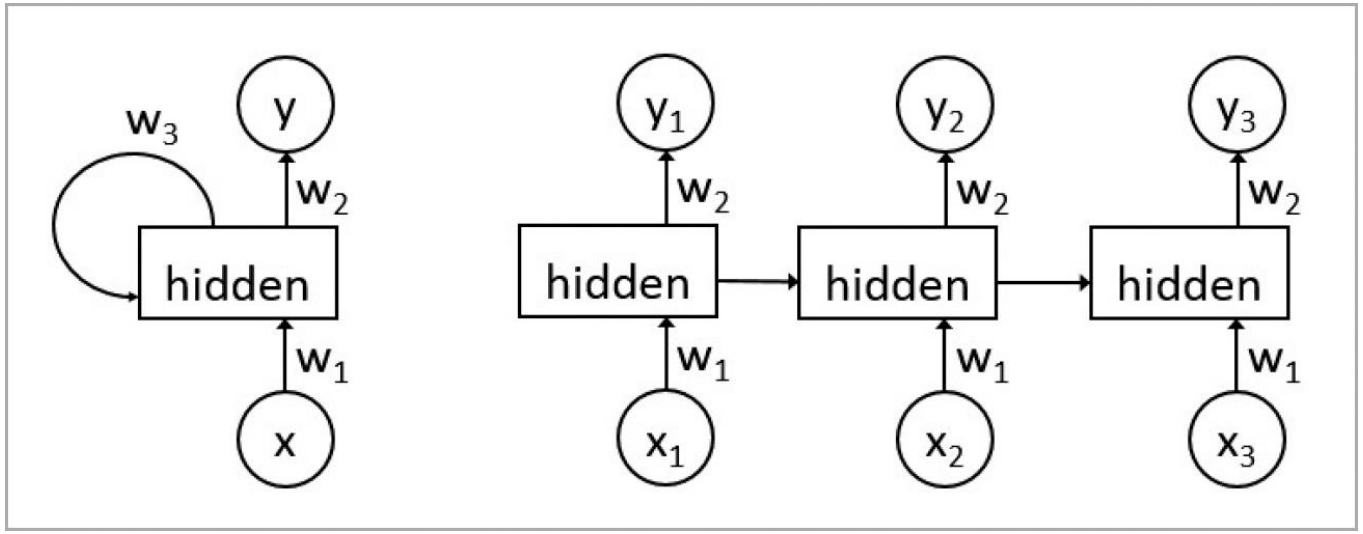
Recurrent Neural Network

A CURA DI LINO POLO

I modelli non ricorrenti con feed forward (propagazione in avanti) operano come un sistema combinatorio, ovvero associano pattern di uscita a pattern di ingresso secondo una relazione del tipo Y(t)=F(X(t)) con funzione di attivazione F.

Come idea di fondo, il modello Recurrent Neural Network (RNN, rete neurale ricorrente) cerca di riprodurre due meccanismi della memoria umana: memoria breve, per appoggio temporaneo in cui o si scorda tutto o il contenuto viene passato alla memoria a lungo termine, e memoria a lungo termine, per conservare le informazioni nel tempo.

Opera come un sistema sequenziale, ovvero associa pattern di uscita a pattern di ingresso, con una dipendenza da uno stato interno che evolve a sua volta nel tempo con gli ingressi presentati,



Architetture

Brain State in a Box, sviluppata da J.A. Anderson e altri nel 1977, completamente ricorrente, funzione di attivazione lineare e neuroni bipolari con valori in [-1,1], assume uno stato all'interno di ipercubo i cui vertici sono definiti dal limite di attivazione dei nodi, permette di recuperare l'informazione originale a partire da informazione con molto rumore;

Elman, creata da Jeffrey L. Elman nel 1990 per tenere conto dello stato precedente; si crea lo strato di contesto avente uscite distribuite in ingresso allo strato hidden con connessioni a maglia completa; le connessioni retroattive si propagano dallo strato intermedio a tali unità contestuali, alle quali si assegna peso costante e pari all'unità; in ciascun istante gli ingressi si propagano nel modo tradizionale e tipico delle reti feed forward, compresa l'applicazione dell'algoritmo di training; le connessioni retroattive fisse mantengono una copia dei precedenti valori dei neuroni intermedi, dal momento che tale flusso avviene sempre prima della fase di apprendimento;

Long Short Term Memory (LSTM): l'output immediatamente precedente potrebbe non essere sufficiente per prevedere ciò che sta per accadere; LSTM permette di fare affidamento sulle informazioni di un ulteriore output precedente. I dati ricorrenti passano attraverso ciò che viene definito Keep Gate o Forget Gate, che in sostanza decide cosa conservare e cosa rimuovere dai dati ricorrenti, quando arrivano i nuovi dati di input determina quali novità aggiungere per conservarli, infine decide il nuovo output.

RNN è utile per predire sequenze temporali, come l'esempio di LSTM nel marketing e viene molto usata nell'analisi del testo, come LSTM, per apprendere una sorta di linguaggio e stile di scrittura simile alla Divina Commedia.

Keras: cos’è?

Keras è una libreria open-source di alto livello per la creazione e l'addestramento di reti neurali in Python.

Fornisce un'interfaccia user-friendly, modulare e estensibile, rendendo più semplice per gli sviluppatori la costruzione di reti neurali complesse senza dover gestire dettagli complessi di basso livello.

Keras è progettato per essere facile da usare, veloce da prototipare e estremamente flessibile, consentendo agli sviluppatori di concentrarsi sulla progettazione della struttura della rete e sulla sperimentazione con diversi modelli di apprendimento profondo.

TensorFlow:

TensorFlow è una piattaforma open-source per il machine learning e l'apprendimento profondo sviluppata da Google.

Essa fornisce un'ampia gamma di strumenti, librerie e risorse per costruire e addestrare modelli di apprendimento automatico. TensorFlow utilizza un approccio di programmazione basato su grafi, in cui le operazioni sono rappresentate come nodi in un grafo e i dati fluiscono attraverso il grafo.

Questo facilita la distribuzione su hardware diversi, come CPU e GPU, nonché su dispositivi mobili. Keras è stato integrato in TensorFlow, rendendolo un alto livello API per la creazione di reti neurali all'interno dell'ecosistema TensorFlow. Ciò ha contribuito a rendere l'uso di TensorFlow più accessibile e amichevole per gli sviluppatori.

Differenza Keras e Tensorflow

Keras come API di alto livello:

Keras Indipendente: Keras originariamente era una libreria separata che forniva un'interfaccia di alto livello per la costruzione di reti neurali.

Integrazione in TensorFlow: Successivamente, Keras è stato integrato direttamente in TensorFlow, diventando l'API di alto livello consigliata per gli utenti di TensorFlow. Dal TensorFlow 2.0 in poi, Keras è il modulo di alto livello predefinito per la creazione di reti neurali all'interno di TensorFlow.

Differenza Keras e Tensorflow

TensorFlow come piattaforma di basso livello:

Piattaforma Completa: TensorFlow è una piattaforma di machine learning completa che comprende non solo Keras come interfaccia di alto livello, ma anche molte altre funzionalità di basso livello per la gestione diretta dei tensori e l'implementazione di algoritmi di apprendimento automatico personalizzati.

Flessibilità e Controllo: TensorFlow offre maggiore flessibilità e controllo per gli sviluppatori che desiderano personalizzare ogni aspetto della loro rete o implementare modelli di apprendimento automatico avanzati.

Differenza Keras e Tensorflow

Scelta in base alle Esigenze:

- **Semplicità con Keras:** Se il tuo obiettivo principale è costruire e addestrare reti neurali in modo rapido e semplice, Keras fornisce un'interfaccia intuitiva e user-friendly.
- **Controllo con TensorFlow:** Se hai esigenze più specifiche o desideri sperimentare con aspetti più avanzati del machine learning, TensorFlow offre un controllo più fine sui dettagli della tua implementazione.
- In breve, Keras all'interno di TensorFlow rappresenta un modo più semplice e ad alto livello per costruire reti neurali, mentre TensorFlow come piattaforma offre una gamma più ampia di funzionalità e maggiore controllo per gli sviluppatori esperti. La scelta tra i due dipende dalle esigenze specifiche del progetto e dalle preferenze dello sviluppatore.