Lezione 10 15/12/2023

Autoencoder

Quando si definisce un problema di apprendimento, questo se supervisionato avrà il target, mentre se non è supervisionato avremo solo l'input senza target.

L'autoencoder era particolare, perchè aveva un task "fake", ovvero veniva dato come output lo stesso input. Ho costruito un task supervisionato dove non era definito il target, perchè è uguale all'input. Vogliamo che la rete venga addestrata riducendo l'errore. Se l'errore nell'approccio supervisionato era che l'output prodotto è diverso dal target, qui invece l'errore è che la rete non sa ricostruire l'ingresso.

Il vantaggio è che se questa struttura di "finta rete in avanti", dove i neuroni in uscita sono tanti quanti quelli in ingresso, se applico la back propagation notiamo che avrò utilità se costruisco di modo da strozzare l'informazione nel collo di bottiglia al centro. Questa struttura a clessidra è quella che ci da valore. Il risultato prodotto dagli autoencoders è dovuto al collo di bottiglia.

Quando produco una codifica, voglio avere la possibilità di codificare (riduzione di dimensionalità) e decodificare. Questo lo otteniamo considerano la parte a sinistra della rete come quella che codifica la rete in un certo codice (stati dei neuroni) e poi verso destra opera come decodifica (da un valore compresso, costruisce l'input che sarà uguale all'output).

Quindi se chiedono "si applica la backpropagation all'autoencoder?" dobbiamo dire si, prendendo ogni singolo neurone in uscita come quello dove posso calcolare un errore che va a retropropagare.

Se chiedono "può avere qualunque struttura la rete autoencoder?" no, perchè va fatta con quel collo di bottiglia.

Se chiede "ci sono nell'autoencoders connessioni che saltano i layers?" no, perchè per forzare la compressione nel collo di bottiglia non voglio connessioni che lo scavalchino.

I neuroni definiscono il modo di fare inferenza in avanti, gli errori tramite backpropagation permettono di fare apprendimento.

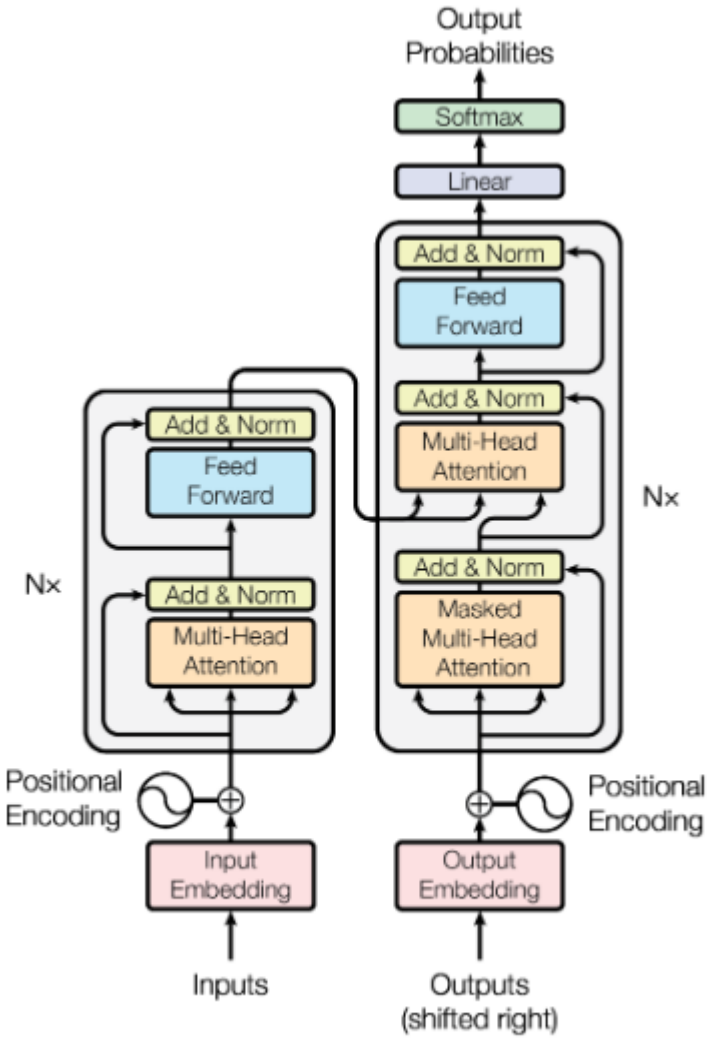
La T di chatgpt sta per "transformer". Nelle architetture odierne il passaggio grosso è stata la struttura basata su neuroni, ma molto più complicata. Nell'immagine i neuroni non sono visibili.

Feedforward è la classica struttura di neuroni connessi in avanti (senza collegamenti all'indietro).

L'input viene trasformato facendo una codifica, l'output produce una dopo l'altra produce una serie di parole probabili in uscita.

Gli output vengono presi e riportati in ingresso, shiftati a destra. Il fatto che un output vecchio ritorni come ingresso che va ad impattare sul calcolo della rete (ma sfasato, l'output vecchio impatta sull'input nuovo).

Il blocco arancione ripetuto "attention" è un elemento nuovo basato su neuroni, che è l'elemento forte delle architetture correnti.



Word2Vec è una macchina neurale capace di produrre codifiche dell'input. Si prendono delle parole, e per ognuna il sistema impara a produrre un vettore. (come

se fosse il bottleneck dell'autoencoder) le parole possono essere quelle del vocabolario, il vettore ha 300 elementi, non posso fare una rappresentazione facile. C'è un vettore che prendendo tutte le 300 componenti fa ??

Anche questo è un sistema di apprendimento non supervisionato. Anche a lui viene dato un "fake task". Per ogni parola che trova in un testo, il sistema deve sapere quali parole sono vicine. Per esempio se vedo "fox" troverò vicine sempre alcune parole tipo "brown" "jumps".

Non supervisionato, fake task, struttura neurale. Il risultato è che la rete ha imparato ad associare vettori a parole.

Questo non è deep learning, ma l'approccio di come portare i neuroni con la solita backpropagation ad imparare qualcosa di utile è nato con il deeplearning, che gioca con la struttura neurale. I vettori che la rete ha prodotto non sono a caso, la rete ha imparato ad associare vettori simili quando il risultato è simile.

Se il vettore ha 300 componenti non lo possiamo vedere in due dimensioni, ma esiste una libreria (TSNE) in cui questi vettori vengono trasformati in due dimensioni, conservando delle proprietà di tenerli uno vicino all'altro. L'immagine a due dimensioni è una proiezione dei vettori.

Sia autoencoder che word2vec producono embedding tramite ottimizzazione degli errori.

Dagli autoencoder si è passati ad una variante chiamata denoising. Il termine evidenzia il fatto che se l'autoencoder ha un collo di bottiglia, e quindi addestrato e seguire un ingresso produce una codifica, ora chiedo di più al sistema, gli chiedo di essere meno sensibile al rumore, anzi "denoising", produce una rimozione del rumore. Viene addestrato con dei dati "rotti", gli viene chiesto di costruire un output pulito, e funziona.

Attention e Transformers

L'attention permette alla macchina di dare maggior peso ad alcuni elementi dell'input. Nel wordtovec prende i vicini senza priorità tra di loro. Con il meccanismo di attention potrei dire che uno degli elementi vicini ha importanza più alta. Attention è un elemento usato nel transformer, che è l'architettura complessiva.

Non consideriamo la ricorrenza. Con i cicli ci sono dei problemi. il transformer con questo loop tra uscita e ingresso, ha preso un'approssimazione della ricorrenza gestibile. Evita la vera e propria ricorrenza e tutti i suoi problemi, permette di considerare elementi vecchi con il meccanismo di attensione, sfasando.

Nel diagramma di transformer vediamo una parte di encoding che poi va ad impattare con l'altra parte di encoding che arriva dall'output precedente, poi è stata ampliata questa possibilità aggiungendo anche decoding.

L'addestramento di una rete moderna viene fatta in due fasi: una è quella "non supervisionata", self training. Prende una grande quantità di dati (internet) e vogliamo avere una macchina simile al wordtovec, è un'apprendimento generico senza un task vero e proprio, ma forzata a ragionare su degli elementi che nascondo. è quindi un pretraining generico, impara a modellare le strutture delle parole, frasi. La seconda fase è chiamata fine-tuning che è supervisionata, posso dopo prendere la rete pre addestrata e specializzarla.

ChatGPT ha aggiunto la possibilità che un umano durante la fase di allenamento vada a trovare cose sbagliate. (reinforcement learning from human feedback) (reinforcement learning è quella dove la rete va penalizzata quando sbaglia)