Lezione 6 27/10/2023

Ora abbiamo più strati di più neuroni. Come possiamo cambiare i pesi degli strati interni (fare apprendimento)? Usiamo l'algoritmo di back propagation.

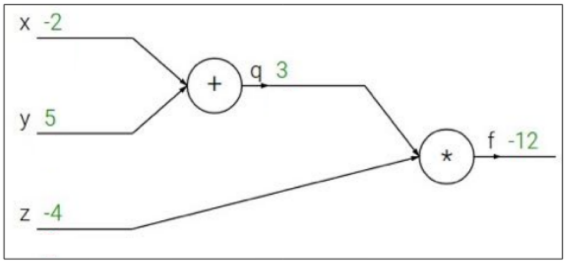
Back propagation

Delta rule: cambio pesi in proporzione al corrispondente cambiamento errore. Il cambiamento errore è ora diventato il gradiente, cioè la derivata su più dimensioni. (non chiederanno la matematica della back propagation)

Ora passiamo a più strati, con la derivata di una funzione composta. Ora l'errore non è più dato solo dall'input e dai pesi (derivata rispetto ai pesi, input costante), ma è la derivata dell'input calcolato dallo strato intermedio e portato in uscita.

Dovrò tornare indietro per calcolare le variazioni dei pesi di ciascuno strato. Il calcolo in avanti diventa una grossa moltiplicazione tra matrici, perchè i pesi di ogni strato creano l'output che diventa l'input dello strato successivo, quindi questo può essere visto e semplificato come una matrice. A noi interessa fare la discesa del gradiente che verrà approssimato e scomposto uno strato alla volta.

La struttura complessiva della rete può essere abbastanza articolata, per esempio alcuni input possono collegarsi al primo strato, altri a strati più interni.



Vengono usati questi diagrammi che rappresentano la struttura della rete neurale. L'immagine presenta un caso molto semplice che non rappresenta bene una rete neurale vera visto che fa solo somme e moltiplicazioni.

Nell'**algoritmo** abbiamo:

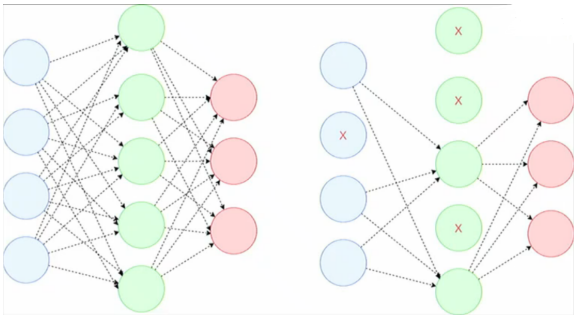
- una inizializzazione con vettori peso casuali
- il passaggio forward, ovvero prendiamo un esempio in ingresso che ci permette di calcolare l'uscita
- dall'uscita, andando all'indietro, calcolo il gradiente approssimato dell'errore

- aggiornamento dei pesi (apprendimento)
- ripeti con gli altri esempi

La rappresentazione sarà con “**computational graph**”.

La discesa stocastica del gradiente viene fatta tramite “mini-batch”. ovvero uso un numero ristretto di istanze alla volta (quante ce ne stanno nella memoria della gpu). Quindi non è ne singola istanza ne batch completo.

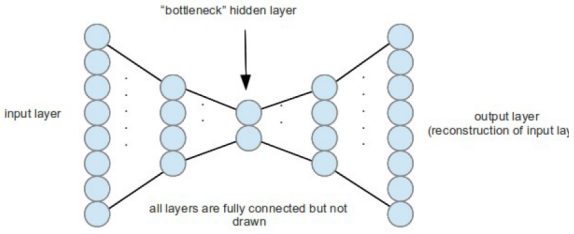
Dropout



Più la rete è ricca di neuroni, più riesce a “fotografare” le istanze. C’è quindi un rischio di **overfitting** nella discesa del gradiente. Limito l’overfitting penalizzando la rete **durante il training** (non parto penalizzato). Quindi durante l’addestramento cerco di ostacolare la memorizzazione, e costringere la generalizzazione. In generale sono tecniche chiamate “regolarizzazione”. In particolare citiamo quella chiamata **dropout**.

Da una rete ricca, passo ad una rete dove rimuovo dei neuroni che vengono rimossi con una certa probabilità. Eliminando un neurone scompaiano anche i collegamenti.

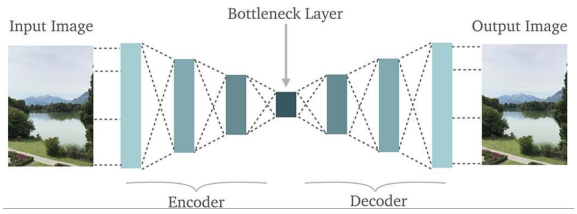
Autoencoders (unsupervised)



Introduciamo la possibilità di estendere questa struttura anche quando non ho dei target, chiamato **metodo non supervisionato di apprendimento**.

L’uscita non viene verificata contro un valore atteso che fornisco insieme all’istanza, ma semplicamente confrontando come errore l’output e

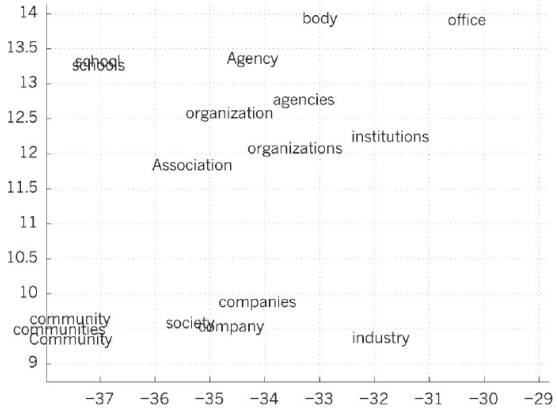
l’input. Ogni istanza è essa stessa il proprio target.

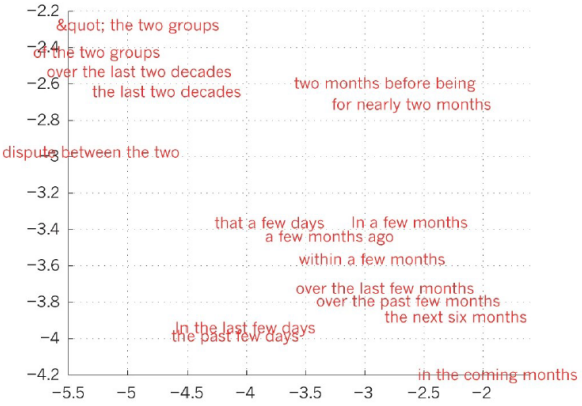


Se usiamo una strozzatura (**bottleneck**) il sistema è costretto ad usare meno neuroni che rende più difficile ricostrutire l’input quando si arriva all’output.

Se abbiamo degli stati che non sono booleani, ma numeri con la virgola (float) e se l’input è regolare (per esempio immagini realistiche, dove per esempio ci sono grosse zone di pixels uguali) allora la rete può farcela.

La rete impara quindi a fare una codifica: ogni immagine in input viene associata un valore in quei 2 piccoli neuroni. E viene quindi effettuata una decodifica da quei 2 neuroni fino all’output.





In questo esempio dall’input costruisco un vettore (x,y) che corrisponde ad una coppia di coordinate che posso rappresentare in un piano. La posizione corrispondente ad una certa parola è associata alla somiglianza del suo significato ad altre parole che le stanno vicine.

Riduzione di dimensionalità

C’è un collegamento tra la riduzione di dimensionalità e l’apprendimento.

Quando io codifico nell’autoencoder delle immagini sopra, con l’apprendimento sono stato in grado di produrre una funzione che passa da n pixel a 2. Il prodotto di

codifica di un autoencoder è analogo alla generale riduzione di dimensionalità.

Esistono algoritmi di riduzione di dimensionalità “non su machine learning” come PCA (Pincipal Component Analysis). L’obiettivo di riduzione di dimensionalità è parente stretto dell’apprendimento, perchè l’apprendimento cerca di cogliere una regolarità dei dati.

Questo può anche diventare una fase di pre elaborazione dei dati prima dell’apprendimento. In questo modo posso apprendere su istanze di cui ho ridotto la dimensione.