Esempio domande per orale di Introduction to Machine Learning

Nicolò Marchini, Patrik Maniu, Silvanus Bordignon

Preambolo

I'm sorry to our English readers but this document started as personal study material so it is written Italian, luckily AI translators are really powerfull these day. The file exam_mentions contains every time the professor mentioned the exam during the 2024 course.

Questo documento contiene delle domande e delle risposte corrette per l'orale, c'è il documento Appunti per Introduction to Machine Learning 2023-2024 per gli appunti completi.

Indice

Domande generali

Cos'è lo spazio delle ipotesi?

il nostro obbiettivo quando facciamo ML è quello di creare un modello che può essere computazionalmente trattato (ovvero che si possa ricavare in un qualche modo ragionevole) che risolva un dato problema,

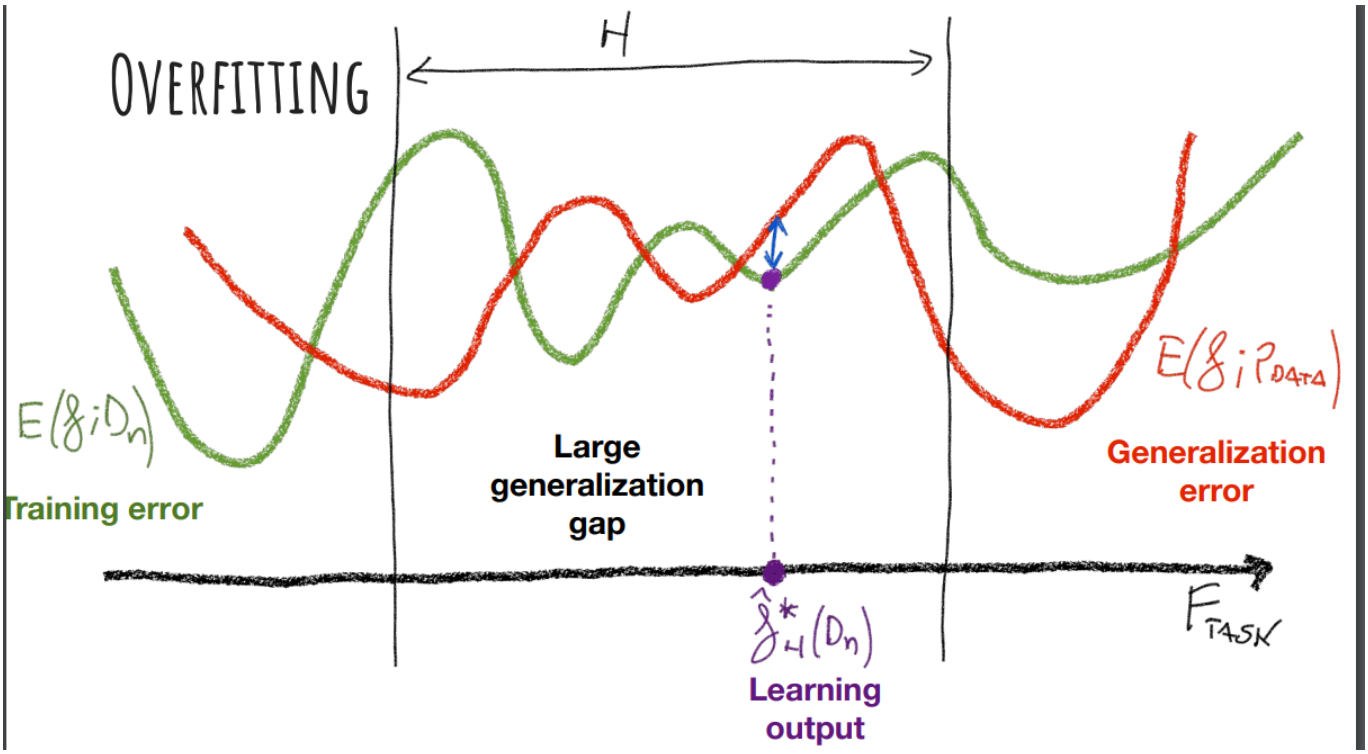
formalmente il set di questi modelli è lo spazio delle ipotesi $H \subset F_{task}$. Esempi:

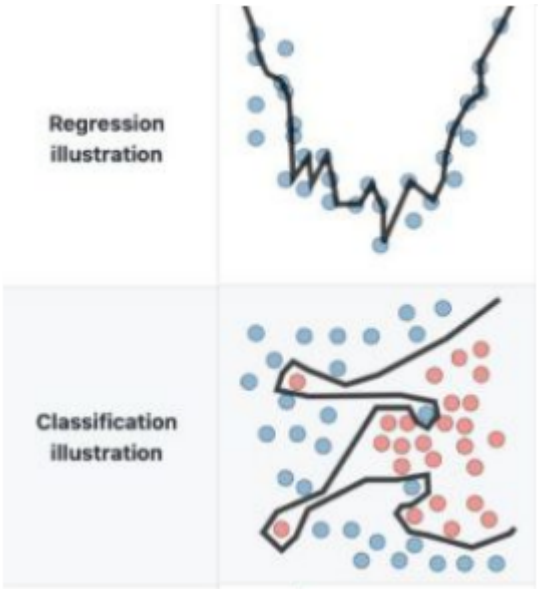
- Nei decision trees, lo spazio delle ipotesi è formato da tutte le possibili strutture dell'albero che risolvono il problema.
- Nelle neural networks, lo spazio delle ipotesi è dato da tutte le architetture possibili e dalle combinazioni di valori che possiamo assegnare ai pesi dei collegamenti fra i neuroni che risolvono il problema.

Cosa sono l'underfitting e l'overfitting?

Entrambi sono dei problemi di bassa generalizzazione che si possono incontrare allenando un modello. **L'underfitting** si verifica quando il modello è scadente sia sul training set che sul test set, è anche un indicatore che il modello è troppo semplice o i dati che sono stati forniti erano insufficenti. **L'overfitting** si verifica quando il modello è perfetto o quasi nei training set, ma continua a non avere risultati sul test set, ancora è un indicatore della scarsità di dati forniti e della complessita del modello (in questo caso il modello è troppo complesso).

Disegnami un esempio di overfitting





è più facile disegnare l'ultimo grafico che mostra overfitting per un problema di classificazione, si ricordi che l'errore di questo modello non si vede col training set (è perfetto o quasi a fare predizioni su di esso), ma usando il validation set.

Cos'è un autovalore? (in inglese eigenvalue)

Un autovalore è un numero scalare associato ad una matrice quadrata che rappresentano le soluzioni di un'equazione caratteristica. Ogni autovalore è associato a un autovettore, che rappresenta un'asse lungo la quale la matrice opera. Vengono usati in analisi e riduzione della dimensionalita.

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

La differenza sta nella fase in cui vengono eseguite le operazioni di calcolo, gli algoritmi lazy learners come il K-NN hanno un training quasi istantaneo perché memorizzano i dati senza operazioni aggiuntive, però durante la fase di inference hanno un costo computazionale molto elevato, la situazione si capovolge per gli

eager learners come decision trees e support vector machines, durante la fase di training devono eseguire delle operazioni che vanno oltre la semplice memorizzazione di un dato, però durante la fase di previsione sono molto più veloci a dare un output.

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 estimare 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?

Il K-NN(K-Nearest-Neighbors) è un algoritmo di classificazione non parametrico (non fa ipotesi sulla struttura dei dati) e lazy-learner. dato un training set, per classificare un nuovo dato, viene calcolata la distanza di quel dato rispetto agli altri (la posizione e il numero di dimensioni è dato dal valore e dal numero di features, non ci possono essere dati che hanno più label di altri per lo stesso modello) tramite una funzione di distanza Es. distanza euclidea, le distanze vengono ordinate e vengono tenute le K distanze minori, tra queste si calcola quale è il label più comune (ci sono versioni in cui danno peso anche a quanto sono distanti i due dati).

Il valore di K viene scelto da chi crea il modello, ci sono però delle euristiche:

- K dovrebbe essere dispari per evitare ambiguità
- K dovrebbe essere al massimo $\sqrt{n}$ per evitare errori causati da sbilanciamenti di cardinalità dei label contenuti nel nostro training set(n è la cardinalità del training set)
- K non dovrebbe essere troppo piccolo per evitare errori causati da outliers

Per ottenere dei buoni risultati con un modello K-NN, il training set dovrebbe essere abbastanza denso, aumentando però il numero di feature che teniamo in considerazione, aumentano anche le dimensioni su cui lavora il modello, per avere la stessa densità di dati, ma con più dimensioni la quantità di dati necessaria diventa esponenzialmente più alta e diventa difficile anche trovare i casi particolari, questo problema prende il nome di curse of dimensionality.

Quali sono le differenze tra K-NN e DT?

- Trattamento delle features
    - K-NN tratta tutte le features in maniera uguale (se non si modifica in modi specifici l'algoritmo base, ma questo avviene a priori del training)
    - DT può dare un peso diverso alle varie features
- Struttura del modello
    - K-NN memorizza il training set (al massimo fa anche un K-D Tree)
    - DT Costruisce un albero che rappresenta le divisioni nel training set
- Costo computazionale
    - K-NN è un lazy learner e il costo computazionale dipende dalla dimensione del training set e dal numero di features, necessità anche di tanta memoria per il training set.

    - DT è un eager learner e il costo dell'allenamento dipende da quanto è complessa la task, dato che non deve ricordarsi tutti i punti del training set è molto più efficente con l'utilizzo della memoria
- Il DT è molto più veloce da interpretare

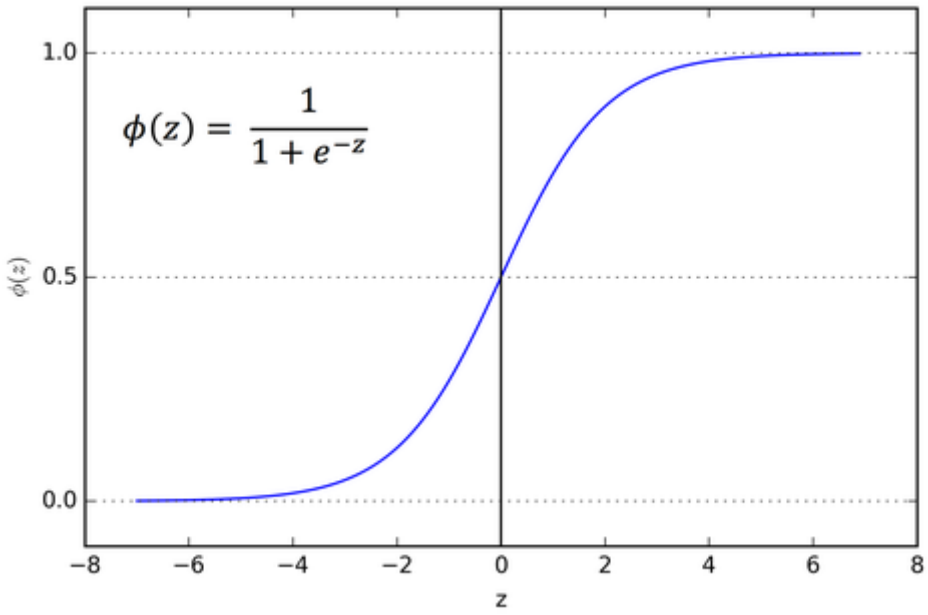
Modelli lineari

Cos'è un perceptron

È un classificatore lineare binario che viene utilizzato come blocco base delle reti neurali. Il perceptron riceve gli input e li moltiplica per i pesi associati a quella feature, somma tutti i risultati e un bias (reale) e fa passare il risultato attraverso una funzione di attivazione.

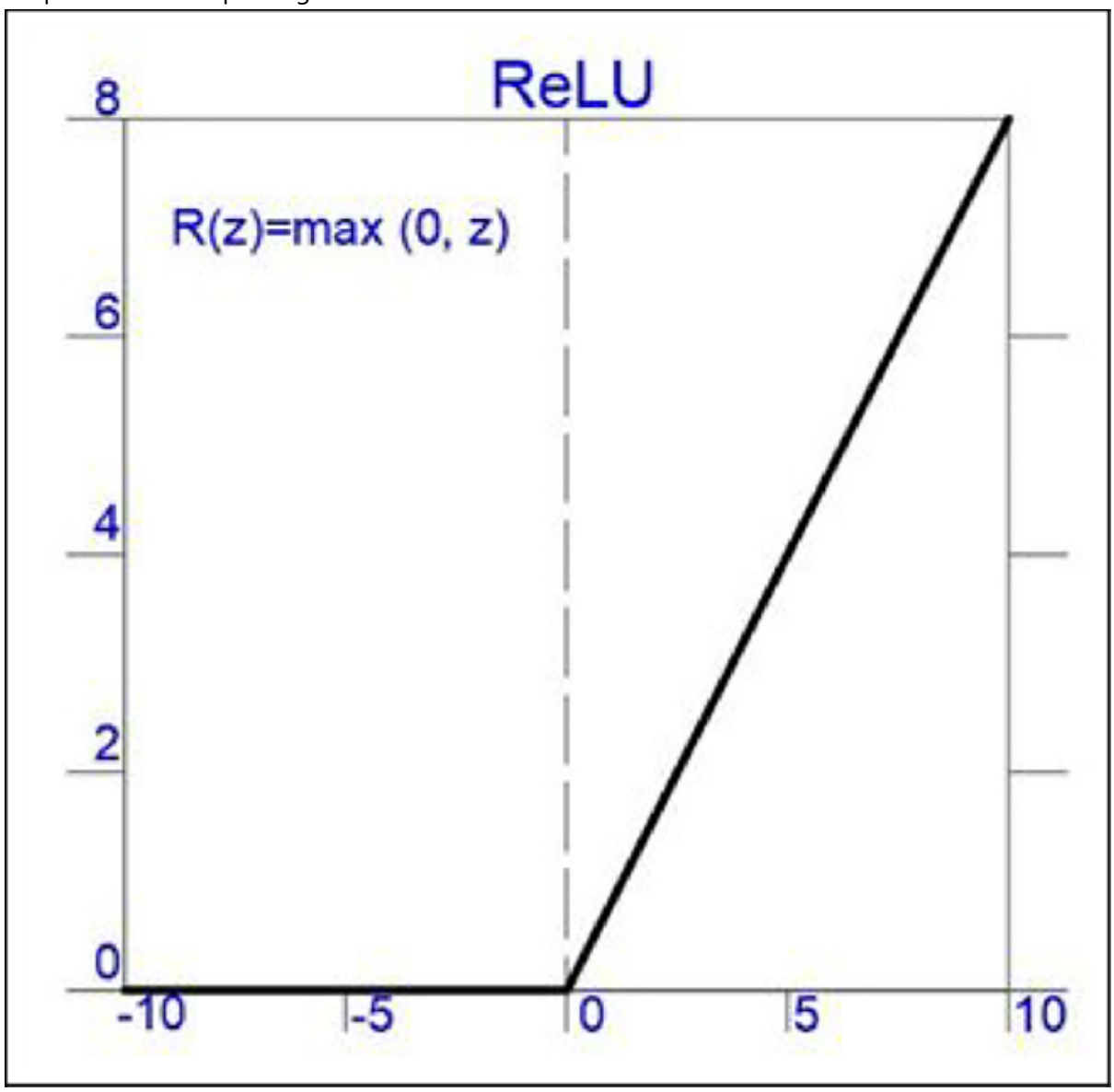
Quali sono i pro e i contro delle funzioni di attivazione? Me le può disegnare?

- 0/1.
    - se il valore è positivo ritorna 1, altrimenti 0
    - Ha il problema di non essere derivabile
- Sigmoide
    - Utile per problemi di classificazione binaria
    - Ha un problema col vanishing gradient



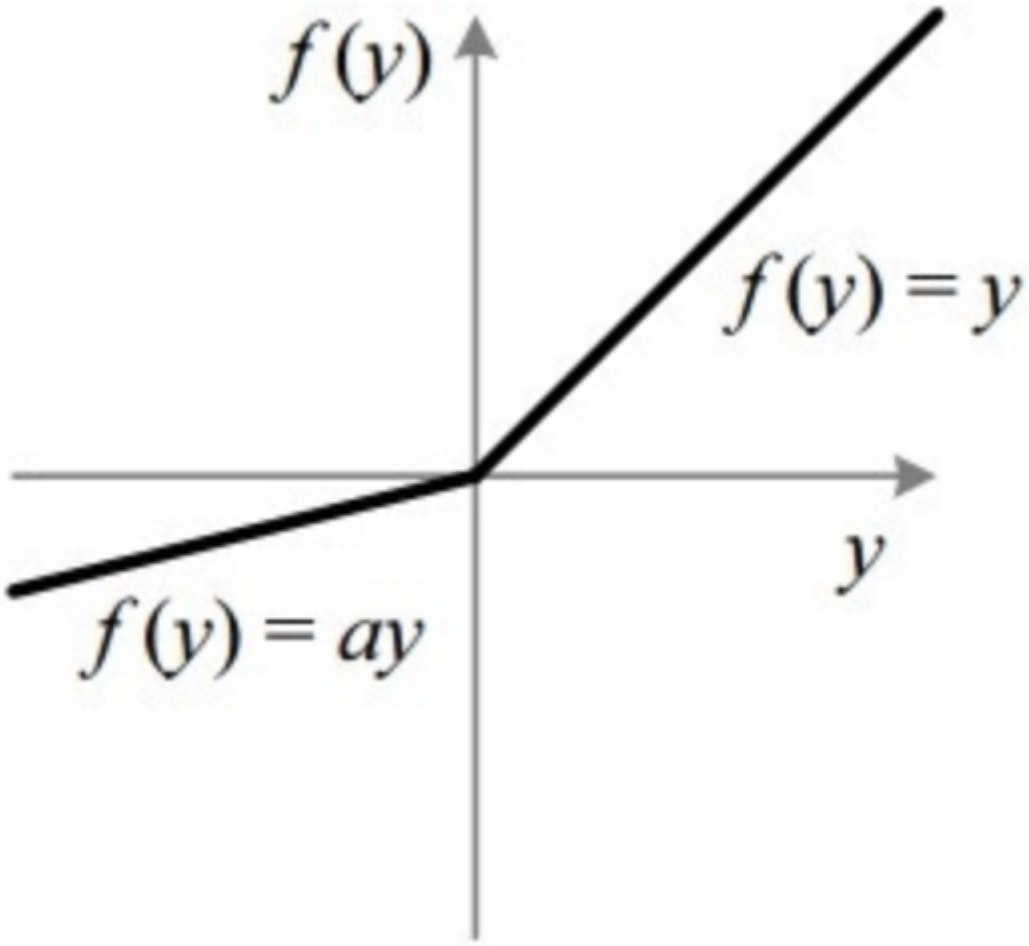
- ReLU
    - Risolve il problema del vanishing gradient

    - Ha problemi con input negativi



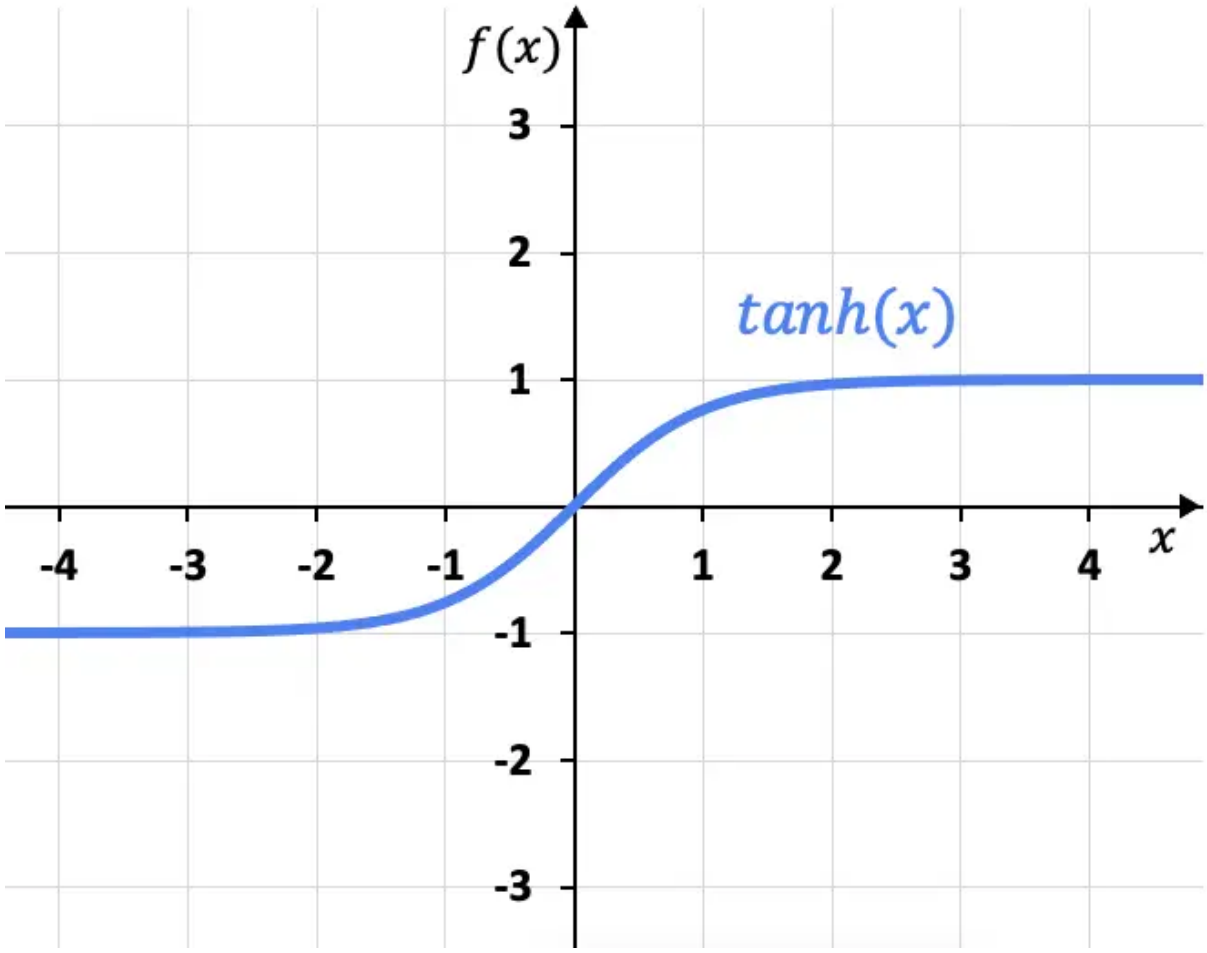
- Leaky ReLU, ELU
    - Risolvono il problema dei numeri negativi della ReLU

    - a è un iperparametro



- Tangente iperbolica
    - Utile per problemi di classificazione e regressione con l'output centrato sullo zero

- Ha problemi col vanishing gradient peggiori della sigmoide

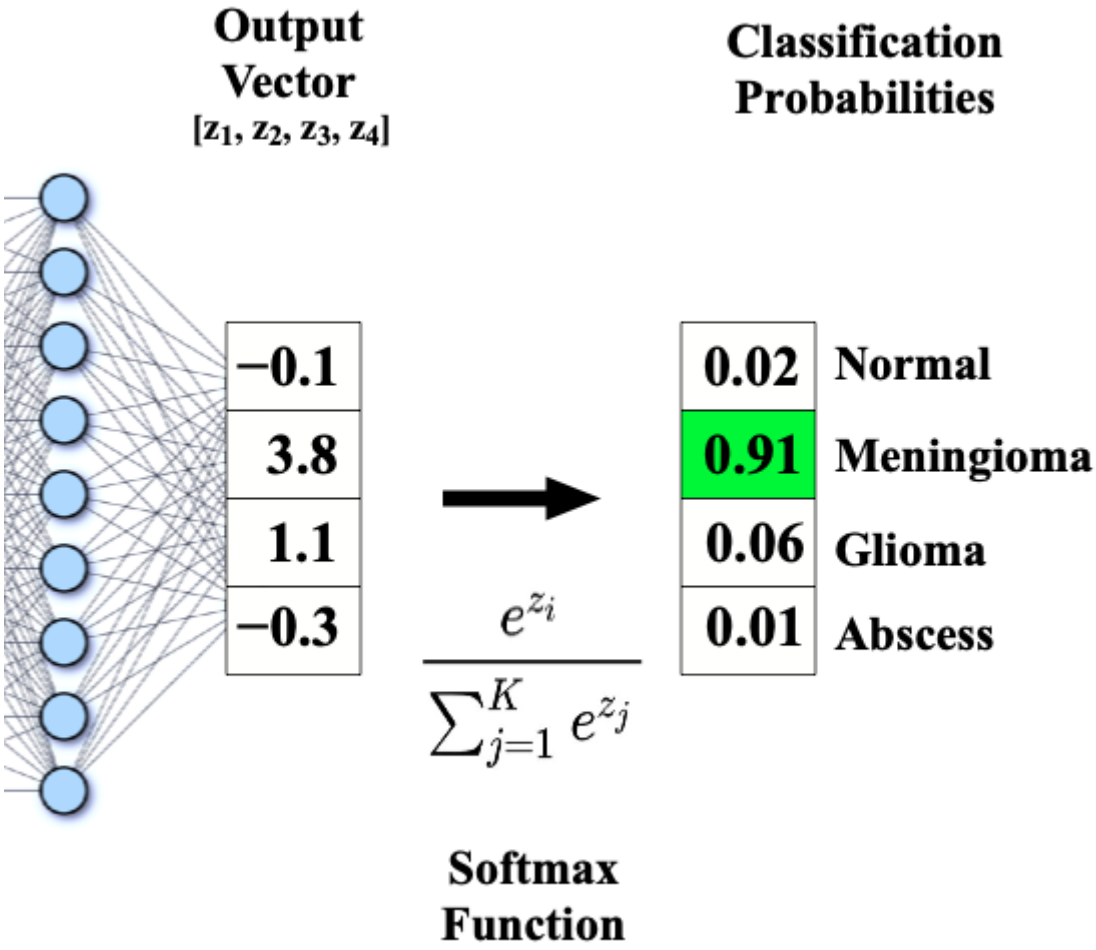


Cos'è il softmax?

È una funzione che converte un vettore di valori reali in un vettore di probabilità in modo che la somma degli elementi del secondo vettore sia 1.

- Generalmente utilizzata per problemi di classificazione multiclasse
- Si usa quando si ha più di un output e garantisce che i singoli valori calcolati siano positivi e che se vengono tutti sommati facciano 1, usati per calcolare output più probabile in una rete neurale.

- Ha problemi col vanishing gradient se un output ha valori troppo alti rispetto agli altri.



Mi disegna una ReLU? + formule

$F(x) = max(0, x)$ Ha lo svantaggio che la funzione non è differenziabile a 0 ed è vulnerabile alla morte dei neuroni nelle NN, se la funzione per qualche motivo arriva a valori negativi o a 0, il neurone non cambierà più i valori dei pesi associati ad esso perché avrà la derivata nulla. Per affrontare questi problemi si possono usare funzioni basate sulla ReLU come la Leaky ReLU e la Randomized Leaky ReLu

Mi descriva la Leaky ReLU

Risolve il problema della morte dei neuroni della ReLU che avviene quando in input vengono forniti tanti valori negativi, questo causa all'output di essere 0 indifferentemente dal valore assoluto, arrestando il flusso dei gradienti, la Leaky ReLU risolve il problema moltiplicando i valori negativi per un iperparametro a generalmente molto piccolo

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

la ReLU risolve il problema del vanishing gradient per i valori positivi,però non va utilizzata quando si hanno valori negativi in quanto porta alla morte dei neuroni e una leaky ReLU sarebbe preferita

Decision trees

Cosa sono i DT?

Sono modelli non parametrici che predicono il label di un input attraverso una struttura ad albero.

Mi farebbe un esempio di ensemble learning?

Random forest, vengono creati più DT partendo da sottoinsiemi diversi dello stesso training set (compresi differenti feature analizzate da ogni albero), questo migliora la accuratezza del modello a costo di un leggero aumento nel training e inference time, permette anche di fare pruning di features se ci si rende conto che eliminando alberi che tengono conto di tale feature la accuratezza non varia, la predizione finale è raggiunta tramite votazione (classificazione) o facendo la media delle predizioni (regressione)

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

L'ensamble learning viene utilizzato anche in generale, consiste nel utilizzare risultati ottenuti da più modelli per evitare l'overfitting.

Come si può evitare l'overfitting nei DT?

- **Ensembling**
- **Regularization**
    - **Limitare la profondità dell'albero**
    - **Mettendo un minimo al numero di valori che una foglia può avere**
    - **Mettendo un limite al numero di features usate durante uno splitting**
    - **Riduzione della complessita**
        - **Pruning**: rimuovendo i rami che forniscono pochi benefici predittivi, può essere fatto durante la fase di training limitando la crescita dell'albero, o dopo eliminando i rami in base a dei criteri.
        - **Early stopping**: limitare il numero di iterazioni dell'algoritmo
    - **Data augmentation**: rimuovendo feature inutili.

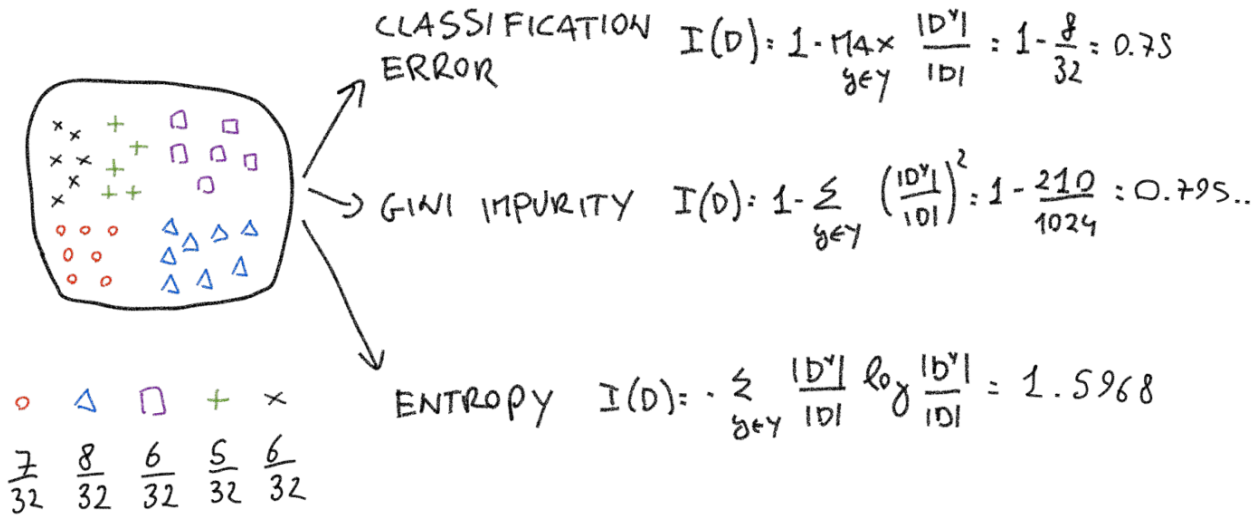
Come si decide se far crescere una foglia?

Crescendo il DT continua a separare il training set in sotto insiemi in cui la misura di impurità continua a diminuire, una volta raggiunto un valore di impurita abbastanza basso il modello crea una foglia al posto di un nodo decisionale. In caso ci fossero delle limitazioni sulla profondita dell'albero, si creerebbe automaticamente una foglia.

Cos'è l'impurita nei DT?

È la misura della eterogeneità del sottoinsieme del training set che sono arrivati ad un nodo del DT. Ci sono diversi metodi per calcolarla, anche in base al tipo di modello che vogliamo creare.

- Classificazione



- Regressione
  - Varianza
    - For regression $\mathcal{Y} \subset \mathbb{R}^d$ and $\mathcal{D} \subset \mathcal{X} \times \mathcal{Y}$
    - If $\ell(f;(x,y)) = \|f(x) - y\|_2$ and $\mathcal{H}_{\text{leaf}} = \cup_{y \in Y} \{y\}^{\mathcal{X}}$ then the impurity measure is the variance:

$$I(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{(x,y) \in \mathcal{D}} \|x - \mu_{\mathcal{D}}\|^2$$

$$\mu_{\mathcal{D}} = \frac{1}{|\mathcal{D}|} \sum_{(x,y) \in \mathcal{D}} x$$

Il DT serve solo per fare classificazione?

No, possono essere utilizzati anche per la regressione (l'output non è un label, ma un numero reale calcolato ad esempio prendendo la media dei valori della foglia in cui risiede la predizione)

Dove si trova la ricorsione in un DT?

Durante la fase di inferenza, se un nodo è una foglia ritorna la predizione, se è un nodo decisionale passerà l'input attraverso la condizione del nodo per vedere il prossimo nodo in cui deve passare. Un discorso simile si può fare per il training dove il processo di decidere un separatore può essere applicato ricorsivamente a ogni nodo generato.

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 functions vengono usate nei nodi decisionali per indicare all'algoritmo dove proseguire sull'albero. ES `if x > 5 L; R`, se x è maggiore di 5 allora si deve proseguire a sinistra se no a destra. Si possono avere split functions oblique, queste funzioni considerano più features alla volta, creare un DT con anche funzioni oblique è piu difficile da ottimizzare

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

Nei nodi non terminali troviamo le split functions, mentre nelle foglie (nodi terminali) troviamo una prediction function.

Cosa sono le split functions?

Separano in 2 o più sottoinsiemi disgiunti l'insieme in input, l'unione del sottoinsiemi ottenuti dovrà dare l'input originale, e ogni sottoinsieme verra passato ai nodi successivi nell'albero.

I DT si possono usare per feature discrete?

Si

Multiclass classification

Mi farebbe un riassunto dei due approcci della multiclass classification?

- OVA (One vs All), avremmo K classificatori, ognuno classifica un data point come appartenente o meno ad una classe, c'è la possibilità di avere ambiguità, facilmente risolvibile se il classificatore offre anche la confidenza della predizione (si sceglie la più confidente classificazione positiva, se nessuna classificazione è positiva allora la meno confidente negativa).
- AVA (All vs All) avremmo $\frac{K(K-1)}{2}$ classificatori, ogni classificatore distingue l'appartenenza di un data point ad una classe o all'altra (stile torneo, classificazioni 1v1), una volta classificato il datapoint con ogni classificatore, si vede quale classificazione è la più comune per vedere il label da assegnare, ci sono anche versioni in cui viene presa in considerazione anche la confidenza dei classificatori oltre alla popolarità dei label.

Mi faccia un contronto tra OVA e AVA

- OVA
  - Maggiore velocità di inference
- AVA
  - Con etichette bilanciate (simile cardinalità fra tutte le etichette all'interno del training set) più veloce a fare training
  - Impara più classificatori
  - Maggiore possibilità di errori (dato il numero di classificatori)

Com'è possibile calcolare l'accuratezza di una predizione?

- Microaveraging: average over examples $Ac = \frac{Pc}{Pc+Pe}$ Ac = accuratezza Pc = predizioni corrette Pe = predizioni errate

- Macroaveraging si calcola l'evaluation score per ogni singolo label (così si può vedere l'accuratezza rispetto ai singoli label e non si hanno problemi causati dalla rarezza di certi label) $Evi = \frac{Pc}{Pc+Pe}$ EVi = Evaluation per un label i Pci = predizioni corrette Pei = predizioni errate $Ac = \frac{\sum_{i=0}^{n} Evi}{n}$ Ac = accuratezza n = numero di label

Cos'è la confusion matrix?

Una matrice quadrata, con lato il numero di label possibili, ogni casella i, j indica il numero di esempi del label i predetti come label j, il modello è un buon modello se la diagonale ha i numeri più alti.

Gradient descent

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

Il gradient descent è il metodo che viene utilizzato per trovare un minimo locale nella loss function di un modello, il learning rate è un iperparametro usato per stabilire quanto velocemente i parametri della funzione cambieranno per arrivare ad un minimo locale. È buon uso cambiare il learning rate nel tempo iniziando con uno alto per avvicinarsi il più velocemente ad un minimo per poi abbassarlo dopo ogni iterazione per evitare che i parametri del modello si spostino oltre il minimo.

Come si calcola il gradiente?

Il gradiente è il vettore del rapporto fra derivata della loss function rispetto a tutte le derivate sui parametri dell'output, ogni componente di questo vettore indicherà in che modo un parametro del modello influisce sul risultato finale. Per calcolare il gradiente delle feature di input nelle reti neurali viene usata la backpropragation.

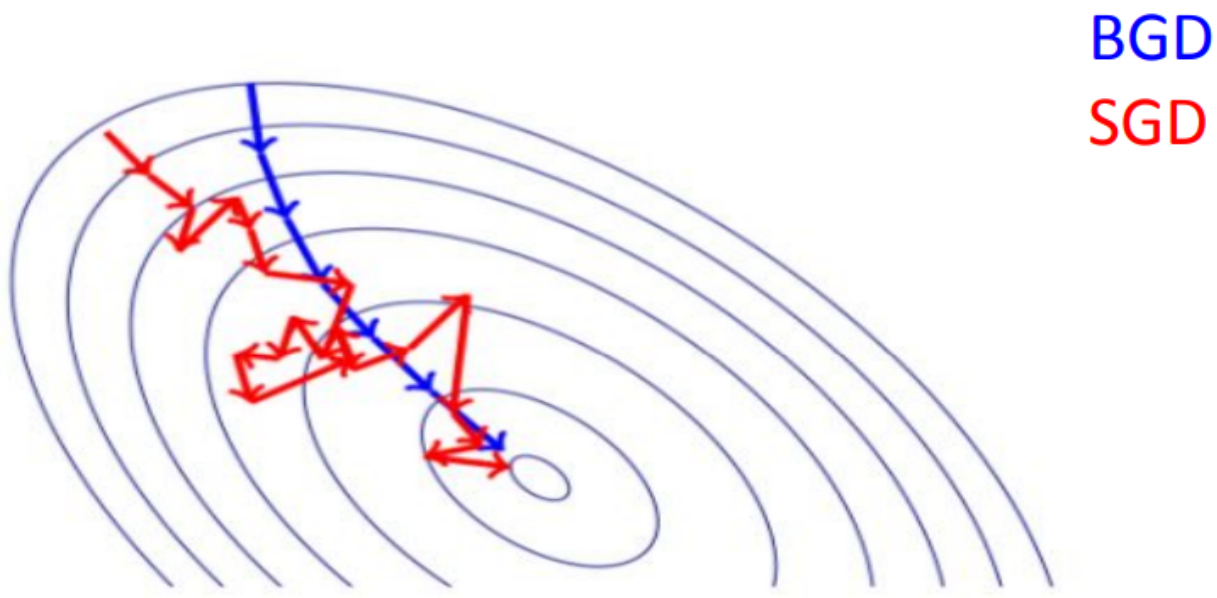
Cos'è il batch gradient descent?

Invece di aggiornare i pesi in base ad un solo data point, il batch gradient descent prende tutto il training set, ne calcola il vettore dei gradienti per ogni data point e poi per ogni posizione ne fa la media, l'aggiornamento averrà seguendo il vettore della media dei gradienti.

Cos'è lo stochastic gradient descent?

Ogni volta viene pescato a caso un data point, viene calcolato il vettore dei gradienti e aggiornato il modello, ha dei vantaggi dal punto di vista dell'efficenza (molto più veloce rispetto a calcolare la derivata per un dataset che può anche contenere milioni di datapoints), ma la sua "discesa" verso un minimo locale è molto irregolare. C'è anche una versione in cui al posto di prendere un singolo componente del dataset vengono

presi dei sotto insiemi chiamati mini-batch dello stesso.



Cos'è il momento?

Il momento è un vettore usato nel SGD che contiene la media pesata dei gradienti passati, dando più importanza ai gradienti utilizzati recentemente, così all'inizio del training il modello può "accelerare" verso un minimo riducendo la irregolarità del percorso, inoltre permette alla GD di essere meno suscettibile ai saddle points

Mi scrive 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 paso $t-1$ moltiplicato per un iperparametro γ che indica quanta importanza dare al momento precedente (per forza coimpreso 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

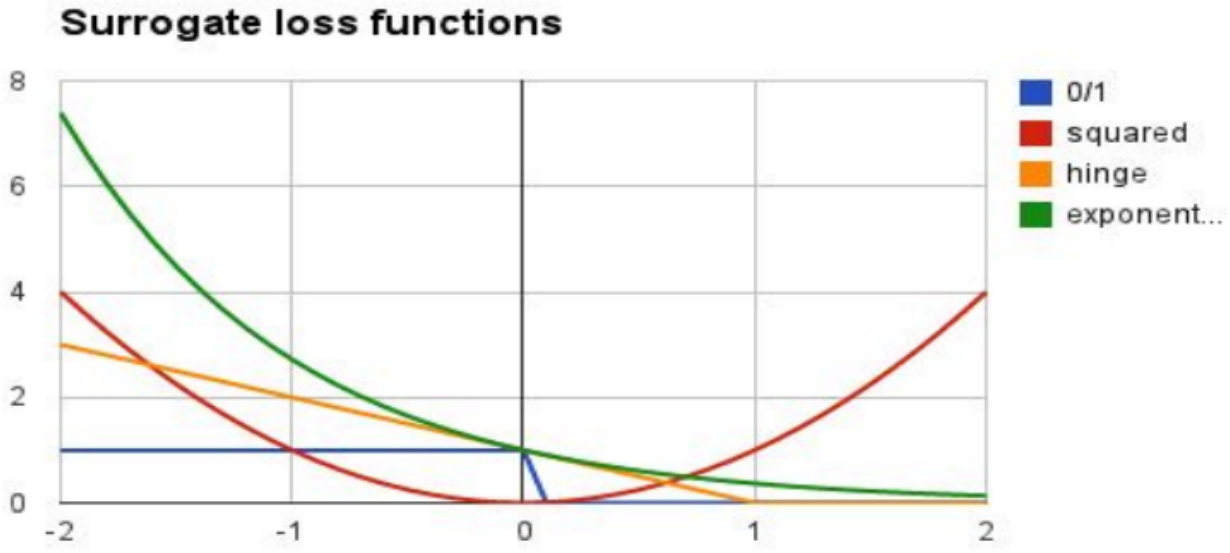
A che cosa serve una loss function? Mi fa un paio di esempi?

Viene utilizzata per misurare quanto è distante l'output ottenuto da quello desiderato, l'obiettivo della fase di training sarebbe quello di minimizzare il valore che otteniamo inserento l'output del modello nella loss function che abbiamo scelto. Una loss function ideale dovrebbe essere differenziabile ovunque, avere un minimo globale e nessun'altro minimo locale e misurare la distanza dell'output dal label reale, non esistendo però una funzione del genere, si usano funzioni che sono convesse (la retta che collega 2 punti appartenenti alla curva sarà sempre sopra la curva stessa) e surrogate (la funzione è un upper bound per la reale loss function)

Es. i è il valore corretto, j è il valore stimato

- 0/1 loss(non è convessa): $L(i,j) = 1[ij <= 0]$ non viene più usato perché non è convesso, infatti ottimizzare una funzione del genere è un problema NP hard

- Hinge loss: $L(i,j) = max(0, 1 - ij)$ utilizzata nelle SVM e nei problemi di classificazione binaria in generale, da notare che non è differenziabile a f(x)=1
- Exponential loss: $L(i,j) = e^{-ij}$ più sensibile ai valori negativi e agli outliers
- Square loss: $L(i,j) = (i-j)^2$ usata in generale, va bene più o meno ovunque anche se più lenta



Cos'è la loss function?

Viene utilizzata per misurare quanto è distante l'output ottenuto da quello desiderato, l'obiettivo della fase di training sarebbe quello di minimizzare il valore che otteniamo inserendo l'output del modello nella loss function che abbiamo scelto

Nelle reti neurali si usa una precisa loss, quale?

Solitamente si tende a usare la square loss (L2 loss) per via della sua propensione a punire fortemente errori grandi, se però si ha una distribuzione iniziale prona ad avere outliers la square loss è sconsigliata in favore della mean absolute error function (L1 loss, la square loss ma l'esponente è uguale a 1 al posto di 2)

Come diventa la loss function quando si aggiunge il regolarizzatore?

Viene aggiunto la funzione di regolarizzazione moltiplicata per il λ che è un iperparametro che indica quanto il regolarizzatore sia influente.

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

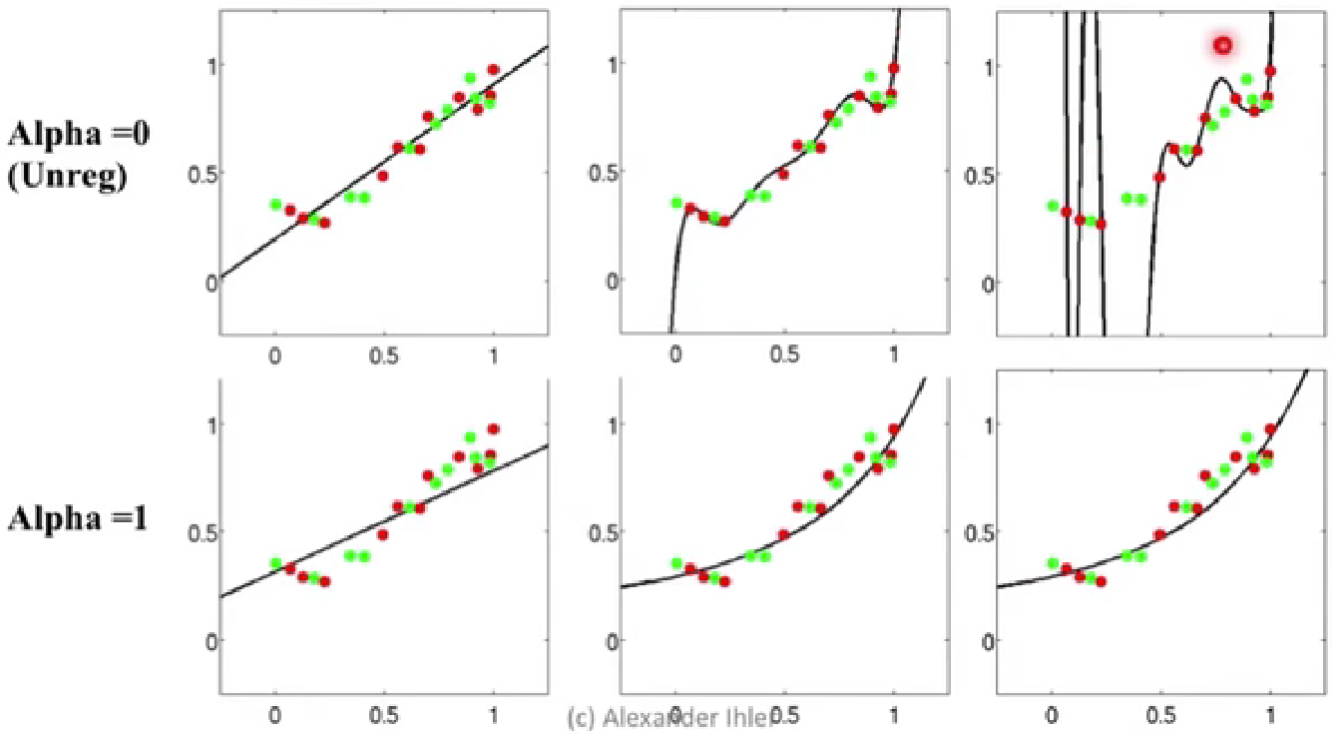
Le loss possono essere usate insieme al gradient descent?

L'obiettivo del gradient descent è quello di trovare un minimo locale nella loss function modificando i parametri del modello, quindi si

Regolarizzazione

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

È un processo per il quale si tenta di evitare l'overfitting, si aggiunge un regolarizzatore alla loss function per mantenere regolari i valori dei pesi evitando di averne di troppo grandi creando funzioni troppo complesse.

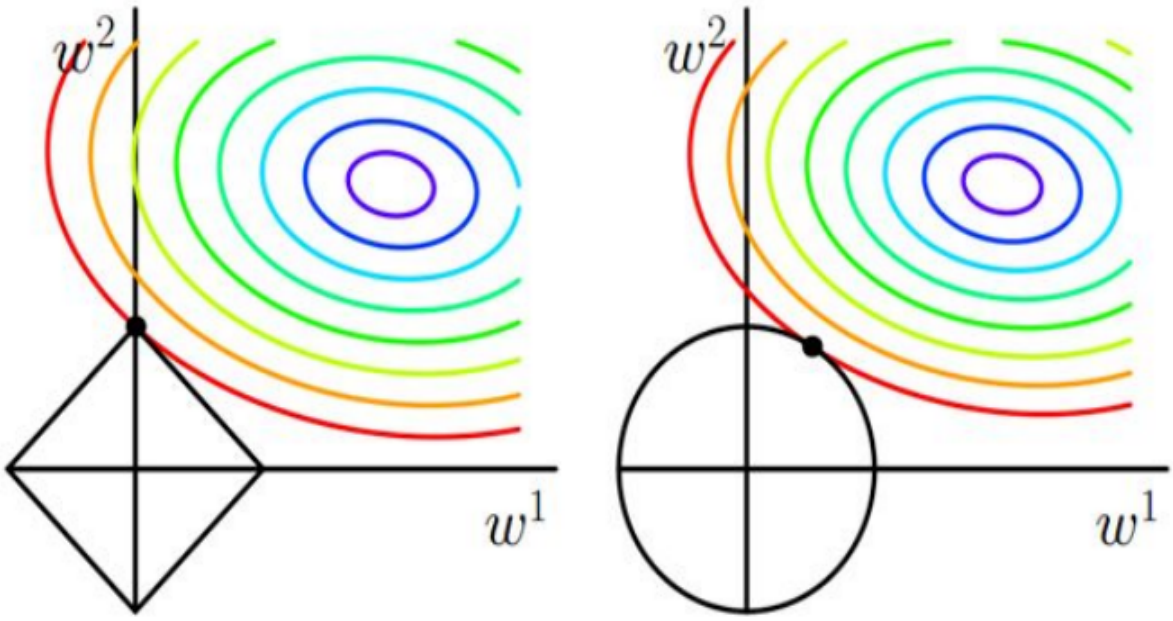


in questa immagine con alpha=0 si ignora il regolarizzatore, andando così a creare funzioni complesse che fanno di tutto per seguire i punti del training set (i rossi) e creano una situazione di overfitting, con alpha=1 invece si tiene conto del regolarizzatore e si ha una funzione molto più adatta a risolvere il problema generale.

Es.

- L1: somma dei valori assoluti dei pesi

- L2: somma dei quadrati dei pesi



(a) ℓ_1-ball meets quadratic function. ℓ_1-ball has corners. It's very likely that the meet-point is at one of the corners.

(b) ℓ_2-ball meets quadratic function. ℓ_2-ball has no corner. It is very unlikely that the meet-point is on any of axes."

Che differenza c'è tra L1, L2 e Lp?

Tutti e tre sono metodi di regolarizzazione p-norm

- L1: tende ad avere pesi uguali a 0 creando funzione sparse, questo comporta la creazione di funzioni che hanno un uso più efficente dello spazio, più efficenti da calcolare e la possibilità di vedere quali pesi siano meno importanti.
- L2: è molto raro che risulti in funzioni sparse ma è più convessa della L1 il che facilita il training.
- Lp: generico per regolarizzatore p-norm, si tende a usare solo $p >= 1$ perché sono le uniche p-norm convesse, più la p è alta più alti saranno i valori dei parametri.

Support vector machines

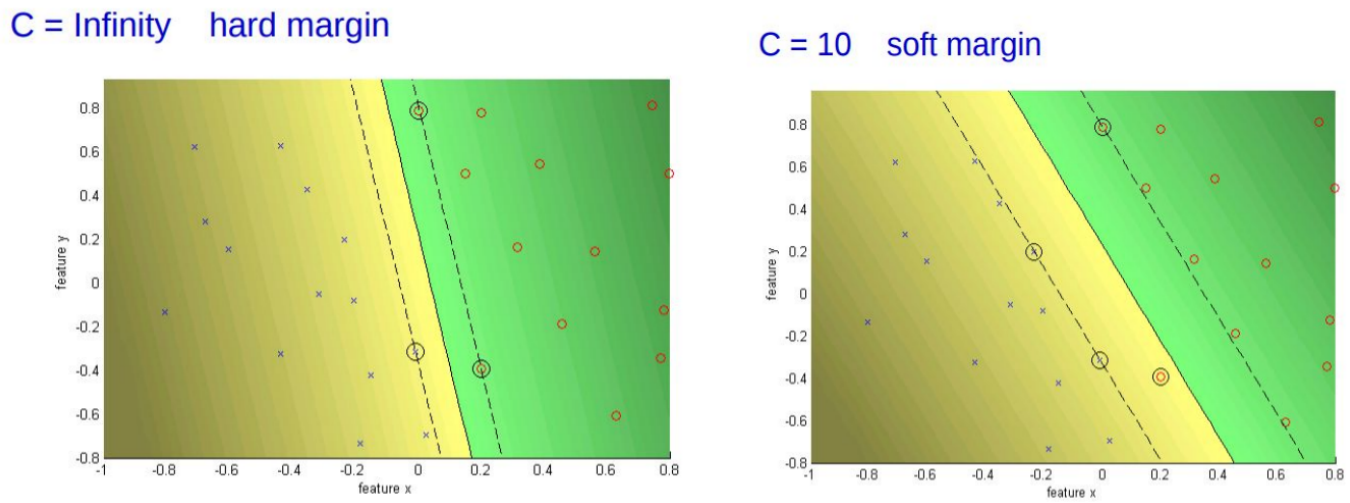
Cosa sono le SVM?

Come il perceptron è un modello di apprendimento supervisionato e un classificatore binario. A differenza del perceptron però che trova uno dei tanti separatori possibili fra i punti del training set, le SVM trovano il separatore che massimizza la distanza(margine) tra il separatore e i punti del training set.

Vi sono due modi per usarlo con 2 loss function diverse:

- hard margin suscettibile a outliers che rendono il dataset non linearmente separabile
  - $min_{w,b}||w||^2 : y_i(w * x_i + b) \geq 1 \forall i$
- soft margin: resistente agli outliers, ζ_i=slack variable dell'input i, C è un iperparametro del regolarizzatore che controlla quanto sono importanti le slack variables, più C è piccolo più grande sarà il margine, C= ∞ è un hard margin.

- $min_{w,b}||w||^2 + C\sum_i \zeta_i : y_i(w * x_i + b) \geq 1 - \zeta_i, \quad \forall i, \zeta_i \geq 0$
- $\zeta_i = max(0, 1 - y_i(w * x_i + b))$ da notare che si sta usando la hinge loss.



i punti circondati e che si incontrano con la linea tratteggiato sono i support vector, nel soft margin viene anche circondato un outlier che viene ammesso.

nel soft margin è come se stessimo aggiungendo un regolizzatore all'hard margin.

da notare che si usa l'esponente=2 per rendere il problema di ottimizzazione uno vecchio che ha avuto molti anni di ricerca dietro e che quindi sappiamo risolvere bene.

Cos'è il dual problem?

Dati i primal problems che sono difficili da risolvere possiamo invece grazie alla magia dei matematici usare in problema equivalente definito come ddual problem che è facile da risolvere e che si può usare facilmente per ricongiungersi al problema originale.

- hard margin: $max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i^T x_j$
  $s.t. \sum_i \alpha_i y_i = 0, \alpha_i \geq 0, \forall i$
- soft margin: $max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i^T x_j$
  $s.t. \sum_i \alpha_i y_i = 0, 0 \leq \alpha_i \leq C, \forall i$

data la soluzione al dual problem si può ritornare al primal problem facendo $w = \sum_i \alpha_i y_i x_i$ e $b = y_k - \sum_i \alpha_i y_i x_i^T x + b$ dove ogni $\alpha_i \neq 0$ sta a significare che il corrispondente x_i è un support vector (ovvero i punti che delimitano il margine)

Quindi la soluzione si basa sul prodotto scalare fra support vector e il nostro input mentre il training consiste nel controllare tutti i prodotti scalari fra tutti i punti nel dataset.

La funzione di classificazione finale sarà: $f(x) = \sum_i \alpha_i y_i x_i^T x + b$

Si spieghi il kernel trick.

Dato un dataset dove usare un soft margin non è abbastanza in quanto la non separabilità del dataset non è causata da del semplice noise si deve usare il kernel trick.

Questo consiste nel mappare i nostri datapoint in una dimensione più alta (un po' come se aggiungessimo delle feature) usando una funzione $\Phi : x \rightarrow \phi(x)$.

Però persone più intelligenti di noi lo hanno già ricercato in passato e quindi invece di cercare manualmente queste funzioni si usano dei kernel che sono già stati dimostrati come corretti che implementano di già le funzioni, matematicamente: $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$

Per controllare che questo venga fatto correttamente si usa il mercer theorem:

- For some functions $K(\mathbf{x}_i,\mathbf{x}_j)$ checking that $K(\mathbf{x}_i,\mathbf{x}_j)= \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j)$ can be cumbersome.
- Mercer's theorem:
  - Every positive semidefinite symmetric function is a kernel
  - A positive semidefinite symmetric functions correspond to a positive semidefinite symmetric Gram matrix:

$$K = \begin{bmatrix} K(\mathbf{x}_1,\mathbf{x}_1) & K(\mathbf{x}_1,\mathbf{x}_2) & K(\mathbf{x}_1,\mathbf{x}_3) & \dots & K(\mathbf{x}_1,\mathbf{x}_n) \\ K(\mathbf{x}_2,\mathbf{x}_1) & K(\mathbf{x}_2,\mathbf{x}_2) & K(\mathbf{x}_2,\mathbf{x}_3) & & K(\mathbf{x}_2,\mathbf{x}_n) \\ & & & & \\ \dots & \dots & \dots & \dots & \dots \\ K(\mathbf{x}_n,\mathbf{x}_1) & K(\mathbf{x}_n,\mathbf{x}_2) & K(\mathbf{x}_n,\mathbf{x}_3) & \dots & K(\mathbf{x}_n,\mathbf{x}_n) \end{bmatrix}$$

- Recap: A symmetric matrix is positive semidefinite if and only if all eigenvalues are non-negative

esempi di kernel:

- Linear: $K(x_i, x_j) = x_i^T x_j$
- Polynomial of power p: $K(x_i, x_j) = (1 - x_i^T x_j)^p$
- Gaussian (radial-basis function): $K(x_i, x_j) = e^{-\frac{||x_i - x_j||^2}{2\sigma^2}}$
  - Mapping Φ: $x \rightarrow \phi(x)$, dove $\phi(x)$ ha dimensioni infinite

Con questi non serve più salvare il dataset ma solo la kernel function, ovvero:

- formulazione del dual problem: $max_\alpha \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j K(x_i, x_j)$
  $s.t. \sum_i \alpha_i y_i = 0, \alpha_i \geq 0, \forall i$
- funzione di classificazione: $f(x) = \sum_i \alpha_i y_i K(x_i, x) + b$
  il resto rimane identico.

Ma i kernel si possono usare anche su dataset e input diversi da punti in un iperspazio ma anche su dati strutturati come i grafi, sono stati studiati modi per trattarli con la kernel function.

Come si riconoscono i support vector in funzione delle alpha?

se le alpha sono diverse da 0 allora il corrispondente punto sarà un support vector.

Quali sono i pro e i contro dei SVM?

Le SVM sono molto efficenti nel trattamento di spazi ad alta dimensionalità, hanno una buona generalizzazione e resistenza a noisy dataset, gestiscono bene grandi set di dati e hanno una buona performance anche con un numero limitato di dati, hanno però problemi con la dipendenza dalla scelta dei parametri, sulla complessità computazionale e su dati non linearmente separabili (non sempre risolvibile aumentando la dimensione dei parametri).

Perché SVM sono migliori del perceptron?

Un perceptron trova uno dei tanti iperpiani che possono dividere il training set nei 2 label, mentre l'SVM trova l'iperpiano che massimizza la distanza tra il piano e i data points, questo perché è stato matematicamente dimostrato che più è grande il margine più è probabile che la suddivisione sia corretta, inoltre usando il soft margin si ha una resistenza ai noisy dataset e con il kernel trick alle volte si può risolvere il problema dei dataset non linearmente separati.

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

L'unico modo e andare a tentativi su kernel pre esistenti controllando quanti errori sono stati compiuti dal validation set (facendo k-fold cross validation). Se invece vogliamo provare a creare un kernel da 0 per vedere se il kernel è valido si controlla usando il mercer theorem.

Il kernel si usa solo nelle SVM?

No, vengono usati anche nelle PCA e nelle CNN.

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

Il margine indica la distanza dei punti più vicini all'iperpiano nel dataset originale (outliers ignorati), questi prendono il nome di support vectors, solo loro verranno consideranti al tempo di inferenza dato che definiscono l'iper piano, inoltre sono sempre uno in più rispetto alle dimensioni del problema

Che tipo di regolarizzatori utilizzano le SVM?

Il regolarizzatore usato dalle SVM non è altro che $C \sum_i \zeta_i$ che viene usato nel softmargin.

SVM Soft margin

Perché vengono aggiunte le slack variables?

Le slack variables sono una penalità che vengono aggiunte per avere un soft margin, così è possibile classificare anche dataset non linearmente separabili (se hanno noise), permettono al modello di commettere errori pesandoli poi sulla objective function, si aggiunge la somma delle slack penalties moltiplicata per C. C è un iperparametro che indica quanto far incidere gli errori sulla funzione, con C basso si avranno margini più larghi mentre con C alto i margini saranno più stretto.

Qual è il range delle slack variables?

Per gli esempi correttamente classificati al di fuori del margine o sul margine il valore è 0, per gli esempi correttamente classificati all'interno del margine sarà la distanza dal punto alla linea del margine e sarà calcolata come un valore tra 0 e 1, per gli esempi classificati come errore è dato dalla somma della distanza dell'iper piano più la distanza dal margine (ancora tra 0 e 1). Matematicamente
$\zeta_i = max(0, 1 - y_i(w * x_i + b))$.

Partendo dalle slack variables, come otteniamo una hinge loss?

la forma che hanno le slack variables è una hinge loss per il singolo datapoint, infatti
$\zeta_i = max(0, 1 - y_i(w * x_i + b))$.

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

Con il soft margin siamo sicuri di trovare una soluzione, anche in casi in cui i dati non siano linearmente separabili inoltre ignoriamo datapoint che potrebbero essere causati da del noise nel dataset originale. Con l'hard margin invece la regione all'interno dei margini è sempre vuota.

Neural networks

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

Attraverso l'algoritmo di backpropagation sono capaci di risolvere problemi molto più complessi rispetto ai metodi classici in quanto le NN classiche non sono altro che un insieme di perceptron che lavorano assieme, è ovvio quindi che rispetto all'usare un singolo perceptron averne di multipli permette di approcciare problemi molto più complessi ignorando il problema della separabilità lineare delle SVM e dei perceptron e ignorando anche la curse of dimensionality della KNN.

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

È preferibile la sigmoide perché la iperbolica è più suscettibile al vanishing gradient. La iperbolica si può utilizzare quando si sa che si è vicini al modello ideale in quanto quando ci si avvicina allo 0 si ha un gradiente maggiore rispetto alla sigmoide.

Nelle reti neurali si usa una precisa loss, quale?

Per problemi di regressione si utilizza la L2-norm (square loss) mentre per la classificazione si utilizza la cross-entropy function che si basa su softmax: S(l_k)= softmax dell'output x, y_k = valore del parametro k del test
Cross-entropy = -$\sum_k y_k log(S(l_k))$

Convolutional neural networks

Come uso una convoluzione in una rete neurale?

In una NN invece di avere solo hidden layer composti da percepetron se ne aggiungono o sostituiscono altri composti da dei filtri convoluzionali.

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

Si vanno a estrarre dall'input delle informazioni spaziali sullo stesso, andando spesso a ridurre la dimensione dell'output rispetto all'input senza perdere informazioni a noi interessanti.

Cosa succede dentro a un layer convoluzionale?

Un layer convoluzionale è diviso in 3 operazioni:

1. Convoluzione: si ha un kernel o filtro che viene applicato in una sliding manner (a seconda di iperparametri come lo stride o se si usa o meno il padding) usando l'operazione di convoluzione (e aggiungendoci un bias costante alla fine)andando così a creare una multidimensional feature map che sarà spesso più piccola dell'input originale.

2. Non-Linearità: viene applicata a ogni dato dentro la multidimensional feature map una funzione di attivazione (solitamente una normale ReLU) aumentando la non-linearità del modello, in questo modo si aumenta la generalizzazione della NN.

3. Pooling: Viene applicato nuovamente un nuovo filtro, ma stavolta viene preso ad esempio solo il valore più alto (max pooling) o la media dei valori (mean pooling) della area di interesse per l'output (receptive field) e si fa in modo che ogni cella dell'input venga sottoposta alla convoluzione solo una volta (selezionando lo stride corretto) per diminuire nuovamente la dimensione dell'output.

Che grado di libertà ho quando genero un layer convoluzionale?

Per pooling:

- Dimensioni del filtro (2x2,3x3, ecc...)
- Tipo di filtro (massimo, minimo, media, etc...)
- Funzione di attivazione (funzione che avviene dopo la convoluzione e prima del pooling)

Per convoluzioni:

- Numero di filtri (si possono fare più convoluzione alla volta)
- Dimensioni del filtro (2x2,3x3, ecc...)
- Stride (di quante celle dell'input si slida quando si applica la convoluzione, valore naturale)
- Funzione di attivazione (funzione che avviene dopo la convoluzione e prima del pooling)
- se utilizzare o meno pooling e qui tornano i gradi di libertà del pooling

Cos'è lo stride?

Di quante celle dell'input si slida quando si applica la convoluzione, valore discreto, per valori più grandi di stride la matrice di output sara minore

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

Il pooling è un operazione deterministica che avviene alla fine di un layer convoluzionale che riduce la dimensione dell'immagine di input, rendendo le rappresentazioni caratteristiche più compatte e concentrando neuroni futuri del modello su informazioni più rilevanti

C'è apprendimento nel pooling?

No, il pooling serve solo a ridurre le dimensioni dell'output.

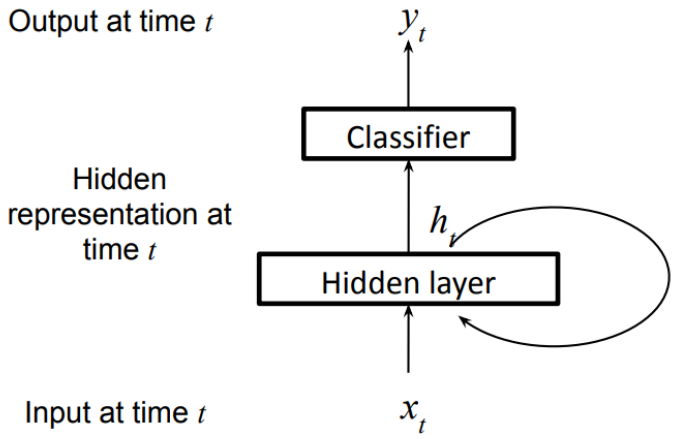
Con lo stesso dataset, cosa succede se utilizzo SVM e CNN?

Con SVM avremo più successo se il dataset è linearmente separabile, con CNN invece se gli input contengono informazoni spaziali utili si avrà un maggior successo.

Recurrent neural networks

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

Sono NN progettate per elaborare sequenze di dati, mantengono uno stato interno che gli permette di tenere conto anche di dati passati per influenzare gli output successivi, risultato ottenuto grazie a layer ricorsivi, vengono usati per il riconoscimento del parlato, traduzione automatica, generazione di testo etc... in generale ambiti in cui ordine e contesto sono fondamentali.



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

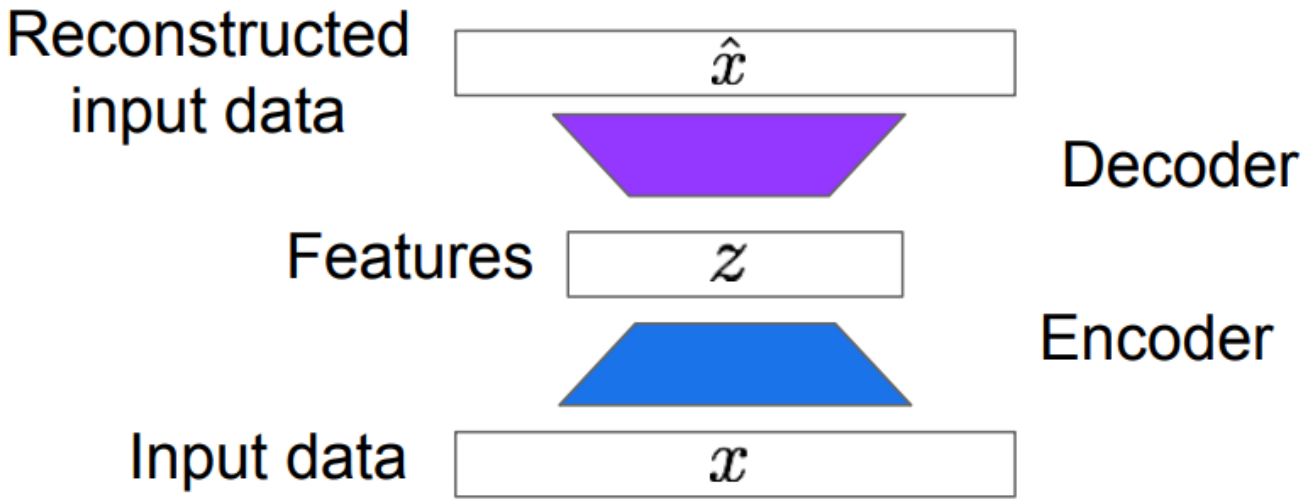
new state | function of W | input at time t | old state

Autoencoders

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

Sono un esempio di unsupervised learning per il problema della dimensionality reduction, è formato da 2 componenti, un encoder che comprime i dati forniti in input e un decoder che decomprime l'output del suo encoder, l'obiettivo è quello di comprimere i dati e poi successivamente riportarli alla dimensione originale se serve, possono anche essere utilizzati per rimuovere rumore dai dati. Spesso sono costituiti da CNN. A test time si fa un confronto fra l'input ricostruito e l'input originale per calcolare la error function, questo non

richiede labels ovviamente il che rende tutto il processo unsupervised.



Per cosa viene usato un autoencoder?

Compressione dati (Dimensionality reduction) Rimozione di rumore

Cosa succede a inference time negli autoencoders?

Viene eliminato il decoder e si utilizza l'encoder da solo ad esempio come input di un modello supervised.

È possibile utilizzare delle immagini con gli autoencoders?

Sì.

Unsupervised learning

Quali sono le differenze tra unsupervised e supervised learning?

- Supervised
  - Il training set ha dati di input e output(es. label dell'input)
  - L'obiettivo è quello di apprendere un modello che mappi gli input ai relativi output
- Unsupervised
  - Il training set contiene solo gli input
  - L'obiettivo è ad esempio trovare relazioni fra set di input (es. clusters) oppure modificare i dati di input per varie ragioni (es. autoencoder for dimensionality reduction) o generare nuovi dati dalla stessa distribuzione di quelli di input (es. GAN e VAE for density estimation)

Cos'è la dimensionality reduction?

È uno dei problemi trattati in unsupervised learning, è il tentativo di diminuire le dimensioni dell'input senza perdere troppe informazioni.

Principal component analysis

Cos'è la PCA?

La principal component analysis è una tecnica lineare usata per la riduzione di dimensioni, si applica così:

1. Il set di datapoints viene centrato (si fa la media delle posizioni in tutte le dimensioni e si considera come nuovo centro il punto risultante dalla media)
2. Creare la matrice delle covarianze tra i datapoints usando le nuove coordinate.
3. Trovare gli autovalori e gli autovettori (componenti) associati alla matrice delle covarianze.
4. Vedere i autovalori minori e decidere quali dimensioni eliminare senza perdere troppe informazioni (una perdità di dati è quasi sempre inevitabile).
5. Gli autovettori rimanenti diventano i nuovi assi di rapresentazione del dataset e si ottengono i nuovi dataset partendo dal dataset originale centrato e moltiplicandolo le matrici con gli autovettori trasposti nella colonna.

Per calcolare la perdità dell'informazione si fa $\frac{\sum_i \lambda_{1,i}}{\sum_j \lambda_{2,j}}$
dove $\lambda_{1,i} \in \Lambda_1$ con Λ_1 l'insieme degli autovalori associati agli autovettori che vogliamo rimuovere e $\lambda_{2,j} \in \Lambda_2$ con Λ_2 è l'insieme di tutti gli autovalori trovati.

Come faccio a dire che per ridurre la dimensione mi bastano le prime x componenti?

Una volta selezionati i primi x componenti (ovvero quelli che hanno maggiore densità di informazione) fai una proiezione dei vecchi datapoints (quelli che usano tutte le dimensioni) su un iperpiano formato dalle prime x componenti come assi per ricavare i nuovi datapoint con dimensioni ridotte, questo si fa moltiplicando le matrici centrate del dataset con la matrice degli autovettori trasposti nella colonna.

Che relazione c'è tra PCA e autoencoder?

Entrambi possono essere usati per dimensionality reduction e sono unsupervised

È possibile utilizzare il kernel nella PCA?

Sì usando il kernel trick si fanno le KPCA, ovvero si prende il dataset originale e si aumenta le sue dimensioni per renderlo lineare prima di applicare la PCA.

Clustering

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

- K-MEANS, Il numero di cluster e la loro forma sono (sferica) vincolati, questo è un hard clustering (gli oggetti possono appartenere solo ad un cluster). L'algoritmo inizia assegnando posizioni casuali ai K centroidi, assegna ogni data point al centroide più vicino e per aggiornare la posizione dei centroidi viene fatta la media delle posizioni degli oggetti che fanno parte dei loro cluster.
    - Per quanto garantisca una convergenza non si garantisce di trovare un minimo globale in quanto la sua loss function non è convessa ma piena di minimi locali.
    - il risultato varia molto in base ai centroidi iniziali
    - necessita di impostare una funzione per la distanza.
    - richiede di normalizzare le feature
    - ha come contro che assume che il cluster sia sferico, il che non è detto

$X = [x_1, \dots, x_n] \in \mathbb{R}^{d*n}$
$min_{C_1,\dots,C_K} \sum_{j=1}^{k} V(C_j)$
$V(C_j) = \sum_{i \in C_j} ||x_i - \mu_j^2||$
$\mu_j = \frac{1}{|C_j|} \sum_{i \in C_j} x_i$

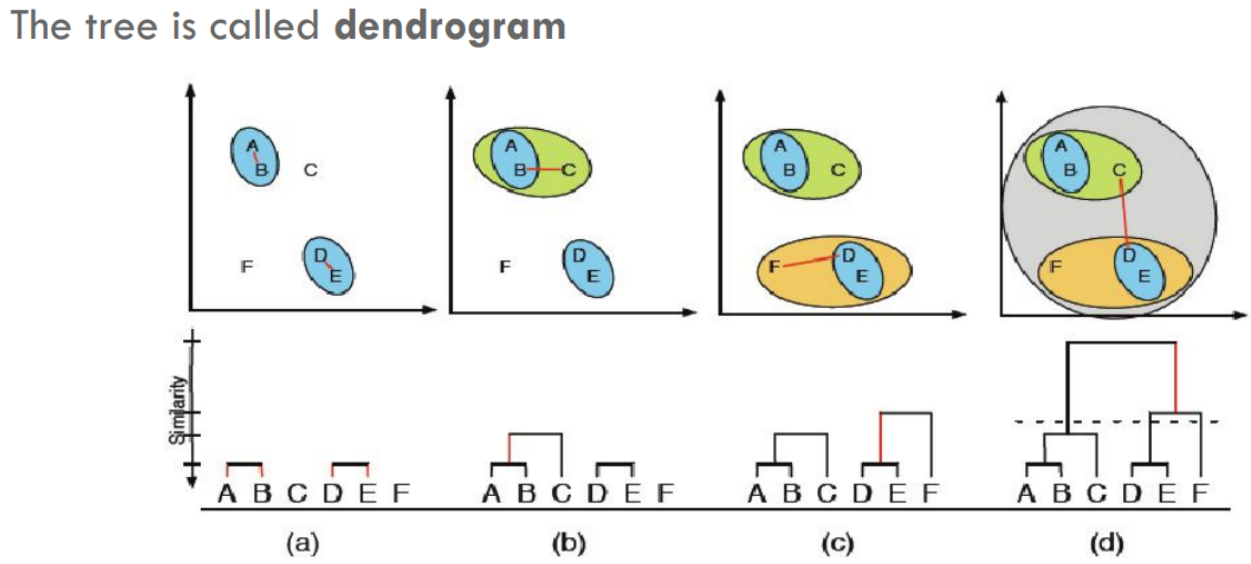
- In Expectation Maximization clustering si usa una distribuzione gaussiana che porta ad avere una forma ellittica dei cluster. È un modello di soft clustering (assegna ad ogni oggetto una probabilita di far parte di un cluster). L'algoritmo inizia posizionando casualmente i cluster, assegna le probabilita che un dato apartenga ad un cluster, si ricalcolano le posizioni dei cluster utilizzando i nuovi parametri appena calcolati, continua finché non converge.
    - migliore di k-means sotto tutti gli aspetti
    - come contro ha che assume che i cluster assumino una forma gaussiana.

$$N[x; \mu, \Sigma] = \frac{1}{(2\pi)^{\frac{d}{2}} \sqrt{det(\Sigma)}} exp[-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)]$$

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

- Spectral clustering, hard clustering, metodo che crea un grafo con alcune nozioni di similarià tra i nodi e utilizza un algoritmo che lo divide andando a creare cluster, esegue dei tagli sugli archi che collegano i nodi che hanno peso minore (collegano nodi meno simili)
    - Riesce a gestire strutture complesse, è robusto ai rumori ed è scalabile e flessibile.
    - Molto dipendente dai parametri, presenta un'elevata complessità di calcolo, una certa sensibilità alla scala dei dati (vanno normalizzati gli input) e difficoltà nella scelta del numero di cluster. Utile per distribuzioni di dati non gaussiane (k means e em non riescono a trattarle)

- Hierarchical clustering, soft clustering, crea un albero gerarchico che può essere top down o bottom up, utilizzato spesso per profili genetici. Nella versione bottom up si parte considerando tutti gli esempi come cluser e poi si uniscono i cluster più simili tra loro finche non si arriva ad avere solo un cluster. Crea così una famiglia di clusters rappresentata da un dendogramma



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

Dover sapere subito il numero di cluster (risolto con gerarchia) Essere limitati alla forma sferica dei cluster (risolto con EM)

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

Principalmente l'assunzione della forma dei cluster, essendo EM centrato su una distribuzione gaussiana risulta molto più flessibile di K-means

Density estimation

Cos'è la density estimation?

La density estimation è un metodo utilizzato per approssimare la funzione di densità di probabilità dei dati senza avere accesso alla distribuzione reale. Permette inoltre la generazione di nuovi dati dalla stessa distribuzione e l'individuazione di anomalie. I modelli generativi svolgono questo tipo di task e sono in grado di generare, per l'appunto, nuovi dati dalle distribuzioni osservate.

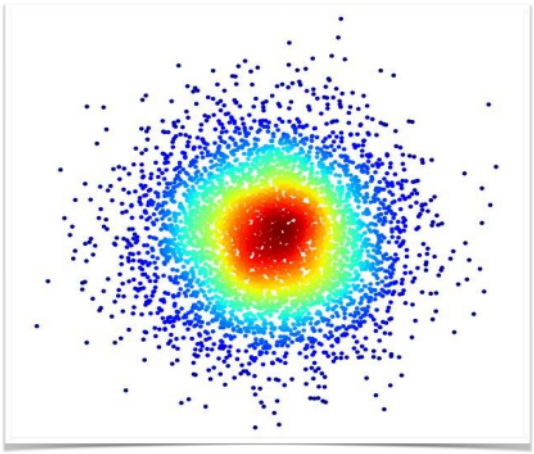
Qual è la differenza tra un modello generativo e un modello discriminativo?

- Generativo, è capace di generare dati nuovi basandosi su dati che sono gia stati osservati dal discriminatore associato
- Discriminativo, è capace di discriminare fra dati reali o dati generati da un modello generativo

Mi descriva la distribuzione di densità esplicita ed implicita

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 $p_{\text{data}} \in \Delta(\mathcal{Z})$

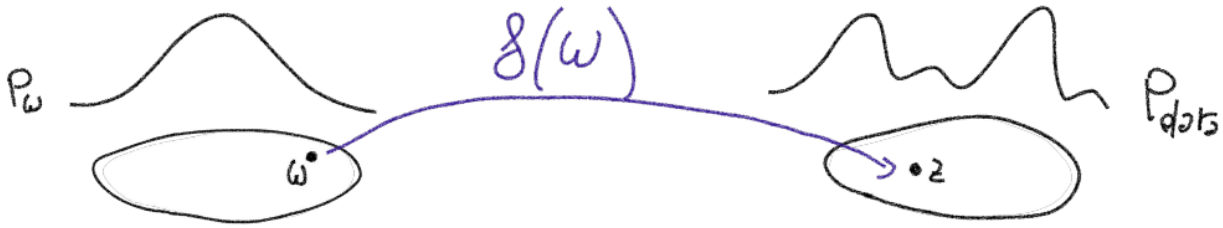


SUPERVISED LEARNING $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$

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

Density estimation (implicit)

Find a function $f \in \mathcal{Z}^{\Omega}$ 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 $p_{\text{data}} \in \Delta(\mathcal{Z})$



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

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

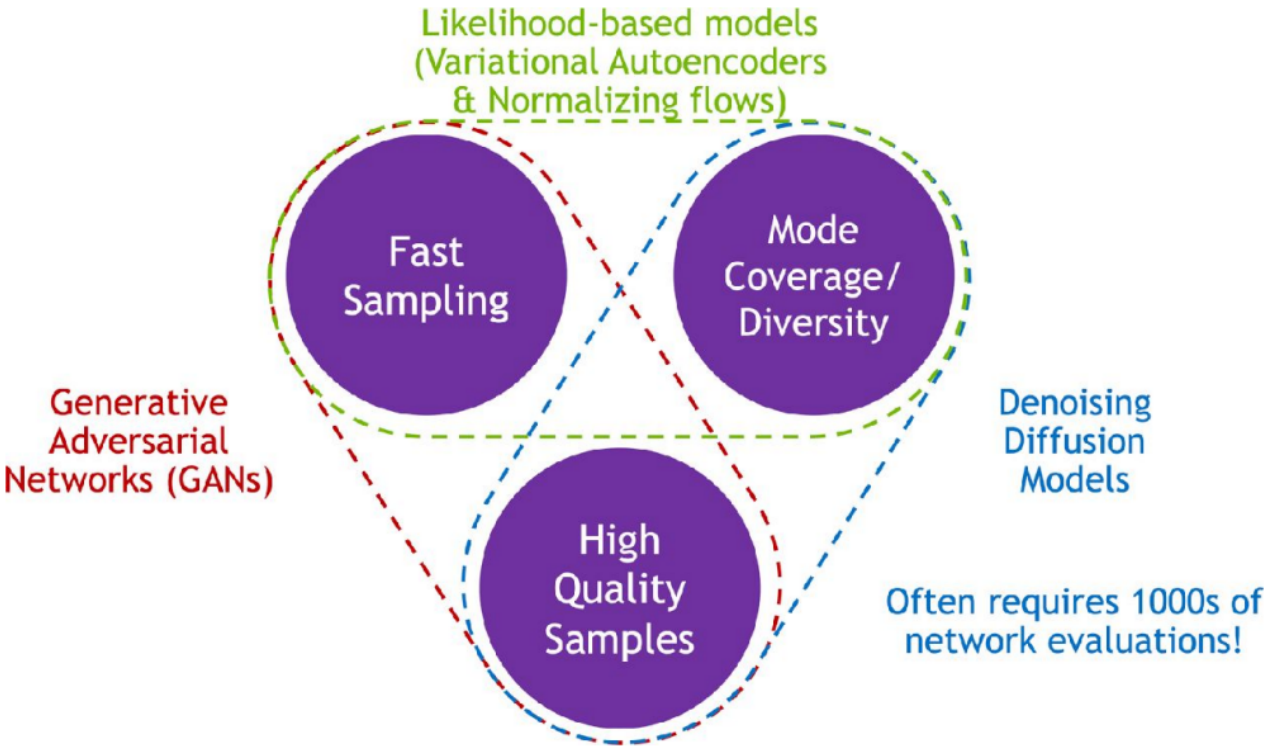
Mi saprebbe dire un'istanza di ognuna? (riferito a sopra)

VAE variational auto encoder per la densità esplicità GAN generative adversarial network per la densità implicita

VAE, GAE e diffusion models

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

questa immagine riassume i loro vantaggi e svantaggi



I VAE e i Diffusion sono modelli espliciti I gan sono impliciti I VAE e i GAN sono veloci a inference time. I VAE e i Diffusion models hanno una mode coverage elevata e stabile del spazio latente I GAN e i Diffusion models sono in grado di produrre output di alta qualità I diffusion models a differenza degli altri hanno il latent space con le stesse dimension dell'output.

I VAE sono veloci a fare sampling in quanto usano una semplice gaussiana normale come latent space su cui è facile far interpolazione.

Quale è la Loss nei VAE? e cosa significa in termine di latent space?

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

Facciamo un attimo un breakdown:

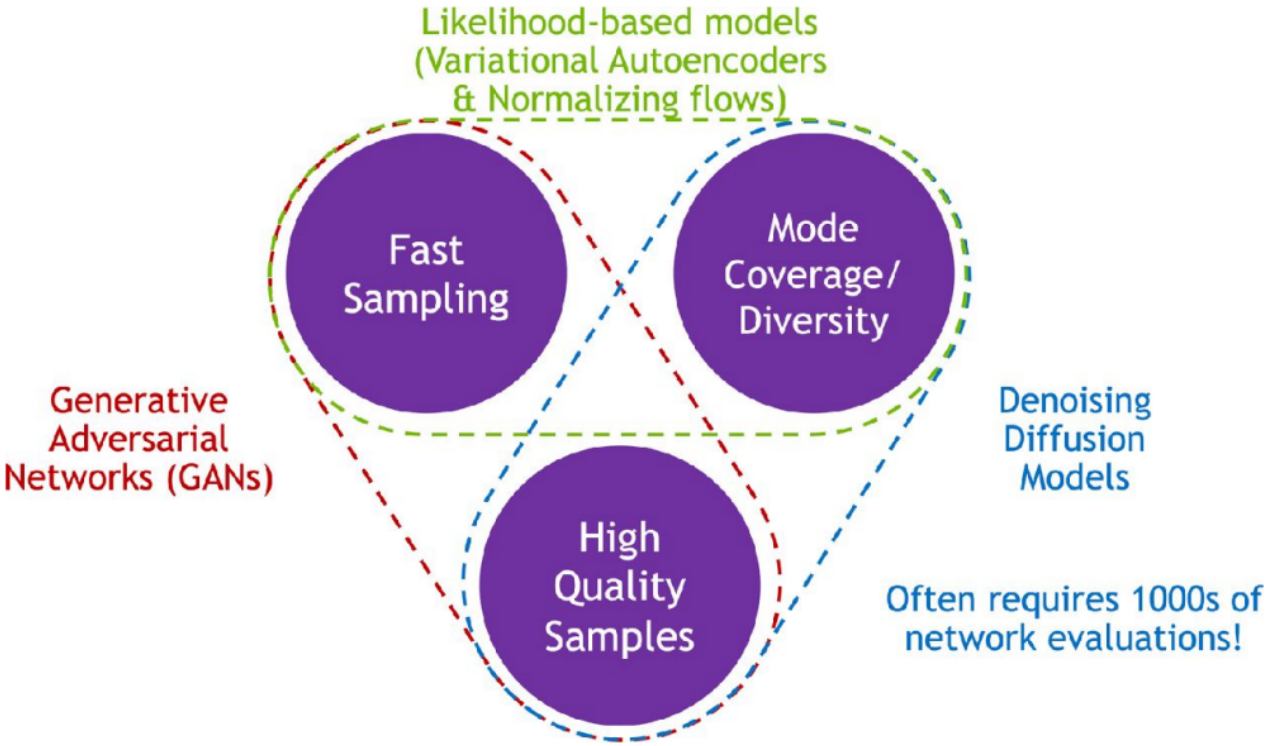
- $E_{\omega \sim q_{\psi(\cdot|x)}}[log(q_\theta(x|\omega))]$ è definito recostruction term ed è essenzialmente l'equivalente del reconstrution error di un AE, si sampla un ω dalla encoding probability distribution $q_\psi(\omega|x)$ e poi lo si dà in input al decoder che darà fuori un output, su questo 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 samplato da una gaussiana normale) e che l'output corrisponda il più possibile alla distribuzione che si vuole estimare (se si usasse solo il regolarizzatore l'output del decoder non avrebbe senso)

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

Perché le GAN soffrono di stabilità del training (è facile che i paramtri oscillino e non convergano mai) e anche di gradient descent quando il discriminatore è troppo bravo rispetto al generatore. Le VAE invece hanno problemi solo nelle fasi iniziali per via del regolarizzatore.

Quando dovrei usare un VAE e quando una GAN?

si usa questo schema che rappresenta i vari vantaggi e svantaggi, si sceglie in base a quello che ci interessa.



Perché è difficile ottimizzare una GAN?

Perché le GAN soffrono di stabilità del training (è facile che i paramtri oscillino e non convergano mai) e anche di gradient descent quando il discriminatore è troppo bravo rispetto al generatore. Come ogni NN inoltre ha che la funzione di errore non è convessa.

Come è fatto lo spazio latente?

Lo spazio latente è la distribuzione generata da un encoder durante il training di una VAE, perché la VAE possa funzionare esso deve essere continuo.

Si parli della funzione di loss della GAN

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

- $E_{x\sim p_{data}}[log(t_\phi(x))]$ è la probabilità che il discriminatore labelli correttamente gli input samplati dalla distribuzione che vogliamo estimare, il discriminatore vorrà massimizzare questo parametro, da notare come θ non abbia modo di influire su questo errore.
- $E_{\omega\sim p_\omega}[log(1-t_\phi(g_\theta(\omega)))]$ 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 dietro al reinforcement learning è quella di avere il nostro modello come un agente dentro un enviroment. Questo agente farà azioni dentro l'ambiente e riceverà feedback dallo stesso in base allo stato attuale, modificando a mano a mano le azioni (settando una policy interna che decide che azioni fare in che stato) che farà in futuro finché non converge nelle migliora azioni possibili.

Il feedback che riceve dall'enviroment viene definito come un reward e il modello si migliora in modo da avere il miglior reward possibile. Un esempio può essere un agente che vuole battere in livello di super mario bros e come reward function ha quanto è andato avanti nel livello.

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

Agente: è il nostro modello Ambiente: lo spazio in cui l'agente agisce Stato: lo stato dell'ambiente Azione: come l'agente interagisce con l'ambiente Reward: per ogni coppia azione, stato, l'ambiente ci restituirà un reward, l'obiettivo dell'agente è massimizare il reward Policy: il gruppo di regole che l'agente segue nelle sue interazioni con l'ambiente, per massimizare il reward si cerca una policy ideale

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

Nel SL l'input futuro non dipende da uno passato o da predizioni passate, ogni singolo step del training è supervisionato è la loss function è differenziabile rispetto ai parametri del modello.

Nel reinforcement learning invece le azioni del agente influenzano lo stato dell'enviroment, non è detto che a ogni azione corrispondi una reward, reward che poi non sono differenziabili rispetto ai parametri del modello.

E' ovvio vedere come queste differenze rendono il SL molto più facile rispetto al RL.

Si spieghi in cosa consiste il Markov Decision Process

Il Markov Decision Process è un framework usato per aiutare a fare decisioni in un ambiente stocastico, consiste nel cercare una policy che dato uno stato dell'enviroment ci da l'azione ottimale. Per risolverlo si usa la equazione di Bellman, I suoi componenti sono:

- Gli stati possibili s di cui s_0 stato iniziale fa parte.
- Le azioni possibili a
- Il modello di transizione P(s'|s,a) che ci dice come cambia lo stato data una azione, questa funziona grazie alla assunzione di Markov
- una funzione di reward r(s) che da all'agente un punteggio in base allo stato attuale
- Policy $\pi(s)$ la funzione che l'agente usa per sapere che azione fare dato uno stato s

NB: ricorda che dato uno stato preciso e una azione precisa non è detto che lo stato successivo sarà sempre lo stesso, ovvero non è detto che l'enviroment sia deterministico.

MDP in particolare è definito dalla tupla (S,A,R,P,y) dove:

- S è l'insieme di tutti gli stati possibili
- A set di tutte le possibili azioni
- R distribuzione delle reward conferite data una tupla (azione, stato)
- P probabilità di transizione, ovvero la distribuzione dello stato successivo dato una tupla (azione,stato)
- $0 < y \leq 1$ fattore di discount, questo valore indicherà quanto sono importanti le reward future rispetto a quelle immediate, più è grande più saranno importanti.

Il Loop operativo è il seguente:

1. time step t=0, l'enviroment da all'agente lo stato iniziale $s_0 \sim p(s_0)$
2. agente seleziona una azione a_t usando una policy $\pi(s)$
3. enviroment da all'agente una reward $r_t \sim R(\cdot|s_t, a_t)$
4. enviroment da all'agente il prossimo stato $s_{t+1} \sim R(\cdot|s_t, a_t)$
5. ripeti 2 to 4 finché non termina la simulazione

Il nostro obbiettivo è quindi quello di trovare $\pi*$ ottimale per massimizzare la cumulative discounted reward che si calcola facendo $\sum_{t>=0} y^t r(s_t)$, la y serve a rendere finito il risultato della CDR aiutando così a convergere durante il training.

Cos'è l'assunzione di Markov?

la probabilità di andare dallo stato s a s' dipende unicamente da s e non da stati passati.

Cos'è l'equazione di Bellman?

E' una tecnica usata nel RF per fare Q-learning, il concetto è simile alla programmazione dinamica dove si riduceil problema in sotto problemi ripetuti finché nonsi trova una soluzione a uno dei sottoproblemi che poi verrà usata per risolvere tutti i sottoproblemi fino ad arrivare a quello originale $Q^{}(s,a) = r(s) + y\sum_{s'}P(s'|s,a) max_{a'}Q^{}(s',a') = E_{s'\sim P(\cdot|s,a)}[r(s) + ymax_{a'}Q_{}(s',a')|s,a]$

Cos'è una policy?

Policy $\pi(s)$ è la mappa che l'agente usa per sapere che azione fare dato uno stato s.