Appunti dai Power Pointe del corso di Intelligenza Artificiale

A.A. 2023/2024

Rosso Carlo

Indice

1 Introduzione

Che cos'è l'intelligenza? La capacità di un agenti di affrontare e risolvere con successo situazioni e problemi nuovi o sconosciuti.
I libri di testo classici definiscono l'IA come lo studio di agenti intelligenti, che percepiscono il loro ambiente e producono azioni volte a massimizzare la probabilità di successo nel raggiungere i loro scopi.

Intelligenza Artificiale stretta si riferisce a qualsiasi intelligenza artificiale in grado di equagliare o superare un essere umano in un compito strettamente definito e strutturato.

Intelligenza Artificiale generale dovrebbe consentire alel macchine di applicare conoscenze e abilità in diversi contesti anche di tipo nuovo. Si tratta di un obiettivo non ancora realizzato e non è detto che sia possibile.

Lo scopo dell'IA può essere definito come quello di costruire "agenti intelligenti". In particolare, l'IA studia come riprodurre in un computer i processi mentali complessi. Questo conduce a due diverse prospettive:

- construire dispositivi più intelligenti, che si avvicinano e superano l'intelligenza umana (prospettiva ingegneristica);
- costruire e testare ipotesi specifiche sui meccanismi all'interno della scatola (cervello), in modo di avvicinarsi il più possibile al comportamento umano anche quando questo non è ottimale (prospettiva delle scienze congnitive).

2 Fondamenti matematici

La modellazione computazionale trae vantaggio dall'utilizzo di strumenti di ricerca precisi, sistematici e, per quanto possibile, chiaramente definiti. In particolare, un avanzamento più rapido della conoscenza scientifica è solitamente supportato dall'adozione di un linguaggio formale (matematico), dove i concetti sono definiti nel modo più esplicito e preciso possibile.
L'idea fondamentale alla base degli algoritmi di apprendimento automatico (machine learning) è di fare in modo che l'algoritmo stesso possa modificare la propria struttura con l'esperienza.

Per esempio, se l'output dell'algoritmo dipende da alcune variabili memorizzate, possiamo programmare l'algoritmo affinché modifichi autonomamente tali variabili a seconda degli input che gli vengono mostrati. In altre parole, il comportamento dell'algoritmo dipende dal valore di alcuni suoi stati interni, e l'algoritmo ha la capacità di modificare in modo autonomo tali stati in base al tipo di esperienza che riceve.
Nel caso di una rete neurale artificiale, il sistema modifica la forza delle connessioni sinaptiche in base alle informazioni rilevate in un particolare insieme di esempi di addestramento (training set).

3 Computazione Neurale

La computazione neurale consiste in un nuovo paradigma di elaborazione dell'informazione, ovvero viene contrapposto alla computazione digitale. La computazione neurale viene eseguita da reti di neuroni biologici o artificiali.
Le reti biologiche sono formate da neuroni che si influenzano attraverso le connessioni (sinapsi) che li collegano:

- ogni neurone rileva un certo insieme di condizioni e segnala ciò che ha rilevato attraverso la sua frequenza di scarica;

- i neuroni possono ricevere segnali da altri neuroni e formare strati di rilevatori più complessi, il cui segnale indica la presenza di qualche struttura più elaborata (letteralmente come somma di segnali più semplici);

- le interazioni tra neuroni sono adattive e si modificano attraverso l'apprendimento.

Ciascun neurone è sintonizzato su uno stimolo preferito, specifico. Per cui se nell'input c'è lo stimolo ricercato dal neurone il neurone emette un segnale molto forte, altrimenti emette un segnale debole. Il campo recettivo del neurone è quella porzione di spazio sensoriale che attiva la risposta del neurone.
Dunque, la codifica dell'informazione diventa più complessa ed astratta passando a livelli più profondi della gerarchia. Infatti ci negli stratti più vicini all'input sono codificate caratteristiche molto semplici, queste sono raggruppate negli strati successivi in maniera sempre più complessa fino ad arrivare a caratteristiche molto complesse.

Le reti neurali artificiali sono sistemi di elaborazione in grado di apprendere compiti complessi e ispirati al funzionamento dei sistemi biologici.
Una rete neurale artificiale ha un'architettura ad elaborazione distribuita e parallela; questo vuol dire che "sullo stesso livello" (layer letteralmente) possono essere presenti più neuroni che lavorano in parallelo. Per esempio, data un'immagine, essa può contenere un quadrato ed una stella, e ci possiamo immaginare che ci siano due neuroni sullo stesso livello che si attivano nello stesso momento e che rilevano la presenza di un quadrato e di una stella. In generale, possiamo assumere che neuroni sullo stesso livello codifichino (segnalino la presenza di) caratteristiche ugualmente complesse. Le reti neurali sono caratterizzate da semplici unità di elaborazione (eseguono solo l'addizione e la moltiplicazione), con un'elevata interconnessione tra le unità. I messaggi che viaggiano tra le unità sono semplici numeri reali (i numeri che studiamo alle superiori, quelli con la virgola per intenderci); infine, le reti neurali possono evolvere nel tempo (ovvero possono imparare) e lo fanno modificando i pesi delle connessioni tra le unità.
Nell'ambito delle scienze cognitive, le reti neurali artificiali sono utilizzate con lo scopo di "ricostruire" le abilità cognitive e il comportamento umano. Infatti per spiegare come funziona una funzione cognitiva, viene costruita una simulazione adeguata, utilizzando un substrato di elaborazione che somigli quanto più possibile al substrato biologico.

La rete neurale si compone di:

- neuroni: sono dei rilevatori, per cui la loro attivazione segnla che hanno rilevato qualche cosa;
- reti neurali: connettono, coordinano, amplificano e selezionano pattern di attivazioni su neuroni;
- apprendimento: organizza le reti per eseguire compiti e per sviluppare modelli interni dell'ambiente. In particolare, viene attuato modificando la rete naurale (in base ai pattern in input).

3.1 Neurone

Un neurone formale (unità di elaborazione o nodo) è un modello matematico che cerca di catturare gli aspetti fondamentali del funzionamento neuronale:

- I neuroni hanno connessioni in ingresso e in uscita con altri neuroni;
- Ogni connessione ha un peso che indica la forza della relazione e può avere valore positivo (eccitatorio) o negativo (inibitorio);
- L'input di un neurone da un altro neurone si calcola come moltiplicazione del segnale di output del neurone in ingresso per il peso della connessione;
- L'input totale di un neurone è la somma di tutti gli input calcolati al passo precedente;
- Lo stato di attivazione finale viene calcolato come funzione di attivazione o di output del neurone (ovvero si prende la somma precedentemente definita e la si passa ad una funzione che restituisce un valore, che è il valore di attivazione del neurone);
- L'output del neurone (il suo valore di attivazione) viene passato a tutti i neuroni a cui è connesso (in uscita e pesando il segnale con la connessione, ovvero moltiplicazione al punto 3).

3.2 Reti di neuroni

L'architettura della rete identifica:

- l'organizzazione in gruppo o strati dei neuroni;
  - le unità che ricevono input dirrettamente dall'ambiente formano lo strato di input;
  - le unità che producono l'output finale della rete formano lo strato di output;
  - gli strati intermedi sono detti strati nascosti.
- il modo in cuii neuroni sono collegati tra di loro;
  - reti feed-forward: ci sono solo connessioni unidirezionali da unità di input a unità nascoste a unità di output;
  - reti ricorrenti: ci sono connessioni bidirezionali, per cui l'attivazione di uno strato può attivare uno strato precedente;
  - reti interamente ricorrenti: come le reti ricorrenti, in più ci sono connessioni anche tra unità dello stesso strato.

3.3 Differenze tra reti neurali biologiche e artificiali

Le reti neurali artificiali catturano i principi base di funzionamento delle reti neurali biologiche, ma non sono copie fedeli (perché non siamo in grado di costruire una copia fedele).
Cosa manca al neurone artificiale:

- organizzazione spaziale dei contatti sinaptici: le connessioni tra neuroni biologici sono molto più complesse di quelle tra neuroni artificiali; inoltre, nel cervello ci sono gruppi di neuroni che lavorano insieme e poi ci sono dei neruoni che collegano questi gruppi, ma non abbiamo ancora compreso questo meccanismo;

- differenziazione tra neuroni eccitatori e inibitori: i neuroni artificiali sono tutti uguali, mentre i neuroni biologici possono essere eccitatori o inibitori;

- tipi diversi di sinapsi: nelle reti artificiali abbiamo solo connessioni con peso positivo o negativo, mentre nel cervello ci sono sinapsi con diversi meccanismi di trasmissione;

- ?: qualcos'altro che non sono in grado di spiegare;

Cosa manca alla rete artificiale:

- struttura laminare e organizzazioen colonnare: le reti neurali artificiali sono bidimensionali, mentre il cervello è tridimensionale;

- organizzazione in mappe topografiche: la rete non si divide in gruppi per rappresentare informazioni diverse o capacità diverse;

- differenza di scala: le reti neurali artificiali sono molto più piccole rispetto al cervello;

Infine, le differenze maggiori sono quelle di cui ancora non ci rendiamo conto, meglio simuliamo il cervello e più ci rendiamo conto di quante cose manchino per rappresentarlo fedelmente, questo elenco non è esaustivo.

4 Apprendimento e memoria nelle reti neurali

Il machine learning consiste nei metodi e negli algoritmi che permettono ad un software di apprendere dall'esperienza. Gli algoritmi di apprendimento automatico apprendono dai dati. La

qualità dell'apprendiemnto tipicamente migliora all'aumentare del numero di esempi disponibili per l'addestramento. Le reti neurali artificiali rappresentano una classe importante di algoritmi di apprendimento, ma non sono gli unici metodi di machine learning.
Il deep learning si riferisce alle reti neurali artificiali (con almeno due strati nascosti) e rappresenta lo stato dell'arte.
Il machine learning si divide in tre categorie a seconda dei dati forniti al modello:

- **Apprendimento supervisionato**: al modello sono presentati dei dati in input e i risultati attesi, lo scopo è di imparare a produrre l'output corretto dato un nuovo input;

- **Apprendimento non supervisionato**: al modello sono presentati solo dati in input, non gli viene detto che cosa cercare. Lo scopo è di costruire una rappresentazione dell'input scoprendo le proprietà più importanti e informative (ci saranno esempi di modelli di questo tipo);

- **Apprendimento per rinforzo**: il modello produce delle azioni a partire da un input. A seconda delle azioni che produce, il modello riceve dei rinforzi o delle punizioni (come carota e bastone). Lo scopo del modello è mangiare più carote possibili e evitare i bastoni nel lungo periodo. Formalmente il modello massimizza i rinforzi a lungo termine.

4.1 Apprendimento

L'apprendimento in una rete neurale consiste nel trovare l'insieme di pesi delle connessioni che permette alla rete di produrre la risposta appropriata ad un certo input.
La forma più semplice di apprendimento è la regola di Hebb: *se due neuroni collegati tra loro sono contemporaneamente attivi, l'efficacia sinaptica della connessione viene aumentata*. L'apprendimento hebbiano è biologicamente plausibile.
I contro della regola di Hebb sono:

- la regola di hebb può solo aumentare il peso delle connessioni;

- i valori dei pesi (le connessioni) possono cresce indefinitamente (all'infinito);

I valori dei pesi sinaptici iniziali sono impostati in modo casuale a valori piccoli. Durante l'allenamento sono presentati più volte gli stessi pattern di addestramento. L'apprendimento consiste nella modifica dei pesi, ovvero il calcolo di Δw_{ij}, rispetto a w_{ij}, dove w_{ij} è il peso della

connessione tra il neurone i e il neurone j.
Se l'aggiornamento dei pesi avviene dopo ogni pattern si chiama on-line learning. Se l'aggiornamento dei pesi avviene dopo ogni epoca si chiama batch learning.
per non stravolgere le conoscenze precedentemente apprese, viene utilizzata solo una frazione della modifica sinpatica calcolata. Il fattore molteplicativo η è chiamato tasso di apprendimento.

$$w_{ij}^t = w_{ij}^{t-1} + \eta \Delta w_{ij}^t$$

Dove t è l'epoca corrente, w_{ij}^t è il peso della connessione tra il neurone i e il neurone j all'epoca t, w_{ij}^{t-1} è il peso della connessione tra il neurone i e il neurone j all'epoca $t-1$, η è il tasso di apprendimento e Δw_{ij}^t è la modifica del peso della connessione tra il neurone i e il neurone j all'epoca t.

4.2 Memoria

La memoria in una rete neurale consiste nella capacità di memorizzare la corrispondenza tra un input e un output. La memoria di un modello è data dall'insieme dei pesi delle connessioni.
L'attivazione dei neuroni è un fenomeno temporaneo ed è specifico per lo stimolo presentato e si esaurisce con la sua scomparsa. Fondamentalmente, da uno stimolo in input corrisponde un'esito in output. Questo vuol dire che c'è un'associazione tra tutti gli stimoli possibili in input e ciascun esito in output. Queste associazioni sono memorizzate nei pesi delle connessioni.
Invece, se l'attivazione non cessa bruscamente, può influenzare l'elaborazione dello stimolo successivo (reti ricorrenti). Alcuni compiti cognitivi richiedono di ricordare per breve tempo informazioni che non sono più presenti utilizzanod la memoria di lavoro o a breve termine.

Esempio del fallimento della regola di Hebb Se una rete neurale viene sottoposta al compito di apprendimento associativo AB-AC si ottiene un effetto di interferenza ancora maggiore che negli esseri umani. Ovvero viene rinforzata la connessione AB e la connessione AC, dunque ogni volta che si riceve A in input, la risposta è arbitraria tra B e C, perché entrambe le connessioni AB e AC sono state rafforzate.
Due fattori determinano l'interferenza in una rete neurale: il grado di sovrapposizione delle rappresentazioni, ovvero quanto sono simili B e C e la correlazione tra i pattern di input; e il tas-

so di apprendimento elevato. Biologicamente, il cervello umano ha sviluppato due sistemi di apprendimento separati e complementari:

- **Ippocampo**: specializzato nell'apprendimento rapido e non soggetto a interferenze, con rappresentazioni sparse;
- **Neocorteccia**: apprende lentamente e integra gradualmente le esperienze estrendo le conoscenze generali sul mondo. In questo caso la rappresentazione è distribuita (sono coinvolti più neuroni e c'è maggiore precisione).

5 Apprendimento supervisionato

L'apprendimento supervisionato si divide in due categorie:

- **Classificazione**: associare ai dati di input una categoria (cane-gatto, a-b-c-...-z, ecc.);
- **Regressione**: associare ai dati di input un valore continuo (per esempio, prezzo di un'abitazione, date tutte le sue caratteristiche, ...).

La rete impara ad associare un dato in input ad uno specifico output, ovvero la risposta corretta, che deve essere fornita esplicitamente durante il training. Per questo motivo, tutti i pattern di input devono essere etichettati.
Algoritmi:

- **Percettrone**;
- **Regola Delta**;
- **Backpropagation**;

Applicazione:

- classificazione;
- regressione.

Apprendimento (overview): il training set è l'insieme dei pattern su cui è addestrato il modello. Per ogni pattern di input sono definiti:

- **Input (x)**: la configurazione dei valori di input, una qualche rappresentazione di un pattern di input;

- **Output (y)**: la risposta corretta, la label associata al pattern di input, deve avere una forma specifica;

- **Target (t)**: il valore di output desiderato, la risposta corrispondente al pattern di input. La soluzione che diamo al modello, che viene confrontata con l'output per vedere se la risposta data (output) è corretta oppure no.

Dunque l'apprendimento consiste nei seguenti passaggi:

1. viene presentato un pattern di input alla rete;
2. la rete produce un output sulla base dei parametri attuali (pesi sinaptici);
3. l'output prodotto viene confrontato con l'output desiderato (target);
4. sono modificati i parametri (pesi sinaptici) per ridurre l'errore;
5. si ripete il processo fino a che l'errore non è accettabile.

Testing: per verificare le prestazioni del modello, si utilizza un test set, che ha la stessa struttura del training set, ma non viene utilizzato per modificare i pesi sinaptici. Semplicemente, per ogni pattern del test set, si calcola l'output corrispondete e si confronta con l'output desiderato. Il rapporto tra il numero di risposte corrette e il numero totale di pattern è la misura di accuratezza del modello.

5.1 Percettrone

Si tratta del primo modello di rete neruale con pesi sinaptici modificabili da un algoritmo di apprendimento. La rete ha n input che codificano l'esempio presentato con valori x_i ed un singolo neurone di output che codifica la risposta in modo bipolare o binario. L'output della rete è calcolato come:

$$y = \begin{cases} 1 & \text{se } \sum_{i=1}^{n} w_i x_i - \theta \geq 0 \\ -1 & \text{altrimenti} \end{cases} \tag{1}$$

Per ogni esempio presentato, l'output y viene confrontato con la risposta desiderata (target) t che codifica la classe di appartenenza:

- se $y = t$, i pesi non vengono modificati;

- se $y \neq t$, i pesi vengono modificati con: $\Delta w_i = \eta t x_i$.

dove η è il tasso di apprendimento.

5.2 Associatore lineare

Un associatore di configurazioni è simile ad un percettrone, ma i neuroni di output utilizzano una funzione di attivazione continua, come la sigmoide. L'output dei neuroni quindi è continuo, questo permette di quantificare l'errore, perché il target vale o 0 o 1, mentre l'output è un valore continuo compreso tra 0 e 1.

5.3 Regola Delta

La regola Delta è applicabile quando le unità di output hanno una funzione di attivazione continua e differenziabile, come la sigmoide. Permette di descrivere la prestazione con una funzione che misura l'errore della rete (funzione di errore o di costo). Si basa sullo scarto quadratico medio tra le risposte desiderate e quelle prodotte dalla rete:

$$E_W = \frac{1}{2} \sum_{\mu} \sum_{i} (t_i^{\mu} - y_i^{\mu})^2$$

L'apprendimento consiste nel minimizzare la funzione di costo E, che dipende unicamente dal valore delle connessioni sinaptiche W (perché tutto quello che cambia sono proprio le connessioni sinaptiche, questa funzione indica in quale modo cambiare, letteralmente indica in quale modo cambiare i pesi sinaptici e quindi come imparare). Quindi si modificano i pesi nella direzione opposta al gradiente della funzione stessa (ovvero si minimizza l'errore):

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}$$

Approfondiamo il modo in cui una rete impara:

1. viene presentato un pattern di input alla rete;

2. viene calcolato l'output della rete;

3. viene calcolata la discrepanza tra l'output della rete e l'output desiderato;

4. sono modificati i pesi sinaptici per ridurre l'errore calcolato al punto precedente;

5. si ripete il processo fino a che l'errore non è accettabile.

I problemi di inserparabilità lineare (AB e AC) possono essere risolti da reti neurali multistrato, ovvero con uno o più strati di neuroni nascosti che utilizzano una funzione di attivazione non-lineare.
La rete multistrato è un approssimatore universale, ovvero una rete neurale con almeno uno strato nascosto composto da un numero appropriato di neuroni con funzione di attivazione non-lineare può, in linea di principio, approssimare qualunque funzione $X \rightarrow Y$; cioè può abbinare opportunamente qualunque input all'output appropriato.

5.4 Backpropagation

L'algoritmo noto come backpropagation è un'estensione della regola Delta che permette di allenare reti multistrato; quindi, può essere applicato a qualunque modello di rete neurale con uno o più strati nascosti e con qualunque tipo di connettività (feedforward o ricorrente).
In effetti, la backpropagation permette di calcolare l'errore per le unità nascoste, ovvero indica come modificare i pesi sinaptici, non solo per le unità di output, ma anche per tutte le connessioni precedenti.
Da un punto di vista biologico, la backpropagation non è plausibile, perché richiede che l'errore sia calcolato e propagato all'indietro attraverso la rete, cosa che non avviene nel cervello.
Fondamentalmente utilizzando la regola delta si calcola l'errore per le unità di output. Utilizzando il peso delle connessioni sinaptiche l'errore è calcolato per ciascuna unità al livello precedente, sono modificati i pesi sinaptici e si ripete il processo fino a che non si arriva allo strato di input.

5.5 Tasso di apprendimento

Il tasso di apprendimento è un parametro che regola la velocità con cui i pesi sinaptici vengono modificati durante l'apprendimento. A tassi di apprendimento alti, l'apprendimento diventa più veloce, ma impreciso; mentre a tassi di apprendimento piccoli, l'apprendimento è più lento, ma più preciso. Infine, a tassi di apprendimento elevati, l'algoritmo di apprendimento potrebbe divergere (non covergere, e quindi non arrivare mai ad una soluzione). Il tasso di apprendimento può anche essere variabile, ovvero può cambiare durante il processo di apprendimento.

5.6 Momentum

Il momento aggiunge all'aggiornamento del peso sinaptico una frazione del precedente cambiamento di valore. Quando il gradiente dell'errore ha la stessa direzione, il momento aumenta la grandezza del passo che viene fatto.
Ci possiamo immaginare i pesi sinaptici attuali come una palla che rotola su una superficie. Il momento è la quantità di moto della palla, che aumenta se la palla va in discesa e diminuisce se la palla va in salita (ovvero la palla prende velocità, il momento è ciò che permette alla pallina di prendere velocità).

5.7 Generalizzazione

La generalizzazione è la capatcità di utilizzare in modo appropriato la conoscenza sul dominio quando si incontrano nuovi esempi del problema.
Ovvero è la produzione di una risposta appropriata a pattern di input non utilizzati per l'addestramento, ovvero un test set indipendente dal training set.
Alla generalizzazione si contrappone l'overfitting: si verifica quando il modello continua a migliorare le prestazioni sul training set, ma peggiora la prestazione in termini di generalizzazione (ovvero le predizioni sul test set sono peggiori del modello precedente).
Come evitare l'overfitting?

- **limitare il numero di unità nascoste**: se il modello è troppo complesso, la rete può memorizzare i pattern di input, piuttosto che generalizzare;
- **Early stopping**: utilizzare un validation set, quando l'errore sul validation set inizia a crescere, si ferma l'addestramento;
- **Decadimento dei pesi**: aggiungere un termine di decadimento ai pesi sinaptici, che li riduce ad ogni iterazione. In questo modo pesi inutili vengono ridotti a zero.

In generale, i metodi per prevenire eccessivo overfitting sono chiamati metodi di regolarizzazione.

5.8 Dataset

Training set : insieme di esempi (pattern) per l'addestramento. Sono utilizzati dall'algoritmo di apprendimento per trovare i valori dei pesi delle connessioni.

Validation set : insieme di esempi utilizzati per ottimizzare apprendimento (e.g. learning rate, momentum, weight decay, ecc.), il numero di unità nascoste, ecc. e per decidere quando fermare l'addestramento (early stopping).

Test set : insieme di esempi utilizzati per valutare le prestazioni finali del modello. Non è utilizzato per l'addestramento, ma solo per valutare le prestazioni quando il modello è pronto.

Perchè utilizzare set diversi per validation e test? Il set di validazione viene utilizzato per selezionare il modello migliore, quindi l'accuratezza sul validation set ha un bias (è sovrastimata).

Cross-validation

Se i dati non sono sufficienti per ripartirli in due insiemi separati di training e test, è essenziale massimizzare il numero di esempi di training utilizzando la tecnica di cross-validation. La k-fold cross-validation consiste nella divisione in k parti uguali del dataset totale. Ad ogni ciclo di cross-validation, la k-esima parte del dataset viene esclusa dal training per essere utilizzata come test.

Valutazione delle prestazioni

La valutazione delle performance di un modello va fatta sul test set. Esistono diverse metriche di performance e tendenzialmente sono specifiche per dominio. La distinzione principale tra metriche di performance è relativa a compiti di regressione e classificazione: un compito di regressione prevede la predizione di un valore continuo, quindi le prestazioni sono misurate in termini di distanza tra l'output del modello e quello desiderato; un compito di classificazione implica output binari, ovvero risposta corretta oppure no. D'altra parte una valutazione in termini di accuratezza non è sufficiente per stabilire la qualità di un classificatore, ma va considerata la matrice di confuzione.

Curva ROC la curva ROC (Receiver Operating Characteristic) per un classificatore binario è una curva che mostra la relazione tra il tasso di veri positivi e il tasso di falsi positivi al variare della soglia di classificazione (di due classi in genere).

Curva AUC l'area sotto la curva ROC (AUC, Area Under the Curve) è una misura della bontà di un classificatore binario, anche in questo caso il valore è compreso tra 0 e 1. In particolare, AUC è invariante rispetto alla soglia di classificazione.

6 Deep Learning Supervisionato

Per risolvere il problema della linearità, ovvero per riuscire a classificare correttamente input che non sono linearmente separabili, ci è venuto in mente di adottare funzioni non lineari. In particolare, abbiamo pensanto ottenere separabilità lineare attraverso proiezioni non lineari dello spazio degli input. Mettendo più livello uno sopra l'altro, si ottiene una rete neurale profonda in grado di approssimare funzioni complesse.
L'elaborazione gerarchica (profonda) è una caratteristica fondamentale della computazione neurale nel cervello. La codifica dell'informazione diventa più complessa ed astratta passando a livelli più elevati (profondi) della gerarchia.
Nel deep learning la rete neurale ha più di uno strato nascosto, quindi si forma una rete profonda. Per poter allenare una rete profonda sono necessari molti pattern di addestramento (già etichettati) e un'enorme potenza di calcolo. Una caratteristica interessante dei modelli che sfruttano il deep learning è che è in grado di imparare da dati grezzi, anche rumorosi, senza bisogno di pre-elaborazione.
Se aggiungiamo molti strati nascosti, il segnale di errore deve passare molti livelli di backpropagation. Con una funzione del tipo sigmoide, la derivata tende a 0 su una rete satura, ovvero quando il valore della sinapsi tende a 0 o a 1. Questo porta a problemi di vanishing gradient, ovvero il gradiente diventa troppo piccolo e la rete non riesce più ad apprendere.
Il problema del vanishing gradient viene arginato con alcune tecniche:

- inizializzazione dei pesi furba (per esempio, nell'intorno dello 0, dove la derivata della sigmoide è massima);

- learning rate adattivo, piuttosto che costante;

- rete di grandi dimensioni insieme a tecniche di regolarizzazione avanzata, come weight decay, dropout, etc.;

- ottimizzatori del secondo ordine, viene stimata la curvatura del gradiente e si aggiusta il learning rate di conseguenza;

- funzione di attivazione ReLU (al posto di sigmoide), che non satura il gradiente.

I metodi sempre adottati per migliorare le prestazioni di una rete neurale sono: l'adozione di dataset più grandi, l'utilizzo di hardware più potente e l'adozione di architetture convoluzionali (per esempio, per immagini).

7 Convolutional Neural Networks

La CNN è una rete profonda che include almeno uno strato convoluzionale, in cui i neuroni nascosti non sono interamente connessi con lo strato precedente, ma hanno campi recettivi locali. Lo strato convoluzionale, di solito, è seguito da uno strato di pooling per ridurre la dimensionalità ed enfatizzare le caratteristiche più interessanti. Nella parte più profonda della rete è inserito almeno uno strato nascosto standard, interamente connesso con quello precedente. L'ultimo strato nascosto è interamente connesso con lo strato di output.
Ogni neurone nascosto dello strato convoluzionale ha un campo recettivo locale, che codifica una feature specifica, per questo motivo i neuroni in questo strato sono anche chimati filtri (o kernel). Il numero di filtri definisce quante feature sono rappresentate in ciascun livello (proprio perché ogni filtro codifica una feature). Ciascun filtro è applicato all'intera immagine attraverso un'operazione di convoluzione. Fondamentalmente, i filtri sono applicati a una porzione piccola dell'immagine, scorrendo su tutta l'immagine, è come se venisse passato l'input poco per volta.

Iperparametri di una CNN:

- **Numero di neuroni nascosti**: specifica quanti filtri usare in ciascun layer;

- **Dimensione del kernel**: definisce il campo recettivo del filtro, ovvero la dimensione del filtro stesso, quindi la dimensione dell'input che il filtro considera alla volta;

- **Stride**: la dimensione del passo con cui il filtro scorre sull'immagine;

- **Padding**: aggiunge zeri attorno ai bordi dell'immagine, in modo che il filtro possa essere applicato anche ai bordi;

7.1 Pooling

Lo strato di pooling viene inserito dopo uno strato convoluzionale per ridurre la dimensione dell'immagine e quindi il numero di parametri.

7.2 Adversarial Examples

Possiamo indurre errori di classificazione in una rete neurale profonda facendo modifiche ad hoc all'immagine di input. Le modifiche non sono distinguibili all'occhio umano. Le modifiche sono studiate a partire dall'immagine di input in modo tale da massimizzare la funzione di errore e tendenzialmente modelli diversi sono ugualmente vulnerabili.

8 Reti ricorrenti

L'informazione in input può arrivare al sistema neurale in modo sequenziale e quindi può possedere una struttura temporale. Anche l'output può avere una struttura sequenziale.
In una classica rete multistrato l'output $O(t)$ al tempo t dipende solo dall'input corrente: $I(t)$.
Possibile soluzione: trasformare il tempo in spazio, ovvero viene fornito al modello alcuni elementi della sequenza S in una finestra temporale $W^T(t)$ che si sposta sopra S : $W^T(t) = S(t : t+T) = [S_t, S_t + 1, \dots S_{t+T}]$.
Questo metodo presenta alcuni problemi:

- i neuroni di input sono replicati per ogni elemento nella finestra, ovvero per ciascun S_t si hanno N neuroni di input che servono per rappresentare l'elemento S_t;

- la dimensione della finestra T è fissa e non si adatta alla lunghezza della sequenza;

Per ovviare a questi problemi si possono usare le **reti parzialmente ricorrenti**, in questo modo gli input alla rete sono il risultato di:

- l'input corrente $I(t)$

- l'output precedente $O(t-1)$

Notiamo che se $O(t)$ dipende da $O(t-1)$, allora $O(t-1)$ dipende da $O(t-2)$ e così via; ma allora $O(t)$ dipende da $O(t-1), O(t-2), \ldots, O(0)$, ovvero dall'intera sequenza di output precedente. Questa è letteralmente la definizione di ricorrenza.
Le reti parzialmente ricorrenti si possono addestrare con l'algoritmo backpropagation, perché la struttura temporale può essere "srotolata", trasformandola in una struttura spaziale, come se l'architettura fosse più o meno grande a seconda della lunghezza dell'input. I pesi della rete sono aggiornati sommando il cambiamento ad ogni passo temporale: $\Delta w_{ij} = \sum_t \Delta w_{ij}(t)$.

8.1 Reti di Elman (SRN - Simple Recurrent Network)

Le reti di Elman sono implementate aggiungendo un layer di neuroni nascosti ricorrenti su se stessi. Ovvero:

1. viene passato $I(t=0)$ al layer di input;
2. viene calcolato $H(t=0)$ a partire da $I(t=0)$ ed è passato al layer di output;
3. viene calcolato $O(t=0)$ a partire da $H(t=0)$;
4. viene passato $I(t+1)$ al layer di input;
5. viene calcolato $H(t+1)$ a partire da $I(t+1)$ e $H(t(=0))$ ed è passato al layer di output;
6. viene calcolato $O(t+1)$ a partire da $H(t+1)$;
7. e così via.

8.2 Self-supervised learning

Questo tipo di apprendimento viene chiamato self-supervised learning perché input e target hanno la stessa struttura (fondamentalmente gli input sono usati anche come target).

8.3 Long-Short Term Memory (LSTM, RNN - Recurrent Neural Network)

Le SRN appena descritte hanno una memoria a breve termine, non riescono ad apprendere dipendenze temporali lontane nella sequenza di input. Questo problema viene risolto nelle reti LSTM, in cui lo strato nascosto è formato da unità LSTM che hanno una struttura più complessa rispetto ai tipici neuroni nascosti. Le unità LSTM operano attraverso tre porte che gestiscono l'informazione di una feature temporale:

- **input gate**: regola l'input nella cella, ovvero se cambiare o meno il contenuto della cella;
- **output gate**: regola l'output della cella, ovvero se usare o meno il contenuto della cella;
- **forget gate**: regola se dimenticare o meno il contenuto della cella.

Overview del funzionamento delle unità LSTM:

- rete profonda con molteplici strati nascosti di unità LSTM;
- l'input è un elemento della sequenza S_t;
- l'output è la previsione del prossimo elemento della sequenza S_{t+1};
- predizione della sequenza: l'output al passo attuale diventa l'input al passo successivo, permettendo alla rete di predire la sequenza $[O_t, O_{t+1}, \ldots, O_{t+T}]$ a partire da S_t.

8.4 Word embeddings

La modellizzazione di sequenze per l'elaborazione del linguaggio naturale (NLP) è più efficiente al livello della parola, per cui bisogna codificare le singole parole come vettori di lunghezza fissa usando algoritmi di embedding.

Word embedding è un termine usato per la rappresentazione di parole in forma di un vettore numerico che codifica il significato della parola. Ci sono molti algoritmi di embedding, alcuni basati su reti neurali.
Gli embedding preservano le relazioni semantiche: parole vicine nello spazio hanno significato simile; inoltre, operazioni lineari sui vettori danno risultati coerenti.

8.5 Transformer

Un transformer è un'architettura neurale che si basa su meccanismi di self-attention per trasformare una sequenza di elementi in input in una sequenza di elementi in output, senza utilizzare convoluzioni o connessioni ricorrenti. I transformer sono divisi in due blocchi principali:

- **encoder**: prende una sequenza di input e genera una rappresentazione interna;

- **decoder**: prende la rappresentazione interna e genera una sequenza di output.

Entrambi i blocchi sono formati da più layer con connessioni solo feedforward (quindi non ricorrenti!).
Il meccanismo di self-attention permette di calcolare l'importanza di ciascun elemento della sequenza rispetto agli altri elementi. Per esempio, i tranformer sono molto usati per lavorare con il linguaggio naturale, il sistema di self-attention tende a scartare gli articoli e le preposizioni, che sono generalmente meno importanti per il significato di una frase.
I tranformer hanno il vantaggio di avere un'architettura altamente parallela, che permette di ridurre i tempi di addestramento. Inoltre sono molto forti nella gestione di lunghe sequenze, in quanto non hanno problemi di memoria a lungo termine come le RNN.

8.6 Large Language Models (LLM)

Gli LLM sono basati su architetture di transformer addestrati su enormi quantità di testo (chatGPT è un LLM). Fondamentalmente, sono transformer molto più grandi e ripetuti in serie. Gli LLM sono allenati con apprendimento self-supervised, ovvero senza target, ma cercando di prevedere la parola successiva in una sequenza di testo.
Successivamente vengono adattati a compiti specifici con apprendimento supervisionato (con target); questa seconda fase di addestramento è detta **fine-tuning**. Ogni tanto è complesso specificare un target per un LLM, per cui viene allenato per rinforzo, ovvero una persona dice sono se la risposta va bene oppure no; questo si contraddistingue all'allenamento supervisionato che è in grado di dire di quanto la risposta è sbagliata. L'allenamento per rinforzo è meno preciso, per questo ha bisogno di più dati.

9 Apprendimento non supervisionato

Pro:

- l'apprendimento non richiede nessuna etichetta negli esempi, quindi si possono sfruttare enormi quantità di informazioni grezze ("as is");
- **transfer learning**: si può allenare un modello di base in questo modo e poi specializzarlo per un compito specifico mediante l'allenamento supervisionato oppure per rinforzo. Per esempio GPT (Generative Pre-trained Transformer) è preallenato, mentra ChatGPT è un modello specializzato per il compito di chatbot;
- biologicamente plausibile: sembra che gli animali (anche l'uomo) sfruttino massivamente questa modalità di apprendimento durante lo sviluppo.

Contro:

- è difficile definire una buona rappresentazione dell'ambiente (che cosa è importante e che cosa no?);
- richiede un'enorme quantità di dati e di potenza di calcolo per ottenere risultati soddisfacenti;
- non è possibile inferire relazioni causali.

Ci sono due approcci principali all'apprendimento non supervisionato:

- **clustering**: si cerca di raggruppare gli esempi in modo che quelli dello stesso gruppo siano simili tra loro e diversi da quelli appartenenti ad altri gruppi. Questo approccio è utile per scoprire nuove categorie o per ridurre la complessità di un problema. In genere viene individuato un prototipo per ogni gruppo al livello più vicino all'output, mentre i livelli sottostanti elaborano l'input per evidenziare le differenze tra gli esempi e rendere l'input via via più simile al prototipo;
- **riduzione della dimensionalità**: si cerca di ridurre la dimensione dello spazio di input, mantenendo le informazioni più rilevanti. Questo approccio è utile per visualizzare i dati, per ridurre il rumore e per velocizzare l'addestramento dei modelli.

9.1 PCA (Principal Component Analysis)

La PCA è una tecnica statistica che cerca di trovare la direzione di massima variabilità in un certo insieme di dati; ovvero, la PCA scropre un insieme di variabili linearmente scorrelate, le componenti principali, che spieghino la maggior parte della varianza osservata nella distribuzione. La scomposizione dei dati della PCA permette di mappare i pattern di input in un nuovo sistema di coordinate a bassa dimensionalità. Naturalmente, minore è il numero di feature che si considerano, minore è la quantità di informazione che si riesce a catturare (peggiore sarà la ricostruzione dell'input originale).

9.2 Autoencoder

Gli autoencoder sono una famiglia di reti neurali che cercano di apprendere una rappresentazione compatta di un insieme di dati. Gli autoencoder sono composti da due parti: un **encoder** che mappa l'input in uno spazio di rappresentazione più compatto e un **decoder** che mappa la rappresentazione compatta in un'immagine ricostruita. Gli autoencoder sono addestrati per minimizzare la differenza tra l'input e l'output.
Utilizzando neuroni nascosti con funzione di attivazione non lineare, effetuano uan riduzione non lineare della dimensionalità, questo aumenta molto la capacità di compressione del modello, non solo a parità di parametri, anche la qualità dell'immagine ricostruita è migliore.
Allenando gli autoencoder con dati rumorosi, si può ottenere un modello che riesce a ricostruire l'input originale anche se questo è stato corrotto.
Gli autoencoders sono modelli generativi, ovvero possono generare nuovi esempi simili a quelli di training.

10 Apprendimento non supervisionato

Pro:

- l'apprendimento non richiede nessuna etichetta negli esempi, quindi si possono sfruttare enormi quantità di informazioni grezze ("as is");
- **transfer learning**: si può allenare un modello di base in questo modo e poi specializzarlo per un compito specifico mediante l'allenamento supervisionato oppure per rinforzo. Per

esempio GPT (Generative Pre-trained Transformer) è preallenato, mentra ChatGPT è un modello specializzato per il compito di chatbot;

- biologicamente plausibile: sembra che gli animali (anche l'uomo) sfruttino massivamente questa modalità di apprendimento durante lo sviluppo.

Contro:

- è difficile definire una buona rappresentazione dell'ambiente (che cosa è importante e che cosa no?);

- richiede un'enorme quantità di dati e di potenza di calcolo per ottenere risultati soddisfacenti;

- non è possibile inferire relazioni causali.

Ci sono due approcci principali all'apprendimento non supervisionato:

- **clustering**: si cerca di raggruppare gli esempi in modo che quelli dello stesso gruppo siano simili tra loro e diversi da quelli appartenenti ad altri gruppi. Questo approccio è utile per scoprire nuove categorie o per ridurre la complessità di un problema. In genere viene individuato un prototipo per ogni gruppo al livello più vicino all'output, mentre i livelli sottostanti elaborano l'input per evidenziare le differenze tra gli esempi e rendere l'input via via più simile al prototipo;

- **riduzione della dimensionalità**: si cerca di ridurre la dimensione dello spazio di input, mantenendo le informazioni più rilevanti. Questo approccio è utile per visualizzare i dati, per ridurre il rumore e per velocizzare l'addestramento dei modelli.

10.1 PCA (Principal Component Analysis)

La PCA è una tecnica statistica che cerca di trovare la direzione di massima variabilità in un certo insieme di dati; ovvero, la PCA scropre un insieme di variabili linearmente scorrelate, le componenti principali, che spieghino la maggior parte della varianza osservata nella distribuzione. La scomposizione dei dati della PCA permette di mappare i pattern di input in un nuovo sistema di coordinate a bassa dimensionalità. Naturalmente, minore è il numero di feature che si considerano, minore è la quantità di informazione che si riesce a catturare (peggiore sarà la ricostruzione dell'input originale).

10.2 Autoencoder

Gli autoencoder sono una famiglia di reti neurali che cercano di apprendere una rappresentazione compatta di un insieme di dati. Gli autoencoder sono composti da due parti: un **encoder** che mappa l'input in uno spazio di rappresentazione più compatto e un **decoder** che mappa la rappresentazione compatta in un'immagine ricostruita. Gli autoencoder sono addestrati per minimizzare la differenza tra l'input e l'output.
Utilizzando neuroni nascosti con funzione di attivazione non lineare, effetuano uan riduzione non lineare della dimensionalità, questo aumenta molto la capacità di compressione del modello, non solo a parità di parametri, anche la qualità dell'immagine ricostruita è migliore.
Allenando gli autoencoder con dati rumorosi, si può ottenere un modello che riesce a ricostruire l'input originale anche se questo è stato corrotto.
Gli autoencoders sono modelli generativi, ovvero possono generare nuovi esempi simili a quelli di training.

11 Dalla rebola di Hebb ai modelli generativi

11.1 Reti di Hopfield

Le reti di Hopfield possono essere utilizzate per memorizzare e recuperare pattern (e niente altro). La memorizzazione, come al solito, avviene modificando i pesi delle connessioni, in particolare, applicando la regola di Hebb. Il recupero avviene in modo dinamico, aggiornando iterativamente lo stato dei neuroni fino a che non si raggiunge uno stato di equilibrio (attrattore). Infatti, si definisce una funzione di energia che che indica in quale modo aggiornare lo stato dei neuroni, ai minimi locali corrisponde uno stato di equilibrio, per cui durante il recupero, la funzione di energia aggiorna lo stato dei neuroni per diminuire l'energia del sistema e raggiungere uno stato di equilibrio. Lo scopo dell'apprendimento è quello di modificare i pesi in modo tale che la funzione di energia abbia minimi locali corrispondenti ai pattern memorizzati. Quando si presenta un pattern parziale (in realtà per ogni input), la rete gradualmente assegna si assesta nella configurazione corrispondente al pattern memorizzato più simile.

Architettura : la rete è ricorrente, ovvero tutte le connessioni sono bidirezionali) con topologia completamente connessa, ovvero tutti i neuroni sono collegati tra loro. Infine, tutti i neuroni sono

visibili; cioè lo strato di input coincide con lo strato di output e non ci sono altri strati.

Funzione di energia : la funzione di energia è definita come:

$$E = -\frac{1}{2}\sum_{i,j} w_{ij}x_i x_j$$

Gli attrattori della rete corrispondono ai punti di minimo locale della funzione di energia, e rappresentano i pattern memorizzati.
Ci sono due metodi per minimizzare la funzione di energia:

- **Recupero di un pattern**: sono aggiornate le attivazioni dei neuroni una alla volta, in modo sequenziale, fino a che non si raggiunge uno stato di equilibrio (è aggiornato x_i o x_j);
- **Apprendimento**: sono aggiornati i pesi delle connessioni per creare minimi locali in corrispondenza dei pattern memorizzati (è aggiornato (w_{ij}).

Recupero dei pattern

I neuroni possono avere solo due stati possibili: -1 e 1 e la loro attivazione è calcolata con la regola:

$$x_i = \begin{cases} 1 & \text{se } \sum_j w_{ij}x_j > \theta_i \\ -1 & \text{altrimenti} \end{cases}$$

Apprendimento

Come sono modificati i pesi delle connessioni per memorizzare i pattern? Si utilizza la regola di Hebb, che in questo caso è:

$$\Delta w_{ij} = \eta x_i y_j$$

Ovvero, la variazione del peso tra due neuroni è proporzionale al prodotto dei segnali in input e in output. Interprentando la formula, il peso della connessione tra i due neuroni aumenta se i due neuroni hanno lo stesso segno (+ per +, - per -), e diminuisce se i due neuroni hanno segno opposto.
In questo modo, l'apprendimento scava il bacino degli attrattori, in modo che la funzione di energia abbia minimi locali corrispondenti ai pattern memorizzati. Allo stesso modo è aumentata

l'energia corrispondente a configurazioni improbabili (non presenti nei pattern).
Questo metodo può imparare solo un numero limitato di pattern in base al numero di neuroni della rete. La dinamica stocastica può aiutare a memorizzare più pattern, per cui la funzione di attivazione diventa:

$$P(x_i) = \frac{1}{1 + e^{-\frac{1}{T}\sum_j w_{ij}x_j}}$$

Fondamentalmente, al posto di usare la funzione a scalino si usa una sigmoide che da un valore di probabilità tra 0 e 1 per l'attivazione del neurone, infatti T è la temperatura e regola la pendenza della sigmoide e quindi a una temperatura elevata, la sigmoide è piatta e quindi la probabilità di attivazione è 0.5; mentre a una temperatura bassa, la sigmoide è molto ripida e quindi la probabilità di attivazione è 0 o 1.
L'idea è quella di diminuire la temperatura durante il recupero, in modo che la rete si assesti in uno stato di equilibrio evitando attratori spuri (fasulli).

11.2 Macchine di Boltzmann

Le macchine di Boltzmann sono una variante stocastica delle reti di Hopfield. che sfruttano unità nascoste per estrarre correlazioni di ordine superiore dei dati (astrazioni maggiori sui dati). Hanno una capacità di memorizzazione superiore, infatti i neuroni nascosti sono usati per comprimere l'informazione. Le macchine di Boltzmann sono molto computazionalmente molto costose.
Fondamentalmente, oltre alle unità visibili, sono presenti unità nascoste totalmente connesse tra loro e con le unità visibili.

11.3 Macchine di Boltzmann Ristrette

In questo caso la complessità computazionale viene enormemente ridotta. Ora ci sono due strati di neuroni: uno visibile e uno nascosto. Gli strati sono completamente connessi tra loro, ma non ci sono connessioni tra neuroni dello stesso strato.
La funzione di energia dipende dalle attivazioni dei neuroni visibili e nascosti e dai pesi delle connessioni.