament determinada per l'estructura interna de les capes i funcions d'activació explicades al capítol 3 i parlarem d'algunes de les seves aplicacions al capítol 6.

Autoencoder dispers

L'autoencoder dispers (Sparse Autoencoder o SAE) és usat per la extracció d'entitats d'una base de dades donada. Té com a objectiu fer que cada element de la base de dades activi no nombre limitat de neurones. Aquesta activació dispersa permet diferenciar la base de dades en diferents subgrups d'elements. Tot i que puguin tenir un nombre elevat de capes i neurones (poden ser OAE) aquest són inclús més ràpids que els UAE degut a que cada element de la base de dades no activa la totalitat de les neurones, aconseguint així un nombre inferior d'operacions a cada iteració i per tant una major velocitat. Degut a aquests fets podem considerar la representació latent per separat, degut a que no tots els elements seran reduits a les mateixes variables latents: podem imaginar-nos que un SAE divideix una base de dades en subgrups, i assigna a cada un d'aquest un autoencoder senzill diferent.

Per minimitzar el nombre de neurones activades s'aplica una penalització a la funció de pèrdua per cada neurona activada. Aquesta penalització afegida a la funció de pèrdua dóna una nova funció a minimitzar:

$$\mathcal{L}' = \mathcal{L} + \Omega(h),$$

on $\Omega(h)$ és la penalització aplicada per neurona activada a la capa que conté la representació latent.

Unes altres funcions de penalització preses són la regularització L1

$$\Omega_{L1}(h) = \lambda \sum_{i=1}^{n} |w_i|$$

amb w_i el vector de pesos associat a la representació latent de l'ièssim element de la base de dades i λ un hiperparàmetre o la funció de regularització L2

$$\Omega_{L2}(h) = \lambda \sum_{i=1}^{n} (w_i)^2.$$

Denoising autoencoder

Un dels problemes dels autoencoders és que pres un nombre de capes, un nombre massa gran de neurones o un nombre massa alt d'iteracions durant l'aprenentatge es pot arribar a obtenir un autoencoder que es converteixi en la identitat, i per tant no aporti cap mena d'informació. Per evitar aquest problema existeixen els denoising autoencoders (DAE). Aquests en lloc de prendre la base de dades com a entrada en prenen una versió corrupta d'aquesta mateixa, per exemple amb soroll gausià. A partir d'aquesta versió corrupta l'autoencoder busca recuperar la base de dades sense soroll. Tenen una estructura similar als autoencoders bàsics però amb una primera capa que afegeix el soroll a l'entrada. Un dels seus avantatges és que la base de dades es torna robusta contra el soroll: degut a que és entrenada amb una base de dades supervisada (i per tant sense soroll) l'autoencoder és capaç de rebre nous elements ara si amb soroll i retornar-ne l'original.

Per últim, tot i que no parlarem d'aquesta funció en aquest treball els DAEs també poden ser usats com a generadors de noves dades de distribució similar.

Autoencoders contractius

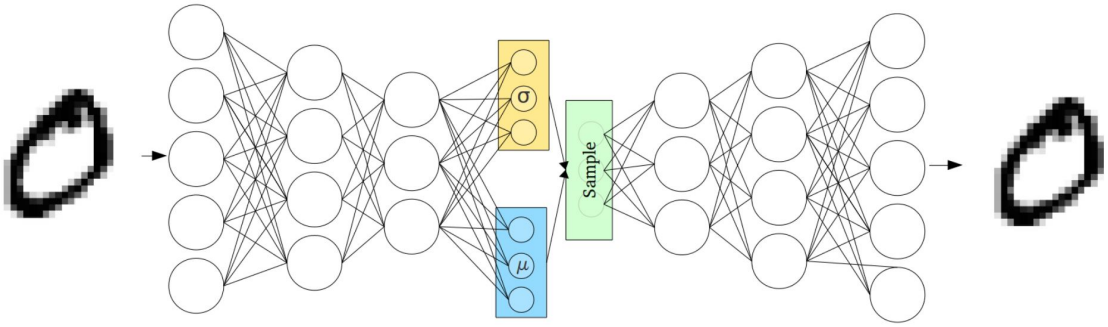
Els autoencoders contractius (CAE) afegeixen una mesura de regularització directament a l'encoder, intentant minimitzar la derivada d'aquest. La funció de penalització pot representar-se com

$$\Omega(h) = \lambda \left\| \frac{\partial f(x)}{\partial x} \right\|_F^2,$$

o el que és el mateix, la funció de penalització és la suma dels elements al quadrat de la matriu Jacobiana de derivades parcials associades amb l'autoencoder al quadrat.

5 Autoencoders Variacionals

Els autoencoders variacionals (Variationals Autoencoders o VAE) són una familia d'autoencoders que tenen com a objectiu generar més dades amb una estructura similar a la de les dades sobre el qual han estat entrenats. Per fer-ho intenten inferir una combinació de distribucions gausianes a les dades d'entrada que queda reservada a la representació latent en forma de paràmetres (la mitjana i la desviació estàndard).



Suposem que la base de dades és una mostra aleatòria extreta a partir d'una funció distribució desconeguda p^* (la distribució real). Aquesta funció real usualment és massa complicada per calcular-la, de manera que el nostre objectiu principal és aproximar aquesta amb un model estadístic $p_\theta(x)$ de paràmetres θ:

$$x \sim p_\theta(x).$$

Durant aquest apartat el procés d'aprenentatge serà el procés d'optimització de paràmetres θ tals que la funció distribució donada pel model $p_\theta(x)$ aproximi de la millor manera possible la funció distribució real, és a dir:

$$p_\theta(x) \approx p^*(x)$$

Trobar una funció distribució complint aquestes característiques és a vegades possible, però amb els autoencoders no busquem només eficàçia, sinó també eficiència. Parlem breument de la inferència bayesiana i veurem com aplicar-la al nostre problema.

La inferència bayesiana sorgeix a arrel del Teorema de Bayes. És un mètode d'inferència estadística on s'utilitza el Teorema de Bayes per actualitzar el valor de la probabilitat d'una hipòtesis a mesura que es tenen més dades. Siguin H unes hipòtesis i D unes dades, la inferència bayesiana es pot plasmar com

$$p(H|D) = \frac{p(D|H)p(H)}{p(D)} \text{ on}$$

- $p(H)$ és la funcidistribució de la probabilitat a priori de les hipòtesis que estima una probabilitat abans d'analitzar les dades.
- $p(H|D)$ és la funció distribució de la probabilitat posterior, que ens dona un valor després d'analitzar les dades. És el valor que volem calcular amb el mètode.
- $p(D|H)$ és la funció distribució de la versemblança. Fixada una hipòtesis dona la compatibilitat de les dades amb la hipòtesis donada.

Degut a que els VAE mantenen l'estructura dels autoencoders parlats anteriorment aquests estan formats per un encoder, que dona una representació latent de la base de dades i un decoder que a partir d'aquesta és capaç de reconstruir la base de dades. És a dir, donada $p_\theta(z)$ la funció distribució de la representació latent podem dir que trobar un bon encoder fa referència a trobar un bon model estadístic per

$$p_\theta(z|x)$$

i el problema del decoder es redueix a trobar un bon model estadístic per

$$p_\theta(x|z).$$

El càlcul del decoder és directe:

$$p_\theta(x|z) = \frac{p_\theta(x,z)}{p_\theta(z)}$$

on ambdues seran conegudes una vegada executat el procés d'aprenentatge.

Ara bé, el càlcul pertanyent a l'encoder no és trivial degut a que el càlcul de $p_\theta(x)$ pot ser extremadament complicat y lent (amb mètodes com per exemple la cadena de Markov de Monte Carlo). Per tant buscarem una funció distribució $q_\phi(x|z)$ tal que

$$q_\phi(x|z) \approx p_\theta(x|z).$$

Per veure que aquesta aproximació és prou bona ens cal parlar d'una mesura de distància entre funcions distribucions.

Definició 5.1. ***Divergència de Kullback-Leibler(D_{KL})***
La divergència de Kullback-Leibler és una distància estadística, és a dir, calcula com és de diferent una distribució de probabilitat d'una altra. Siguin P i Q distribucions diferents de variables aleatòries contínues, aleshores

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)}\, dx \ .$$

El nostre interès resideix per tant en minimitzar la divergència de Kullback-Leibler entre $q_\phi(z|x)$ i $p_\theta(z|x)$. Ara

$$\begin{aligned}
D_{KL}(q_\phi(z|x)||p_\theta(z|x) &= \int q_\phi(z|x) \log \frac{q_\phi(z|x)}{p_\theta(z|x)}\, dz \\
&= \int q_\phi(z|x) \log \frac{q_\phi(z|x) p_\theta(x)}{p_\theta(z,x)}\, dz \\
&= \int q_\phi(z|x) (\log p_\theta(x) + \log \frac{q_\phi(z|x)}{p_\theta(z,x)})\, dz, \\
&= \log p_\theta(x) + \int q_\phi(z|x) \log \frac{q_\phi(z|X)}{p_\theta(z,x)}\, dz, \\
&= \log p_\theta(x) + \int q_\phi(z|x) \log \frac{q_\phi(z|x)}{p_\theta(x|z) p_\theta(z)}\, dz,
\end{aligned}$$

Per minimitzar respecte els paràmetres ϕ cal minimitzar el segon terme del sumand ja que el primer no depèn de ϕ i per tant és constant. Aquest terme es conegut com a ELBO (Evidence Lower Bound).

$$\begin{aligned} L = \int q_\phi(z|x) \log \frac{q_\phi(z|x)}{p_\theta(x|z)p_\theta(z)} &= E_{z\sim q_\phi(z|x)} \log(\frac{q_\phi(z|x)}{p_\theta(x|z)p_\theta(z)}) \\ &= E_{z\sim q_\phi(z|x)}(\log q_\phi(z|x) - \log p_\theta(x|z) + \log p_\theta(z)) \end{aligned}$$

i minimitzar aquesta última expresió equival a maximitzar l'oposada, és a dir maximitzar

$$-L = E_{z\sim q_\phi(z|X)}(\log q_\phi(z|X) - \log p_\theta(X|z) + \log p_\theta(z))$$

respecte θ i ϕ. A més L és fàcilment computable degut a que $p_\theta(X|z)$, $p_\theta(z)$ i $q_\phi(z|X)$ són conegudes.

Un cop parlat del cas general, encara podem afegir noves hipòtesis al model. Primerament, podem prendre la funció distribució de versemblança com una gaussiana i $p_\theta(z)$ com els paràmetres d'aquesta gaussiana (variància i mitjana) per cada una de les dimensions de l'espai latent triat, essent així la funció distribució conjunta una mixtura de gaussianes. Aquest serà el cas triat per ser més eficient tot i que es podrien prendre altres formes (com per exemple funcions distribucions Bernoulli).

Usualment es pren com a funció de penalització

$$\sum_{j=1}^{s} D_{KL}(\rho||\widehat{\rho}_j)$$

essent s el nombre variables latents, ρ és el paràmetre de dispersió (un valor petit, usualment $\rho = 0.05$) que pren una distribució Bernoulli i $\widehat{\rho}_j$ la mitjana de valors rebuts per la j-èssima neurona del layer de la representació latent, és a dir

$$\widehat{\rho}_j = \sum_{i=1}^{n} z_i^j$$

amb z_i^j els valors rebuts per la j-èssima neurona del layer latent de l'i-èssim element de la base de dades. Es pot provar que si ρ i $\widehat{\rho}_j$ són els parametres d'una distribució Bernoulli aleshores

$$D_{KL}(\rho||\widehat{\rho}_j) = \sum_{j=1}^{s} \rho \log(\frac{\rho}{\widehat{\rho}_j}) + (1-\rho) \log(\frac{1-\rho}{1-\widehat{\rho}_j})$$

Truc de reparametrització

Per introduir aquest truc cal fer una observació prèvia respecte la funció de pèrdua presa. Hem vist anteriorment que quan fem ús de la retropagació hem de calcular el gradient de la funció de pèrdua feta servir, (en el nostre cas una combinació lineal de l'ELBO i la funció triada per l'autoencoder, com per exemple el MSE), respecte els paràmetres desitjats. Degut a que és combinació lineal i sabem que les funcions de pèrdua de l'autoencoder té un gradient calculable cal veure si el gradient de l'ELBO també ho és En el nostre cas

tenim dos paràmetres diferents sobre els que minimitzar: sobre ϕ i sobre θ. Veiem que minimitzar sobre θ és senzill

$$\nabla_\theta E_{z\sim q_\phi(z|X)}(\log q_\phi(z|X) - \log p_\theta(X|z) + \log p_\theta(z)) \quad =$$

$$E_{z\sim q_\phi(z|X)}\nabla_\theta(\log q_\phi(z|X) - \log p_\theta(X|z) + \log p_\theta(z)) \quad = E_{z\sim q_\phi(z|X)}\nabla_\theta(\log p_\theta(X|z) + \log p_\theta(z))$$

i podem aproximar aquesta derivada amb les dades que tenim al problema. Per altra banda minimitzar respecte ϕ és diferent ja que no podem invertir l'esperanç (una integral) amb el jacobià per ser sobre la mateixa variable. És per aquesta raó que cal efectuar el truc de reparametrització:

$$z \sim q_\phi(z|X) = \mu + \sigma \odot \epsilon$$

on μ i σ són els vectors de mitjanes i desviacions estàndard respectivament de $q_\phi(z|X)$, ϵ és un vector aleatori de components seguint una distribució gausiana de mitjana 0 i desviació 1 i $\odot$ el producte per elements.

Teorema 5.2. *Sigui g(X) una funció d'una variable aleatòria X amb $X \sim f_X$ aleshores*

$$E(g(X)) = \int_{-\infty}^{\infty} g(x) f_X(x)\, dx$$

Aquest teorema ens permet escriure per funcions qualsevols

$$\nabla_\theta E_{p_\theta(x)}(f(X)) = \nabla_\theta f(x) p_\theta(x)\, dx \& = \nabla_\theta \int f(g(\epsilon,\theta)) p(\epsilon)\, d\epsilon \; = \int \nabla_\theta f(g(\epsilon,\theta)) p(\epsilon)\, d\epsilon \; =$$

$$E_{p_\epsilon}(\nabla_\theta f(g(\epsilon,\theta)) p(\epsilon))$$

Vist aquest resultat i prenent

$$p(\epsilon) \sim \mathcal{N}(0,1)$$
$$z = g(\phi, x, \epsilon)$$

només cal reescriure la funció depenalització com

$$L = E_{z\sim q_\phi(z|X)}(\log q_\phi(z|X) - \log p_\theta(X|z) + \log p_\theta(z)) \quad = E_{p(\epsilon)}(\log q_\phi(z|X) - \log p_\theta(X|z) + \log p_\theta(z))$$

i derivant respecte ϕ

$$\nabla_\phi L = \nabla_\phi E_{p(\epsilon)}(\log q_\phi(z|X) - \log p_\theta(X|z) + \log p_\theta(z)) = E_{p(\epsilon)}\nabla_\phi(\log q_\phi(z|X) - \log p_\theta(X|z) + \log p_\theta(z))$$

i novament aquestes són calculables a partir de la base de dades. Per últim cal mencionar que quan apliquem un VAE no prendrem com a representació latent la mitjana aritmètica μ i la desviació estàndard σ, sinó μ i $log(\sigma)$. Degut a que la variància és sempre positiva i $log(\sigma)$ no, aquesta segona millora notablement l'entrenament del VAE.

6 Aplicació dels autoencoders

Una vegada ja definits els diferents tipus d'autoencoders i les funcionalitats que poden tenir cal parlar de quina manera aplicar-los, per aconseguir una major efectivitat. Parlarem primerament dels diferents tipus de funcions d'activació que podem trobar i en una segona instància parlarem dels tipus de layers que poden formar els autoencoders. Per últim afegirem un breu apartat dedicat als VAE, on parlarem de trucs fets servir per millorar la velocitat dels programes.

6.1 Tipus de capes dels autoencoders

En capítols anteriors hem parlat d'alguns exemples de capes, però no n'hem mirat la part matemàtica. És important fer una distincions segons sigui la base de dades. Per una banda tenim les bases de dades numerals. Aquestes ens aporten dades numèriques dels elements a estudiar o qualitatives que podem representar amb numerals (per exemple, si tractessim el color de cabell com a variable podem assignar a cada color un valor numèric) i queden representades mitjançant un vector de dimensió fixada. Per altra banda tenim les bases de dades composades per imatges. Les imatges en blanc i negre queden representades per una matriu amb dimensió igual als píxels de la imatge, on cada element ve donat per un número enter entre 0 i 255, on aquest valor representa la intensitat de la imatge (0 és la total absència d'intensitat, negre i 255 blanc). Les imatges a color queden representades en una matriu amb un mateix nombre de files i columnes com pixels té la imatge i cada element ve donat per un vector de tres elements entre 0 i 255. Aquests marquen la intensitat de vermell, verd i blau de la imatge (sistema RGB). En ambdos casos usualmen es divideixen entre 255 aquestes dades, així fent que els valors quedan a l'interval [0,1], i podent tractar-los com si fossin una probabilitat. En funció del tipus de base de dades es faran servir unes families de capes o unes altres.

Començarem parlant dels tipus de capes aplicable a bases de dades totalment numèriques i més endevant parlarem de les aplicades a bases de dades formades per imatges.

Capes per bases de dades numèriques

Capa densa

Com hem dit abans les neurones d'una capa d'aquest tipus està connectada a totes i cada una de les neurones de la capa anterior. Els pesos d'aquesta queden reservats a una matriu de n files i m columnes on n és el nombre de neurones de la capa i m el nombre de neurones de la anterior. És el tipus de capa bàsic als autoencoders i és la que conserva la major part de la informació durant la fase d'aprenentatge.

Capa de normalització per lots

La capa de normalització per lots (o batch normalization layer) és una capa que retorna la entrada amb dades normalitzades. Aquesta capa no té cap paràmetre a entrenar durant la fase d'aprenentatge, sinó que es tracta simplement d'una reparametrització. Té la característica de que es comporta diferent a la fase d'aprenentatge i una vegada ja entrenat l'autoencoder.

Sigui $\Theta_{(i)}$ el conjunt format pel lot fet servir a l'i-èssima iteració d'una donada època de la fase d'aprenentatge, la sortida d'aquesta capa a aquest iteració és

$$o(i) = \frac{\gamma * (\Theta_{(i)} - \overline{\Theta}_{(i)})}{Var(\Theta_{(i)}) + \epsilon} + \beta$$

on $\overline{\Theta}_{(i)}$ és la mitjana aritmètica del lot, $Var(\Theta_{(i)})$ la variància mostral, i γ, ϵ i β hiperparàmetres de la capa. Presos $\gamma = 1$ i $\epsilon = \beta = 0$ aleshores aquesta normalització és la estàndard.

Per altra banda, durant la inferència i mantenint la notació anterior la sortida durant l'ièssima iteració ve donada per

$$o(i) = \frac{\gamma * (\Theta_{(i)} - \widehat{\Theta}_{(i)})}{\sqrt{\widehat{Var(\Theta_{(i)})} + \epsilon}} + \beta,$$

on $\widehat{\Theta}_{(i)}$ i $\widehat{Var(\Theta_{(i)})}$ són variables inicialitzades a 0 que durant cada iteració s'actualitza donant un valor aproximat de la mitjana i variància dels elements ja normalitzats amb la capa. Aquestes variables són actualtizades mitjançant les següents fòrmules:

$$\widehat{\Theta}_{(i)} = \widehat{\Theta}_{(i-1)} * m + \overline{\Theta_{(i)}}) * (1 - m)$$
$$\widehat{Var(\Theta_{(i)})} = \widehat{Var(\Theta_{(i-1)})} * m + \overline{Var(\Theta_{(i)}} * (1 - m),$$

amb $m \in (0, 1)$ un hiperparàmetre anomenat moment.

Aquesta capa es fa servir o bé en una primera capa de l'autoencoder(tenir unes dades amb distribució normal facilita el tractament d'aquestes) o bé en capes intermitges, on actua com a punt de control d'estabilitat de l'autoencoder. En autoencoders formats per moltes capes ajuda a que l'interval d'hiperparàmetres a triar vàlids per al correcte funcionament de la xarxa neuronal sigui més ampli.

Capa dropout

Aquesta capa és una capa auxiliar que anul·la aleatòriament algunes neurones d'entra i reescala la resta, de manera que la suma de neurones en valor absolut es manté constant. Aquesta capa només actua durant la fase d'aprenentatge i es fa servir per evitar l'overfitting, és a dir, es fa servir per evitar que l'autoencoder aprengui entitats generades per anomalies. És especialment útil en autoencoders amb una base de dades petita, ja que aquests són els més vulnerables a irregularitats.

Capes per a bases de dades d'imatges

Abans de parlar de families de layers diverses cal introduir un seguit de conceptes.

Definició 6.1. *Un sistema lineal és un sistema tal que per qualsevol $\mathcal{S}$ operador, qualsevols parell d'elements d'entrada $f(x_1)$, $f(x_2)$ i les seves corresponents sortides $g(x_1) = \mathcal{S}\{f(x_1)\}$, $g(x_2) = \mathcal{S}\{f(x_2)\}$ es compleix la següent igualtat:*

$$\mathcal{S}\{\alpha f(x_1) + \beta f(x_2)\} = \alpha g(x_1) + \beta g(x_2)$$

Aquest principi ens permet desglossar l'entrada en petites peces, passar-la pel sistema i veure com afecta a cada peça individualment i finalment ajuntar-les en una sortida única final.

Definició 6.2. *Un sistema lineal és invariant per translacions (lineal shift invariant-LSI) si per qualsevol operador $\mathcal{S}$ i funció, $g(x) = \mathcal{S}\{f(x)\}$ es compleix*

$$\mathcal{S}\{f(x - x_0)\} = g(x - x_0)$$

A continuació donarem una definició informal per a la Delta de Dirac **Delta de Dirac** Sigui Ω un subconjunt qualsevol real, Delta de Dirac (o unitat d'impuls) es defineix com:

$$\delta(x) \simeq \begin{cases} +\infty & si \quad x = 0 \\ 0 & si \quad x \neq 0 \end{cases}$$

$$\int_\Omega \delta(x)\, dx = \begin{cases} 1 & si \quad 0 \in \Omega \\ 0 & si \quad 0 \notin \Omega \end{cases}$$

$$\int_{-\infty}^{+\infty} f(x)\delta(x)\, dx = f(0).$$

per tota funció $f(x)$ integrable.

Observació 6.3. Fent el canvi de variable $y = x - x_0$, $dy = dx$ a

$$\int_{-\infty}^{+\infty} f(x)\delta(x - x_0)\, dx$$

és directe veure que

$$\int_{-\infty}^{+\infty} f(y + x_0)\delta(y)\, dy = f(x_0).$$

Hem vist que en un sistema LSI la sortida ve definida per l'entrada i per l'operador. No obstant això quan el sistema ve alterat per impulsos diem que la sortida és la resposta impulsiva del sistema o kernel i ho denotem com

$$\mathcal{S}\{\delta(x)\} = h(x)$$
$$\mathcal{S}\{\delta(x - x_0)\} = h(x - x_0)$$

Un cop introduits aquests termes tornem al nostre camp. En diversos papers (en matèria més aviat física que no treballarem a aquest TFG) s'ha vist que es pot considerar l'òptica i el tractament d'imatges com a LSI. Per la última propietat vista, donada una entrada $f(x)$, podem reescriure-la com

$$f(x) = \int_{-\infty}^{+\infty} f(x)\delta(x - \alpha)\, d\alpha\ ,$$

i aplicant a aquesta transformació un operador $\mathcal{S}$:

$$\begin{aligned} \mathcal{S}\{f(x)\} = \mathcal{S}\left\{ \int_{-\infty}^{+\infty} f(\alpha)\delta(x - \alpha)\, d\alpha \right\} &= \int_{-\infty}^{+\infty} f(\alpha)\mathcal{S}\{\delta(x - \alpha)\}\, d\alpha \\ &= \int_{-\infty}^{+\infty} f(\alpha)h(x - \alpha)\, d\alpha \end{aligned} .$$

Aquesta darrera integral és coneguda com a integral de convolució.

En el cas de les imatges treballem a $\mathbb{R}^2$ així que ho reformularem a

$$g(x,y) = \mathcal{S}\{f(x,y)\} = \int_{-\infty}^{+\infty} f(\alpha,\beta)\delta(x-\alpha, y-\beta)\, d\alpha d\beta \ .$$

Notació D'ara en endevant aquesta operació de convolució la denotarem com

$$g(x,y) = f(x,y) * h(x,y)$$

Proposició 6.4. *Siguin f, g i h operadors d'un sistema LSI, aleshores es compleixen les següents propietats:*

1) ***Propietat associativa:*** $f * (g * h) = (f * g) * h$

2) ***Propietat commutativa:*** $f * g = g * f$

3) ***Distributiva respecte la suma:*** $f * (g + h) = f * g + f * h$

4) ***Commuta amb la diferenciació*** $(f * g)' = f' * g = f * g'$

Pel cas discret Delta de Kronecker. Denotarem la resposta impulsiva com H(x,y) tal que

$$H(x,y) = \mathcal{S}\{\delta(x,y)\} \text{ amb } \begin{cases} \delta(x,y) = 1 & si (x,y) = (0,0) \\ \delta(x,y) = 0 & si (x,y) \neq (0,0) \end{cases}$$

Aleshores l'operació de convolució ve donada per

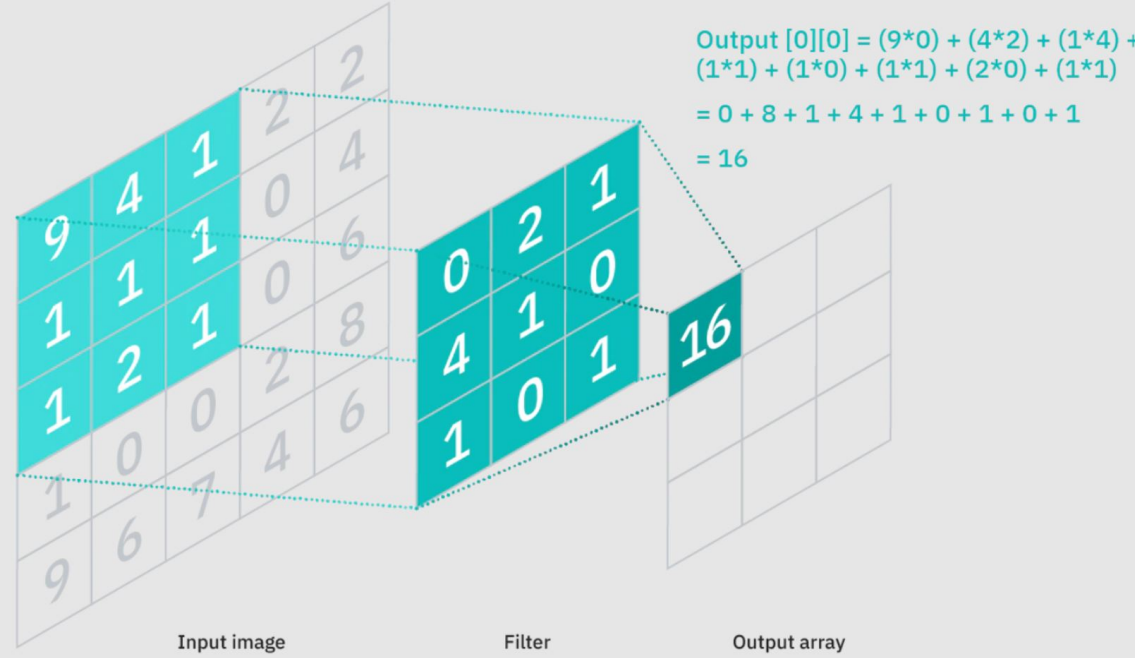
$$G(x,y) = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} I(x,y)H(i-x, j-y)$$

Geomètricament és el producte de dos senyals, un dels quals està rotat 180 graus. A la pràctica la resposta impulsiva $H(x,y)$ és una matriu de dimensió i rep el nom de filtre o kernel.

Quan apliquem la convolució aleshores la sortida conté a cada cel·la un valor computat amb els valors de l'entorn. Per aconseguir un output complet llisquem el filtre per tota la imatge i guardem els valors a una matriu diferent (no es pot fer servir la mateixa degut a que si així fos es propagaria un error per tota la matriu).

Capa convolutiva

Aquesta tipologia de capa aplica la operació de convolució a una imatge. Primerament designem el nombre de convolucions diferents que farem a la imatge i la mida del filtre. Cal destacar que cada filtre és diferent i actua sobre un únic entorn de la imatge i conté tants pesos com mida el filtre. És la capa bàsica dels CNN.



La seva contrapart (feta servir al decoder) és la capa deconvolutiva o capa de la convolució transposada. Aquesta capa el que fa és ampliar la qualitat de la imatge afegint píxels. Mentre que la capa de convolució extreu informació i la condensa en els mínims píxels possibles la convolució transposada regenera la imatge a partir de la informació extreta. Aquesta capa afegeix 0 a l'entrada fins a una dimensió prefixada (aquests 0 s'intercalen entre els valors o es posen al voltant dels valors segons la configuració desitjada) i després actua com una capa de convolució usual.

Capa de mitjana local Aquesta capa fa funció de reparametrització. Pren com a filtre una matriu on tots els valors són iguals i sumen 1. Fa la mitjana de les intensitats de la imatge en un entorn d'un donat píxel. Serveix per anul·lar el soroll, però provoca una pèrdua de definició arreu de tota la imatge, podent fer desaparèixer detalls petits i vèrtexos. Com més gran és el filtre més detalls es perden, però més uniformes poden ser les regions: pot fer-se servir pel reconeixement i localització d'elements.

Capa de màxim local Aquesta capa ajuda a conservar els elements més grans o de més importància (més lluminosos o de colors més forts). Essencialment el que fa és reduir la dimensió de la imatge, i per tant la qualitat. Si es pren una mida de filtre de 2x2 aleshores per el valor màxim dels quatre píxels i ho transforma en un únic píxel amb la intensitat del valor màxim.

Capa d'upsampling Aquesta capa és la contrapart de la capa de màxim local. Pren l'entrada i duplica els valors en ambdos eixos segons triem. Per exemple, la matriu

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

amb paràmetres $(2, 3)$ es transformaria a

$$\begin{pmatrix} 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \\ 1 & 1 & 2 & 2 \\ 3 & 3 & 4 & 4 \\ 3 & 3 & 4 & 4 \\ 3 & 3 & 4 & 4 \end{pmatrix}.$$

6.2 Exemples pràctics

Durant aquest subapartat parlarem dels exemples pràctics treballats i en veurem alguns resultats.

Reducció de dimensionalitat

Atletes d'heptatlon

Un dels avantatges de PCA respecte als autoencoders és que la seva interpretació és més directe és millor per bases de dades petites. En aquest primer cas farem servir una base de dades amb 20 atletes i els seus resultats a una heptatló.

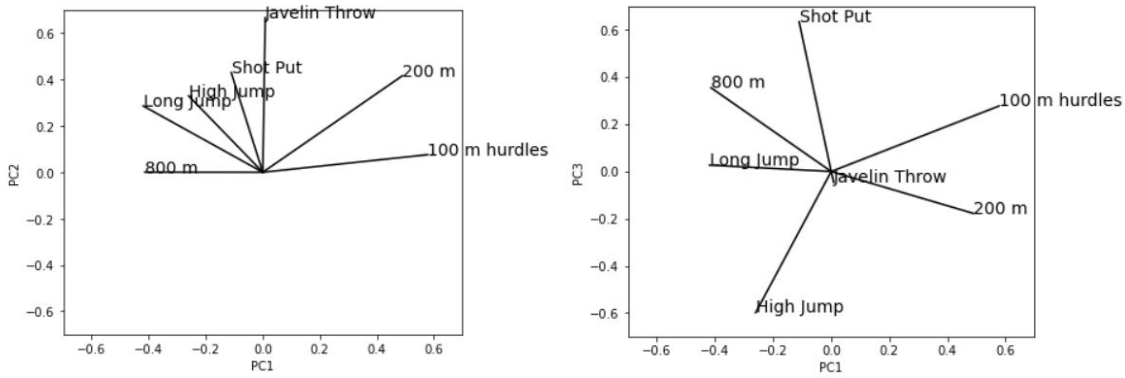
	Name	100 m hurdles	High Jump	Shot Put	200 m	Long Jump	Javelin Throw	800 m	Position
0	Nafissatou Thiam	13.54	1.92	14.82	24.90	6.60	54.68	135.98	1
1	Anouk Vetter	13.09	1.80	15.29	23.81	6.47	51.20	138.72	2
2	Emma Oosterwegel	13.36	1.80	13.28	24.25	6.29	54.60	131.09	3
3	Noor Vidts	13.17	1.83	14.33	23.70	6.32	41.80	129.05	4
4	Kendell Williams	12.97	1.80	12.41	24.00	6.57	48.78	136.91	5

Degut a que en una heptatló està formada per 7 proves diferents és interessant veure com alguns atletes destaquen en unes proves o en altres. En aquesta tasca ens pot ajudar PCA.

Primerament hem de netejar les dades: el nom dels participants i la posició no ens serà útil en aquest cas, així que podem no considerar-les durant la resta de l'exercici.

El segon pas dut a terme consisteix a estandaritzar les dades. Si aquest pas s'ometÃ©s les relacions trobades no serien fiables: per exemple la prova de 800 m prendria més importància que la de salt d'altura (High Jump) degut a que la magnitud dels resultats és superior.

Apliquem un llindar a la variabilitat aportada per cada PC del 15% de la variabilitat total del problema, el que ens dona un valor de PCs a fer servir de 3, mantenint aproximadament el 70% de la variabilitat total. Enfrontant gràficament la PC1 amb la PC2 i la PC1 amb la PC3 obtenim les gràfiques següents:



Observant les gràfiques podem pensar que la PC1 representa la potència en els seus valors més negatius i la velocitat màxima en els valors positius. Això concorda amb que les proves de salt prenen valors negatius mentre que el llançament de javelina pren un valor nul. Aquesta darrera prova pren un valor màxim a la PC2 i nul també a la PC3, així que podem dir que PC2 representa la potència del tren superior.

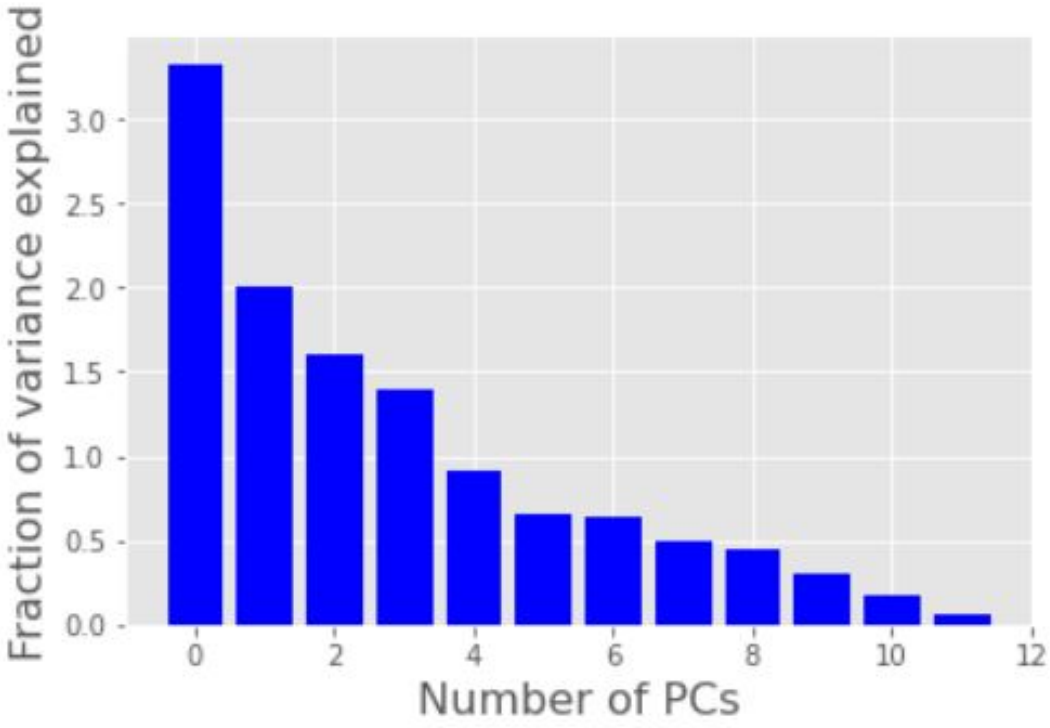
Per altra banda, per aquesta base de dades la tria d'un autoencoder no és útil: durant la fase d'aprenentatge hi ha overfitting, el que fa que l'error al training set sigui gran.

Varietat de vi

En un segon programa tractarem una base de dades amb 1500 varietats de vins diferents amb algunes de les seves qualitats numèriques.

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

Primer de tot dividim la nostra base de dades en el conjunt d'aprenentatge (1000 elements) i en cinc conjunts de tests (100 elements cada). Començarem fent PCA sobre el conjunt d'aprenentatge. En buscar les PCs veiem la següent gràfica de barres amb la variabilitat aportada amb cada PC.



Es pot observar com hi ha dos canvis de pendent: un primer entre la PC4 i la PC5 i un segon entre la PC5 i la PC6. Les variabilitats totals explicades mantingudes per 4 i 5 PCs són 75% i 80% respectivament, aportant la cinquena PC un 5% de variabilitat, el qual és petit. Per aquesta raó durem a terme el PCA amb 4 PC. Aquesta tria redueix la dimensió de la base de dades en un 60% (de 12 variables diferents a només 4).

Per altra banda

$$Y = XV^T$$

on X és el conjunt de test i V^T la matriu de canvi de base.

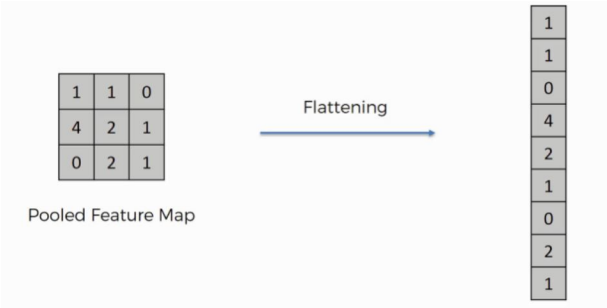
Per altra banda preparem un autoencoder amb dimensió latent 3 i podem comprovar com no només fa que la base de dades ocupi menys espai (la dimensió latent és més petita que el nombre de PCs triades), sinó que té un error menor (amb MSE). A més a més com més gran és el percentatge de conjunt d'entrenament respecte l'error disminueix en major mesura a l'autoencoder (disminuint un 37% mentre que el de PCA només disminueix un 22%).

Generació de dades mitjançant VAEs

A aquest apartat parlarem de com generar dades mitjançant els autoencoders variacionals (VAE). Utilitzarem la base de dades del mnist. Aquesta està formada per imatges electròniques de $28x28$ píxels, on cada píxel conté un valor numèric entre 0 i 255 (inclosos). Aquest valor representa la intensitat de la llum al píxel: el 0 és el negre absolut i el 255 el blanc. Aquestes imatges representen números entre 0 i 9 escrits a mà per persones.

El primer que farem serà dividir els valors de la matriu entre 255, així els nous valors estaran entre 0 i 1, poguen ésser considerats com la probabilitat de que un píxel donat estigui il·luminat. Hi ha diferents maneres d'afrontar aquest problema: des del punt de vista purament unidimensional usant capes denses, des del punt de vista bidimensional fent servir capes convolutives o un híbrid de les dues. Aquesta darrera opció és la recomanable, agafant la millor part de l'unidimensional i del bidimensional.

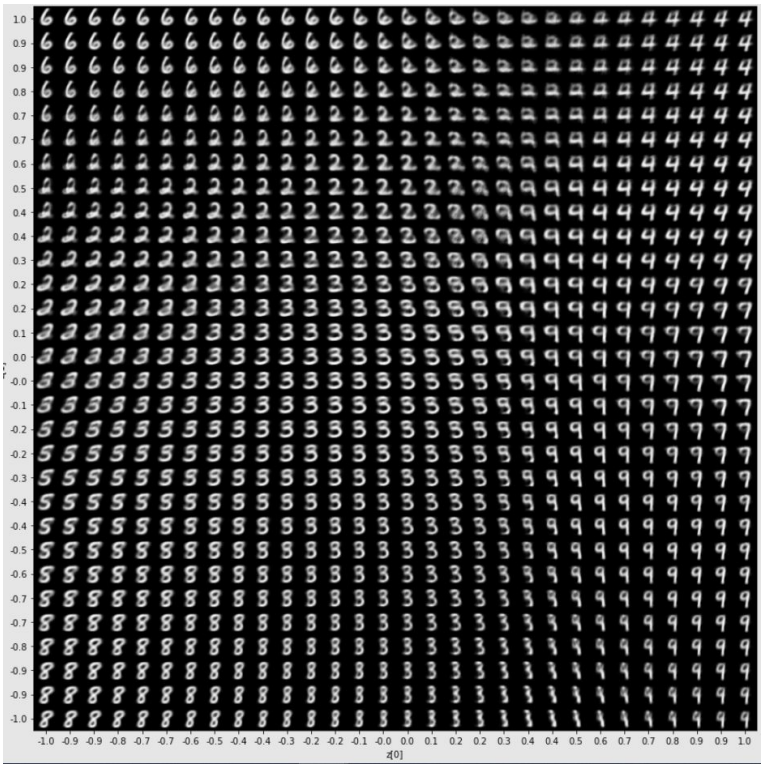
Primerament crearem l'encoder amb un seguit de capes convolutives, seguides d'una capa de reparametrització aplanadora (flatten layer). Aquesta capa és l'encarregada de moure les neurones de manera que deixin de estar en forma de matriu i les posa en la forma de la capa densa sense cambiar-ne els valors.



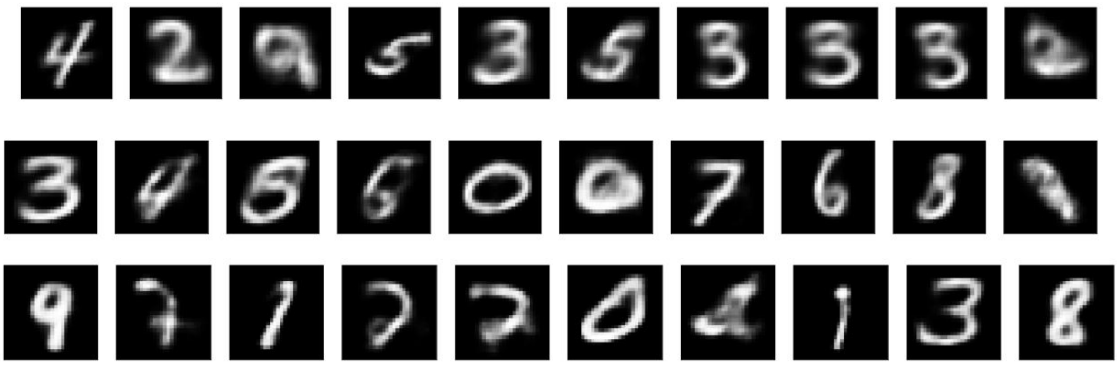
La sortida de l'encoder seran dos vectors de dimensió latent fixada, un contenint les mitjanes de les distribucions gaussianes triades i un altre amb el logaritme de la desviació estàndard. Els algoritmes amb aquest logaritmes són més precissos que la desviació estàndard degut a que estem prenent dades seguint una distribució gaussiana estàndard, i per tant valors entre 0 i 1. Recordem que treballem amb ordinadors, els quals cometen errors d'arrodoniment i allà on són més rellevants és al voltant del 0, justament on el logaritme pren valors més distints.

Tot seguit hi apliquem el truc de reparametrització, permet que l'autoencoder aprengui de distribució i per últim apliquem el decoder, que busca reconstruir les dades a partir dels paràmetres de la gaussiana trobada amb el truc de reparametrització. Abans de passar als resultats passem de la funció de pèrdua.

Primerament veiem el cas de dimensió latent 2, degut a que el podem plasmar al pla bidimensional, és a dir a un paper.



Veiem alguns exemples de dades generades una vegada acabada la fase d'aprenentatge. Aquestes imatges no són reconstrucció de cap altra, sinó creades amb dades aleatòries de dimensió latent 2, 4 i 8 respectivament.



Com podem veure en aquest cas degut a la simplicitat del traç és recomanable prendre valors de dimensió latent petita, ja que la més gran és aquesta més confuses es tornen les imatges.

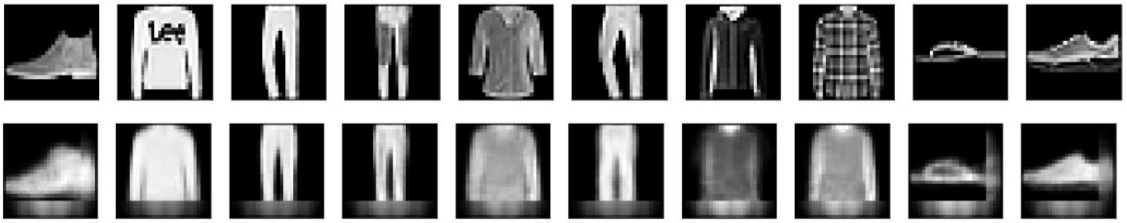
Reducció de dimensionalitat amb autoencoders

A aquest apartat parlarem de la reducció de dimensionalitat amb autoencoders regulars. La base de dades presa està formada per imatges electròniques de mida 28×28 amb valors entre 0 i 255 (inclosos) que representen les imatges electròniques de peces de roba.

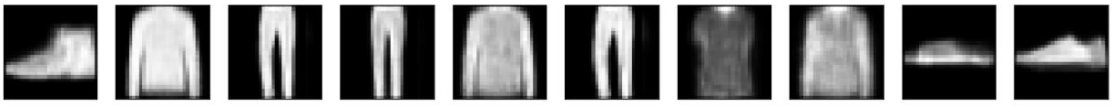
L'autoencoder creat pren una dimensió de entrada de 28×28 i ho redueix fins a una dimensió latent 2. La base de dades inicialment estaba formada per 60.000 imatges electròniques amb 784 variables cada una, ocupant un espai total a la memòria de 47.040.000 de valors numèrics diferents.

Per altra banda, una vegada ja entrenat l'autoencoder conté un total de 77.306 paràmetres i una vegada transformades les dades amb l'encoder només ocuper l'espai per 2 valors numèrics. Sumant tot aquest valors amb aquesta base de dades l'autoencoder ocupa un espai a la memòria equivalent a 119.306 valors numèrics, una quantitat aproximadament 238 vegades més petita que la de la base de dades inicial. A més la nova base de dades obtinguda amb l'encoder només incrementa l'espai utilitzat per noves dades en 2 per cada imatge, mentre que a la inicial s'incrementa en 784. Fent tendir a l'infinit el nombre de imatges de la base de dades acabaríem tenint una nova base de dades 392 vegades més petita.

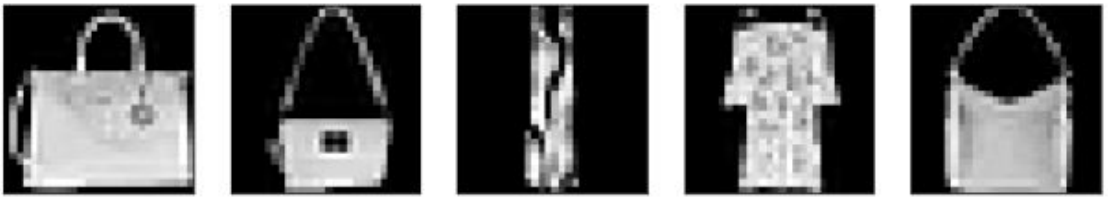
Veiem els resultats obtinguts. La primera fila conté les imatges inicials i la segona les reconstruccions.

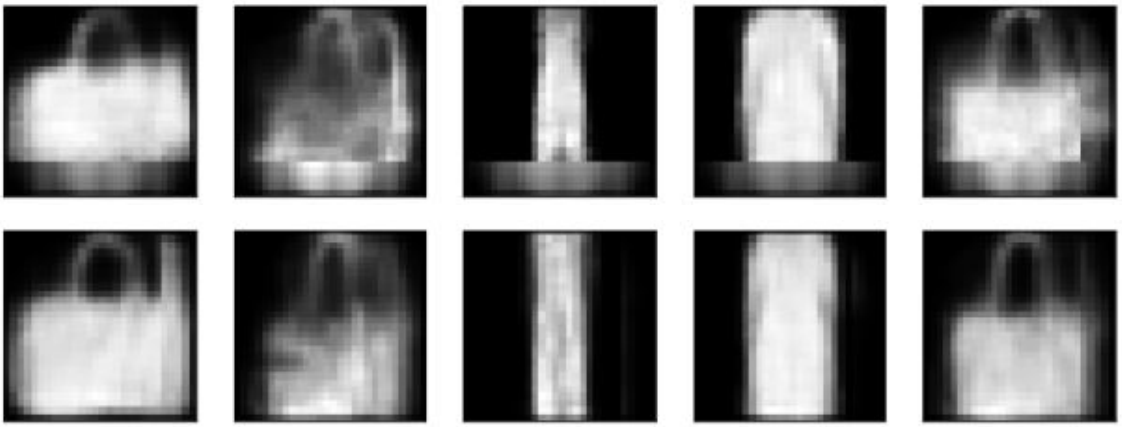


Es perd detall a les imatges, però les reconstruccions permeten saber a quina peça de roba feien referència, inclus a la tonalitat de color. Comparem-ho amb un autoencoder similar amb dimensió latent 4.

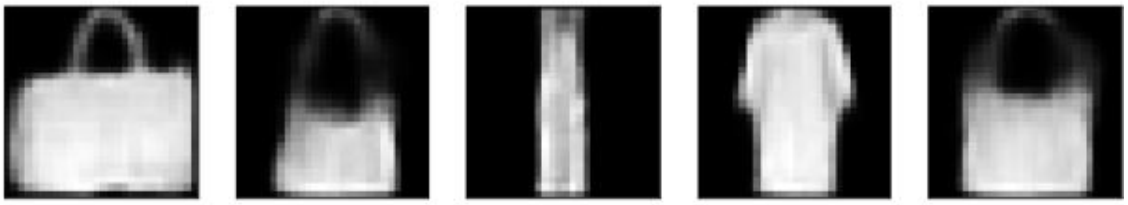


Com veiem en aquest cas desapareix la franja grisa inferior i les imatges tenen contorns més definit. Veiem una altra mostra de reconstruccions per veure com millora l'augment de dimensió de l'espai latent.





Com veiem en el cas de bolsos la representació tot i millorar en el cas de dimensió 4 no és suficient. Les reconstruccions són imatges poc nítides on es pot arribar a intuir que es tracta d'un bolso, però no és similar al de la imatge inicial. Quan provem amb un autoencoder amb dimensió latent 8 obtenim les següents imatges.



En aquest cas si que podem distingir l'objecte i inclús mantenir les dimensions.

Separació d'imatges amb autoencoders

Per aquest programa treballarem amb una base de dades alterada per nosaltres. Prendrem la base de dades dels nombres escrits a mà i la de les peces de roba (ambdues de 60.000 imatges) i farem la mitjana per elements, és a dir, prendrem la primera imatge d'ambdues bases de dades, les sumarem i en farem la mitjana. Repetirem aquest procés per tots els elements de la base de dades i obtindrem així la nova base de dades.

El nostre objectiu és crear dos autoencoders: un que reconstrueixi la part numèrica de la imatge ignorant les peces de roba com si fossin soroll i un altre que reconstrueixi les peces de roba ignoralt la part numèrica. Per fer-ho farem servir un autoencoder sobre-complet

Veiem els resultats.

Referències

[1] Bengio, Y. and Courville, A. and Goodfellow, I (2016): *Deep Learning*, MIT Press, Cambridge.

[2] Bishop, C. (2006): *Pattern Recognition and Machine Learning*, Springer, Cambridge.

[3] Calin, O. (2020): *Deep Learning architectures: a mathematical approach*, Springer, New York.

[4] Géron, A.(2022): *Hands-on Machine Learning with Scikit-Learn, Keras and Tensorflow*, O'Reilly Media.

[5] Nalwa, V.(1994): *A guided tour of Computer Vision*, Addison Wesley Publishing Company.