

Domande generali

Cos'è lo spazio delle ipotesi?

È lo spazio di possibili soluzioni in cui restringiamo la nostra ricerca.

Non è fattibile cercare una soluzione nello spazio di tutte le soluzioni possibili.

Quindi cerchiamo all'interno di uno spazio da noi scelto.

Questo non ci permette di trovare la soluzione perfetta e introduce dell'errore che accettiamo in cambio di riuscire a trovare una soluzione.

Prendendo come esempio il curve fitting potremmo cercare una curva tra tutte le possibili funzioni esistenti ma invece decidiamo di ridurre la nostra ricerca alle funzioni polinomiali rendendo la trattazione molto più semplice.

Nei decision trees lo spazio delle ipotesi sono tutti gli alberi che risolvono il problema.

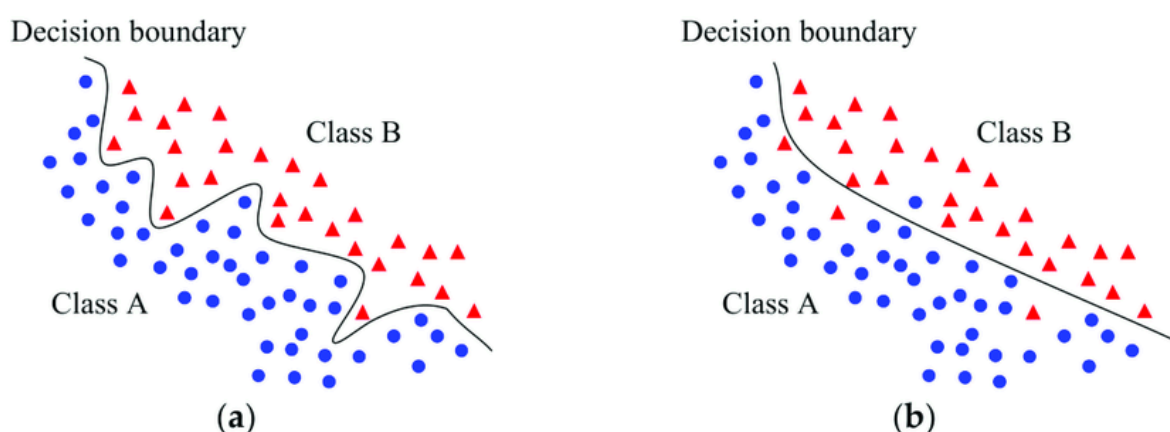
Cosa sono l'underfitting e l'overfitting?

Entrambi sono errori derivanti da uno scorretto training del modello.

Il primo si ha quando le prestazioni sono insufficienti sia durante la training phase che la testing phase. Il risultato ottenuto è troppo distante da quello ideale.

Nel caso dell'overfitting invece otteniamo ottimi risultati durante la training phase e pessimi risultati durante la testing phase. Questo significa che il modello non è in grado di generalizzare. Il modello ha imparato a memoria i dati che ha ricevuto nel training e non è in grado di adattarsi a nuovi dati.

Disegnami un esempio di overfitting



Cos'è un autovalore?

Un autovalore o eigenvalue in inglese è uno scalare associato ad un'autovettore che indica il fattore di ridimensionamento in seguito ad una trasformazione lineare.

Indica per quale fattore vengono scalati i punti che si trovano su l'autovettore

Quali sono le differenze tra gli algoritmi lazy learners e gli algoritmi eager learners?

La differenza principale sta nel tempo di training dei modelli.

Un Algoritmo lazy è un algoritmo il cui training è molto semplice e veloce a discapito della inference phase che invece risulta più lunga e complessa.

Un Algoritmo eager invece presenta le caratteristiche contrarie.

Esempi di lazy learners sono: KNN

Esempi di eager learners sono: Decision-trees

Come si passa da un empirical error a uno reale(generalization error)?

Interpretazione nostra della domanda: Dato un nostro modello (che avrà solo un empirical error, ovvero un errore che è causato dal non avere la distribuzione reale ma solo una approssimata via sampling) come facciamo a stimare il generalization error? Usando il test set per calcolare il generalization error del nostro modello, non possiamo usare elementi del training set o del validation set perché sono biasati dal nostro training.

K-NN

Cos'è il K-NN? Cos'è la curse of dimensionality?

K-NN è un modello supervised, lazy learner e non parametrico(ovvero non fa ipotesi sulla struttura dei dati) per la classificazione dei dati.

Si basa sul mappare gli elementi in base alle loro features su un piano n dimensionale. La classificazione avviene cercando i K elementi più vicini al nuovo dato. Per trovare gli elementi più vicini viene calcolata la distanza(tipicamente distanza euclidea).

Una volta trovati al nuovo dato viene assegnata la classe più popolare.

K è un hyper parameter. Le euristiche da seguire sono:

- K dispari per evitare ambiguità
- K molto minore della cardinalità del training set
- K non troppo piccolo per evitare errori dovuto agli outliers

La curse of dimensionality è un problema derivante dalla necessità di K-NN di avere dataset molto densi per assegnare correttamente le classi. In breve in uno spazio con sempre più dimensioni gli oggetti tra loro diventano in media sempre più lontani.

Inoltre anche i casi particolari diventano sempre più difficili da trovare

Diventa quindi difficile creare dei decision boundaries che siano corretti.

Quali sono le differenze tra K-NN e DT?

- Features
 - DT tratta le feature in modo diverso, dando ad alcune maggiore importanza di altre. DT sviluppa l'albero in base alla feature che minimizzano l'impurità.
 - K-NN tratta tutte le feature in modo uguale, questo può essere un male.
- Struttura
 - DT costruisce un'albero
 - K-NN memorizza il training set
- Modello
 - DT eager learner molto più veloce in inference phase. L'albero costruito è molto più leggero a livello di memoria
 - K-NN lazy learner lento durante la inference phase. Richiede molta memoria.

Modelli lineari

Cos'è un perceptron?

Un perceptron è un modello eager learner e parametrico (dati linearmente separabili) per la classificazione dei dati.

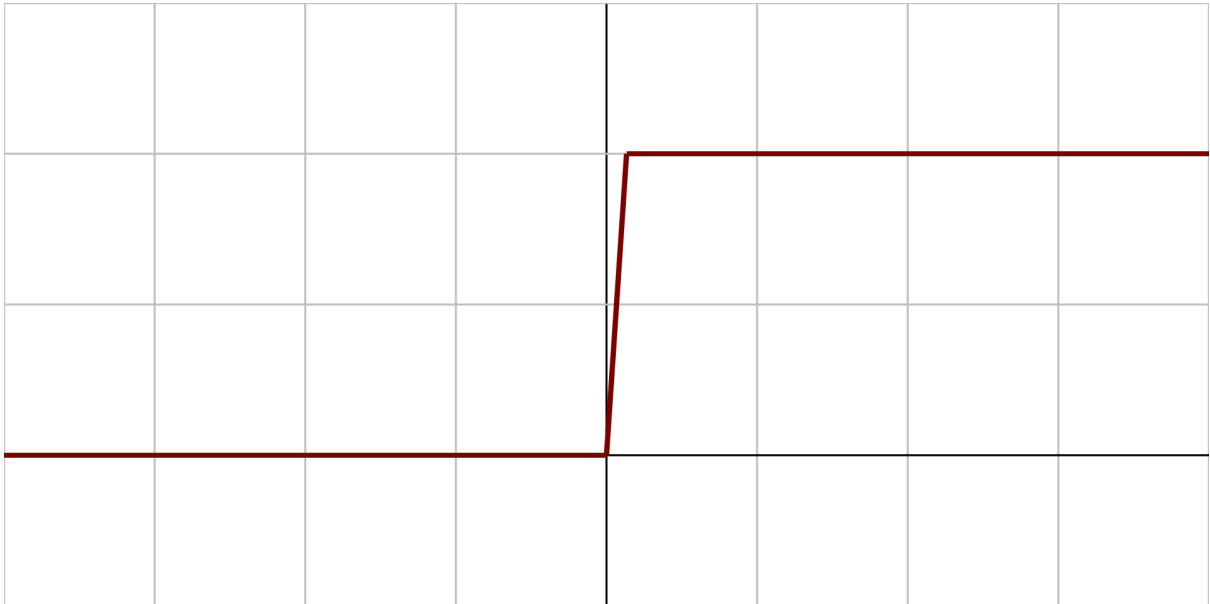
Impara una separazione lineare tra due classi di dati.

Viene utilizzato come base delle reti neurali dove imita il funzionamento di un reale neurone.

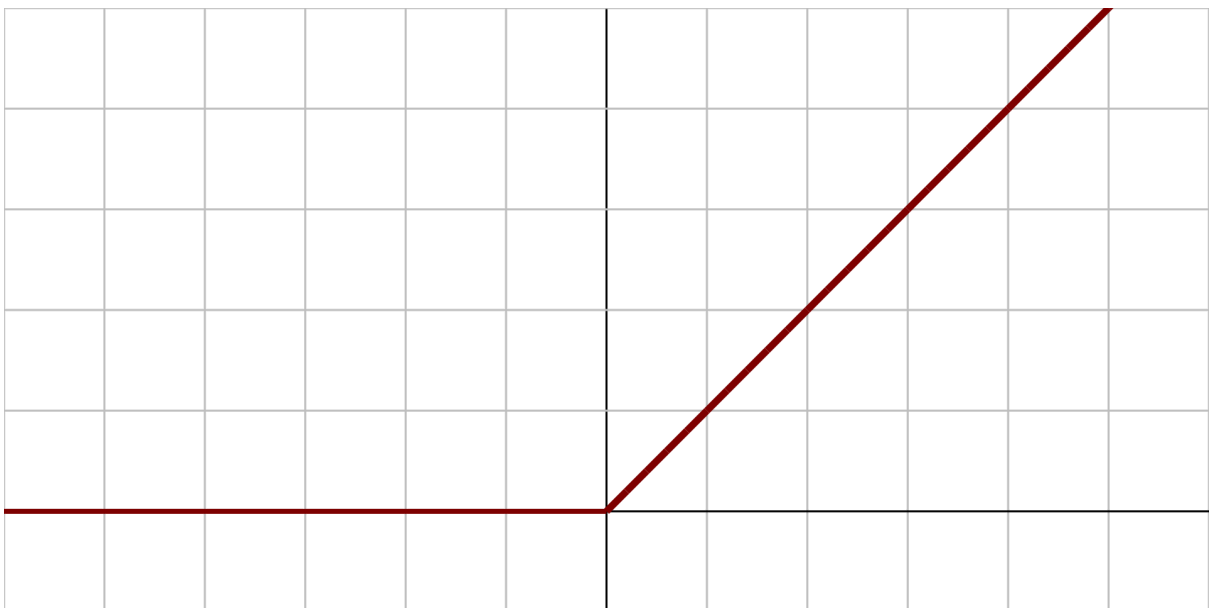
Inference phase: $\sigma(\sum w_i \cdot x_i + b)$ σ = activation function

Quali sono i pro e i contro delle funzioni di attivazione? Me le può disegnare?

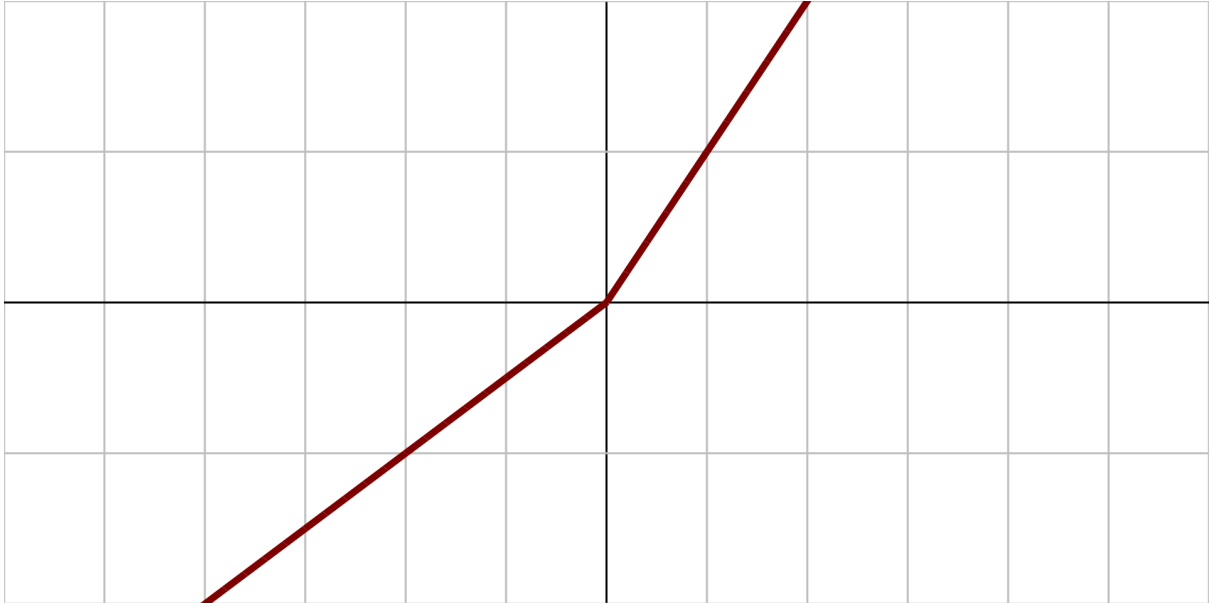
- 0/1
 - if $(x > 0)$ 1 else 0
 - non è derivabile



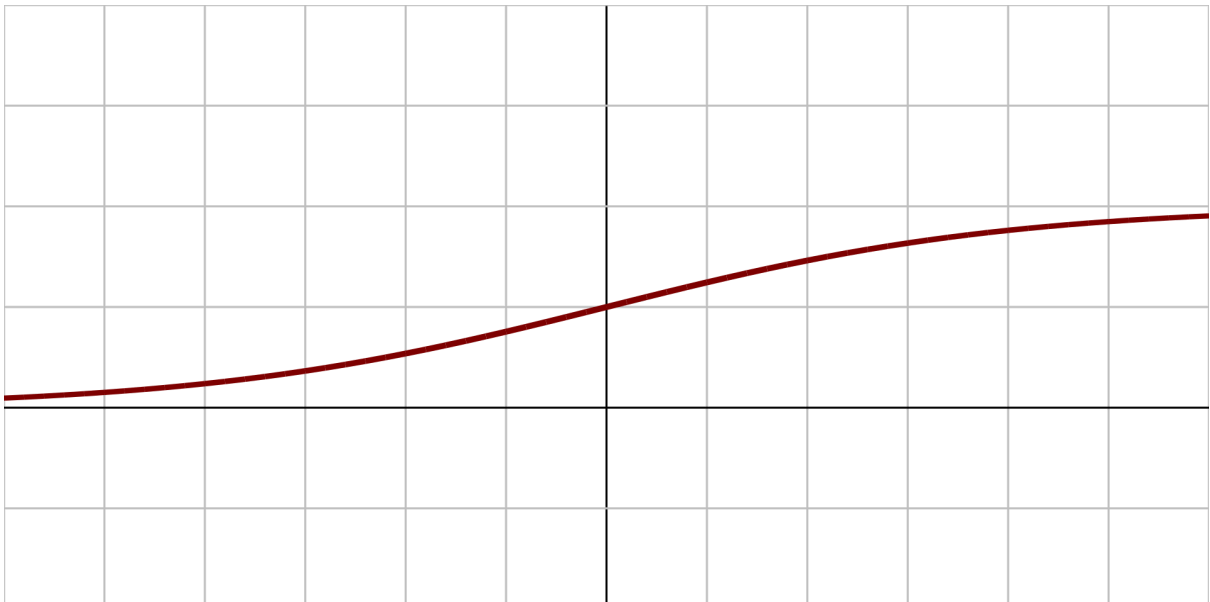
- ReLU
 - $\max(0, x)$
 - non è derivabile nello zero
 - vanishing gradient con input negativi
 - più che di vanishing gradient bisogna parlare di morte del neurone perché a valore negativi la derivata è zero e quindi i gradienti dei pesi rimangono invariati



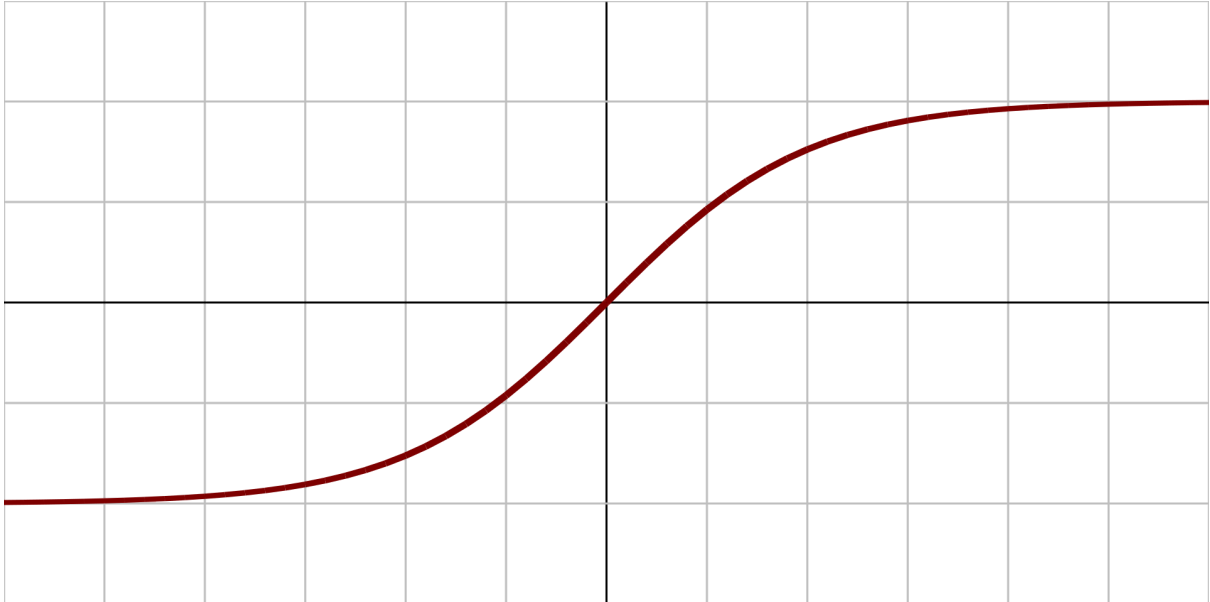
- Leaky ReLU
 - if $(x > 0)$ x else αx



- Sigmoide
 - range 0-1
 - vanishing gradient alle estremità. derivata cambia significamente sono attorno allo 0



- Tangente iperbolica
 - range $(-1) - 1$
 - problema uguale alla sigmoide




Cos'è il softmax?

È una funzione che converte un vettore di numeri reali in una distribuzione di probabilità. Ovvero la somma degli elementi deve essere 1. Viene utilizzata nelle NN quando si fa multiclass classification. Ci permette di capire la predizione della rete.

Mi disegna una ReLU? + formule

Mi descriva la Leaky ReLU

È una forma leggermente modificata di ReLU che ha lo scopo di evitare la morte dei neuroni. Anche a valori minori di zero il gradiente rimane diverso da zero e permette il continuo aggiustamento dei pesi. Il fattore di moltiplicazione α è un hyperparameter che va scelto. Generalmente si tratta di un valore molto piccolo.

Perché dovrei utilizzare una ReLU invece di altre funzioni di ottimizzazione? 

Decision trees

Cosa sono i DT ?

Sono un modello eager learner, non parametrico, supervised utilizzato per classification e regression.

I DT si basano sul costruire un albero da attraversare per ottenere il risultato.

Mi farebbe un esempio di ensemble learning?

Un esempio sono le random forests.

L'idea dell'ensemble learning è quella di trainare diversi modelli uguali contemporaneamente e ottenere un risultato come somma delle predizioni dei vari modelli. Il training viene effettuato su batch random dello stesso training set

Nel caso delle random forests vengono trainati diversi decision tree e viene fatta una media degli output(regression) oppure nel caso della classificazione si sceglie il label più popolare. Questa tecnica ha lo scopo di mitigare l'overfitting e migliorare l'accuratezza delle previsioni.

L'ensemble learning viene usato solo nei DT o in generale?

Non non è una tecnica esclusiva ai DT. Possiamo pensare di fare un ensemble di NN che fanno classificazione per essere sicuri di classificare correttamente un'immagine.

Come si può evitare l'overfitting nei DT?

- Ensemble
- Pruning
 - possiamo decidere di sostituire un ramo con una foglia
- Migliorando i criteri per creare una foglia
 - aumento livello di impurità
 - decidendo un numero minimo di oggetti a cui fermare la classificazione
 - limitando la profondità dell'albero

Come si decide se far crescere una foglia?

Mano a mano che l'albero cresce e il training set viene suddiviso in parti sempre più piccole l'impurità e il numero di elementi diminuiscono.

Le foglie vengono create una volta raggiunto un certo livello di impurità o una cardinalità di elementi oppure un'altro criterio scelto utile al nostro scopo.

Cos'è l'impurità nei DT?

È una misura di purezza di un set di elementi. Misura quanto è eterogeneo un set. Viene utilizzata per determinare le split function dato che dà una misura dei migliori split. Ci sono diversi modi per calcolarla. Ogni modo ci dà delle informazioni diverse più o meno adatte al nostro scopo.

Il DT serve solo per fare classificazione?

No è utile anche per fare regressione. Invece di avere come output un label abbiamo un numero reale ottenibile per esempio calcolando la media dei valori presenti nella foglia

Dove si trova la ricorsione in un DT?

Durante la training phase ricorsivamente creiamo nodi e sottoalberi finché il livello di impurità non è soddisfacente.

Durante la inference phase attraversiamo l'albero ricorsivamente finché non raggiungiamo una foglia.

Dove abbiamo visto le split function? a cosa servono? mi sa fare un esempio di split function? che vantaggio c'è nell'usare una split function semplice? posso avere comunque delle funzioni oblique?

Le split function risiedono nei nodi dei DT.

Servono a suddividere il training set in parti minimizzando l'impurità.

Ex: if $x < 5$ L else R

Le split function parallele agli assi rendono il training più semplice.

Le funzioni oblique possono esserci, sono quelle che valutano più features contemporaneamente per effettuare lo split.

Che genere di funzioni troviamo all'interno di un DT?

All'interno dei nodi troviamo le split functions e all'interno delle foglie le prediction function. Solitamente per la classificazione le prediction function ritornano una costante che è il label.

Cosa sono le split functions ?

Sono funzioni che separano i dati in input in due insieme disgiunti cercando di minimizzare l'eterogeneità.

I DT si possono usare per feature discrete?

Sì. Un esempio può essere dividere i vestiti in base alla taglia.

ex: if $t < m$ L else R

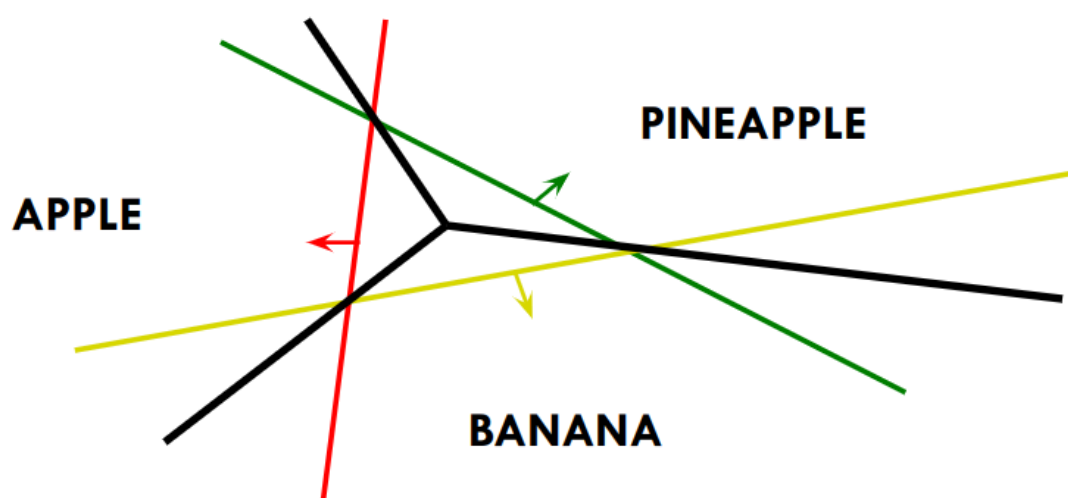
$$K = \{XS, S, M, L, XL\}$$

XS	S	M	L	XL
L	R	R	R	R
L	L	R	R	R
L	L	L	R	R
L	L	L	L	R

Multiclass classification

Mi farebbe un riassunto dei due approcci della multiclass classification?

- OVA - One Vs All
 - In questo caso abbiamo k classificatori. Ognuno per classe.
 - I classificatori sono allenati per riconoscere solo gli elementi della loro classe assieme ad un fattore di confidenza
 - La classe scelta è quella riconosciuta con più confidenza



- AVA - All vs All
 - Abbiamo un classificatore per ogni classe rispetto ad un'altra
 - Una volta classificato il dato si verifica quale classificazione è più popolare per definire il risultato

		apple vs orange		orange vs banana	
	apple		+1		+1
	orange		+1		-1
	apple		-1		-1
	banana	apple vs banana			
	banana		+1		
			+1		
			-1		
			-1		

Mi faccia un confronto tra OVA e AVA

- OVA
 - alleniamo meno classificatori e otteniamo un inference time minore
- AVA
 - Alleniamo più classificatori e quindi introduciamo più complessità che rende le cose sempre più difficili da gestire. Introduciamo tante possibilità di errore
 - Dovremmo avere un training più rapido

Com'è possibile calcolare l'accuratezza della predizione?

- Micro Averaging
 - $Ac = \frac{\text{Predizioni}_{corrette}}{\text{Predizioni}_{totali}}$
 - calcolo delle predizioni corrette sulle predizioni totali
- Macro Averaging
 - $Ac = \frac{\sum_i^n Ac_i}{n}$
 - calcolo dell'accuratezza per ogni singolo label e media delle accuratèzze

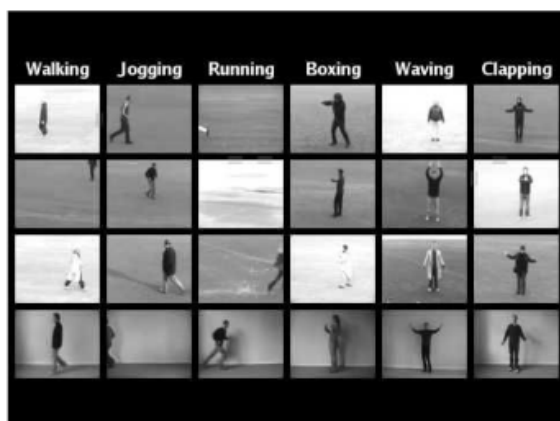
Cos'è la confusion matrix?

È una matrice che rappresenta il numero di prediction corrette per ogni label.

Un buon modello dovrebbe assomigliare ad una matrice diagonale.

Alti score sulla diagonale e praticamente zero il resto delle celle.

Questa matrice è una buona visualizzazione delle performance del nostro modello. Rende facile spottare le classi in cui il modello performa male.



Box	100	0	0	0	0	0
Clap	0	94	6	0	0	0
Wave	0	1	99	0	0	0
Jog	0	0	0	91	7	2
Run	0	0	0	10	89	1
Walk	0	0	0	0	6	94
	box	clap	wave	jog	Run	Walk

Gradient descent

Cos'è il gradient descent? cos'è il learning rate?

Il gradient descent è una tecnica di ottimizzazione per la funzione di costo. Il nostro scopo è quello di minimizzare l'errore aggiustando i nostri parametri e il gradient descent ci permette di farlo. Si basa sul calcolo delle derivate che ci dà un'indicazione dell'andamento della funzione di costo. Gradient descent funziona meglio con le funzioni convesse ma può essere applicato anche con altri tipi di funzioni.

Il learning rate è un parametro che modifica la grandezza dello step di ottimizzazione nella discesa del gradiente. questo iper parametro solitamente viene usato come inerzia, quindi parte con un valore grande che va poi a ridursi esponenzialmente per gli ultimi aggiustamenti necessari.

Come si calcola il gradiente?

Il gradiente è un vettore contenente tutte le derivate parziali della loss function per tutti i parametri del nostro modello. Nel caso dei perceptron i parametri sarebbero i weights e i bias.

$$\nabla C = \begin{bmatrix} \frac{\partial C}{\partial w^{(1)}} \\ \frac{\partial C}{\partial b^{(1)}} \\ \vdots \\ \frac{\partial C}{\partial w^{(L)}} \\ \frac{\partial C}{\partial b^{(L)}} \end{bmatrix}$$

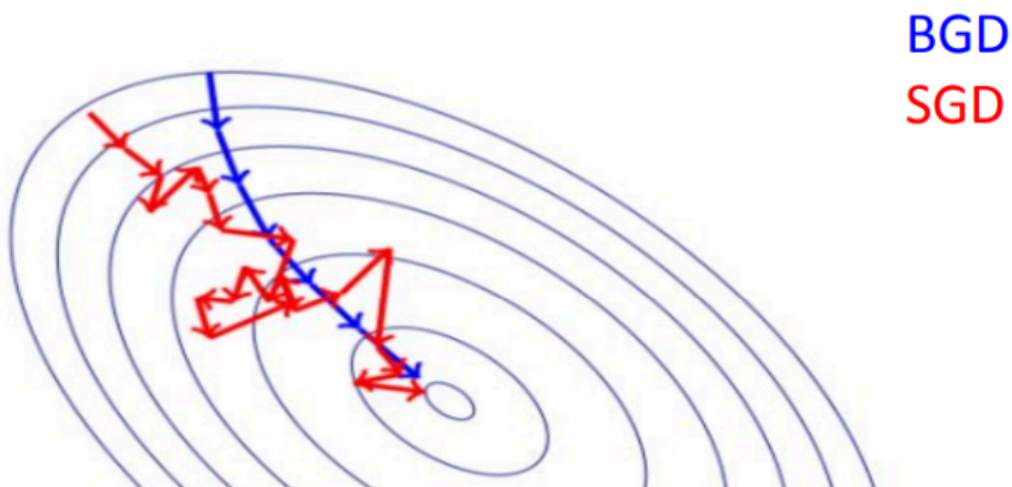
Cos'è il batch gradient descent?

Il gradient descent di norma prende in considerazione tutti i training sample prima di procedere con uno step. Il vettore dei gradienti si compone della media di tutti i gradienti per ogni singolo sample.

Questa procedura però è molto lenta e spesso si preferiscono altre tecniche tipo il mini batch gradient descent in cui vengono prese batch randomizzate di samples per calcolare il gradiente. Questo rende il processo molto più veloce anche se magari più erratico.

Cos'è lo stochastic gradient descent?

Invece di valutare ogni sample o un sottogruppo di questi viene valutato un sample randomico per ogni step. Questo ci permette di trovare un minimo molto più velocemente a discapito di una discesa molto più irregolare



Cos'è il momento?

Il momento è un vettore utilizzato per dare inerzia al gradient descent. Contiene la media pesata dei gradienti passati, dando più importanza ai gradienti recenti. In questo modo la discesa iniziale è accelerata e poi va rallentando. È molto utile per evitare minimi locali e saddle points.

Mi scrivi la formula del momento ?

$v_t = \gamma v_{t-1} - \eta_k \nabla w_t$ Il vettore del momento al passo t è uguale al vettore del momento al passo $t - 1$ moltiplicato per un iperparametro γ che indica quanta importanza dare al momento precedente (per forza compreso tra 0 e 1), meno il gradiente dei pesi ∇w_t per il learning rate η_k . $w_{t+1} = w_t + v_t$ Questa invece è la formula per l'aggiornamento dei pesi.

Loss function

A che cosa serve una loss function? Mi fa un paio di esempi?

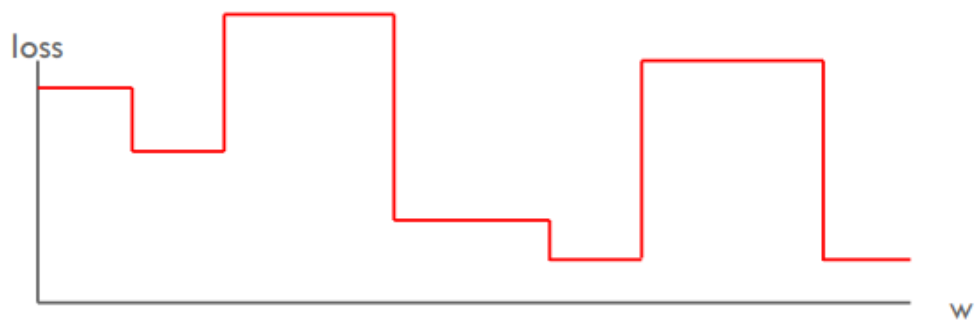
La loss function è necessaria per avere una misura oggettiva della performance del nostro modello. Ci permette di guidare il training verso un'errore sempre minore.

Misura la distanza dell'output ottenuto da quello corretto.

Una loss function ideale dovrebbe essere sempre derivabile e avere un solo minimo di tipo globale.

Utilizziamo funzioni convesse o surrogate ovvero che sono upperbound della reale loss function.

- 0/1 loss
 - non è convessa



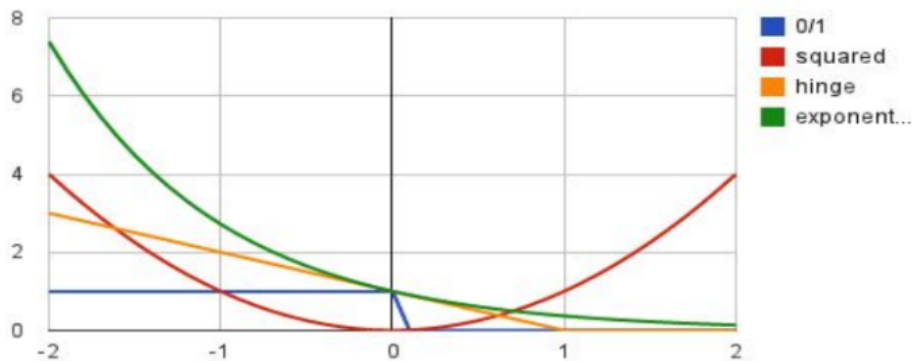
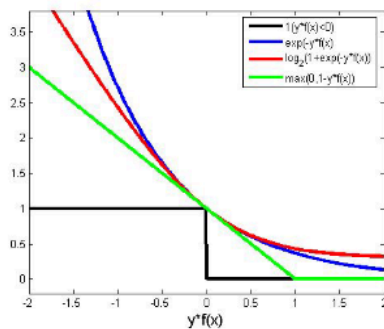
- Square loss
 - $L(i, j) = (i - j)^2$
 - usata più o meno ovunque
- Hinge loss
- Exponential loss

0/1 loss: $l(y, y') = 1[y y' \leq 0]$

Hinge: $l(y, y') = \max(0, 1 - y y')$

Squared loss: $l(y, y') = (y - y')^2$

Exponential: $l(y, y') = \exp(-y y')$



Cos'è la loss function?

Nelle reti neurali si usa una precisa loss, quale?

Tendenzialmente si usa la square loss perchè è molto punitiva con errori grandi

Come diventa la loss function quando si aggiunge il regolarizzatore?

Il regolarizzatore viene moltiplicato per un iper parametro λ e poi sommato alla loss.

λ rappresenta un trade-off tra la diminuire la loss ed evitare l'overfitting

$$\operatorname{argmin}_{w,b} \sum_{i=1}^n \text{loss}(y y') + \lambda \text{regularizer}(w, b)$$

Le loss possono essere usate insieme al gradient descent?

L'obiettivo con gradient descent è quello di trovare un minimo in una funzione che può appunto essere la loss function. Quindi sì è possibile ed è proprio ciò che vogliamo fare

Regolarizzazione

Che cosa significa regolarizzazione? Che strategia abbiamo visto nella regolarizzazione?

La procedura di regolarizzazione tenta di evitare l'overfitting. La regularization che viene aggiunta alla loss serve a regolarizzare i valori dei pesi in modo che non diventino troppo grandi. Pesi grandi creano funzioni poco adatte a generalizzare.

I regularizer principali sono L1 e L2

sum of the weights (1-norm) $r(w, b) = \sum |w_j|$

sum of the squared weights
(2-norm) $r(w, b) = \sqrt{\sum |w_j|^2}$

p-norm = family of norms

p-norm $r(w, b) = \sqrt[p]{\sum |w_j|^p} = \|w\|^p$

Smaller values of p ($p < 2$) encourage sparser vectors
Larger values of p discourage large weights more

Che differenza c'è tra L1, L2 e Lp?

(Utilizziamo regularizer con $p \geq 1$ perché sono convessi)

Sono tutti e 3 regularizer della famiglia p-norm

- L1
 - crea vettori sparsi con pesi vicini zero
 - le funzioni risultano più veloci da calcolare ed è più facile capire quali feature siano meno importanti.
- L2
 - ha una forma più convessa che facilita il training
- LP
 - valori di p grandi sfavoriscono pesi grandi

Support vector machines

Cosa sono le SVM?

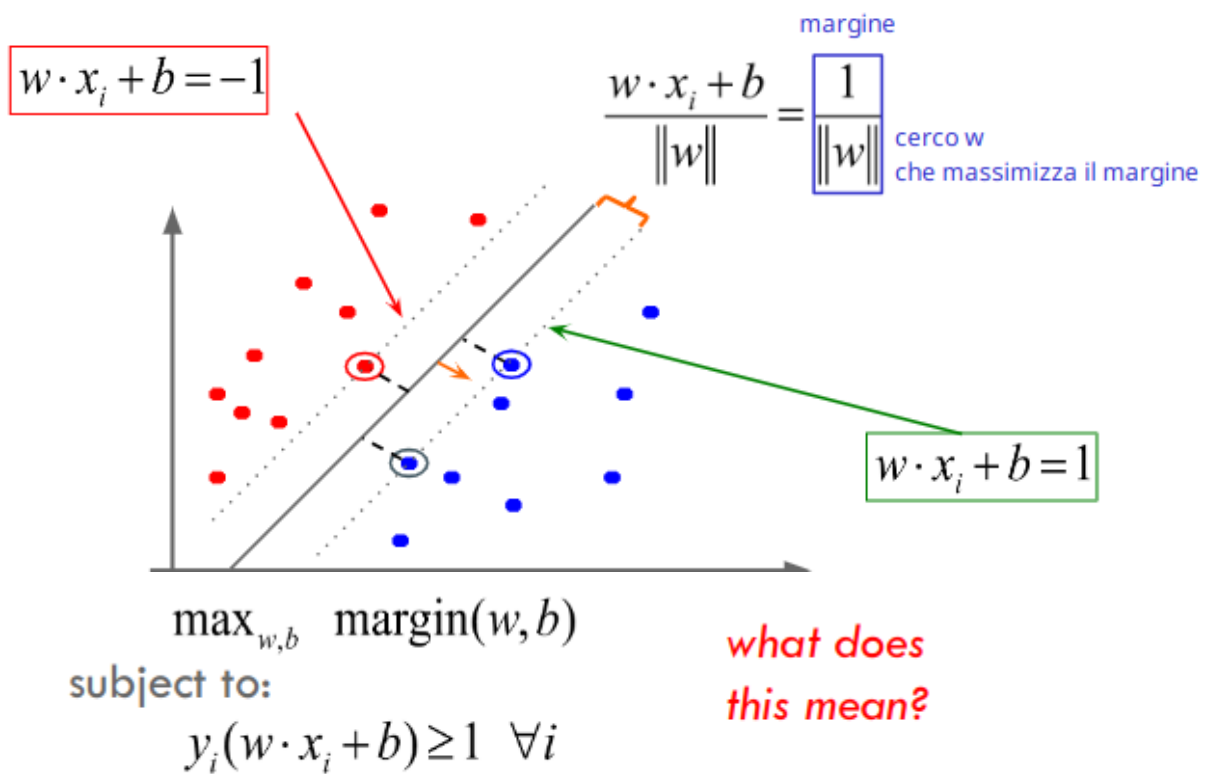
È un modello supervised che cerca una separazione lineare con il massimo margine.

Il margine è la distanza tra il separatore e i punti.

SVM si basa su importanti garanzie teoriche. Trovare la separazione lineare con il miglior margine ci permette di trovare il modello che generalizza meglio.

I punti che si trovano al limite del margine vengono chiamati support vectors e sono gli unici punti che vengono tenuti in considerazione durante la inference phase. Questo perchè sono gli unici a definire l'hyperplane.

Questo è un grande vantaggio di calcolo.



questo constrain ci assicura che tutti i punti sono fuori dal margine

Il margine può essere di due tipi

- Hard margin
 - la separazione lineare deve essere assoluta e non c'è spazio per errori

$$\min_{w,b} ||w||^2 : y_i(w * x_i + b) \geq 1 \forall i$$

- Soft margin
 - la separazione lineare ammette degli errori
 - ogni punto ha una slack variable associata che rappresenta l'errore di classificazione per quel punto
 - un hyperparameter C definisce quanta importanza dare alle slack variables nel processo di ottimizzazione

$$\min_{w,b} ||w||^2 + C \sum_i \zeta_i : y_i(w * x_i + b) \geq 1 - \zeta_i, \quad \forall i, \zeta_i \geq 0$$

Cos'è il dual problem?

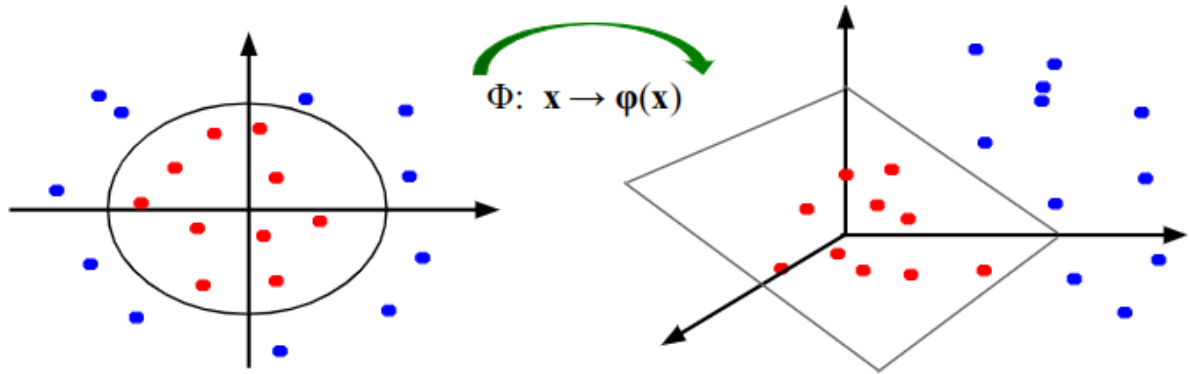
Per trainare un SVM dobbiamo risolvere un quadratic optimization problem ovvero minimizzare una funzione quadratica rispettando dei vincoli.

I problemi quadratici sono molto noti e ben studiati. Possiamo quindi costruire un problema duale che ci dà la soluzione al nostro problema iniziale.

La soluzione ci porta ad un inference phase in cui il calcolo si basa solo sui support vectors e il training nel calcolo dei prodotti scalari per ogni punto.

Si spieghi il kernel trick

Il kernel trick serve a trovare una soluzione in caso i dati non siano linearmente separabili. Possiamo pensare di mappare i punti ad uno spazio di maggiori dimensioni dove possiamo trovare un dataset che li separa



Solitamente si utilizzano kernel già scoperti senza doverne cercare uno noi.

Il kernel trick è particolarmente potente perché le kernel function ci permettono di calcolare il dot product in uno spazio con più dimensioni senza dover effettivamente proiettare i punti in tale spazio.

I kernel possono essere usati su ogni cosa, anche i grafi, non per forza punti su uno spazio.

Come si riconoscono i support vector in funzione delle alpha?

Domanda relativa alla formula del dual problem

Quali sono i pro e i contro dei SVM?

- Pro
 - Le SVM sono modelli molto flessibili e capaci per la classificazione binaria supervised.
 - Grazie alle loro basi teoriche riguardo al margine siamo sicuri di ottenere un buon grado di generalizzazione.
 - Sono ottime per gestire grandi dataset in quanto la predizione deve tenere conto solamente dei support vector.
 - Performano bene anche con dataset ridotti
- Contro
 - Non sempre è possibile trovare una separazione lineare dei dati

Perché SVM sono migliori del perceptron?

I perceptron trovano una tra le tante possibili separazioni lineari dei dati. Le SVM invece trovano la separazione lineare che garantisce il margine più grande e quindi la massima generalizzazione. Il soft margin inoltre ci permette di lavorare con dati rumorosi e il kernel trick ci permette di trovare una soluzione dove il perceptron fallirebbe.

Come si sceglie un kernel? come faccio a sapere qual è il kernel più adatto?

Purtroppo non possiamo sapere a priori quale sia il kernel più adatto. Dobbiamo provare quelli conosciuti e scoprire quale performa meglio con il nostro dataset.

Il kernel si usa solo nelle SVM?

No si usano anche nelle CNN nei layer di convolution

Che cos'è il margine in un contesto di SVM e perché è importante?

Il margine è la distanza tra il separator e i support vectors. È importante perché il nostro obiettivo è quello di massimizzarlo. Le SVM si appoggiano a importanti basi teoriche per cui siamo sicuri di trovare un buon modello massimizzando il margine.

Che tipo di regolarizzatori utilizzano le SVM?

Regolarizziamo il risultato tenendo conto degli slack values.

Facciamo la somma degli slack values e li sommiamo ad un parametro C che rappresenta il trade off tra la dimensione del margine e gli errori di classificazione.

SVM Soft margin

Perché vengono aggiunte le slack variables?

Le slack variables vengono aggiunte per trovare il soft margin. Il soft margin è un margine che permette gli errori. Può essere molto utile perché ci permette di separare dati che non sono perfettamente linearmente separabili a causa di rumore per esempio.

Le slack variables rappresentano l'errore di classificazione dei punti. Gli errori vengono conteggiati nella fase di training e si cerca un compromesso tra ottenere un grande margine e fare meno errori possibili. La variabile C che viene moltiplicata per la somma delle slack variables funge da parametro di trade off.

Un C molto alto comporta un hard margin mentre un C molto basso comporta margini molto più ampi.

Qual è il range delle slack variables?

Per i dati classificati correttamente o sul margine il valore è 0.

Per i dati interni al margine il valore corrisponde alla distanza dal punto alla linea del margine, range 0 - 1

Per i dati classificati scorrettamente il valore è la distanza dall'iperpiano + la distanza dal margine.

Partendo dalle slack variables, come otteniamo una hinge loss?

La formula per trovare il valore delle slack variables è già di per sé una hinge loss.

$\text{loss} = \max(0, 1 - y \cdot y')$

$$\zeta_i = \max(0, 1 - y_i(w \cdot x_i + b))$$

Qual è il vantaggio di utilizzare soft margin rispetto ad utilizzare un hard margin?

Soft margin ci consente sempre di trovare una soluzione anche con dati non linearmente separabili a causa di noise per esempio. Hard margin ci garantisce di non avere errori nella classificazione ma non è sempre possibile trovare una separazione lineare perfetta e un margine troppo piccolo porta a una cattiva generalizzazione.

Neural networks

Perché le NN risolvono i problemi che si avevano negli approcci classici?

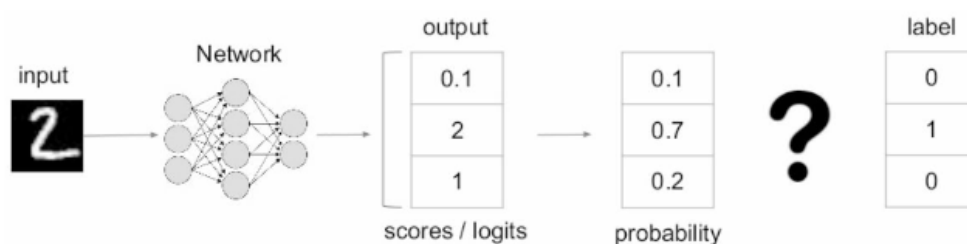
Le NN sono modelli molto versatili (non fanno alcuna assunzione sui dati) e complessi che mettendo assieme un alto numero di perceptron ed altri tipi di layer permettono di trovare soluzioni più avanzate. Per esempio non è necessario che i dati siano linearmente separabili o simili. Più perceptron assieme sono in grado di ricavare separazioni non lineari. Inoltre le NN lavorano direttamente sui dati. Non è necessario fare un processo di feature extraction fatto da umani che possono commettere errori. La NN autonomamente troverà dei pattern e delle feature che ritiene utili a risolvere il problema.

Che differenza c'è tra una sigmoide e una retta iperbolica? perché si preferisce la sigmoide?

La retta iperbolica ha un problema di vanishing gradient molto forte, presente anche nella sigmoide ma in maniera inferiore

Nelle reti neurali si usa una precisa loss, quale?

possiamo usare la square loss ma per i problemi di classification è tipico trovare la Cross-Entropy loss che si appoggia alla funzione Softmax. Essenzialmente misura la distanza tra la distribuzione ottenuta e quella target.



$$\mathcal{L}_i = - \sum_k y_k \log(S(l_k)) = - \log(S(l))$$

Convolutional Neural Networks

Come uso una convoluzione in una rete neurale?

Aggiungo dei layer di convoluzione ai classici layer di perceptron

Dentro a una rete neurale cosa succede quando si applica una convoluzione?

Avviene una sorta di feature extraction basata sulla località spaziale dei dati.

Un layer di convolution per esempio potrebbe fare edge detection.

I layer di convoluzione estraggono informazioni e riducono la dimensione dell'input conservando le informazioni importanti.

Cosa succede dentro a un layer convoluzionale?

1. Convolution
 - a. un filtro detto kernel viene applicato, tramite l'operazione di convoluzione, alla matrice di input facendolo scorrere. Lo scorrimento è influenzato dallo stride e dal padding.
 - i. convoluzione: (media pesata dell'input con il kernel)
 - ii. stride: quante celle spostare il kernel ad ogni step
 - iii. padding: aggiunta di zeri attorno alla matrice di input per mantenere la stessa dimensione nell'output
2. Non linearity
 - a. all'output detto feature map viene applicata una funzione di attivazione per aumentare la non linearità
3. Spatial Pooling
 - a. un nuovo filtro che ha il solo scopo di ridurre la dimensione dell'input viene applicato. si presta attenzione ad usare uno valore di stride corretto in modo che ogni cella venga sottoposta all'operazione di pooling una sola volta.
 - i. max pooling
 1. viene selezionato il valore più alto
 - ii. average pooling

Che grado di libertà ho quando genero un layer convoluzionale?

- Convolution
 - numero di filtri
 - dimensione del filtro
 - valore di stride
 - funzione di attivazione
- Pooling
 - dimensioni del filtro
 - tipo di filtro

Cos'è lo stride?

Lo stride è il numero di celle per cui viene spostato il filtro ad ogni iterazione. È un valore discreto. Più grande è il valore di stride e minore è la dimensione dell'output.

Cos'è il pooling nelle CNN? È un processo deterministico?

Il pooling è un processo di riduzione dell'input che avviene nella fase finale dei layer convoluzionali. Riduce l'input mantenendo le caratteristiche. È un processo deterministico.

C'è apprendimento nel pooling?

No, il pooling è un passaggio puramente meccanico che serve solamente a ridurre la dimensione dell'input.

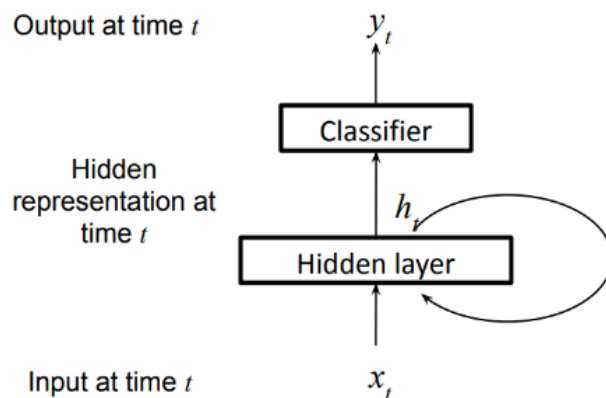
Con lo stesso dataset, cosa succede se utilizzo SVM e CNN?

- SVM
 - ottengo un buon risultato se il dataset è linearmente separabile
- CNN
 - se i dati contengono informazioni spaziali avrò un migliore risultato. Evito che l'immagine venga unrollata in un vettore perdendo informazioni importanti

Recurrent Neural Networks

Disegni il ciclo caratteristico delle RNN + cosa sono le RNN?

Sono NN pensate per elaborare sequenze di dati. Dati che quindi non hanno una lunghezza predefinita. Mantengono uno stato interno che permette di tenere conto dei dati passati nel calcolo dell'output. Sono usate per il riconoscimento vocale, la traduzione automatica, la generazione del testo, ecc..



$$h_t = f_W(x_t, h_{t-1})$$

new state function of W input at time t old state

Autoencoders

Disegnare un autoencoder spiegando cosa sia e come è possibile usarlo a test time

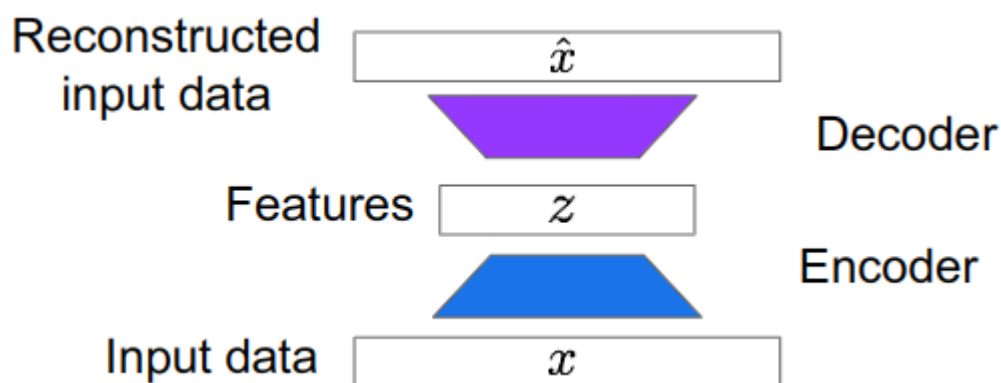
Un'autoencoder è una NN unsupervised che effettua dimensionality reduction.

Un'autoencoder è formato da due componenti

1. Encoder
 - a. performa la dimensionality reduction mappando l'input in uno spazio più piccolo
2. Decoder
 - a. cerca di ricostruire l'input originale a partire dal sottospazio intermedio

Encoder e decoder sono due NN separate che vengono trainate assieme. La objective function misura la distanza tra la distribuzione dell'input ricostruito e quello originale

Una volta terminato il training il decoder può essere rimosso



Per cosa viene usato un autoencoder?

Viene usato per fare dimensionality reduction

Cosa succede a inference time negli autoencoders?

Il decoder viene rimosso perché utile solamente durante il training.

L'encoder rimane utilizzabile come input di un modello per la classificazione

È possibile utilizzare delle immagini con gli autoencoders?

Sì certo, coder e decoder possono essere delle CNN

Unsupervised learning

Quali sono le differenze tra unsupervised e supervised learning?

- Supervised
 - Il training set deve avere i label
 - Producono risultati migliori ma è complesso raccogliere grandi quantità di dati con relativi label
 - Il modello impara un mapping tra gli input e gli output
- Unsupervised
 - Il training set non deve avere alcun label
 - Producono risultati peggiori ai modelli supervised e quando presenti i label è sempre meglio usarli.
 - Il modello impara pattern o strutture nascoste nei dati

Cos'è la dimensionality reduction?

È un problema di unsupervised learning.

È il lavoro di ridurre la dimensione dell'input perdendo il minimo numero di informazioni

Principal component analysis

Cos'è la PCA?

La principal component analysis è una tecnica per la dimensionality reduction

1. I punti vengono centrati e standardizzati.
 - a. centramento: nuovo centro è la media di tutte le posizioni in tutte le dimensioni
2. Calcolo della matrice delle covarianze
 - a. questa matrice contiene le informazioni di varianza e covarianza delle feature(dimensioni). Le varianze sulla diagonale e le covarianze nelle altre posizioni.
 - i. varianza: rappresenta la varianza di ogni singola feature. quindi quanta informazione racchiude
 - ii. covarianza: misura la correlazione della varianza tra due feature
3. Calcolo degli autovettori e autovalori associati alla matrice delle covarianze
 - a. Gli autovettori rappresentano le direzioni in cui i dati variano di più
 - b. Gli autovalori rappresentano la quantità di varianza sugli autovettori
4. Decido quali dimensioni scartare partendo da quelle con autovalori minori
5. Creo un nuovo spazio con gli autovettori selezionati come assi e mappo il mio dataset.

Che relazione c'è tra PCA e autoencoder?

Entrambi sono modelli unsupervised utilizzati per fare dimensionality reduction

È possibile utilizzare il kernel nella PCA?

Si è possibile utilizzare il kernel trick. In questo caso si parla di KPCA.

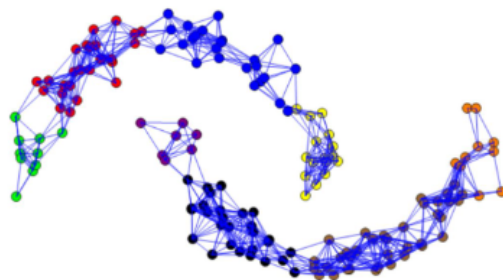
Ovvero si applica il kernel al dataset originale e poi si esegue PCA.

Questo perché PCA è una tecnica che utilizza trasformazioni lineari e se il dataset non è lineare non può essere applicata.

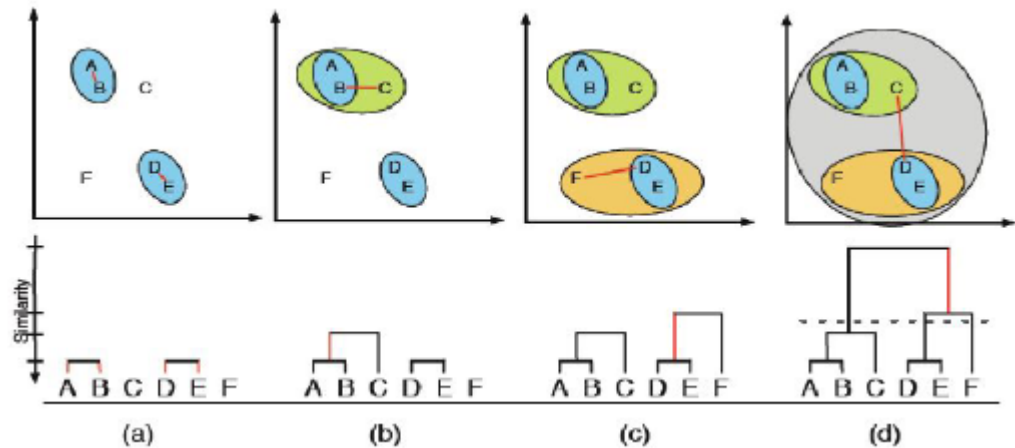
Clustering

Che modelli abbiamo visto nel clustering? Quali sono i vari pro e contro?

- K-MEANS
 - assumo il numero k clusters da trovare di sforma sferica
 - assumere la forma sferica è un'importante assunzione
 - hard clustering, quindi un oggetto può appartenere ad un solo cluster per volta.
 - Algoritmo
 - creo k centroidi in posizioni casuali
 - assegno i centroidi ai punti più vicini
 - calcolo la nuova posizione del centroide faccio la media delle posizioni della relativa classe
 - converge ma non assicura di trovare il minimo in quanto la posizione iniziale dei centroidi può variare molto il risultato
 - bisogna trovare una funzione per la distanza
 - i dati devono essere standardizzati
- Expectation Maximization
 - assumo cluster gaussiani, quindi con forma ellittica
 - soft clustering, l'appartenenza di un oggetto ad una classe è espressa come probabilità. un oggetto può essere associato ad più classi contemporaneamente.
 - Algoritmo
 - Inizializza randomicamente i cluster
 - Calcola la probabilità che un punto appartenga a un cluster
 - Ricalcola il centro del cluster
- Spectral clustering
 - assumo cluster come grafi. lavora sull'affinità locale
 - Hard clustering
 - Algoritmo
 - Collego tutti i nodi affini
 - Taglio gli archi meno importanti
 - I grafi connessi rimasti rappresentano i cluster



- Hierarchical clustering
 - assumo struttura a dendrogramma. Creo un'albero gerarchico.
 - Posso creare quanti cluster voglio andando a tagliare l'albero all'altezza che desidero
 - Soft clustering
 - Algoritmo
 - Creo un'albero bottom up o top down
 - bottom up: considero tutti i punti come grafi a se e poi li unisco in base alla loro affinità finchè non arrivo ad avere 1 solo cluster



Quali sono i problemi del K-MEANS e cosa si può fare per risolverli?

Problemi di K-MEANS

- devo sapere a priori quanti cluster voglio individuare
- K-MEANS assume che i cluster siano circolari

Perché l'EM clustering è migliore del K-means?

Pro

- clusters di forma gaussiana
- oggetti possono avere più label

Density estimation

Cos'è la density estimation?

Si intende il problema di riuscire a stimare e quindi replicare una distribuzione sconosciuta di dati senza avere accesso alla distribuzione reale.

Una volta stimata la distribuzione posso usare questa mia conoscenza per trovare outliers oppure generare nuovi dati che seguono la distribuzione osservata.

Qual'è la differenza tra un modello generativo e un modello discriminativo?

- Generativo
 - È in grado di generare nuovi dati associati ad una distribuzione trovata
- Discriminativo
 - Cerca di individuare i dati appartenenti alla distribuzione reale e quelli generati da un modello generativo tentando di imitare la distribuzione reale

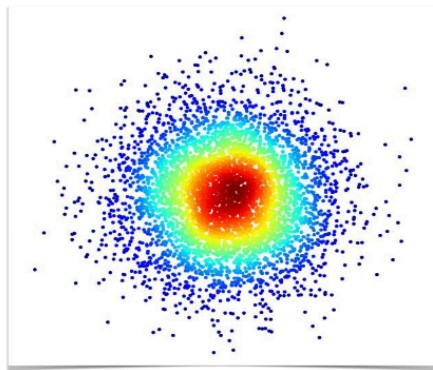
Mi descriva la distribuzione di densità esplicita ed implicita

- Esplicita

- Voglio scoprire qual è la distribuzione di probabilità che fitta i dati samplati da una data distribution sconosciuta

DENSITY ESTIMATION (EXPLICIT)

Find a probability distribution $f \in \Delta(\mathcal{Z})$ that fits the data, where $z \in \mathcal{Z}$ is sampled from an unknown data distribution $p_{\text{data}} \in \Delta(\mathcal{Z})$



SUPERVISED
LEARNING

$$\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$$

UNSUPERVISED
LEARNING

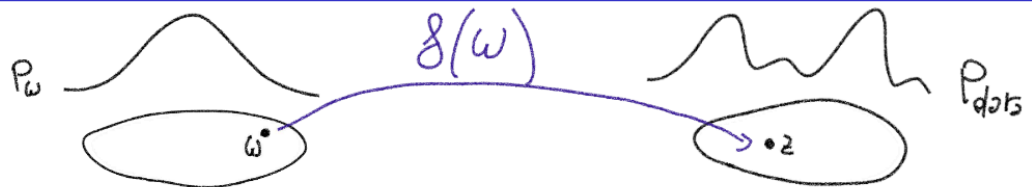
$$\mathcal{Z} = \mathcal{X}$$

- Implicita

- Voglio trovare una funzione f che genera dati appartenenti alla distribuzione target a partire da un input di una distribuzione conosciuta più semplice

DENSITY ESTIMATION (IMPLICIT)

Find a function $f \in \mathcal{F}^\Omega$ that generates data $f(\omega) \in \mathcal{Z}$ from an input ω sampled from some predefined distribution $p_\omega \in \Delta(\Omega)$ in a way that the distribution of generated samples fits the (unknown) data distribution $p_{\text{data}} \in \Delta(\mathcal{Z})$



posso applicare la stessa idea
al caso in cui ho i labels

SUPERVISED
LEARNING

$$\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$$

UNSUPERVISED
LEARNING

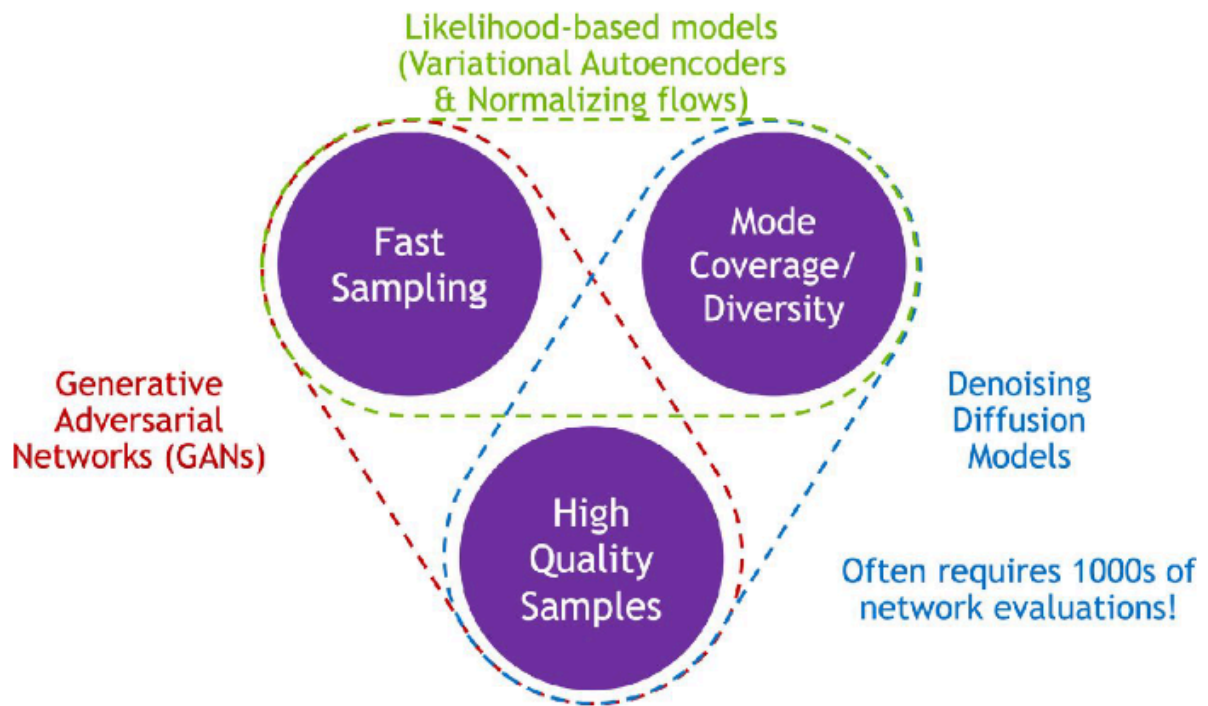
$$\mathcal{Z} = \mathcal{X}$$

Mi saprebbe dire un'istanza di ognuna? (riferito a sopra)

Esplicita: VAE, Variational Auto Encoders. Implicita: GAN, Generative Adversarial Network

VAE, GAE e diffusion models

Quali sono le differenze tra VAE, GAN e diffusion models? Perché i VAE sono veloci a fare sampling?



- explicit
 - VAE
 - good coverage
 - Diffusion
 - good coverage
 - output di alta qualità
 - latent space ha le stesse dimensioni dell'output
- implicit
 - GAN
 - output di alta qualità

I VAE sono veloci perchè lo spazio latente è una semplice gaussiana in cui è anche facile fare interpolazione

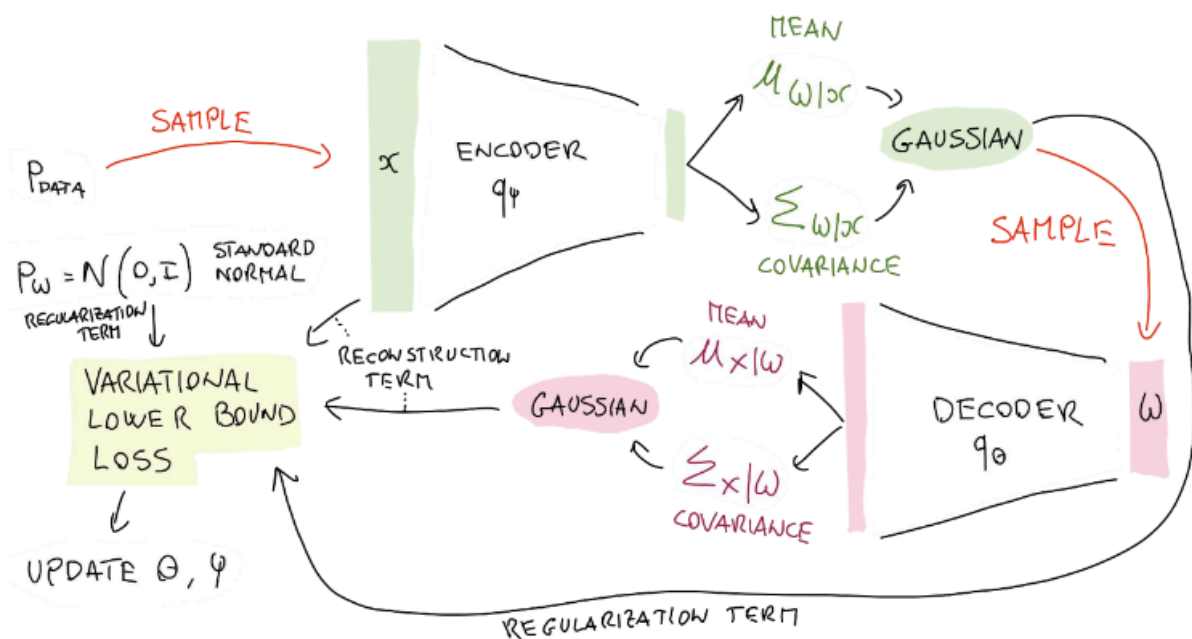
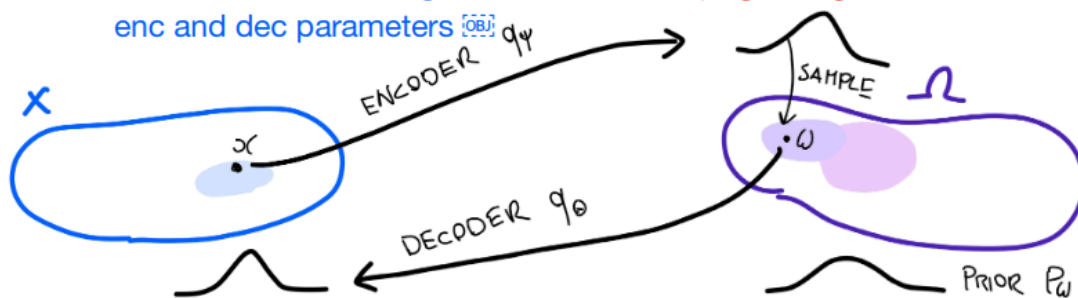
Qual'è la Loss nei VAE ? e cosa significa il termine latent space ?

VARIATIONAL BOUND

$$d_{\text{KL}}(p_{\text{data}}, q_{\theta}) \leq \mathbb{E}_{x \sim p_{\text{data}}} \left[\underbrace{-\mathbb{E}_{\omega \sim q_{\psi}(\cdot|x)} [\log q_{\theta}(x|\omega)]}_{\text{RECONSTRUCTION TERM}} + \underbrace{d_{\text{KL}}(q_{\psi}(\cdot|x), p_{\omega})}_{\text{REGULARIZATION TERM}} \right] + \text{const}$$

- Still intractable to compute but we can estimate the gradients of enc and dec parameters [OBJ] q_{ψ}

- It might have closed-form solution (e.g. using Gaussian distributions)



$$E_{\omega \sim q_{\psi}(\cdot|x)} [\log(q_{\theta}(x|\omega))] - d_{KL}(q_{\psi}(\cdot|x), p_{\omega})$$

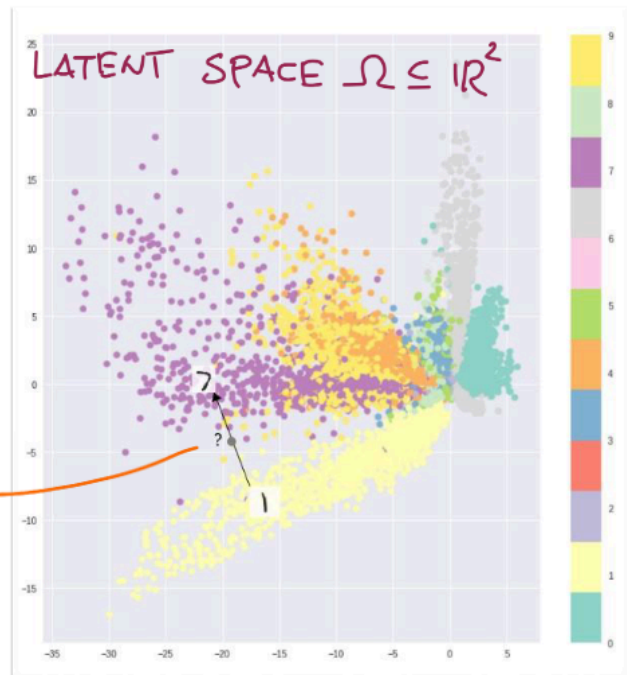
Facciamo un attimo un breakdown:

- $E_{\omega \sim q_{\psi}(\cdot|x)} [\log(q_{\theta}(x|\omega))]$ è definito reconstruction term ed è essenzialmente l'equivalente del reconstruction error di un AE, si sampla un ω dalla encoding probability distribution $q_{\psi}(\omega|x)$ e poi lo si dà in input al decoder che darà fuori un output, su questo output verrà poi calcolato l'errore
- $d_{KL}(q_{\psi}(\cdot|x), p_{\omega})$ è definito come regularizer, il suo compito è quello di spingere la $q_{\psi}(\omega|x)$ a essere simile a una distribuzione p_{ω} che solitamente è una gaussiana normale, questo ci permetterà a inference time di usare $w \sim p_{\omega}$ come input della VAE.
- Si usano entrambi questi due termini per ottimizzare la funzione così che il latent space dell'encoder che il decoder andrà a usare come input sia continuo (se si usasse solo il reconstruction term sarebbe discontinuo) (il che permette al decoder di saper agire su qualsiasi input campionato da una gaussiana normale) e che l'output corrisponda il più possibile alla distribuzione che si vuole stimare (se si usasse solo il regolarizzatore l'output del decoder non avrebbe senso)

EXAMPLES OF LEARNED LATENT SPACE ON MNIST

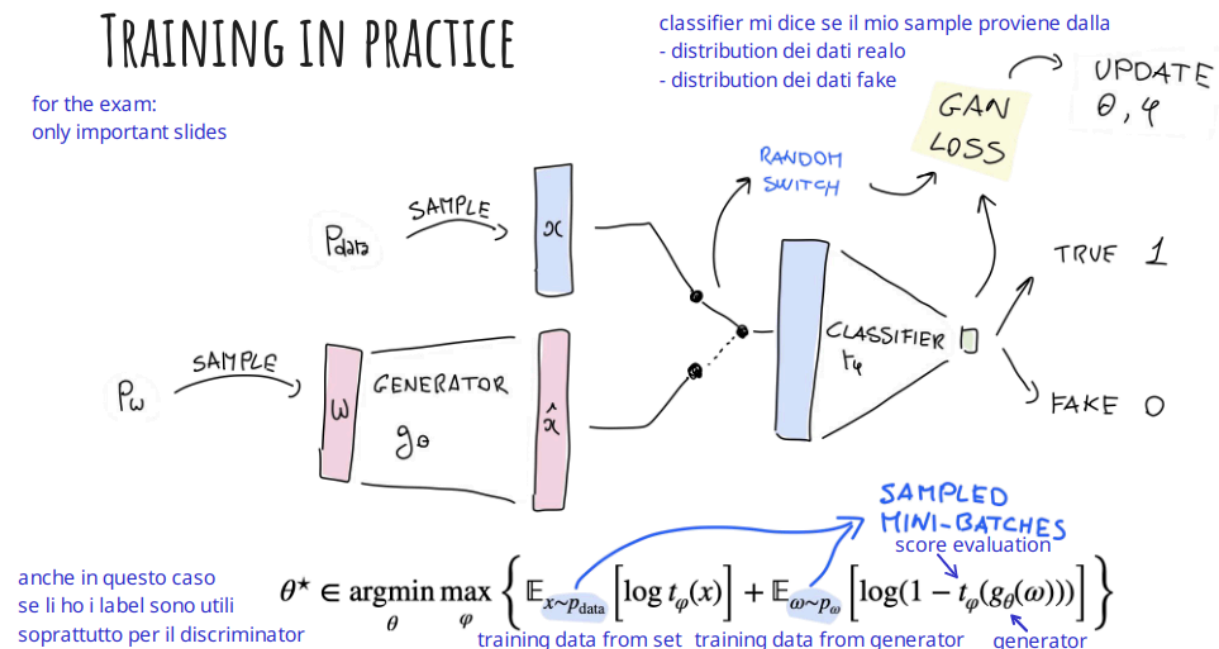
USING ONLY RECONSTRUCTION LOSS

THIS MIGHT GENERATE MEANINGLESS DATA !



Perché è più difficile allenare una GAN invece che una VAE?

La GAN soffre di stabilità nel training. Dato che si allenano contemporaneamente due reti avversarie non è sempre detto che convergano. Inoltre se il classifier diventa troppo bravo prematuramente il generator rimane come bloccato e senza possibilità di imparare.



Quando dovrei usare un VAE e quando una GAN?

↑ + le VAE producono output di molta maggiore qualità

Perché è difficile ottimizzare una GAN? ↑

Come è fatto lo spazio latente?

Nel caso della VAE è una distribuzione gaussiana generata dall'encoder.

Si parli della funzione di loss della GAN

l'errore è $\mathbb{E}_{x \sim p_{data}} [\log(t_\phi(x))] + \mathbb{E}_{\omega \sim p_\omega} [\log(1 - t_\phi(g_\theta(\omega)))]$ dove ϕ è un classificatore (discriminatore) che cerca di massimizzare l'errore del generatore ϕ , facciamo un breakdown:

- $\mathbb{E}_{x \sim p_{data}} [\log(t_\phi(x))]$ è la probabilità che il discriminatore labelli correttamente gli input samplati dalla distribuzione che vogliamo stimare, il discriminatore vorrà massimizzare questo parametro, da notare come θ non abbia modo di influire su questo errore.
- $\mathbb{E}_{\omega \sim p_\omega} [\log(1 - t_\phi(g_\theta(\omega)))]$ è la probabilità che il discriminatore labelli correttamente (il 1- è perché il valore corretto sarebbe 0) i datapoint generati dal generatore, il discriminatore vorrà massimizzare questo termine mentre il generatore vorrà minimizzarlo.

Reinforcement learning

Che idea c'è dietro al reinforcement learning? Cos'è il reward? Mi fa un esempio?

L'idea è quella di trainare un agente, il nostro modello, all'interno di un environment con cui interagisce costantemente. Vogliamo allenare il modello nel saper predire qual'è la prossima migliore azione in un costante feedback loop con l'environment. Vogliamo quindi che impari una policy per decidere la migliore azione in ogni stato.

Il reward è un incentivo che viene dato al modello per misurare la performance e spingerlo a scegliere azioni migliori. Nel reinforcement learning non c'è una funzione di loss.

Un agente può essere un AI che tenta di battere un videogioco e il reward è dato dal punteggio raggiunto.

Quali sono le parole chiave parlando di reinforcement learning? a cosa si riferiscono?

- Agente
 - è il modello
- Ambiente
 - lo spazio in cui il modello agisce
- Stato
 - lo stato dell'ambiente
- Azione
 - interazione dell'agente con l'ambiente
- Reward
 - l'ambiente ci restituisce un reward per ogni stato e azione che compiamo
- Policy
 - regole che l'agente segue per decidere qual'è la migliore azione

Perché fare reinforcement learning è più difficile del fare supervised learning?

Innanzitutto nel reinforcement learning le azioni future dipendono da quelle passate.

Nel caso del supervised learning il modello è continuamente guidato. Per ogni predizione viene calcolata una loss e quindi un'indicazione oggettiva delle performance.

Nel caso del reinforcement learning invece i reward possono essere sparsi e quindi il modello deve compiere molte azioni prima di poter valutare la sua performance.

Inoltre nel reinforcement learning i parametri non sono differenziabili e quindi non possiamo applicare tecniche classiche come gradient descent.

Si spieghi in cosa consiste il Markov Decision Process

Si tratta di un framework per prendere decisioni in un ambiente stocastico.

Un ambiente stocastico è un ambiente in cui non posso predire con certezza lo stato futuro dato uno stato di partenza e una possibile azione.

MDP cerca una policy che dato lo stato ci ritorni l'azione ottimale.

- Gli stati possibili s , con stato iniziale s_0
- le possibili azioni a
- il modello di transizione $P(s'|s, a)$
- una funzione di reward $r(s)$
- Policy $\pi(s)$, l'azione da fare in ogni possibile stato

MDP può essere definito con la tupla

$$(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma)$$

\mathcal{S} : set of possible states

\mathcal{A} : set of possible actions

\mathcal{R} : distribution of reward given (state, action) pair

\mathcal{P} : transition probability i.e. distribution over next state given (state, action) pair

γ : discount factor

Vogliamo trovare una policy che massimizzi la cumulative discounted reward

γ fa sì che il reward attuale abbiamo più valore di quelli futuri

$$r(s_0) + \gamma r(s_1) + \gamma^2 r(s_2) + \gamma^3 r(s_3) + \dots$$

$$= \sum_{t \geq 0} \gamma^t r(s_t), \quad 0 < \gamma \leq 1$$

Algoritmo

1. $t=0$ l'agente riceve lo stato iniziale
2. agente seleziona a dalla policy π
3. environment dà all'agente un reward
4. environment dà all'agente il prossimo stato
5. ripeti finché 2-4 finché l'obiettivo finale non viene raggiunto

Note:

- Value Function
 - Invece di ottimizzare la policy ottimizziamo una value function che rappresenta il valore atteso a partire da un particolare stato. Il massimo reward che possiamo ottenere a partire da quello stato
 - $V(s)$
- Q-Value Function
 - Stessa idea della Value function ma valutiamo assieme stato e azione
 - $Q(s, a)$
 - Calcolo della reward matrix conosciuta solo dall'environment e che l'agente deve costruire da sé e seguire per trovare la policy ottima

ALGORITHM

-
- Initialize the Q matrix with zeros
 - Select a random initial state
 - For each episode (set of actions that starts on the initial state and ends on the goal state)
 - While state is not goal state
 - Select a random possible action for the current state
 - Using this possible action consider going to this next state
 - Get maximum Q value for this next state
 - $Q^*(s,a) = R(s,a) + \gamma \max_{a'} [Q^*(s',a')]$
- Deep Q-Learning
 - passiamo ad una NN che approssima Q perchè lo spazio degli stati e delle azioni diventa troppo grande.

Cos'è l'assunzione di Markov?

Lo stato futuro dipende solamente da quello immediatamente precedente

Cos'è l'equazione di Bellman?

Il problema originario del reinforcement learning è complesso da calcolare. Utilizziamo quindi l'equazione di Bellman che sfrutta dynamic programming in cui il problema originale si può suddividere in problemi più semplici la cui soluzione può essere usata per risolvere il problema originale.

$$\begin{aligned} Q^*(s, a) &= r(s) + \gamma \sum_{s'} P(s'|s, a) \max_{a'} Q^*(s', a') \\ &= \mathbb{E}_{s' \sim P(\cdot|s, a)} [r(s) + \gamma \max_{a'} Q^*(s', a') | s, a] \end{aligned}$$

Cos'è una policy?

Una policy è una funzione utilizzata dall'agente per sapere quale azione compiere ad ogni stato per massimizzare il reward