Nozioni apprendimento automatico

kocierik

17 maggio 2024

Indice

1 Entropia

L'**entropia** misura il grado di impurità dell'informazione. E' massimizzata quando la X è uniforme su tutti i valori (1) e minima (0) quando è concentrata su un singolo valore.

$$H(X) = -\sum_{i=1}^{n} P(X = i) log_2 P(X = i)$$

Quando gli attributi sono continui prendiamo decisioni attraverso delle **soglie**

2 Generative techniques

Le tecniche generative sono una categoria di metodi utilizzati in diverse discipline per creare contenuti artificiali che imitano il comportamento umano o il processo di generazione di una certa classe di oggetti. Ad esempio, le tecniche generative possono essere utilizzate per **creare immagini**, testo o musica artificiali che sembrano essere stati creati da esseri umani. In campo medico, le tecniche generative possono essere utilizzate per **generare dati** di laboratorio o per **prevedere** il comportamento di una determinata malattia. Nel campo dell'intelligenza artificiale, le tecniche generative vengono spesso utilizzate per **creare modelli** di dati o per sintetizzare dati da fonti esterne.

3 Linear Classification

La classificazione lineare è un tipo di algoritmo di classificazione utilizzato in intelligenza artificiale e in altri campi per **prevedere** a quale di diverse **categorie appartiene** un oggetto o un dato, in base a una serie di caratteristiche note. In altre parole, la classificazione lineare cerca di **stabilire** una **relazione** lineare tra le **caratteristiche** di un oggetto e la sua **categoria** di appartenenza.

Per esempio, immaginiamo di avere un insieme di dati che descrivono diverse specie di fiori in base a loro petali, lunghezza dello stelo e colore. Un algoritmo di classificazione lineare potrebbe essere utilizzato per prevedere a quale specie appartiene un fiore in base a queste caratteristiche.

La classificazione lineare è un metodo **semplice** e intuitivo, ma può essere **meno efficace** di altri metodi di classificazione in presenza di **relazioni complesse** tra le caratteristiche e le categorie. Tuttavia, è ancora ampiamente utilizzato in molti campi per la sua semplicità e la sua capacità di fornire risultati interpretabili.

4 Linear regression

La regressione lineare è un modello di statistica utilizzato per stabilire una relazione tra una variabile dipendente (che si vuole prevedere) e una o più variabili indipendenti (che vengono utilizzate per fare la previsione). Si basa sull'assunzione che esista una relazione lineare tra la variabile dipendente e le variabili indipendenti.

In italiano, la regressione lineare viene spesso utilizzata per fare previsioni sulla base di dati del passato o per comprendere come le varie variabili indipendenti influiscono sulla variabile dipendente. Ad esempio, potrebbe essere utilizzata per prevedere il prezzo di una casa sulla base della sua dimensione, della sua posizione e di altri fattori, o per capire come il numero di ore di esercizio fisico settimanale influisce sulla pressione sanguigna di una persona.

5 Bias

Il bias si riferisce alla tendenza di un modello a fare **previsioni** o a prendere decisioni in modo **scorretto** o sfavorevole per alcune **categorie** di input.
Il bias può essere presente in diversi modi: ad esempio, un modello di classificazione potrebbe avere un **bias verso** una particolare **classe**, **classificando** in modo **scorretto** gli input appartenenti a quella classe. **Oppure**, un modello di previsione potrebbe avere un bias **verso** alcune variabili, ignorando altre **variabili** che potrebbero essere rilevanti per la previsione.
Il bias può avere diverse cause, ad esempio la presenza di dati di **training sbilanciati** o la scelta di un **modello non adeguato** per il problema da risolvere.
È importante individuare e correggere il bias nei modelli di intelligenza artificiale, in modo da evitare decisioni o previsioni scorrette o sfavorevoli per alcune categorie di input.

6 Epoch

Un'epoca rappresenta un **ciclo** di **addestramento** del modello sui dati di training. Durante un'epoca, il modello viene esposto a tutti i dati di training e viene **aggiornato** in base agli **errori** commessi nella previsione.
Il numero di epoche è un parametro che viene **scelto** dal **progettista** del modello e **rappresenta** il numero di volte che il modello viene **addestrato** sui **dati** di training. Maggiore è il numero di epoche, **maggiore** è il **tempo** impiegato per addestrare il modello, ma anche maggiore è la possibilità che il modello raggiunga una buona **performance**.

7 Batch

Il termine "batch" si riferisce a un **insieme** di **dati** che vengono utilizzati per **aggiornare** i **pesi** di un modello durante il processo di addestramento.

Il processo di addestramento di un modello può essere **suddiviso** in diverse **epoche**, ognuna delle quali può essere suddivisa a sua volta in più batch. Ad esempio, se il dataset di training è composto da 1000 esempi e si utilizza un batch size di 100, il processo di addestramento sarà suddiviso in 10 batch, ognuno dei quali conterrà 100 esempi.

8 Optimizer

Un ottimizzatore (o "optimizer") è un algoritmo utilizzato per **aggiornare** i **pesi** di un modello durante il processo di addestramento. L'**obiettivo** dell'ottimizzatore è quello di **minimizzare** l'**errore** del modello, ovvero la differenza tra le previsioni del modello e i valori osservati nei dati di training.
Esistono diverse strategie di ottimizzazione, ognuna delle quali si basa su un approccio specifico per aggiornare i pesi del modello. Ad **esempio**, l'ottimizzatore "gradient descent" utilizza il **gradiente** dell'errore per **aggiornare** i **pesi** del modello

9 Decision trees

Gli alberi di decisione sono una tecnica di apprendimento automatico utilizzata per fare **previsioni** o **classificare** i dati. Funzionano **creando** un **modello** basato su una serie di regole di decisione, dove ogni nodo dell'albero **rappresenta** una **decisione** da prendere e ogni **ramo** rappresenta una possibile conseguenza di quella decisione.
Per costruire un albero di decisione, si inizia dall'alto verso il basso, **selezionando** la **feature** che **massimizza** l'informazione sulla variabile da prevedere. A ogni passo, si seleziona la feature che fornisce il massimo guadagno di informazione, ovvero quella che meglio separa gli esempi positivi da quelli negativi. Si **continua** finché non si raggiunge una **soglia** di **purezza** desiderata o non ci sono più feature da considerare.
Gli alberi di decisione sono **facili** da **comprendere** e da **interpretare**, ma possono essere **sensibili** al **rumore** nei dati e tendono a **sovrastimare** le prestazioni durante il training. Tuttavia, questi **svantaggi** possono essere **mitigati** utilizzando tecniche come il **pruning** (potatura) degli alberi o l'utilizzo di un insieme di alberi di decisione come nel caso del **random forest**.

10 Random forest

Un random forest è costituito da un **insieme** di **alberi** di **decisione**, dove ogni albero viene allenato utilizzando un sottoinsieme casuale delle features (caratteristiche) presenti nel dataset di input e da un sottoinsieme casuale di esempi del dataset. Quando si deve effettuare una previsione, **ciascun albero** del foresto **fornisce** la sua **previsione** e il risultato finale viene ottenuto mediante una votazione tra le previsioni dei singoli alberi.

Il random forest è un **algoritmo** robusto e generalmente molto **preciso**, ed è particolarmente utile quando si hanno **datasets** di **grandi** dimensioni o con un elevato numero di features. Tuttavia, poiché utilizza un **gran numero** di **alberi** di decisione, può essere piuttosto **lento** durante il training e durante l'utilizzo in fase di predizione.

11 No free lunch

Il principio "No free lunch" (NFL) afferma che, in assenza di informazioni specifiche sui dati, **nessun algoritmo** di apprendimento automatico è intrinsecamente **migliore** di un **altro**. In altre parole, non esiste un algoritmo "universale" che funzioni bene in tutte le situazioni.
Questo principio si basa sul fatto che ogni **algoritmo** di apprendimento automatico ha delle **proprietà specifiche** e delle ipotesi implicite sui dati, e che queste proprietà possono renderlo adatto a funzionare bene in alcune situazioni ma meno performante in altre. Ad esempio, un algoritmo di classificazione basato sulla regressione logistica potrebbe funzionare bene su dati linearmente separabili, ma potrebbe avere prestazioni peggiori su dati non linearmente separabili.

12 Gradient ascendent

La tecnica di "gradient ascent" (o "ascesa del gradiente") è un algoritmo utilizzato in intelligenza artificiale e in particolare in apprendimento automatico per **ottimizzare** i **pesi** di un modello.
In generale, la tecnica di gradient ascent consiste nel **calcolare** il gradiente della **funzione** di **perdita** (ovvero l'errore) rispetto ai pesi del modello, e quindi nel modificare i pesi del modello in modo da aumentare il valore della funzione di perdita. Ciò viene **ripetuto** iterativamente fino a quando la funzione di perdita non **raggiunge** il massimo o un punto di **convergenza**.
La tecnica di gradient ascent viene utilizzata in particolare quando la funzione di perdita è monotona crescente rispetto ai pesi del modello. Tuttavia, può essere utilizzata anche in altri contesti, ad esempio per ottimizzare modelli di intelligenza artificiale per la generazione di testi o immagini.

13 Convolutional layer

Ogni neurone al layer $k-1$ è connesso parametricamente al **kernel** ad un sottoinsieme di neuroni al layer k. Il kernel è convoluto dal precedente. La dimensione dell'output dipende dal numero di volte che il **kernel è applicato**. L'input è **strutturato** e riflette sull'output.

14 Kullback-Leibler divergence

La divergenza di Kullback-Leibler (KL divergence) è una misura di **disuguaglianza** tra due distribuzioni di **probabilità**. Si basa sulla **differenza** tra l'entropia di una distribuzione di probabilità e l'**entropia** dell'altra distribuzione, tenendo conto della prima distribuzione.
Può essere **utilizzata** per valutare quanto un modello di apprendimento automatico si **avvicini** a una distribuzione di probabilità **ideale**.
La KL divergence è spesso utilizzata nell'ottimizzazione dei modelli di apprendimento automatico, dove viene **utilizzata** come termine di perdita per **minimizzare** la differenza tra la distribuzione del modello e la distribuzione ideale. Inoltre, viene spesso utilizzata nell'inferenza bayesiana per stimare la verosimiglianza di un modello rispetto a un altro.

15 Vanish gradient problem

Nel Vanishing gradient, il gradiente assume un valore **nullo** o asintotico allo zero **impedendo** cosi l'**aggiornamento** dei **pesi**. In una rete convoluzionale gli strati in prossimità dell'input **estraggono** dall'immagine **features** spaziali. Le reti **profonde** sono però **difficili** da **addestrare**. Uno dei maggiori problemi è la scomparsa del gradiente. Durante la fase di **backpropagation** i pesi degli strati in prossimità dell'**input** restano **costanti** o si **aggiornano** molto **lentamente** al contrario di quanto **accade** per gli **strati vicini** all'output.

16 Logistic regressor

La regressione logistica **stima** la **probabilità** del verificarsi di un evento, come ad esempio voto espresso o non espresso, **sulla base** di uno specifico **dataset** di **variabili** indipendenti. Poiché il **risultato** è una **probabilità**, la variabile dipendente è vincolata tra 0 e 1. Nella regressione logistica, viene applicata una trasformazione logit sulle probabilità - ossia la **probabilità** di **successo divisa** per la probabilità di **fallimento**. Ciò è anche comunemente noto come **probabilità logaritmica**, o logaritmo naturale delle probabilità.

17 Activation function

Quindi fondamentalmente una funzione di attivazione viene utilizzata per **mappare** l'**ingresso** all'**uscita**. Questa funzione di attivazione aiuta una **rete neurale** ad **apprendere** relazioni e schemi complessi nei dati.
Un uso importante della funzione di attivazione è di **mantenere** l'uscita limitata a un **intervallo** particolare. Un altro uso della funzione di attivazione è l'aggiunta di non linearità nei dati. **Scegliamo** sempre **funzioni non lineari** come funzioni di attivazione.

Se usiamo le funzioni di **attivazione lineare** in una **rete** neurale **profonda**, indipendentemente dalla profondità della nostra rete, sarà **equivalente** ad una semplice **rete** neurale lineare **senza strati** nascosti perché quelle funzioni di attivazione lineare possono essere **combinate** per formare un'altra **singola funzione lineare**.

Quindi fondamentalmente tutta la nostra rete sarà **ridotta** ad un **singolo neurone** con quella funzione lineare combinata come sua funzione di attivazione e quel singolo neurone

18 Feed-forward network

Una rete neurale feed-forward è una rete neurale artificiale dove le connessioni tra i nodi **non formano cicli**, differenziandosi dalle reti neurali ricorrenti. Questo tipo di rete neurale fu la prima e più semplice tra quelle messe a punto. In questa rete neurale le **informazioni** si muovono solo in **una direzione**, avanti, rispetto a nodi d'ingresso, attraverso nodi nascosti (se esistenti) fino ai nodi d'uscita. Nella rete non ci sono cicli. Le reti feed-forward **non hanno memoria** degli input avvenuti a tempi precedenti, per cui l'**output** è **determinato** solamente dall'attuale **input**.

19 Dense layer

In ogni rete neurale un layer denso è connesso al precedente neurone. Questo layer è il più comunemente usanto nelle reti neurali. Ogni **neurone** al layer $k-1$ è **connesso** agli **altri neuroni** al layer k.
Esso è uno strato di una rete neurale artificiale che si occupa di effettuare una **trasformazione lineare** dei dati di input. Ciò significa che ogni **neurone** dello strato denso **riceve** input da tutti i neuroni dello strato **precedente** e produce un output calcolando una combinazione lineare di questi input.
Gli strati densi sono spesso utilizzati come "building block" di reti neurali artificiali perché possono essere **facilmente combinati** tra loro per creare modelli di apprendimento automatico più complessi. Ad esempio, è possibile creare una rete neurale con più strati densi, dove ciascuno strato riceve come input i dati prodotti dallo strato precedente e li trasforma in modo lineare. Oppure, è possibile combinare strati densi con altri tipi di strati, come gli strati di pooling o di dropout, per creare modelli di apprendimento automatico ancora più sofisticati.

20 Convolution layer

Ogni neurone al layer $k-1$ è **connesso parametricamente** al **kernel** in un numero **fisso** di neuroni al layer k. Il kernel è **convoluto** al layer precedente. La dimensione dell'output dipende soolo dal numero di volte che il kernel è applicato. Questo layer consente di estrarre caratteristiche **specifiche**. Ad

esempio, un layer convoluto potrebbe utilizzare un **filtro** per rilevare **bordi** o linee nell'immagine di ingresso.
A differenza di un layer denso, che connette ogni unità a tutte le unità del layer precedente, un layer convoluto utilizza solo un **sottoinsieme** di **connessioni**, che permette di conservare la struttura spaziale dei dati.

21 Current loss

In campo dell'apprendimento automatico, il "loss" è una **misura** dell'**errore** commesso da un modello durante la sua fase di training. Questo valore viene calcolato utilizzando una **funzione** di **perdita**, che confronta le previsioni del modello con i valori desiderati (noti anche come etichette). La funzione di perdita restituisce un valore che indica quanto l'**errore** sia grande, e questo valore viene utilizzato dall'algoritmo di ottimizzazione per **aggiustare** i **pesi** del modello in modo da ridurre l'errore nelle previsioni.

22 Backpropagation

Un metodo semplice per risolvere il problema delle reti neurali è quello di muoversi a piccoli passi sulla superficie della funzione (Discesa del gradiente) e via via **ricalcolare** i pesi della rete neurale **partendo** da quelli già **noti** e dall'ultimo errore calcolato per ridurre lo stesso. (Back Propagation).
Il meccanismo di addestramento, inizia con un'operazione di **inizializzazione casuale** dei **pesi**, ed **itera** passando le coppie di input-output appartenenti ad un prefissato insieme di dati al modello neurale, fin quando, grazie ad operazioni di aggiornamento dei pesi, non si **raggiunge** il **minimo** (assoluto) della funzione costo, che **esprime** la **misura** della **distanza** che intercorre tra gli output desiderati e i corrispondenti output che il modello neurale calcola, ovvero l'errore. La backpropagation viene fatta dopo la **feed-forward** dei dati per aggiustare l'errore.
La backpropagation ci dice solo in quale **direzione** il **gradiente** dovrebbe essere **aggiornato**.

23 Learning rate

L'algoritmo di backpropagation ci dà solo la **direzione** del gradiente. Il numero di aggiornamenti è dato dalla moltiplicazione per uno scalare fisso **learning rate**. Un valore del learning rate troppo **piccolo** può **rallentare** il processo di training, poiché l'algoritmo di ottimizzazione compie passi molto piccoli verso la soluzione ottimale. D'altro canto, un valore troppo **alto** può causare **instabilità** o divergenza del modello

24 Overfitting

Se in fase di training un modello di machine learning predice al 98% il contenuto di un'immagine, e in testing lo stesso modello ha un'accuratezza del 60%, o comunque di molto inferiore, possiamo parlare di overfitting. In Machine Learning l'overfitting è causato da una serie di fattori:

- il modello che si desidera creare contiene **troppi parametri** da individuare, ergo è troppo complesso
- i **dati** di training sono **pochi**, o di **cattiva qualità**.

Per ovviare a questo problema:

- **semplificazione** del **modello**: optare per uno con meno parametri, ridurre il numero di attributi del dataset, o limitare l'apprendimento del modello (regularization)
- **nuova raccolta dati**: aumentare la dimensione del dataset di training
- riduzione rumore: correggere errori di struttura e gestire gli outliers

25 Chain rule

La regola della catena è uno strumento molto utile nell'analisi matematica e nell'apprendimento automatico, poiché permette di **calcolare** facilmente il **derivato** di **funzioni complesse**.
Ad esempio, date due funzione derivabili f e g con derivate $f^{'}$ e $g^{'}$, la derivata della funzione composta $h(x) = f(g(x))$ è:

$$h^{'}(x) = f^{'}(g(x)) * g^{'}((x)) = f^{'}(y) * g^{'}(x)$$

La derivata della **composizione** di una serie di **funzione** è il **prodotto** delle derivate delle **singole** funzioni.

26 Transfer learning

Il transfer learning è una tecnica utilizzata nell'apprendimento automatico per **trasferire** le **conoscenze** acquisite da un **modello** pre-addestrato su un problema a un nuovo problema. In altre parole, il transfer learning permette di riutilizzare un modello pre-addestrato su un problema simile, **anziché** addestrarne uno **da zero**, al fine di ridurre i tempi di addestramento e migliorare le prestazioni del modello.

27 Receptive field

Il campo di ricezione (receptive field) di un neurone è la porzione dello spazio di input che il **neurone** può "vedere". In altre parole, è la regione dello **spazio** di **input** che ha un **effetto** sull'**output** del neurone.

28 Cos'è un tensore:

Un tensore è un **array** multidimensionale. Un tensore tipico per immagini 2D ha 4 **dimensioni**:

$$batchsize \cdot width \cdot height \cdot channels$$

29 Feature map

Una feature map è una **rappresentazione** di un **input**, come ad esempio un'immagine, in modo da enfatizzare determinate caratteristiche dell'input che sono rilevanti per una determinata attività. Ad esempio, nel caso di una rete neurale per l'elaborazione delle immagini, la **feature map** potrebbe **evidenziare** i **bordi** e le forme degli oggetti nell'immagine, rendendo più facile per la rete identificare questi oggetti. Per una immagine a colori inizialmente abbiamo r, g, b.

30 Pooling layer

Il pooling layer è un tipo di strato presente in una rete neurale. Serve a **ridurre** le **dimensioni** delle **immagini** o dei **dati** in ingresso, rendendo la **rete** più **efficiente** e veloce nell'elaborazione delle informazioni. In pratica, il pooling layer opera una sorta di "sottomissione" dei dati, **scegliendo** i **valori** più **importanti** e ignorando quelli meno significativi. Ciò aiuta a **eliminare** il **rumore** nei dati e a rendere la rete più resistente agli errori.

31 Residual learning

Il residual learning è una tecnica utilizzata nell'addestramento delle reti neurali. Consiste nell'**aggiungere** un ulteriore **strato** di connessioni alla rete, chiamato "**strato residuo**", che permette alla rete di **imparare** le **informazioni mancanti** nei dati in ingresso. In questo modo, la rete è in grado di "**raddrizzare**" i **dati** non lineari e di migliorare le prestazioni di apprendimento. Il residual learning è particolarmente utile per le **reti profonde**, poiché permette di evitare il problema del "**vanishing gradient**", che impedisce alla rete di imparare in maniera efficiente.

32 Loss function

In generale, una loss function misura il grado di **accuratezza** con cui un certo **modello** statistico descrive un set di dati empirici di un certo fenomeno. Per valutare la sua efficacia e performance, usiamo una loss function.

33 Why classification techniques are vulnerable

Le tecniche di classificazione sono soggette a **vulnerabilità** per diversi motivi. Una delle principali ragioni è che spesso si **basano** su **modelli** che sono stati addestrati su **dati limitati** o non rappresentativi dell'intero spettro di possibili casi. Ciò può portare a risultati **imprecisi** o addirittura sbagliati quando si cerca di utilizzare il modello per classificare nuovi dati.
Inoltre, le tecniche di classificazione possono essere **influenzate** dall'**ordine** in cui vengono presentati i dati durante il processo di addestramento. Se i dati vengono presentati in modo non casuale, il modello potrebbe sviluppare una certa **dipendenza** dall'**ordine** e non essere in grado di generalizzare correttamente a nuovi dati.

34 Difference between generative and discriminative approaches in machine learning

In ambito di apprendimento automatico, i modelli generativi e discriminativi sono due approcci diversi per l'**addestramento** di **modelli** di classificazione.
Un modello **generativo** cerca di apprendere una distribuzione probabilistica sui **dati** di **input**, ovvero cerca di capire come sono distribuiti i dati all'interno dello spazio di input. Una volta che il modello ha appreso questa distribuzione, può essere utilizzato per **generare nuove istanze** di dati simili a quelle presenti nel training set. Ad esempio, un modello generativo potrebbe essere utilizzato per generare immagini di animali mai visti prima, basandosi su un training set di immagini di animali reali.
Un modello **discriminativo**, al **contrario**, cerca di imparare a distinguere tra **diverse classi** di dati di **input**. Una volta addestrato, il modello può essere utilizzato per **prevedere** a quale **classe** appartiene un nuovo esempio di dati in base alle sue caratteristiche. Ad esempio, un modello discriminativo potrebbe essere utilizzato per prevedere se una persona ha una malattia o no in base ai suoi sintomi.
In sintesi, i modelli **generativi** cercano di capire come sono **distribuiti** i **dati** all'interno dello spazio di input e possono essere utilizzati per **generare** nuove istanze di **dati**, mentre i modelli **discriminativi** cercano di **distinguere** tra **diverse classi** di dati e possono essere utilizzati per prevedere a quale classe appartiene un nuovo esempio di dati.

35 Example Neural Network for image processing

Una rete neurale per il processing delle immagini ha una struttura del tipo:

- Una lunga sequenza di **convolution layer**, organizzata in **moduli**
- una sequenza di **dense layer** (2-3) alla **fine** del training

Con i **convolutional layers estraiamo** le **feature** interessanti dall'immagine di input, generando diverse rappresentazioni. Con i **dense layer** andiamo ad **analizzare** queste **feature** mirando al problema da risolvere (e.g. classification)

36 Manifold of data

Manifold si riferisce alla struttura di base di un insieme di dati. In altre parole, un manifold è la **rappresentazione** di come i **dati** sono **organizzati** nello spazio di input.
Ad esempio, immaginiamo di avere un insieme di dati costituito da punti in uno spazio bidimensionale. Se questi **punti formano** una **curva**, possiamo dire che la curva è il **manifold** dei dati. Se invece i punti formano una rete di punti distribuiti in modo **casuale**, possiamo dire che il **manifold** dei dati è più **complesso** e difficile da rappresentare.
Il concetto di manifold è importante in apprendimento automatico perché può influire sulla capacità di un modello di **generalizzare** a nuovi **dati**. Ad esempio, se il manifold dei dati è lineare, un modello lineare potrebbe essere sufficiente per rappresentare i dati e ottenere buone prestazioni. Se invece il manifold è più **complesso**, potrebbe essere necessario utilizzare **modelli** più **sofisticati**, come reti neurali artificiali, per rappresentare i dati in modo accurato.

37 Fooling networks

Il "fooling" di una rete neurale consiste nel **fornire** alla rete un'**immagine** o un input che viene **interpretato** in modo **errato** dalla rete stessa, nonostante sia stato progettato per essere riconosciuto correttamente da un essere umano. Ciò può accadere perché la **rete** neurale ha imparato a **riconoscere pattern specifici** in modo troppo stretto, il che significa che può essere **facilmente ingannata** da immagini o input che presentano piccole variazioni rispetto a quelli che ha visto durante il suo processo di apprendimento.
Il fooling di una rete neurale può essere **utilizzato** per testare la **robustezza** della **rete** e per identificare eventuali punti deboli nella sua capacità di riconoscimento.

38 Latent space

Il latent space viene spesso utilizzato in riferimento a uno spazio di rappresentazione nascosto o "latente" all'interno di un modello. Ad esempio, in un modello di autoencoder o di VAE (variational autoencoder), il "latent space" è uno spazio di **rappresentazione** nascosto che viene utilizzato per **codificare** l'**input** in una rappresentazione **compatta**. Lo scopo del "latent space" è di catturare le caratteristiche principali dei dati in modo da poterli utilizzare per la generazione di nuove campionature o per la classificazione.

39 Autoenconders

L'architettura di un Autoencoder prevede due componenti: **encoder** e **decoder**. Il **primo** riceve i dati in **input** e ne fornisce una rappresentazione **compressa** o latent space representation. Il **decoder** riceve in input la versione **compressa** del dato e lo **riproduce** con il minor errore possibile.
Esso è una rete allenata a **ricostruire dati** in **input** imparando dalla rappresentazione interna. Solitamente essa ha una **dimensione** in input **più piccola**.

40 What Autoencoders can do

Non molto utili per comprimere dati, per la perdita naturale. ma per:

- data denoising
- anomaly detection
- feature extraction (generalization of PCA)
- generative models (VAE)

Especially, an amusing and simple to understand topic.

41 Compression

La "compressione" in machine learning si riferisce all'utilizzo di **tecniche** per **ridurre** la quantità di **dati** che un modello di apprendimento automatico deve elaborare. Ciò può essere utile per **migliorare** le **prestazioni** del modello, ad esempio aumentando la **velocità** di **addestramento** o rendendo il modello più facile da utilizzare su dispositivi con risorse limitate. Se l'input ha una **struttura** molto **casuale** (entropia alta) allora la **compressione non** è **possibile**.
Esistono diverse tecniche di compressione che possono essere utilizzate in machine learning, ad esempio:

- **Compressione dei dati**: si tratta di ridurre la quantità di dati di input utilizzati dal modello, ad esempio eliminando duplicati o dati non rilevanti.
- **Compressione del modello**: consiste nel ridurre il numero di parametri o la complessità del modello stesso, ad esempio utilizzando modelli più piccoli o semplificando la struttura del modello.
- **Compressione delle feature**: si tratta di ridurre il numero di variabili di input utilizzate dal modello, ad esempio eliminando variabili non rilevanti o utilizzando una rappresentazione più compatta delle feature.

42 Anomaly detection

L'anomaly detection, o rilevamento di anomalie, è una tecnica utilizzata per **individuare elementi** che si **discostano** in modo significativo da un **modello** o da una tendenza stabilita. In altre parole, si tratta di un processo che cerca di **identificare** eventi o **dati atipici** all'interno di un insieme di dati.

Esistono diversi metodi per effettuare il rilevamento di anomalie, tra cui:

- **Modelli di distribuzione**: si basano sulla definizione di una distribuzione **probabilistica** dei dati e sulla individuazione di punti che si discostano significativamente da essa.
- **Algoritmi di apprendimento supervisionato**: si basano sull'utilizzo di un set di dati di allenamento per costruire un modello che può essere utilizzato per **classificare** i **dati** in categorie predefinite (ad esempio, anomali o normali).
- **Algoritmi di clustering**: si basano sulla suddivisione dei dati in **gruppi omogenei** (cluster) in base a determinate caratteristiche, individuando poi gli elementi che non appartengono a nessun cluster o che si discostano significativamente dai cluster vicini.

43 Latent encoding

L'encoding latente consiste nel **mappare** un **dato** o un'informazione in un insieme di **caratteristiche nascoste** che possono essere utilizzate per rappresentare il dato in modo più **efficiente** o per eseguire operazioni di machine learning.
L'encoding latente viene spesso utilizzato in tecniche di machine learning come le reti neurali ricorrenti (RNN) o le reti generative profonde (GAN). In questi casi, le **caratteristiche** latenti vengono **apprese** dai dati **durante** il processo di **addestramento** del modello e possono essere **utilizzate** per eseguire diverse operazioni, ad esempio la **generazione** di nuovi **dati** o la classificazione di dati di test. I valori latenti sono i valori significativi delle feature, essi rendono più facile il rilevamento di errori osservando dei valori latenti anomali.

44 Convolutional Neural Network (CNN)

Essa infatti è costituita da un blocco di **input**, uno o più blocchi nascosti (**hidden layer**), che effettuano calcoli tramite **funzioni** di **attivazione** (ad esempio **RELU**) e un blocco di output che effettua la classificazione vera e propria. La **differenza**, infatti, rispetto alle classiche reti feed forward è rappresentata dalla **presenza** dei **livelli** di **convoluzione**.
A differenza pertanto di una feed forward tradizionale che lavora "sull'informazione generale dell'immagine" una **CNN** lavora e classifica l'immagine **basandosi** su **particolari caratteristiche** della stessa.

45 U-Net (FCN Fully Convolutional Network)

In sintesi, l'architettura di una U-Net è costiuita da:

- Un **encoder** che **riduce** (down-sample) l'**immagine** in ingresso in una feature map, attraverso pooling layers, estraendone gli elementi chiave
- Un **decoder** che **amplifica** (up sample) la **feature map** in una immagine, usando i livelli di deconvoluzione, impiegando cioè i pooling layers appresi per permettere la localizzazione degli elementi.

46 Deep Object Detection approce

Esistono diverse architetture di Convolution Network Neural (CNN) che possono essere utilizzate per il **rilevamento** degli oggetti profondi, ad esempio:

- Reti di rilevamento degli oggetti basate su **regioni di interesse** (R-CNN): in questo caso, la CNN viene utilizzata per individuare regioni di interesse nell'immagine, che vengono poi analizzate in modo più dettagliato per identificare gli oggetti.
- Reti di rilevamento degli oggetti basate sulla **condivisione delle feature** (Fast R-CNN): in questo caso, la CNN viene utilizzata per estrarre le feature da diverse regioni dell'immagine, che vengono poi utilizzate per individuare gli oggetti.
- Reti di rilevamento degli oggetti unificate (**YOLO**): in questo caso, la CNN viene utilizzata per individuare gli oggetti direttamente nell'immagine, senza la necessità di analizzare le regioni di interesse in modo separato.

47 YOLO

Il concetto è di **ridimensionare** l'**immagine** in modo da ricavarne una griglia di quadrati. YOLO fa **predizioni** su 3 **diverse scale**, riducendo l'immagine

rispettivamente di 32, 16 e 8.
Queste griglie, a loro volta, prevedono le **coordinate** del riquadro di delimitazione B relative alle coordinate della cella, nonché il nome dell'elemento e la **probabilità** che l'oggetto sia presente nella cella. A causa del fatto che molte celle predicono lo stesso elemento con varie previsioni del riquadro di delimitazione, questa tecnica **riduce** notevolmente il **calcolo** perché sia il rilevamento che il **riconoscimento** sono **gestiti** dalle **celle** dell'immagine. Tuttavia, produce molte previsioni **duplicate**. YOLO lo risolve esaminando i punteggi di **probabilità** legati a ciascuna **opzione**.

48 Anchor boxes

Le anchor boxes sono utilizzate in alcuni algoritmi di **rilevamento** degli oggetti per indicare le regioni di un'immagine in cui potrebbe essere presente un **oggetto**.
Le anchor boxes vengono utilizzate in algoritmi di rilevamento degli oggetti basati sulla **condivisione** delle **feature**, come le reti di rilevamento degli oggetti basate sulla condivisione delle feature (Fast R-CNN) o le reti di rilevamento degli oggetti unificate (YOLO). In questi casi, le anchor boxes vengono utilizzate per **selezionare** le **regioni** dell'immagine che vengono **analizzate** in modo più **dettagliato** dal modello di rilevamento degli oggetti.
Le anchor boxes possono essere definite in modo statico o dinamico. Nel primo caso, le anchor boxes vengono definite **prima** dell'**addestramento** del modello e rimangono **fisse** durante l'intero processo di addestramento. Nel secondo caso, le anchor boxes vengono definite dinamicamente **durante** l'**addestramento** del modello in base alle **caratteristiche** dei dati.

49 Objectness score

L'objectness score è una misura della **probabilità** che una determinata regione di un'immagine **contenga** un **oggetto**.
Lo score di oggettività viene utilizzato in alcuni algoritmi di rilevamento degli oggetti per determinare quali regioni dell'immagine meritano di essere analizzate in modo più dettagliato.
Lo score viene inolte **passato** ad una funzione **sigmoide** per interpretare la **probabilità**.

50 Class confidence

La class confidences si riferisce alle **probabilità** assegnate dal **modello** a ciascuna **classe** per un dato input. Ad esempio, se il modello è addestrato per riconoscere diverse categorie di animali (ad esempio gatti, cani e uccelli), per un'immagine di un gatto il modello potrebbe assegnare una probabilità del $0,7$

per la classe "gatto", una probabilità del $0,2$ per la classe "cane" e una probabilità del $0,1$ per la classe "uccello". In questo caso, il modello ha una "alta confidenza" nella classe "gatto" per questo input.

51 YOLO's loss function

Nel caso di YOLO, la funzione di perdita viene calcolata utilizzando una **combinazione** di due diverse **funzioni** di **perdita**: la perdita di localizzazione (**localization loss**) e la perdita di classificazione (**classification loss**). La perdita di **localizzazione** viene utilizzata per **misurare** quanto l'**output** della rete neurale **differisce** dalla posizione dell'**oggetto** nell'immagine, mentre la perdita di **classificazione** viene utilizzata per misurare quanto l'**output** della rete neurale **differisce** dall'**etichetta** della classe dell'oggetto. La funzione di perdita complessiva viene quindi calcolata come una **combinazione ponderata** di queste due **perdite**.

52 Non Maximum Suppression

La Non Maximum Suppression (**NMS**) è un algoritmo utilizzato per **rimuovere** le **sovrapposizioni** tra le regioni di interesse (RoI) all'interno di un'**immagine**. Viene comunemente utilizzato in combinazione con i sistemi di rilevamento degli oggetti basati sull'intelligenza artificiale, come ad esempio YOLO (You Only Look Once), per individuare gli oggetti all'interno di un'immagine e tracciare i loro contorni.
In pratica, la NMS viene utilizzata per **rimuovere** le **regioni** di interesse che sono troppo **simili** tra loro o che si **sovrappongono** in modo significativo. Ciò è importante perché può evitare la doppia conta degli oggetti e garantire una maggiore accuratezza nel rilevamento degli oggetti.
Per utilizzare la NMS, il sistema di rilevamento degli oggetti prima **individua** tutte le **regioni** di interesse all'interno dell'immagine e quindi utilizza l'algoritmo di NMS per selezionare solo le regioni di interesse più significative, **eliminando** quelle che sono troppo **simili** o che si sovrappongono in modo significativo.

53 Generative model

Un modello generativo è un **modello** che è in grado di **generare** nuove **output**, come ad esempio parole, frasi, immagini o suoni, a partire da un input. I modelli generativi sono spesso utilizzati in campi come il machine learning e l'intelligenza artificiale, dove possono essere utilizzati per **prevedere** il **comportamento** di un sistema o per creare contenuti sintetici.
Essi provano a imparare la distribuzione dei dati imparando dagli esempi disponibili.

54 Multi-modal output

Output multi-modale si riferisce a un **output** che viene **generato** o prodotto attraverso più modalità o **canali**. Ad **esempio**, un sistema di intelligenza artificiale potrebbe generare un output multi-modale che **include** sia un'**immagine** che un **testo**, oppure una combinazione di parole, suoni e movimenti. Un output multi-modale può essere utilizzato per fornire informazioni in modo più completo e coinvolgente, rendendo più facile per gli utenti comprendere e interpretare l'output del sistema.

55 Latent variable models

I modelli a variabili latenti sono modelli che tentano di **spiegare** la **relazione** tra una o più **variabili** osservate e una o più variabili nascoste o latenti. Le variabili latenti sono considerate "nascoste" perché **non** sono **direttamente osservabili**, ma possono essere inferite a partire dalle osservazioni. Ad esempio, in un modello a variabili latenti potrebbe essere osservata solo la quantità di tempo che una persona trascorre sui social media, mentre la variabile latente potrebbe essere il livello di dipendenza dai social media di quella persona.

56 Generative models

I modelli generativi sono modelli che sono in grado di generare nuove output, come ad esempio parole, frasi, immagini o suoni, a partire da un input.
alcune delle principali **classi** di **modelli** generativi sono:

- **compressive models**
  - Variational Autoencoders (VAEs)
  - Reti neurali generative (GAN): utilizzate per generare immagini, suoni e altre forme di dati utilizzando una rete neuronale.
- **dimension preserving models**
  - Normalizing Flows
  - Denoising Diffusion Models

57 VAE (Variation AutoEnconder)

Il Variational Autoencoder (VAE) è un tipo di **modello** di rete neuronale generativa utilizzato per **generare** nuove **output**, come ad esempio immagini, a partire da un input. Il VAE funziona utilizzando una tecnica chiamata "**codifica** e **decodifica**" per **apprendere** la **distribuzione** di probabilità dei dati di input e quindi generare nuove osservazioni che seguono la stessa distribuzione.

Caratteristica fondamentale dei VAE 'e l'obiettivo di **creare** una **rappresentazione latente** del **dataset** utilizzato per la calibrazione (o training) della rete, tale per cui per ogni punto del dataset X esista una configurazione delle variabili latenti che porta il modello a generare qualcosa di simile ad X.

58 GAN (Generative Adversarial Networks)

La GAN è composta da **due reti neurali** artificiali: il network **generatore** e quello **discriminatore**. Entrambe sono **reti** neurali **convoluzionali**. Come in tutte le reti neurali artificiali, anche in ciascuno dei due network che compongono la GAN l'elaborazione procede a **livelli** di **astrazione** progressiva, dai più semplici ai più complessi: l'**output** del livello **precedente** diventa l'**input** di quello **successivo**.
I due network vengono messi in **competizione** tra **loro**, in un gioco a somma zero: l'obiettivo del **generatore** è **ingannare** il **discriminatore**, che a propria volta è programmato per controllare e scoprire le immagini false. Le loro funzioni-obiettivo sono opposte: una vince, l'altra perde. Ma i **feedback** sono ovviamente **collegati** tra loro: il **generatore migliora** la produzione di immagini sulla base dei **riscontri** del **discriminatore**, che a propria volta raffinerà i propri mezzi di detection.

59 Normalizing Flows

I Normalizing Flows sono una **classe** di **modelli** di machine learning che sono stati progettati per rappresentare **distribuzioni** di **probabilità complesse** in modo efficiente. In particolare, i Normalizing Flows utilizzano una serie di trasformazioni deterministiche per **mappare** una distribuzione di **probabilità semplice**, nota come distribuzione di base, su una distribuzione di probabilità **complessa**.

- **pro** Consente di ottenere una computazione con probabilità logaritmica
- **Contro**: Limita l'espressività del modello

60 Diffusion model

In sintesi, i modelli di diffusione possono essere utilizzati in machine learning per **prevedere** il comportamento di una **caratteristica** o di un fenomeno in un sistema dinamico, per identificare i fattori che influiscono sulla diffusione di una caratteristica o di un fenomeno e per **costruire modelli** di simulazione di sistemi **dinamici**.

61 The problem with the deterministic autoencoder

Un **problema** comune con gli autoencoder deterministici è che possono avere **difficoltà** a **rappresentare distribuzioni** di probabilità complesse in modo efficiente. Ciò significa che potrebbero essere meno accurati nella rappresentazione di dati rispetto ai modelli che utilizzano distribuzioni di probabilità per rappresentare i dati, come i modelli di generative adversarial networks (GAN) o i Normalizing Flows.
Inoltre, gli **autoencoder** deterministici tendono a essere **meno robusti** alle **perturbazioni** nei dati di input rispetto ai modelli basati su distribuzioni di probabilità. Ciò significa che potrebbero essere **meno affidabili** nel caso in cui ci siano **rumori** o variazioni nei dati di input.
Infine, gli autoencoder deterministici possono avere difficoltà a generalizzare a dati nuovi o sconosciuti, poiché non hanno la capacità di rappresentare incertezza o variabilità nei dati. Ciò può limitare la loro capacità di fare previsioni accurate su dati che non sono stati visti durante il processo di addestramento.

62 Problems with VAE

- bilanciamente della probabilità logaritmica e **regolarizzazione** KL nella loss function
- collasso delle variabili
- marginal inference vs prior mismatch
- blurriness (aka variance loss)

63 Problems with Gans

Ci sono alcuni problemi potenziali che possono insorgere quando si utilizzano i GAN:

- **Instabilità dell'addestramento**: i GAN possono essere difficili da addestrare perché l'obiettivo di ottimizzazione coinvolge un gioco minimax a due giocatori. Ciò può portare a un addestramento instabile, in cui il generatore e il discriminatore possono rimanere bloccati in un equilibrio subottimale o oscillare tra diverse soluzioni.
- **mode collapse**: i GAN possono soffrire di collasso dei modi, in cui il generatore produce solo un numero limitato di modi o variazioni dei dati, anziché catturare la piena diversità dei dati di formazione.
- **Spazi ad alta dimensione**: i GAN possono essere difficili da utilizzare in spazi ad alta dimensione, come le immagini, perché il generatore e il

discriminatore possono avere difficoltà a imparare la struttura complessa dei dati.

64 RRN (Recurrent Neural Networks)

Le reti neurali ricorrenti (RNN) sono un tipo di rete neurale artificiale che è particolarmente adatta per l'**elaborazione** di **sequenze** di **dati**. Sono chiamate "ricorrenti" perché utilizzano feedback interni che **permettono** loro di "**ricordare**" le **informazioni** per un certo periodo di tempo.
Sono in grado di prendere in input una sequenza di dati, come una serie di parole in una frase o una sequenza di frame di un video, e di utilizzare le informazioni presenti in questi dati per fare previsioni o classificare l'input.
Una delle caratteristiche distintive delle RNN è la loro capacità di "memorizzare" le informazioni per un certo periodo di tempo, utilizzando uno stato nascosto che viene aggiornato ad ogni passo della sequenza. Ciò le rende ideali per l'**elaborazione** di **sequenze** di **dati** in cui l'ordine è importante, come nel **linguaggio naturale** o nella generazione di musica.

65 LSTM (Long Short Term Memory networks)

Uno dei vantaggi principali delle LSTM è la capacità di apprendere da **lunghe sequenze** temporali e **conservarne la memoria**.
L'idea alla base dell'architettura LSTM è quella che ad ogni time step alcuni gate sono utilizzati per controllare il passaggio di informazioni lungo le sequenze che sono in grado di catturare **dipendenze** sul **lungo termine** più accuratamente.
In una LSTM, ad ogni step, **l'hidden state** è **aggiornato** dallo stesso dato al time step, dall'hidden state al **precedente** time step, dall'input gate, dal forget gate, dall'output gate e da una cella di memoria.

66 Attention

L'attenzione in machine learning è un concetto che si riferisce alla **capacità** di un modello di **prestare attenzione** a specifici elementi di un'**input** durante il processo di apprendimento o di elaborazione delle informazioni.

Un modello che utilizza l'attenzione può **concentrarsi** su **determinati elementi** dell'input in modo più accurato e dettagliato, rispetto a un modello che tratta tutti gli elementi dell'input allo stesso modo. Ciò può aiutare il modello a fare **previsioni** o **classificazioni** più **accurate**, soprattutto in caso di input complessi o astratti.

L'attenzione viene spesso utilizzata in **combinazione** con le reti neurali ricorrenti (**RNN**) per consentire al modello di prestare attenzione a specifici elementi di una sequenza di dati nel tempo. Ad esempio, in un modello di

traduzione automatica, l'attenzione può essere utilizzata per fare in modo che il modello presti maggiore attenzione alle parole chiave della frase di origine durante la traduzione.

67 Attention: the key-value approach

Il concetto di "Attention: the key-value approach" può essere tradotto come "Attenzione: l'approccio **chiave-valore**". Si tratta di una tecnica utilizzata in alcuni modelli di intelligenza artificiale, in particolare nei modelli di linguaggio, per implementare l'attenzione a sé stessi (self-attention).
L'approccio chiave-valore consiste nell'utilizzo di due **vettori**, chiamati "chiave di attenzione" (**attention key**) e "valore di attenzione" (**attention value**), per calcolare l'importanza di ogni parola o sequenza di parole nell'input e trasformare le informazioni contenute nell'input in un formato che può essere utilizzato dal modello.
La chiave di attenzione è un **vettore** di **pesi** che viene utilizzato per **calcolare** la "punteggiatura" di attenzione (**attention score**) per ogni parola o sequenza di parole. Il valore di attenzione è un vettore di trasformazione che viene utilizzato per trasformare le informazioni contenute nell'input in un formato utilizzabile dal modello.
La punteggiatura di attenzione viene calcolata utilizzando la chiave di attenzione e il valore di attenzione insieme, e viene utilizzata per **determinare** l'**importanza** di ogni parola o sequenza di parole per il modello. In questo modo, il modello può prestare maggiore attenzione alle parole o alle frasi più rilevanti per il significato del testo e fare previsioni più accurate.

68 Multi head attention

Il concetto di base dell'attenzione multipla è quello di utilizzare più "teste di attenzione" per fare **riferimento** a diverse parti del vettore di **input** in modo **indipendente**. Ognuna di queste teste di attenzione può essere vista come un modello di attenzione separato che cerca di **catturare relazioni diverse** tra gli elementi del vettore. Il risultato finale dell'attenzione multipla è quindi ottenuto dall'**aggregazione** dei **risultati** delle diverse teste di attenzione.
L'utilizzo dell'attenzione multipla può essere utile in molti casi, ad esempio per **catturare relazioni** tra parole in un testo che potrebbero essere trascurate da un **modello** di attenzione **tradizionale**, che considera solo una parte del testo alla volta. Inoltre, l'utilizzo di diverse teste di attenzione permette di catturare relazioni diverse tra gli elementi del vettore, il che può essere particolarmente **utile** in casi in cui il **vettore** di input è **complesso** o contiene informazioni di diverse nature.

69 Gating function

Le funzioni di ingate (o "gating functions") sono una tecnica utilizzata in machine learning per **gestire** l'**informazione** all'interno di un **modello**. Sono spesso utilizzate nelle reti neurali ricorrenti (RNN) per **controllare** quali **informazioni** vengono **trasmesse** da un passo all'altro nella sequenza di dati e quali invece vengono dimenticate.

Le funzioni di ingate utilizzano una serie di valori di "gate" per determinare quali informazioni vengono trasmessi e quali invece vengono ignorati. Ad esempio, in una rete neurale **LSTM** (Long Short-Term Memory), ci sono **tre porte** di ingate: l'ingate di input (input gate), l'ingate di dimenticanza (forget gate) e l'ingate di uscita (output gate). Ognuna di queste porte utilizza una **funzione** di ingate per **determinare** quali informazioni vengono **trasmessi** o **ignorati** durante il processo di elaborazione della sequenza di dati.

Le funzioni di ingate sono utilizzate per **gestire** l'**informazione** nel tempo e per fare in modo che il modello presti maggiore attenzione agli elementi più importanti dell'input durante il processo di elaborazione.

70 Squeeze and excitation

Uno strato di riduzione spaziale (SE, dall'inglese Squeeze-and-Excitation) è un tipo di strato utilizzato in alcuni modelli di deep learning per **aumentare** la **capacità** di un modello di **concentrarsi** su elementi specifici dell'input.

Uno strato SE consiste in **due parti**: una parte di "pressione" (squeeze) e una parte di "eccitazione" (excitation). La parte di **pressione** utilizza una **funzione** di riduzione per **comprimere** le informazioni presenti nell'input in uno spazio di dimensioni ridotte. La parte di **eccitazione** utilizza quindi queste **informazioni** compresse per **determinare** quali elementi dell'input meritano maggiore **attenzione**.

Gli strati SE vengono spesso utilizzati in combinazione con le reti neurali convoluzionali per consentire al modello di prestare maggiore attenzione alle caratteristiche più importanti dell'input.

71 Transformer

I transformer sono composti da diverse componenti, come le seguenti:

- **Encoder**: è responsabile di codificare l'input in una rappresentazione a basso livello.
- **Decoder**: è responsabile di decodificare la rappresentazione dell'encoder e di produrre l'output desiderato.

- **Moduli di attenzione**: sono responsabili di calcolare i pesi di attenzione che indicano l'importanza di ogni elemento dell'input rispetto agli altri.
- **Strati di feed-forward**: sono strati di reti neurali standard che elaborano l'input in modo sequenziale.
- **Strati di normalizzazione**: sono responsabili di normalizzare i valori dell'input in modo da evitare il problema dell'esplosione del gradiente.
- **Strati di addizione e moltiplicazione**: sono responsabili di combinare i risultati degli altri strati in modo da ottenere l'output finale.