Domande di teoria sull'intelligenza artificiale

- Intelligenza artificiale, Machine Learning e Deep Learning.
  - Intelligenza artificiale: Riproduzione parziale dell'attività intellettuale propria dell'uomo (con particolare riguardo ai processi di apprendimento, di riconoscimento, di scelta) realizzata o attraverso l'elaborazione di modelli ideali, o, concretamente, con la messa a punto di macchine che utilizzano per lo più a tale fine elaboratori elettronici.
    - L'intelligenza Artificiale forte riguarda sistemi che sono progettati per risolvere problemi specifici in un settore particolare.
    - L'intelligenza Artificiale debole riguarda sistemi che sono progettati per risolvere problemi generici.
  - Machine Learning: "Il Machine Learning è quella branca dell'informatica che permette a una macchina di imparare ad eseguire un compito senza essere esplicitamente programmata per farlo" - 1959 Herbert Simon
  - Deep Learning: o deep neural learning , è un sottoinsieme del machine learning , che utilizza le reti neurali per analizzare diversi fattori con una struttura simile al sistema neurale umano.
- Storia dell'intelligenza artificiale, Origini, I due Inverni, Tempi moderni: 2011 ad oggi Deep Learning
  - La storia
- Paradigma del machine Learning vs Paradigma di programmazione tradizionale.
  - «Il Machine Learning è il campo di studi che fornisce ai computer la capacità di imparare a risolvere i problemi senza essere esplicitamente programmati» - Artur Samuel 1959. Prima era necessario istruire la macchina su come risolvere un specifico problema, mentre ora grazie al machine learning la macchina è in grado di risolvere problemi di cui non è stata istruita.
- Preparazione dei dati (Training Set, Validation Set, Testing Set) – Modello – Predizione
  - Training Set: è un insieme di dati utilizzati per addestrare un modello di machine learning. Il training set è un sottoinsieme del dataset utilizzato per addestrare il modello.
  - Validation Set: su questi dati vengono messi a punto gli iperparametri del modello.
  - Testing Set: è un insieme di dati utilizzati per valutare le prestazioni di un modello di machine learning.
  - Modello: è il nucleo di un sistema di machine learning. È il risultato dell'addestramento del modello su un training set. Il modello rappresenta la conoscenza acquisita dal sistema di machine learning.
  - Predizione: è il processo di utilizzo di un modello per effettuare previsioni su dati sconosciuti.
- Task del Machine Learning: Classificazione, Regressione e Clustering

  - Classificazione: il concetto di classe è correlato al concetto di etichetta. L'obiettivo è quello di assegnare un'etichetta a un oggetto in base alle sue caratteristiche.
  - Regressione: viene utilizzata per modellare la relazione tra una variabile dipendente e una o più variabili indipendenti, al fine di fare previsioni su nuovi dati.
  - Clustering: è una tecnica di apprendimento non supervisionato che raggruppa gli oggetti in base alle loro caratteristiche.
- Necessità di una fase di feature extraction nel Machine learning
  - è la procedura di estrazione delle caratteristiche dai dati in modo da creare un nuovo e più contenuto insieme di dati in grado di descrivere in modo più efficace il problema che si vuole risolvere.
- Deep Learning vs Machine Learning
  - Il machine learning è un campo che si occupa dello sviluppo di algoritmi che permettono ai computer di imparare dai dati e di fare previsioni o prendere decisioni senza essere esplicitamente programmati. Questi algoritmi analizzano i dati di input e identificano i modelli o le relazioni in essi contenuti per fare previsioni o prendere decisioni.
  - Il deep learning è una sottocategoria del machine learning che utilizza reti neurali artificiali profonde per l'apprendimento automatico delle caratteristiche dai dati. Queste reti neurali profonde sono composte da molti strati di neuroni artificiali e possono apprendere direttamente dai dati senza la necessità di estrarre manualmente le caratteristiche.
  - In sintesi, il deep learning è un approccio più avanzato in cui le reti neurali profonde possono imparare automaticamente le caratteristiche dai dati, eliminando la necessità di una pre-elaborazione manuale. Tuttavia, richiede una grande quantità di dati di addestramento e risorse computazionali per l'addestramento.
- Classificazione degli algoritmi di Machine Learning: Algoritmi Supervisionati (Regressione e classificazione), Algoritmi non supervisionati (Clustering ed Associazione), Apprendimento Semi-supervisionato, Apprendimento con rinforzo.
  - Supervisionato: è un tipo di apprendimento nel quale vengono presentati i dati di input e gli output che essi dovrebbe generare, con lo scopo di apprendere una regola generale in grado di mappare la relazione tra gli stessi.
    * Regressione: è un tipo di apprendimento supervisionato che si occupa di predire un valore numerico continuo. In altre parole, si tratta di modellare una relazione tra le variabili di input e una variabile di output continua.
    * Classificazione: è un tipo di apprendimento supervisionato che si occupa di predire un valore discreto.
  - Non supervisionato: è un tipo di apprendimento nel quale non vengono presentati gli output attesi, ma il sistema deve essere in grado

di trovare da solo la struttura dei dati. In questo modo si vanno a creare classi con dati più utili per le analisi successive e permettendo si scoprire nuove informazioni nei dati che non erano state considerate.
    * Clustering: è un tipo di apprendimento non supervisionato che si occupa di raggruppare gli oggetti in base alle loro caratteristiche.
    * Associazione: è un tipo di apprendimento non supervisionato che si occupa di trovare le relazioni tra gli oggetti.
  - Semi-supervisionato: è un tipo di apprendimento nel quale vengono presentati sia dati di input che gli output attesi, ma solo per una parte dei dati. Il sistema deve essere in grado di trovare da solo la struttura dei dati.
  - Rinforzo: è un tipo di apprendimento nel quale il sistema deve imparare a prendere decisioni in base alle interazioni con l'ambiente. Il sistema riceve un feedback in base alle sue azioni e deve imparare a massimizzare il feedback ricevuto.
- Reti Neurali Artificiali: Neurone Biologico e Neurone artificiale. Funzioni di attivazione. Percettrone a soglia e limiti.
  - Neurone Biologico: è un tipo di cellula del sistema nervoso che è in grado di ricevere, elaborare e trasmettere informazioni. I neuroni sono collegati tra loro tramite sinapsi. Quando un neurone riceve un segnale da altri neuroni, elabora le informazioni e trasmette il segnale ai neuroni collegati tramite sinapsi. E' composto da:
    * Dendriti: sono le estremità del neurone che ricevono i segnali da altri neuroni.
    * Corpo cellulare: elabora i segnali ricevuti dai dendriti e trasmette il segnale ai neuroni collegati tramite sinapsi.
    * Assone: trasmette il segnale ai neuroni collegati tramite sinapsi.
    * Sinapsi: sono le connessioni tra i neuroni. Quando un neurone riceve un segnale da altri neuroni, elabora le informazioni e trasmette il segnale ai neuroni collegati tramite sinapsi.
  - Neurone artificiale: è un modello matematico che si ispira al neurone biologico. E' composto da:
    * Input: sono i segnali in ingresso al neurone.
    * Pesi: sono i parametri che vengono utilizzati per modificare l'importanza dei segnali in ingresso.
    * Funzione di attivazione: è una funzione che viene utilizzata per calcolare l'output del neurone.
    * Output: è il segnale in uscita dal neurone.
  - Funzioni di attivazione: serve per derminare se un neurone deve essere attivato o meno. Si tratta di una funzione non lineare che prende in input la somma pesata dei segnali in ingresso al neurone e restituisce un output.
  - Percettrone a soglia: è un modello matematico di neurone artificiale che prende in input un vettore di valori numerici e restituisce un output binario (0 o 1). Si tratta di una funzione non lineare che

prende in input la somma pesata dei segnali in ingresso al neurone e restituisce un output.
    * Limiti del Percettrone a soglia: il percettrone a soglia non è in grado di modellare funzioni non lineari. Per ovviare a questo problema sono state introdotte le reti neurali feedforward.
- Reti Neurali artificiali: Input Layer, Hidden Layer, Output Layer. Reti FeedForward, Reti ricorrenti.
  - Una rete neurale artificiale è costituita da neuroni artificiali collegati tr loro. Ogni connessione, come in un cervello biologico, può trasmettere un segnale da un neurone all'altro. I neuroni sono organizzati in strati. Ogni connessione ha un peso che determina l'importanza del segnale trasmesso. Le reti neurali artificiali sono composte da gruppi di neuroni artificiali organizzati in strati:
    * Input Layer: è lo strato di input della rete neurale. I neuroni di questo strato ricevono i dati in input alla rete neurale.
    * Hidden Layer: è uno strato intermedio tra lo strato di input e lo strato di output. I neuroni di questo strato elaborano i dati ricevuti in input e trasmettono il segnale ai neuroni dello strato successivo.
    * Output Layer: è lo strato di output della rete neurale. I neuroni di questo strato restituiscono il risultato della rete neurale.
  - Reti FeedForward: è un tipo di rete neurale artificiale in cui i segnali si muovono in una sola direzione, dallo strato di input allo strato di output.
  - Reti ricorrenti: è un tipo di rete neurale artificiale in cui i segnali si muovono in entrambe le direzioni, dallo strato di input allo strato di output e viceversa.
- MultiLayer Preceptron (MLP)
  - E' un tipo di rete neurale artificiale feedforward in cui i neuroni sono organizzati in strati. Si tratta di uno dei modelli più utilizzati per il deep learning. L'MLP è in grado di apprendere relazioni complesse e non lineari tra i dati di input e l'output desiderato grazie alla sua struttura multistrato e alla capacità di modellare rappresentazioni gerarchiche dei dati. L'addestramento dell'MLP avviene utilizzando algoritmi di ottimizzazione come la retropropagazione dell'errore, che calcola e aggiorna i pesi delle connessioni in base all'errore tra l'output previsto e l'output desiderato. Il MLP più comune è l'FFNN (FeedForward Neural Network), ed è composto da:
    * un input Layer: è lo strato di input della rete neurale. I neuroni di questo strato ricevono i dati in input alla rete neurale.
    * uno o più Hidden Layer: è uno strato intermedio tra lo strato di input e lo strato di output. I neuroni di questo strato elaborano i dati ricevuti in input e trasmettono il segnale ai neuroni dello strato successivo.
    * un output Layer: è lo strato di output della rete neurale. I neuroni di questo strato restituiscono il risultato della rete neurale.

- Training di una rete neurale. Forward Propagagation e Backward Propagation.
  - Training di una rete neurale: è il processo di addestramento di una rete neurale artificiale. Iterativamente, la rete neurale viene esposta a un insieme di dati di addestramento e viene aggiornata in base all'errore tra l'output previsto e l'output desiderato. L'obiettivo del training è quello di ridurre l'errore tra l'output previsto e l'output desiderato.
    - Forward Propagation: è il processo di calcolo dell'output di una rete neurale artificiale. L'output di ogni neurone viene calcolato utilizzando la funzione di attivazione e i pesi delle connessioni.
    - Backward Propagation: è il processo di aggiornamento dei pesi delle connessioni di una rete neurale artificiale. L'aggiornamento dei pesi delle connessioni avviene utilizzando l'algoritmo di ottimizzazione della retropropagazione dell'errore.
- Loss Function e Funzione costo.
  - Loss Function: è una funzione che misura l'errore tra l'output previsto e l'output desiderato. L'obiettivo del training di una rete neurale artificiale è quello di ridurre l'errore tra l'output previsto e l'output desiderato. La loss function viene utilizzata per calcolare l'errore tra l'output previsto e l'output desiderato.
  - Funzione costo: è una funzione che misura l'errore tra l'output previsto e l'output desiderato. L'obiettivo del training di una rete neurale artificiale è quello di ridurre l'errore tra l'output previsto e l'output desiderato. La funzione costo viene utilizzata per calcolare l'errore tra l'output previsto e l'output desiderato. La funzione costo viene calcolata come: $C = \frac{\sum_{i=1}^{n} L(y_i, \hat{y_i})}{n}$ dove n è il numero di esempi di addestramento, y_i è l'osservazione effettiva dell'esempio di training i-esimo, $\hat{y_i}$ è la previsione dell'esempio di training i-esimo.
- Reti neurali Convoluzionali.
  - Le reti MLP mancano di invarianza per traslazione, il che significa che sono sensibili ai cambiamenti di posizione dei pixel all'interno di un'immagine. Le reti neurali convoluzionali (CNN) sono reti profonde (deep) ispirate alle ricerche biologiche di Hubel e Wiesel durante lo studio del cervello dei gatti. Utilizzano due tipi di celle:
    - celle semplici : specializzate nella rilevazione di caratteristiche locali dell'input visivo (feature extractor), (Convoluzioni)
    - celle complesse : specializzate nell'integrazione (pooling) delle informazioni provenienti da diverse posizioni retinotopiche per formare una rappresentazione globale dell'input visivo, preservando le caratteristiche invarianti per posizione
- Architettura di una rete CNN: parte convoluzionee parte fully-connected
  - La parte convoluzionale consiste di strati convoluzionali seguite da funzioni di attivazione non lineare tipo(RELU) e di pooling. Questa parte costituisce il componente essenziale dell'estrazione di feature

  - La parte fully-connected consiste in un'architettura di rete neurale completamente connessa. Questa parte esegue il compito di classificazione in base all'input dalla parte convoluzionale.
- Sotto quali condizioni, il metodo di discesa del gradiente con passo fisso converge a un punto stazionario della funzione costo, che può essere un minimo globale se la funzione è convessa?
  - La discesa del gradiente trova il minimo globale se la funzione costo C è convessa e se sono soddisfatte le seguenti condizioni:
    * Condizione di Lipschitz per il gradiente: deve esistere una costante tale che il gradiente della funzione costo sia limitato superiormente da tale costante. In altre parole, il gradiente della funzione costo non deve essere troppo grande.
    * Step size: il passo di discesa deve essere scelto in modo che sia sufficientemente piccolo da garantire che la funzione costo decresca in ogni iterazione. In altre parole, il passo di discesa non deve essere troppo grande. E' importante anche sapere che la scelta del passo n può essere limitante in termini di efficienza e convergenza in problemi di grandi dimensioni.
- Non convessità della funzione di costo.
  - La funzione costo di una rete neurale è non convessa. Questo significa che la discesa del gradiente non trova il minimo globale. Tuttavia, la discesa del gradiente trova un minimo locale. In altre parole, la discesa del gradiente trova un minimo locale che potrebbe non essere il minimo globale. Per gestire la non convessità della funzione di costo, è necessario eseguire più volte la discesa del gradiente con diversi valori iniziali dei pesi.
- Importanza del learning rate nei metodi di discesa.
  - Il learning rate è un iperparametro che controlla la dimensione del passo di discesa del gradiente. Determina quanto velocemente o lentamente il modello si adatta ai dati. Esso determina la dimensione dei passi che vengono compiuti durante l'aggiornamento dei parametri del modello lungo il gradiente della funzione di costo. L'importanza del learning rate risiede nel fatto che esso può influenzare significativamente la convergenza del metodo di discesa del gradiente e la qualità della soluzione ottenuta. Un learning rate troppo grande può causare oscillazioni o divergenza del processo di ottimizzazione, mentre un learning rate troppo piccolo può rallentare notevolmente la convergenza, richiedendo più iterazioni per raggiungere una soluzione accettabile. Un learning rate adeguato è essenziale per ottenere una convergenza stabile e rapida del metodo di discesa del gradiente. Un learning rate ottimale dipende dalla natura del problema, dalla forma della funzione di costo e dalla scala dei dati.
- Exact line search.
  - Gli algoritmi di ottimizzazione unidimensionale per trovare la dimensione del passo ottimale sono genericamnete chiamati exact line search.

- Inexact line search rule , Armijo rule.
  - La regola di Armijo, o regola di Armijo-Goldstein, è un criterio utilizzato nell'ottimizzazione numerica per determinare la dimensione del passo (o step size) da utilizzare durante la ricerca lineare in una direzione di discesa all'interno di un algoritmo di ottimizzazione. Questa regola garantisce una riduzione significativa della funzione costo durante l'ottimizzazione, assicurando nel contempo una convergenza adeguata dell'algoritmo. La regola di Armijo si basa sul concetto di "backtracking" o retrotracciamento, che consiste nel ridurre gradualmente la dimensione del passo fino a trovare un valore che soddisfi determinate condizioni. L'obiettivo è trovare un passo che garantisca una riduzione sufficiente del valore della funzione costo durante la ricerca lineare. Il processo inizia con un punto iniziale e riduce progressivamente la dimensione del passo, moltiplicandola per un parametro compreso tra 0 e 1. Questo processo viene iterato fino a quando non viene raggiunta una riduzione sufficiente nella funzione obiettivo, rispettando una specifica condizione di Armijo. La condizione di Armijo richiede che la riduzione della funzione costo sia proporzionale alla riduzione prevista calcolata in base al gradiente e al passo corrente. Questo criterio permette di trovare uno step size adeguato che assicura una significativa diminuzione della funzione costo. Utilizzando la regola di Armijo, è possibile controllare la dimensione del passo durante l'ottimizzazione, bilanciando la velocità di convergenza con la riduzione della funzione costo. In questo modo, si può garantire una progressione efficace verso la soluzione ottimale del problema di ottimizzazione.
- Metodo di ottimizzazione del gradient descent con momento. Perchè è stato studiato e formula di aggiornamento dei pesi.
  - Il Gradient Descent aggiorna i parametri con: $w_{k+1} = w_k - \eta \nabla C(w_k)$. La discesa del gradiente trova il minimo globale se la funzione costo C è convessa e se sono soddisfatte le condizioni di Lipschitz per il gradiente e lo step size. Questo metodo è stato studiato per affrontare due problematiche comuni nell'ottimizzazione: la lentezza della convergenza e la possibilità di rimanere bloccati in minimi locali. Il concetto chiave del metodo del gradiente con momento è l'introduzione di un termine di momento, che rappresenta l'accumulo di informazioni sulle iterazioni precedenti per guidare l'aggiornamento dei pesi. Il momento agisce come una sorta di "inerzia" che accelera l'ottimizzazione nella direzione in cui la discesa del gradiente è costante e rallenta quando la direzione cambia bruscamente.
- Iperparametri di una rete neurale.
  - Gli iperparametri sono parametri esterni al modello di machine learning che devono essere impostati prima dell'avvio del processo di addestramento. A differenza dei parametri del modello, che vengono appresi durante il processo di addestramento stesso, gli iperparametri influenzano il comportamento del processo di addestramento e la con-

figurazione del modello. Gli iperparametri devono essere scelti con cura, in quanto possono influenzare significativamente la qualità della soluzione ottenuta e la velocità di convergenza del processo di addestramento. Gli iperparametri di un modello di rete neurale includono: - numero di epoche di addestramento; - dimensione del batch di addestramento; - learning rate; - funzione di costo; - architettura della rete neurale (numero di layer, numero di neuroni per layer, tipo di layer); - funzione di attivazione dei neuroni; - regolarizzazione (early stopping, dropout, L1, L2); - ottimizzatore (Gradient Descent, SGD, Adam, Adagrad, RMSProp); - ecc.

- learning rate scheduling: step decay, decadimento esponenziale, decadimento dipendente dal tempo.
  - Durante l'allenamento di un modello di machine learning, la regolazione del learning rate, o tasso di apprendimento, è spesso altrettanto importante della scelta dell'ottimizzatore. La selezione adeguata del learning rate può influire sulla velocità e sulla qualità della convergenza dell'algoritmo di ottimizzazione. All'inizio dell'allenamento, un learning rate elevato è auspicabile poiché i pesi del modello sono lontani dai minimi. Un learning rate alto consente un rapido aggiornamento dei pesi, accelerando l'apprendimento iniziale. Tuttavia, nella fase finale dell'apprendimento, quando i pesi si avvicinano ai minimi, un learning rate basso è più appropriato. Riducendo gradualmente il learning rate, si aumenta la probabilità di raggiungere il minimo globale senza oscillazioni eccessive. La regolazione del learning rate richiede la considerazione di diversi aspetti. In primo luogo, è importante selezionare una dimensione adeguata per il learning rate. Se il learning rate è troppo grande, l'ottimizzazione può divergere, mentre se è troppo piccolo, l'allenamento richiederà molto tempo o si otterrà un risultato subottimale. Inoltre, è importante considerare il tasso di decadimento del learning rate nel corso dell'allenamento. Se il learning rate rimane elevato per troppo tempo, si può rimbalzare intorno al minimo senza raggiungerlo. Un decadimento graduale del learning rate può essere vantaggioso per raggiungere un compromesso tra la velocità di convergenza e la precisione del risultato finale.
- Learning rate adattivo: Adagrad, RMSProp, Adadelta, Adam.
  - La regolazione del learning rate è una sfida nell'ottimizzazione dei modelli di machine learning, poiché gli iperparametri devono essere definiti in anticipo e dipendono dal tipo di modello e problema. Inoltre, applicare lo stesso learning rate a tutti gli aggiornamenti dei pesi potrebbe non essere ottimale, specialmente quando si hanno dati sparsi e si desidera aggiornare i pesi in modo diverso. Per affrontare queste sfide, sono stati sviluppati quattro metodi rappresentativi che modificano in modo adattivo il learning rate:
    * Adagrad: Adagrad adatta il learning rate per ogni parametro del modello in base alla frequenza degli aggiornamenti prece-

denti. Ciò significa che i parametri che vengono aggiornati più frequentemente hanno un learning rate ridotto, mentre i parametri meno frequentemente aggiornati hanno un learning rate maggiore. Questo approccio aiuta a gestire i problemi di sparsity dei dati.
    * RMSProp: RMSProp (Root Mean Square Propagation) modifica il learning rate in base alla media mobile degli aggiornamenti precedenti dei gradienti. Questo metodo attenua l'impatto dei gradienti di grandi dimensioni, consentendo una regolazione più stabile del learning rate nel corso dell'allenamento.
    * Adadelta: Adadelta è un'altra tecnica che adatta il learning rate in base alla media mobile degli aggiornamenti dei gradienti precedenti. Tuttavia, a differenza di RMSProp, Adadelta utilizza anche una stima della varianza dei gradienti per regolare il learning rate. Questo approccio permette un aggiornamento più stabile dei pesi nel corso dell'allenamento.
    * Adam: Adam (Adaptive Moment Estimation) combina le caratteristiche di Adagrad e RMSProp. Utilizza la media mobile dei gradienti e la varianza per adattare il learning rate. Inoltre, incorpora un termine di momento che tiene conto della storia degli aggiornamenti precedenti dei gradienti. Adam è uno dei metodi più utilizzati in pratica ed è efficace in una vasta gamma di problemi.
- Altro
  - Il vantaggio di jacobi è che si può parallelizzare. Ma nel caso non si parallelizza risulta essere più lento di gauss seidel