Metodi

luca carabini

July 2023

Indice

Capitolo 1

Intelligenza artificiale, Machine Learning e Deep Learning

1.1 Intelligenza artificiale

Un' intelligenza artificiale è la 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.

1.1.1 Intelligenza artificiale debole

si riferisce a sistemi che sono progettati per risolvere problemi specifici in un settore particolare. Questi sistemi sono in grado di imparare da dati passati e di prendere decisioni in base a questi dati, ma non hanno una comprensione più ampia del mondo o dell'esperienza umana. Esempi di AI debole includono chatbot, raccomandazioni di prodotti, analisi dei dati, e così via.

1.1.2 Intelligenza artificiale forte

si riferisce a sistemi in grado di replicare tutte le funzioni cognitive umane, comprese la comprensione del linguaggio naturale, la percezione visiva, l'apprendimento, il ragionamento e la creatività. Un sistema di AI forte sarebbe in grado di risolvere qualsiasi problema che un essere umano potrebbe risolvere. Tuttavia, un tale sistema non esiste ancora ed è stato oggetto di dibattito filosofico.

1.2 Machines Learning

Il Machine Learning come un programma che apprende soltanto dopo aver migliorato le proprie prestazioni a seguito dello svolgimento di un compito. Secondo l'idea di Machine Learning è possibile rappresentare la realtà come una funzione matematica di cui l'algoritmo non è a conoscenza, ma che è in grado di ricavare dopo aver elaborato i dati forniti dalla stessa

1.3 Deep Learning

Il 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.

Capitolo 2

Storia dell'intelligenza artificiale e Deep Learning

2.1 Storia del Machines Learning

2.1.1 Tra il 1940 e il 1974:

- Primi calcolatori
- teoria della computazione di Turing, Test di Turing
- Nel 1943 primo neurone artificiale
- Nel 1957 creazione di G.P.S. un programma al fine di risolvere problemi (formalizzati)
- 1965 Eliza un chatbot
- Grande entusiasmo e predizioni troppo ottimistiche

2.1.2 1974-1984 : Primo Inverno

- Risultati non all'altezza delle aspettative, drastica riduzione dei finanziamenti
- Problemi: scarsa capacità computazionale, esplosione combinatoria e non trattabilità, data set di piccole dimensioni. Ridimensionamento dell'approccio connessionistico (reti neurali)

2.1.3 1980-1987 Nuova Primavera

- Nascita dei sistemi aperti : conoscenza + regole logiche
- Nuova linfa alle reti neurali dell'algoritmo BackPropagation(1986)
- Finanziamento da parte del governo Giapponese per la Quinta Generazione

2.1.4 1987-1993 Il secondo inverno

- Flop della Quinta Generazione e nuovo stop ai finanziamenti
- Risultati concreti dei sistemi esperti solo in campi specifici. Reti neurali non scalano a problemi complessi

2.1.5 1993-2011 Tempi moderni

- Hardware sempre più potenti
- Classificatori robusti(SVM), Multi Classificatori
- Maturità tecniche di feature extraction(fatte a mano) per esempio SIFT
- Successi in numerose discipline(visione, sistemi biometrici ecc..)

2.2 dal 2011 a oggi(Deep Learning)

- CNN: Conviltional Neural Network
- 2012 rivoluzione in Computer Vision
- 2016 Speech Recognition (es: Siri, Google) e Language Translation 2023 ChatGPT

Capitolo 3

Paradigma del machine Learning vs Paradigma di programmazione tradizionale

3.1 Paradigma di programmazione tradizionale

- Cambia radicalmente l'approccio del programmatore : non bisogna più scrivere dettagliate righe di codice per istruire la macchina su cosa fare situazione per situazione, oggi è l'algoritmo a sviluppare una sua logica e conseguentemente a compiere determinate azioni, a seconda del set di dati a disposizione
- Apprendimento automatico (ML) è un sottoinsieme del campo AI che cerca di sviluppare sistemi in grado di apprendere automaticamente da esempi specifici (training data) e di generalizzare la conoscenza su nuovi campioni (test data) dello stesso dominio.
- Procedimento :
  1 Disponiamo di alcuni dati che rappresentano il nostro dominio applicativo
  2 Implementiamo un algoritmo in grado di apprendere da quei dati (fase di training)
  3 Utilizziamo i dati (diversi dal set di addestramento) per capire se il modello addestrato ha imparato qualcosa (fasi di validazione e test) -¿ deployment del modello

3.2 Paradigma di programmazione tradizionale

3.2.1 Acquisizione dati

I dati sono l'elemento fondante di qualsiasi applicazione correlata al ML. L'acquisizione di grandi quantità di dati è oggi uno degli obiettivi principali delle migliori aziende. I dati si possono ottenere in due modi:

- Usare set di dati o database di dati pubblicamente disponibili
- Acquisendo un nuovo set di dati

3.2.2 Data processing

Tutte quelle tecniche con cui vengono elaborati i dati per adattarli al meglio al modello ML che intendiamo sviluppare

3.2.3 Modello

Questo è il nucleo principale del sistema AI. Un modello può essere visto come un insieme di tecniche matematiche e statistiche (ma non solo), in grado di apprendere da una certa distribuzione di dati (o caratteristiche) forniti in input e di generalizzare su nuovi dati.

3.2.4 Predizione

L'output del modello può assumere molte forme a seconda dell'applicazione sviluppata. È importante valutare l'efficacia del sistema sviluppato

3.2.5 Organizzare i dati:

- Training set, I dati sui quali il modello apprende automaticamente durante la fase di apprendimento.
- Validation set, parte del training set. Su questi dati, vengono messi a punto gli iperparametri .
- Testing set, dati su cui il modello viene testato durante la fase di test. La fase di test verifica l'efficacia del modello, anche attraverso misure numeriche qualitative e quantitative.

Capitolo 4

Task del Machine Learning: Classificazione, Regressione e Clustering

4.1 Classificazione

Dato un input specifico, il modello (classificatore) emette una classe Se ci sono solo 2 classi, chiamiamo il problema classificazione binaria Se ci sono più classi (¿2), chiamiamo il problema classificazione multiclasse. Una classe è :

- Un set di dati con proprietà comuni
- Il concetto di classe è correlato al concetto di "etichetta" precedentemente introdotto
- Il concetto di classe è semantico, in quanto strettamente dipendente dal contesto

4.2 Regressione

La regressione viene utilizzata per modellare la relazione tra le variabili indipendenti e la variabile dipendente, in modo da poter fare previsioni su nuovi dati. Esempi di compiti di regressione:

- Stima dell'altezza di una persona in base al peso
- Stima dei prezzi di vendita degli appartamenti nel mercato immobiliare
- Stima del rischio per le compagnie assicurative
- Previsione dell'energia prodotta da un sistema fotovoltaico
- Modelli di previsione dei costi sanitari

4.3 Clustering

Il clustering è spesso applicato in un ambiente di apprendimento non supervisionato, in cui i dati non sono etichettati e/o le classi del problema non sono note in anticipo.

Di solito, la natura non supervisionata del problema lo rende più complesso rispetto alla classificazione.

Spesso, anche il numero di cluster non è noto a priori. I cluster identificati possono essere usati come classi.

Esempi di clustering

- Definizione di gruppi di utenti basati sul consumo nel marketing
- Raggruppamento di individui in base alle analogie del DNA nella genetica
- Partizionamento dei geni in gruppi con caratteristiche simili nella bioinformatica
- Segmentazione non supervisionata nella visione

Capitolo 5

Necessità di una fase di feature extraction nel Machine learning

La Feature Extraction èla procedura di estrazione delle feature dai dati (caratteristiche dei dati): è un modo per creare un nuovo e più piccolo insieme di dati che cattura la maggior parte delle informazioni dei dati grezzi.

Capitolo 6

Deep Learning – vs Machine Learning

Nel Machine learning la feature extraction è svolta da un umano, mentre nel deep learning viene svolta dall'algoritmo

Capitolo 7

Classificazione degli algoritmi di Machine Learning:

Gli algoritmi del Machines learning possono essere :

- Supervisionati
- Non supervisionati
- Semi-supervisionati
- Con rinforzo

Capitolo 8

Algoritmi Supervisionati(Regressione e classificazione)

Algoritmi supervisionati, il modello di apprendimento nel quale vengono presentati i dati di input e gli output che essi dovrebbero generare, con lo scopo di apprendere una regola generale in grado di mappare la relazione tra gli stessi. Per usare l'apprendimento supervisionato è necessario avere dati con output noti.

8.1 Classificazione

Modello di apprendimento automatico in grado di stimare se c'è una mela o una banana su un nastro trasportatore.

- Dobbiamo quindi utilizzare immagini sia di mele che di banane, e per ogni immagine si deve indicare se si tratta di una mela o di una banana che viene raffigurata.
- Una volta che il modello di machine learning è stato addestrato, può essere utilizzato per fare previsioni su nuovi dati che non hanno un'etichetta.

8.2 Regressione

La regressione viene utilizzata per modellare la relazione tra le variabili indipendenti e la variabile dipendente, in modo da poter fare previsioni su nuovi dati. L'obiettivo della regressione è quello di trovare la migliore funzione che descriva la relazione tra le variabili. Ad esempio, nel caso della regressione lineare, l'obiettivo è quello di trovare la migliore linea retta che rappresenti la relazione tra le variabili. Il processo di regressione inizia con la raccolta dei dati, che sono divisi in un set di training e un set di test. Il set di training viene utilizzato per addestrare il modello di regressione, mentre il set di test viene utilizzato per valutare la qualità del modello. Una volta addestrato il modello, è possibile utilizzarlo per fare previsioni su nuovi dati. Ci sono molti algoritmi di regressione disponibili, tra cui la regressione lineare, la regressione logistica, la regressione polinomiale e molte altre. La scelta dell'algoritmo dipende dalle caratteristiche dei dati e dall'obiettivo della regressione.

Capitolo 9

Algoritmi non supervisionati (Clustering ed Associazione)

L'algoritmo deve imparare a riconoscere pattern nell'input senza che sia data nessuna indicazione specifica dei valori in uscita.

Lo scopo è quello di apprendere in autonomia la struttura che possiedono i dati in ingresso.

Questo approccio va a ristrutturare i dati in maniera diversa, creando classi di dati più utili per le analisi successive, e permettendo di scoprire nuovi significati nei dati che prima non erano stati presi in considerazione.

Un agente che è in grado di apprendere in modo totalmente non supervisionato non è in grado di imparare osa fare, dal momento che non possiede informazioni su cosa costituisce un'azione corretta o uno stato desiderabile al quale arrivare. L'apprendimento non supervisionato è usato per gestire problemi di clustering andando a trovare gruppi di dati in base alle caratteristiche che hanno in comune.

Utilizzato anche per l'association learning, cioè per trovare regole associative tra i dati inseriti.

Inoltre può essere utilizzato per trovare transazioni fraudolente, prevedere vendite e sconti o analizzare le preferenze dei clienti in base alla loro cronologia delle ricerche. Il programmatore non sa cosa sta cercando di trovare, ma ci sono sicuramente alcuni schemi e il sistema può rilevarli.

Capitolo 10

Apprendimento Semi-supervisionato

Nell'apprendimento semi-supervisionato i dati di input sono una miscela di campioni etichettati e non etichettati. È possibile utilizzare l'approccio semi-supervisionato (semi-supervised learning) nel quale si combinano le caratteristiche dell'apprendimento supervisionato e non supervisionato. In un primo momento si utilizza l'approccio supervisionato andando a fornire gli input con l'output associato, e successivamente si inseriscono altri input, simili, senza l'output di riferimento. Gli input e gli output inseriti forniscono il modello generale dal quale è possibile estrapolare gli output degli input rimanenti. Si può utilizzare l'apprendimento per trasferimento (transfer learning) trasferendo la conoscenza creata per affrontare un determinato problema, per usarla nell'affrontare un problema simile. Il vantaggio che si ha dal riutilizzo della conoscenza è notevole, anche se quest'approccio non è sempre praticabile perché molti algoritmi di Machine Learning devono essere adattati al caso specifico. Utilizzato per: auto a guida autonoma, giochi, robot, gestione delle risorse

Capitolo 11

Apprendimento con rinforzo

Nell' Apprendimento con Rinforzo l'algoritmo interagisce con l'ambiente dinamico al fine di raggiungere un determinato obiettivo.

Gli algoritmi agiscono in base alle ricompense che gli vengono date in funzione degli obiettivi raggiunti, senza le quali non sarebbe possibile capire cosa è auspicabile fare e cosa è da evitare, e l'algoritmo non potrebbe prendere decisioni.

Il reinforcement learning viene utilizzato quando l'algoritmo deve prendere delle decisioni da cui discendono delle conseguenze. L'algoritmo non è semplicemente descrittivo ma è prescrittivo, cioè indica cosa bisogna fare. È un tipo di apprendimento molto innovativo nel Machine Learning, ed è predisposto ad imparare il funzionamento dell'ambiente.

Con questo tipo di approccio si riesce ad addestrare un algoritmo senza definire delle regole che vengono dedotte a seconda dei feedback ricevuti. È un approccio utilizzato in domini complessi perché è l'unico modo per addestrare un programma e ottenere buone prestazioni.

Nell'apprendimento con rinforzo di tipo attivo l'agente deve imparare cosa deve fare, e tramite l'esplorazione, deve sperimentare come comportarsi, dato l'ambiente in cui è inserito. È possibile implementare questa strategia facendo giocare l'algoritmo ad un gioco senza fornirgli le regole. Ogni volta che compie un'azione che è consentita dalle regole, o che è utile, gli verrebbe conferito un feedback positivo, altrimenti uno negativo. Questo permetterebbe all'algoritmo di imparare le mosse che può e che è meglio fare durante la partita, e imparerebbe a giocare senza che gli siano date le regole del gioco. L'agente deve imparare a comportarsi con successo nell'ambiente in cui è posto senza conoscere niente, ma apprendendo dai feedback ricevuti.

Esempi di algoritmi:

- Q-learning ,
- Algoritmi genetici,
- SARSA, DQN, A3C.

Utilizzato per:

- auto a guida autonoma
- giochi
- robot
- gestione delle risorse

Capitolo 12

Reti Neurali Artificiali

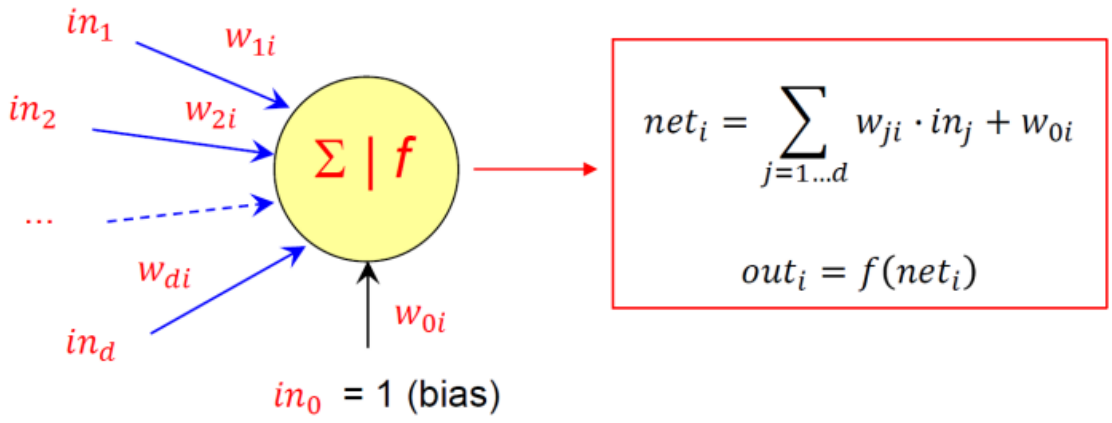
12.1 Neurone Biologico

Il neurone bilogico è formato da 4 parti:

- `Soma:` elabora i segnali in ingresso nel tempo e converte il valore elaborato in un output
- `Dendriti:` ricevono (input) segnali elettrici e chimici dagli altri neuroni e li trasmettono al soma.
- `Assone:` trasmette il segnale elettrico dal soma ad altri neuroni o, a cellule muscolari o ghiandolari.
- `Sinapsi:` collegano il neurone ad altri neuroni per trasmettere il segnale in uscita. Ogni singola sinapsi può modificare la propria risposta e variare, in questo modo, l'efficienza di trasporto dell'informazione.

12.2 il neurone artificiale

Primo modello del 1943 di McCulloch and Pitts. Con input e output binari era in grado di eseguire computazioni logiche.



- $in_1, \ldots, in_d$ sono i d ingressi che il neurone i riceve da assoni di neuroni afferenti
- $w_{1i}, \ldots, w_{di}$ sono i pesi (weight) che determinano l'efficacia delle connessioni sinaptiche dei dendriti (agiremo su questi valori durante l'apprendimento), l'importanza dell'input i-esimo sull'output.
- w_{0i} (detto bias) è un ulteriore peso che si considera collegato a un input fittizio con valore sempre 1 questo peso è utile per «tarare» il punto di lavoro ottimale del neurone.

- net_i è il livello di eccitazione globale del neurone (potenziale interno).
- $f()$ è la funzione di attivazione che determina il comportamento del neurone (ovvero il suo output out_i in funzione del suo livello di eccitazione net_i). La funzione di attivazione simula il comportamento del neurone biologico di attivarsi solo se i segnali in ingresso superano una certa soglia.

12.3 Funzioni di attivazione

Una funzione di attivazione determina se un neurone deve essere attivato o meno . Si tratta di alcune semplici operazioni matematiche per determinare se l'input del neurone alla rete è rilevante o meno nel processo di previsione. Le funzioni di attivazione possono essere di diversi tipi, ma in generale devono essere non lineari per consentire alla rete di apprendere relazioni complesse tra le sue variabili di input, e derivabili

12.4 Percettore a soglia

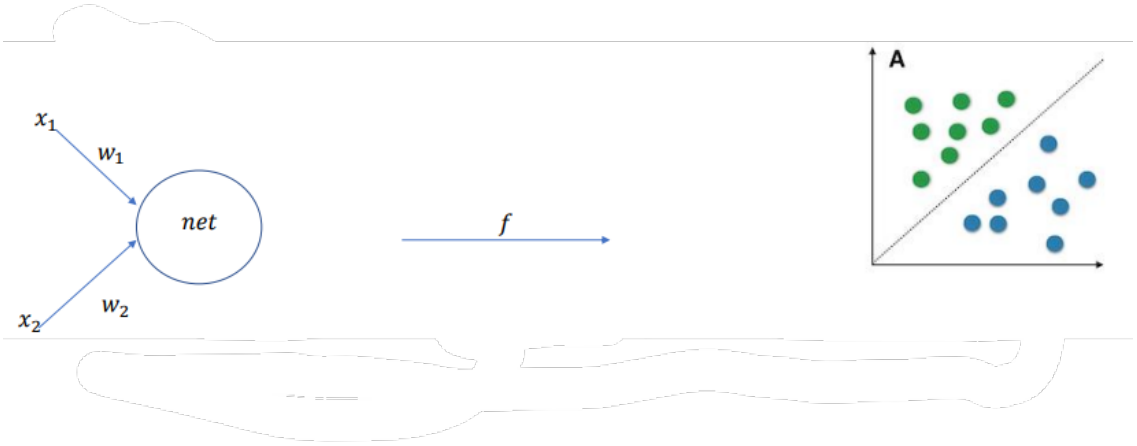
Perceptron: Modello single-neuron con funzione di attivazione a soglia. Effettua una separazione lineare tra due classi

$$f(net) = \begin{cases} 1 & se\ net < 0 \\ 1 & se\ net > 0 \end{cases} \tag{12.1}$$

12.4.1 Esempio

Per semplicità consideriamo il caso di soli 2 input.

$$net = w_1 x_1 + w_2 x_2 \tag{12.2}$$



$$\begin{aligned} w_1 x_1 + w_2 x_2 < 0 \rightarrow w_1 x_1 + w_2 x_2 - 0 < 0 \quad & (x_1, x_2)\ sta\ sopra\ la\ retta \\ w_1 x_1 + w_2 x_2 > 0 \rightarrow w_1 x_1 + w_2 x_2 - 0 > 0 \quad & (x_1, x_2)\ sta\ sotto\ la\ retta \end{aligned} \tag{12.3}$$

Equazione esplicita della retta $w_1 x_1 + w_2 x_2 - 0 = 0$

$x_2 = -\frac{-w_1}{w_2} x_1 + \frac{0}{w_2}$

12.5 Layer

Le reti neurali sono composte da gruppi di neuroni artificiali organizzati in livelli.

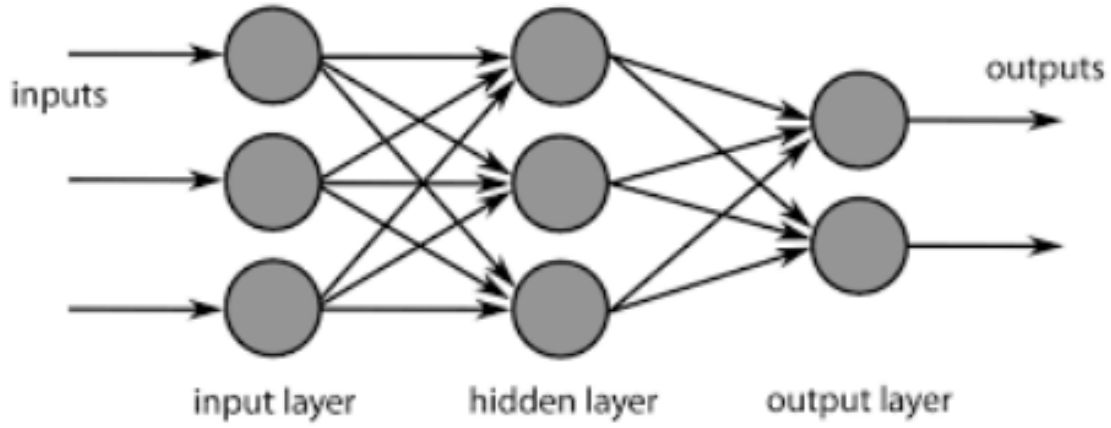
Tipicamente sono presenti un livello di input, un livello di output, e uno o più livelli intermedi.

Ogni livello contiene uno o più neuroni. I layer intermedi sono chiamati hidden layer in quanto restano "invisibili" dall'esterno della rete, la quale si interfaccia all'ambiente solo tramite il layer di ingresso e quello di uscita.

12.6 Reti FeedForward

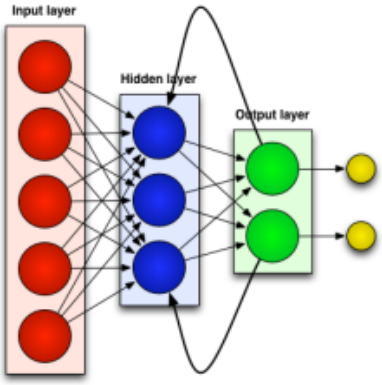
Nelle reti feedforward («alimentazione in avanti») le connessioni collegano i neuroni di un livello con i neuroni di un livello successivo.

Non sono consentite connessioni all'indietro o connessioni verso lo stesso livello. È di gran lunga il tipo di rete più utilizzata



12.7 Reti ricorrenti

Nelle reti ricorrenti sono previste connessioni di feedback, in genere verso neuroni dello stesso livello, ma anche all'indietro. Questo tipo di rete crea una sorta di memoria di quanto accaduto in passato e rende quindi l'uscita attuale non solo dipendente dall'ingresso attuale, ma anche da tutti gli ingressi precedenti



Capitolo 13

Multi Layer Preceptron

Multi Layer Perceptron (MLP) è l'FFNN più comune costituito da tre o più layer (strati):

- un layer di input
- uno o più layer nascosti
- un layer di output

Le MLP sono fully-connected: ogni neurone in uno strato è connesso con ogni neurone nello strato successivo

Un teorema noto come universal approximation theorem asserisce che ogni funzione continua che mappa intervalli di numeri reali su un intervallo di numeri reali può essere approssimata da un MLP con un solo hidden layer. Questa è una delle motivazioni per cui per molti decenni (fino all'esplosione del deep learning) ci si è soffermati su reti neurali a 3 livelli.

Capitolo 14

Training di una rete neurale

Il processo di apprendimento è una caratteristica chiave delle ANN ed è strettamente correlato al modo in cui il cervello umano apprende:

- eseguiamo un'azione
- siamo approvati o corretti da un trainer o coach per migliorare in un determinato compito.

Iterativamente

- i dati di addestramento vengono presentati alla rete (forward),
- quindi i pesi vengono aggiustati (backward) sulla base di quanto sono simili i valori restituiti dalla rete rispetto a quelli desiderati (loss)
- Dopo che tutti i casi sono stati presentati, il processo spesso ricomincia da capo

Durante la fase di apprendimento, i pesi vengono regolati per migliorare la performance sui dati di allenamento

14.1 Forward propagation

Con forward propagation (o inference) si intende la propagazione delle informazioni in avanti dal livello di input a quello di output.
Una volta addestrata, una rete neurale può semplicemente.
Processare pattern attraverso forward propagation.

Capitolo 15

Funzioni

15.1 Loss Function

La loss function, o funzione di perdita, è una misura dell'errore della previsione prodotta da un modello di machine learning rispetto ai dati di training. Essa rappresenta la discrepanza tra l'output previsto dal modello e l'output reale associato ai dati di training. // L'obiettivo del modello di machine learning è quello di minimizzare la loss function, ossia di trovare i valori dei parametri del modello che producono la previsione migliore possibile sui dati di training. // Una loss function è per un singolo esempio di addestramento

15.2 Cost Function

Una cost function è la perdita media sull'intero set di dati di addestramento. Le strategie di ottimizzazione mirano a minimizzare la funzione di costo.

$$C = \frac{\sum_{i=1}^{n} L(y_i, \hat{y_i})}{n} \tag{15.1}$$

Dove

- n è il numero di esempi di training
- y_i è l'osservazione effettiva dell'esempio di training i-esimo
- $\hat{y_i}$ la previsione dell'esempio di training i-esimo
- L è la loss function

Capitolo 16

Reti Convoluzionali

16.1 Limiti delle reti MLP nell'elaborazione delle immagini, mancanza di invarianza per traslazione

Le reti MLP non hanno alcuna invarianza per traslazione.

Una MLP reagirà in modo diverso ad un'immagine ed ad una sua versione traslata, anche se le immagini sono visivamente simili. Ciò è dovuto al fatto che la posizione dei pixel nell'immagine è un fattore importante per determinare l'output della rete.

Le informazioni spaziali più importanti vengono perse quando l'immagine viene appiattita in un MLP.

Conoscere le relazioni di vicinanza tra i pixel è importante perché aiuta a definire le caratteristiche di un'immagine

16.2 CNN

Le reti neurali convoluzionali (CNN) sono reti profonde (deep) ispirate alle ricerche biologiche di Hubel e Wiesel durante lo studio del cervello dei gatti.

- 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

16.3 Architettura di una rete CNN

16.3.1 Parte convulsione

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

16.3.2 Parte fully-connected

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.

Capitolo 17

Algoritmo di backpropagation

Capitolo 18

Tecniche di Ottimizzazione

18.1 Batch Gradient Descent

Per il calcolo della funzione costo C(w) vengono usate tutte gli n_T campioni del training set . Quindi:

- si considera l'intero set di addestramento,
- si esegue la Forward Propagation e si calcola la funzione di costo.
- si aggiornano i parametri usando il gradiente di questa funzione di costo rispetto ai parametri.

Epoca: l'intero set di addestramento viene passato attraverso il modello, vengono eseguite la propagazione in avanti e la propagazione all'indietro e i parametri vengono aggiornati.

Nel batch Gradient Descent poiché stiamo utilizzando l'intero set di addestramento, i parametri verranno aggiornati solo una volta per epoca.

18.2 Discesa del gradiente stocastico (SGD)

Se si utilizza una singola osservazione per calcolare la funzione di costo, si parla di Stochastic Gradient Descent, comunemente abbreviato SGD. Passiamo una sola osservazione alla volta, calcoliamo il costo e aggiorniamo i parametri

18.3 Mini-batch Stocastic Gradient Descent:

Per calcolare la funzione di costo si considera un sottoinsieme dell'intero set di dati. Quindi, se ci sono n_T osservazioni, il numero di osservazioni in ciascun sottoinsieme o mini-batch sarà maggiore di 1 e minore di n_T .

Capitolo 19

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?

- Condizione di Lipschitz per il gradiente: La norma del gradiente della funzione costo deve essere limitata da una costante di Lipschitz. Cioè , esiste una costante L ¿ 0 tale che per ogni vettore di pesi x ed y, si abbia$||\nabla C(x) - \nabla C(y)|| \leq L||x - y||$, dove $\nabla C(x)$ rappresenta il gradiente della funzione obiettivo f in x. Questa condizione impone che il gradiente non cresca troppo velocemente, e che la funzione costo sia sufficientemente regolare.
- Step-size (o learning rate) : Il passo di aggiornamento n deve essere scelto in base alla costante di Lipschitz L. In particolare, per garantire la convergenza, il passo n deve essere minore o uguale a 1/L. Questa condizione assicura che il passo non sia troppo grande rispetto alla crescita del gradiente, limitando così la possibilità di oscillazioni e garantendo la convergenza del metodo.

Capitolo 20

Non convessità della funzione di costo

Introducendo la non linearità nella rete (aggiungendo strati nascosti), la funzione costo diventa non convessa e compaiono i minimi locali

Capitolo 21

Importanza del learning rate nei metodi di discesa

21.1 Valori bassi di n

- possono far sì che sia necessario un numero elevato di passi prima che l'allenamento sia completato.
- possono far sì che i pesi rimangano bloccati in un minimo locale, che è sub-optimal se confrontato con il global minimum.

21.2 Valori alti di n

Valori alti di learning rate permettono di mitigare questi problemi passando oltre i minimi locali e raggiungendo il minimo in numero limitato di passi. i pesi possono però anche finire per superare il minimo target Queste osservazioni fanno intuire che la scelta del learning late è cruciale ed in pratica può richiedere molta sperimentazione per scoprire quale può essere il valore ottimale del learning rate.

21.3 Exact line search

La dimensione ottimale del learning rate può essere trovata risolvendo un problema di ottimizzazione unidimensionale

$$n^{(k)} = arg\ min_{n \geq 0} C(w^{(k)} + np^{(k)}) \tag{21.1}$$

dove $p^{(k)}$ è la direzione di discesa. Gli algoritmi di ottimizzazione unidimensionale per trovare la dimensione del passo ottimale sono genericamente chiamati exact line search

Capitolo 22

Armijo rule

la regola di Armijo consente di trovare uno step-size n appropriato che garantisca una riduzione significativa della funzione costo, garantendo al contempo un'adeguata convergenza dell'algoritmo di ottimizzazione,

La regola di Armijo, chiamata anche regola di Armijo-Goldstein, è un criterio utilizzato nell'ottimizzazione numerica per determinare la dimensione del passo o lo step size da utilizzare durante la ricerca lineare in direzione di discesa all'interno di un algoritmo di ottimizzazione.

La regola di Armijo si basa sul concetto di "backtracking", che prevede di ridurre gradualmente la dimensione del passo fino a trovare un valore che soddisfi determinate condizioni. In particolare, la regola di Armijo si concentra sulla scelta di un passo che garantisca una riduzione sufficiente del valore della funzione costo durante la ricerca lineare.

Capitolo 23

Gradient Descent con momento

Il Gradient Descent con momento è stato studiato per risolvere i problemi di Armijo rule in particolare rende la scelta del learning rate meno cruciale. Si definisce la velocità $v_k = \beta v_k + \nabla C(w_k)$ β prende il nome di momento e può variare tra 0 ed 1. v_0 viene inizializzato a zero. La formula di aggiornamento dei pesi diventa:

$$w_{k+1} = w_k - nv_k \tag{23.1}$$

Capitolo 24

Iperparametri

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 configurazione del modello.

Capitolo 25

learning rate scheduling

25.1 Step decay

Riduce il tasso di apprendimento iniziale n_0 di un fattore δ ogni numero predefinito di epoche s $n = n_0 * \delta^{\frac{n}{s}}$ dove n_0, δ, s sono iperparametri

25.2 decadimento esponenziale

Il decadimento esponenziale ha la forma matematica $n = n_0 * e^{-kt}$ dove n_0 e k sono iperparametri e t è il numero di iterazione corrente

25.3 decadimento basato sul tempo

divide il learning rate iniziale n_0 in funzione del numero di iterazioni eseguite (t) $n = \frac{n_0}{1+kt}$ dove n_0 e k sono iperparametri

Capitolo 26

Learning rate adattivo

26.1 Adagrad

Adagrad adatta il Learning Rate ai parametri, eseguendo aggiornamenti più grandi per i parametri poco frequenti e aggiornamenti più piccoli per quelli frequenti.

Regola il tasso di apprendimento per i parametri in proporzione alla loro cronologia di aggiornamenti, più aggiornamenti più decadimento

$$s_k = s_{k-1} + (\nabla C(w_k))^2 \tag{26.1}$$

$$w_{k+1} = w_k - \frac{\eta}{\sqrt{s_k + \xi}} \nabla C(w_k) \tag{26.2}$$

dove $\xi > 0$ è un iperparametro scalare con valore molto piccolo (in genere $1e-7$) che serve per evitare che il denominatore si annulli.

Funziona bene in presenza di sparsi gradienti o feature sparse. Tuttavia, può diminuire troppo velocemente il tasso di apprendimento nel corso dell'addestramento, rendendolo inefficiente nelle fasi successive.

26.2 RMSProp

RMSProp è stato introdotto per ridurre la diminuzione aggressiva del learning rate di Adagrad

Modifica la parte di accumulo del gradiente di Adagrad con una media ponderata esponenziale dei gradienti al quadrato invece della somma dei gradienti al quadrato

$$s_k = \gamma s_{k-1} + (1-\gamma)(\nabla C(w_k))^2 \tag{26.3}$$

$$w_{k+1=w_k} - \frac{\eta}{\sqrt{s_k + \xi}} \nabla C(w_k) \tag{26.4}$$

Dove $\xi > 0$ è un iperparametro scalare con valore molto piccolo (in genere $1e-7$) che serve per evitare che il denominatore si annulli.
È particolarmente utile quando i parametri del modello hanno diverse scale di aggiornamento. Mentre $\gamma > 0$ 0 è un iperparmetro, il cui valore suggerito dagli autori è $\gamma = 0.9$

26.3 Adadelta

Adadelta è un'altra variante di Adagrad proposta per superare il suo principale inconveniente.
Come RMSProp calcola l'accumulo del gradiente come una media ponderata esponenziale dei gradienti al

quadrato ma, a differenza di RMSProp, non richiede di impostare un learning rate in quanto utilizza la quantità di cambiamento stessa come calibrazione per il cambiamento futuro

$$s_k = \gamma s_{k-1} + (1-\gamma)(\nabla C(w_k))^2 \tag{26.5}$$

$$\breve{\nabla} C = \frac{\sqrt{\triangle w_{k-1} + \xi}}{\sqrt{s_t + \xi}} \nabla C(w_k) \tag{26.6}$$

$$w_{k+1} = w_k - \breve{\nabla} C \tag{26.7}$$

$$\triangle w_k = \gamma \triangle w_{k-1} + (1-\gamma)(\breve{\nabla} C)^2 \tag{26.8}$$

26.4 Adam

L'obiettivo principale di ADAM (Adaptive Moment Estimation) è quello di combinare i vantaggi di due altri algoritmi di ottimizzazione: RMSprop e Momento.

Utilizza la media pesata esponenziale dei gradienti ai passi precedenti, per ottenere una stima del momento del gradiente

$$v_k = \beta_1 v_{k-1} + (1-\beta_1)\nabla C(w_k) \tag{26.9}$$

e del momento secondo del gradiente

$$s_k = \beta_2 s_{k-1} + (1-\beta_2)(\nabla C(w_k))^2 \tag{26.10}$$

dove β_1, β_2 iperparametri positivi ed gli autori consigliano di scegliere $\beta_1 = 0.9, \beta_2 = 0.999$,

Se si inizializza $v_0 = 0, s_0 = 0$, i momenti iniziali "sbilanciati" all'inizio del processo di apprendimento, per compensare questo sbilanciamento si usano le seguenti normalizzazioni:

$$\hat{v}_k = \frac{v_k}{1-(\beta_1)^k} \tag{26.11}$$

$$\hat{s}_k = \frac{s_k}{1-(\beta_2)^k} \tag{26.12}$$

e l'equazione di aggiornamento diventa:

$$w_{k+1} = w_k - -\frac{\eta}{\sqrt{\hat{s}_k + \xi}} \hat{v}_k \tag{26.13}$$