Università degli Studi di Milano - Bicocca
Scuola di Scienze
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di laurea in Informatica

Reti neurali: dall'algoritmo di backpropagation alle reti profonde

Relatore: Prof. Alberto Dennunzio

Co-relatore: Dott. Luca Manzoni

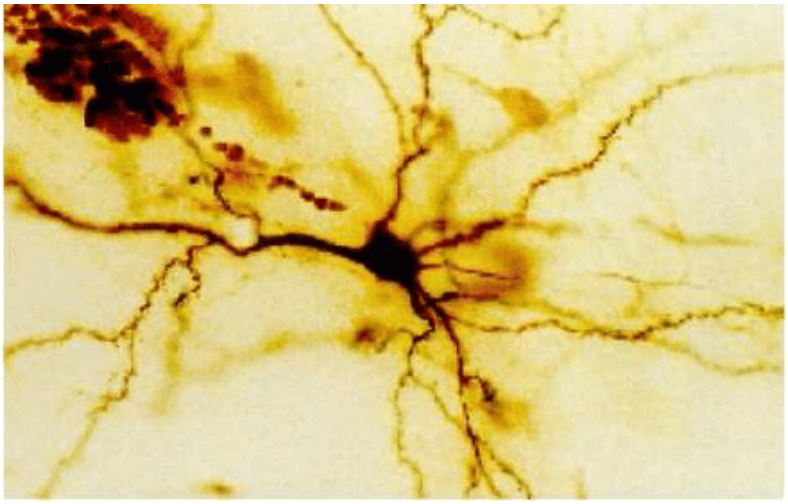
Relazione della prova finale di:
Alessandro Sassi
Matricola 807001

Anno Accademico 2017-2018

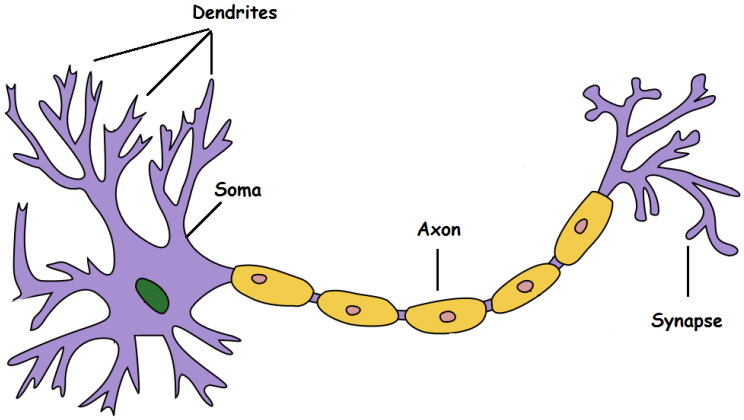
Introduzione

L'ispirazione biologica

Il cervello umano è composto da circa 10 miliardi di neuroni. Nella figura riportata in bassa attorno al nucleo di colore verde si estende il corpo cellulare delle cellule neurali , dette *soma* e i canali di input ed output che la circondano la collegano a circa altri 10000 (dieci mila) neuroni. Ogni neurone riceve degli stimoli elettrochimici dai neuroni circostanti attraverso i proprio dendriti. Se la somma degli input elettrici è potente abbastanza il neurone trasmette a sua volta un segnale elettrochimico lungo l'assone , che viene diffuso a tutti quei neuroni i cui dendriti sono connessi alle terminazione del neurone in oggetto. Per quel che vedremo in seguito è importante sottolineare come il neurone attivi il segnale in uscita solo se il segnale in ingresso totale raggiunge un certo livello, a quel punto il neurone attiva il proprio impulso, secondo un unico livello di segnale, ovvero, non ci sono sfumature di attivazione, o il segnale in uscita viene propagato o non viene propagato, non esistono segnali di potenza variabile da distribuire verso i neuroni connessi. Il nostro intero cervello è quindi composto interamente da questa fitta rete di cellule, i neuroni, che comunicano fra loro attraverso segnali elettrochimici. E' sorprendente vedere come la struttura alla base molto semplice, costituita da una cellula che sommando dei segnali in ingresso si attiva , o meno, comunicando con altre cellule attraverso segnali in uscita, riesca nel suo complesso reticolo di interazioni a svolgere funzioni complicate come quelle del cervello nella sua totalità. Lo studio del cervello e la sua comprensione ha potuto ispirare i modelli scientifici e matematici che oggi danno vita a quelle che sono dette oggi *reti neurali artificiali (ANN).*



(a) *An impression of a real neuron. This image is a light microscope photograph by John Anderson, Institute for Neuroinformatics, Zurich, Switzerland.*



(b) *A biological neuron model*

La storia delle reti neurali

La capacità dei computer di andare oltre la programmazione verso delle primordiali forme di apprendimento si sviluppò inizialmente come il tentativo di due scienziati di meglio comprendere il funzionamento dei nuroni nel cervello. Era il 1943 e il neurofisiologo Warrent McCulloch e il matematico Walter Pitts svilupparono con dei circuiti elettrici una semplice rete neurale che modellava i neuroni biologici. Nel 1949 Donald Hebb scrisse *L'organizzazione del Comportamento* in cui affermava di come i percorsi neurologici tendessero a rinforzarsi e a diventare più solidi mano mano che questi venivano allenati e utilizzati. Un lavoro che troverà conferme in studi successivi e documentati nel saggio di Nicholas Carr del 2010, in cui si fa riferimento alla plasticità del nostro cervello e alla capacità dei nostri neuroni di rinforzare i loro collegamenti man mano che vengono "allenati". (PENSARE A DOVE METTERE LA CITAZIONE DI CARR CHE TROVO NELLA MAIL A CABITZA) Nel frattempo i calcolatori diventavano più potenti e sul finire degli anni '50 Bernard Widrow e Marcian Hoff della Stanford University svilupparono "ADALINE" and "MADALINE". La prima era in grado di riconoscere dei

pattern binari nel linee telefoniche e predire i successivi bit in ingresso; la seconda fu la prima rete neurale ad essere applicata in un problema del mondo reale, ovvero era in grado , grazie a dei filtri adattivi (COSA SONO000??) di eliminare gli echi nel chiamate al telefono. Questo dispositivo , per quanto datato e primitivo funziona così bene da avere ancora oggi una sua validità commerciale. Nel corso degli anni , nonostante i primi buoni risultati delle reti neurali l'architettura di von Neumann si impose sulla scena dello sviluppo dei calcolatori, nonostante von Neumann stesso suggerisse di apprezzare l'approccio dell'imitazione delle funzionali neurali. Inoltre all'epoca furono pubblicati degli articoli scientifici, basati però su un presupposto erroneo, ovvero la non derivabilità delle funzioni di apprendimento (CERCARE DI CAPIRE MEGLIO), che raffreddarono gli entusiasmi per questa tecnologia e con questi gli investimenti per ulteriori sviluppi. Inoltre i primordiali successi condussero ad esagerare potenziale ed aspettative nei confronti delle reti neurali, generando in un primo momento clamore e diffidenza, paura anche rispetto a quallo che avrebbe potuto costituirsi come rapporto uomo-macchina e in seguito in delusione verso le aspettative non ripagate. Arrivando agli anni '70: nel 1972 furono sviluppate indipendentemente da Kohonen e Anderson delle reti molto simili che si basavano su una matematica matriciale ; del 1975 è invece la prima rete multistrato non supervisionata.

L'interesse verso le reti neurali artificiali si rinnovò nel 1982 quando John Hopfield della Caltech (USA) presentò un paper all'Accademia Nazionale delle Scienze. In questo documento presentava l'opportunità di rendere le macchine più utili ed efficienti usando reti bidirezionali rompendo con la tradizionale idea di reti in cui i sengali si propagassero in un'unica direzione. Nello stesso anno altri due ricercatori : Reilly e Cooper usarono reti "ibride" che applicavano differenti strategie di risoluzione per ognuno dei diversi strati della loro rete. Ancora nel 1982 una spinta ai finanziamenti in questo settore arrivò da una conferenza congiunta tra studiosi statunitensi e giapponesi sul tema. Vedendo i nipponici in vantaggio sullo sviluppo di queste tecnologie gli americani decisero di aumentare le risorse a disposizione della ricerca.

Nel 1986 arriviamo ad una svolta che diventa centrale nel corso di questa relazione, ovvero, lo sviluppo dell'idea del backpropagation error. Tre gruppi indipendenti di sviluppatori, tra i quali compariva David Rumelhart, un ex professore di Stanford del dipartimento di psicologia, arrivarono sempre indipendentemente all'idea di distribuire gli errori prodotti dalla computazione della rete su tutta la rete stessa in modo da riprogrammarne i parametri e farla apprendere dai propri errori. Quelle che prima abbiamo chiamato reti "ibride" avevano solo due layer mentre queste nuove reti a retropropagazione dell'errore presentavano molti livelli e , dato lo stato dell'arte degli hardware del tempo, erano ritenute molto lente; e così hanno proseguito ad essere, tanto che negli anni 2000, le computazioni potevano richiedere settimane intere. Così è stato finché i recenti sviluppi degli ultimi anni hanno messo a disposizione delle reti hardware performanti e notevoli quantità di dati da cui le reti possano trarre informazioni ed imparare [1].

I diversi tipi rete neurale

Al giorno d'oggi le reti neurali sono tante e diverse tra loro, vengono riconosciute con l'utilizzo di sigle sempre più lunghe e in questa sezione vedremo alcune delle principali. Ad alcune accenneremo solamente mentre altre non verranno nemmeno citate per via del

loro numero elevato. Le reti neurali biologiche hanno ispirato le reti neurali artificiali che vengono utilizzate per approssimare delle funzioni che non sono conosciute a priori [4] e questo costituisce, semplicisticamente, un ribaltamento del classico paradigma con cui usiamo i computer: la programmazione infatti usa funzioni impostate dal programmatore per computare dei dati in output mentre le reti neurali cercano di creare una funzione a partire da dati forniti. e questo costituisce, semplicisticamente, un ribaltamento del classico paradigma con cui usiamo i computer: la programmazione infatti usa funzioni impostate dal programmatore per computare dei dati in output mentre le reti neurali cercano di creare una funzione a partire da dati forniti. Le reti feedforward furono le prime reti ad essere pensate e sono anche le più semplici. Sono costituite da diversi strati (*layers*), che possono essere di input per i dati di apprendimento, nascosti o di output. Il perceptrone può essere visto come modello di un neurone biologico e può approssimare nella sua semplicità dei circuiti logici. Gli strati delle FF sono tutti collegati fra loro e il flusso dei dati viaggia in una sola direzione, senza mai formare cicli, dallo strato di input, attraverso gli strati nascosti fino allo strato di output. Questa rete viene definita profonda all'aumentare degli *hidden layers* che si nascondono fra input ed output e prendono il nome di *deep feedforward neural networks* (SPECIFICARE SE ESISTE UNA DIFFERENZA SOSTANZIALE, DEL TIPO : UNA VOLTA C'ERANO SOLO QUELLE NON PROFONDE O PERKE HANNO CREATE QUELLE PROFONDE, QUALE ERA L'ESIGENZA CHE LE DISTINGUE).

(c) *un perceptrone* (d) *una FF* (e) *una DFF*

Figura 1: Confronto tra tipologie diverse di algoritmi

Esistono poi anche le *radial basis network* (RBF)che altro non sono che reti feedforward (e che possiamo schematizzare proprio come la FF in figura sopra) ma con una funzione di attivazione radiale di base. Non tutte le reti prendono il nome dalla propria funzione di attivazione ma questa sembrava avere particolari speranze nel 1988, quando uscì il documento che la presentava poiché la funzione che stava alla sua base era in grado di interpolare in spazi di molte dimensioni.(PERCHÈ QUESTA SI ALLORA ????)

Un altro caso speciale o un evoluzione delle FF sono le *reti convoluzionali profonde* (*deep convolutional network , DCN*) il cui utilizzo primario è il processing delle immagini e in alcuni casi anche degli input audio. Si ispirano a quelle che è il funzionamento del sistema visivo degli uomini. Le reti convoluzionali sono in grado di riconoscere soggetti nelle immagini sapendo quindi fare una classificazione in un insieme di immagini che gli vengono

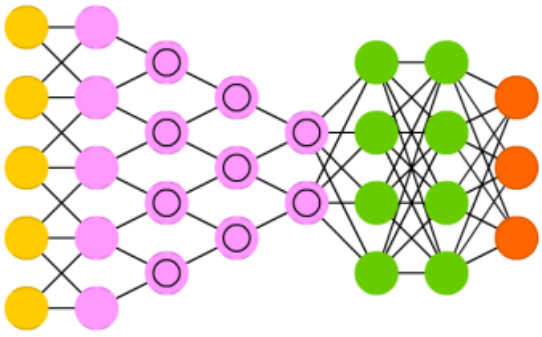


Figura 2: Un perceptrone (P).

sottoposte. Le DCN scansionano le immagini partendo da piccoli blocchi di pixel vicini tra loro e individuano le caratteristiche delle immagini porzione dell'immagine per porzione, fino a classificare l'immagine nella sua totalità. Diversamente delle tradizionali FF i layer di mezzo non sono collegati nodo per nodo fra loro, piuttosto i nodi vicini sono collegati ad un intorno dei nodi vicini nello strato successivo. Questo vuole ispirarsi al modo in cui il nostro campo visivo funziona e reagisce agli stimoli esterni. Man mano che il segnale fluisce lungo la rete attraversa strati costituiti da sempre meno nodi fino ad arrivare alla terminazione della rete in cui viene "attaccata" una piccola rete FF che consente di processare ulteriormente i dati e fare nuove astrazioni.

Figure 2 is a wrapped figure.

Un recente sviluppo , del 2014, delle reti che sembra essere promettente è la combinazione di due reti insieme. Di solito vengono associate tra loro FF e reti convoluzionali e si impostano in modo tale che esse si contrappongano nello svolgere dei compiti. Immaginiamo un pittore che dipinge su una tela un gatto , in sua compagnia un giudice verifica la qualità del lavoro confrontandolo che un insieme di fotografie di gatti reali; il pittore si allena a ritrarre sempre più fedelmente i gatti e il giudice viene sfidato nel giudicare via via se il gatto sottopostogli dall'amico sia una fedele riproduzione artistica o una foto reale. Eccom nelle reti *GAN*, come sono abbreviate, una delle due fa un lavoro di generazione mentre l'altra fa un lavoro di giudizio, in pratica una genera contenuti che vengono giudicati dall'altra e si allenano vicendevolmente nel giudicare una l'operato dell'altra, spingendo la seconda a produrre contenuti sempre più difficili da predire mentre la prima viene allenata dalla seconda , che producendo via via lavori sempre più approssimabili a quelli dell'insieme di confronto.

Abbiamo parlato fin'ora solamente di reti in cui i segnali partendo da uno strato di input si propagano uniderezionalmente verso la fine della rete. E' giusto annoverare brevemente anche quelle reti che invece formano dei cammini ciclici fra i loro nodi.

L'algoritmo di Backpropagation

La matematica dietro a backpropagation

Il nostro algoritmo viene applicato in modo da riprogrammare ad ogni iterazione i parametri della rete , apprendendo via via la loro configurazione migliore per approssimare la funzione che risolve il nostro problema. Nell'apprendimento supervisionato delle reti FF di cui ci occupiamo alleniamo la rete fornendole un paragone con i risultati aspettati del set di training. Nel set di training sono quindi presenti i vettori di soluzione per ogni problema. Indichiamo con $y(x) = (y_1, y_2, \dots, y_n)^T$ il vettore di n dimensioni di output che ci aspetteremmo in uscita dalla nostra rete; $y(x)$ è la funzione che la nostra rete vogliamo approssimi, la forma di questa funzione non è data e non si conosce a priori, conosciamo soltanto i suoi valori. BP valuta ad ogni iterazione (o EPOCH ?) quanto lontano dal valore atteso è il risultato della rete, computa per cui una funzione dei costi $C(w,b) \equiv \frac{1}{2n} \sum_x \parallel y(x) - a \parallel^2$ che chiamiamo *errore quadratico medio (MSE)*; a invece è a sua volta una funzione dei pesi w e dei bias b e rappresenta il vettore output della rete: in questo modo abbiamo quindi una funzione che mette in relazione il risultato atteso con quello effettivo e che diventa tanto più piccola nel suo valore quanto sono simili quando a approssima $y(x)$. Quindi, se l'intento della rete è produrre un output che approssimi al meglio le soluzione del set di apprendimento, e C ne è la misura, dobbiamo cercare di capire per quali valori questa funzione si minimizza, ovvero trovare per quali valori delle sue w e b variabili l'errore è minimo.
Per far questo abbiamo bisogno di un metodo. Potremmo pensare di trattare questa minimizzazione con un approccio analitico ma considerando che le variabili in gioco nelle reti reali possono essere anche miliardi questo non è fattibile. Facciamo allora ricorso ad un algoritmo, che in diversi passi successivi ci aiuta a trovare il punto di minimo che cerchiamo. Questo algoritmo è il *metodo della discesa del gradiente*, spieghiamo in primis cosa è il gradiente e poi come funziona l'algoritmo. Data una funzione di due o più variabili, il suo gradiente in un punto $x = (x_1, x_2, \cdots, x_n)$del dominio è il vettore delle sue derivate parziali rispetto a tutte le variabili: $\nabla C \equiv (\frac{\partial C}{\partial x_1}, \frac{\partial C}{\partial x_2}, \cdots, \frac{\partial C}{\partial x_n})^T$. Questa è solo una formula matematica e non ci da bene idea di cosa stiamo parlando. Dobbiamo allora rifarci ad uno scenario reale per capire meglio cosa sia il gradiente; la realtà è fatta di tre dimensioni quindi rimaniamo in uno spazio tale per spiegarci meglio: se fossimo in una valle e avessimo una palla perfettamente tonda , il gradiente è il vettore della direzione lungo la quale la funzione cresce maggiormente e , di contrario, $-\nabla C$ è la direzione in cui la palla si muoverebbe se, appoggiata a terra, venisse lasciata libera di rotolare lungo il pendio della valle. In una tale ipotesi le variabili libere non sono n come abbiamo generalizzato, ma sono soltanto due v_1, v_2 (usiamo le v e non le x per separare la spiegazione dalla sua metafora). Possiamo vedere una figura che rappresenta quanto detto.

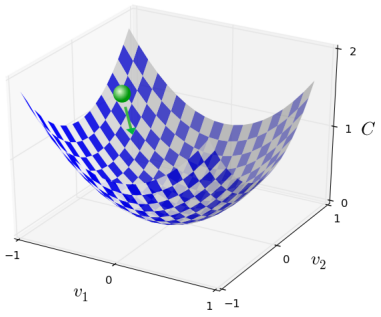


Figura 3: Il grafico della funzione $C(v_1, v_2)$ con la pallina verde che rappresenta il punto in analisi e il vettore gradiente spiccato da esso, ad indicare in quale direzione moverebbe nella discesa

Se ora abbiamo un'idea più chiara di cosa sia il vettore gradiente spieghiamo in cosa consiste il metodo della discesa che lo utilizza. Ipotizzando la stessa metafora della valle vogliamo sfruttare il fatto che la discesa della pallina si arresterà nella parte più bassa dalla depressione dipinta a scacchi della Baviera. Il nostro metodo computa una discesa simile ma diversa nel fatto che il percorso che farà la nostra pallina non approssimerà la discesa *fisica* che avverrebbe nella realtà. Per cui il nostro metodo non si rifà alle equazioni della dinamica di Newton, è un pò diverso. Per ora sappiamo che, dato un punto sulla superficie di una valle, la direzione opposta a quella del gradiente è quella in cui la palla si muoverebbe se fosse libera di scendere; nel nostro metodo seguiamo un percorso di discesa fatto di tanti piccoli passi che prevedono , partendo da un punto, di muoversi verso $-\nabla C$ per un piccolo tratto. Mossi nel nuovo punto, che chiamiamo P_2, ricalcoleremmo il gradiente $\nabla C(P_2)$ e faremmo altri piccoli passi in direzione $-\nabla C(P_2)$. Si capisce che questo metodo va iterato a lungo per scendere lungo la valle fino ad arrivare nel suo punto più basso. Tra un iterazione k e la successiva $k+1$, ci saremo spostati dal punto P_k al punto P_{k+1} per cui $\Delta P = P_{k+1} - P_k = -\eta \nabla C$ dove η è un piccolo parametro positivo detto il *learning rate* che fornisce un indicazione di quanto grande sia la distanza percorsa fra un iterazione e l'altra nell'algoritmo; va da se che se il η cresce l'algoritmo procede più speditamente , ma questo potrebbe portare anche a degli errori. Ritorniamo dal parallelo con la valle, la palla e le variabili v alla nostra funzione dei costi $C(w, b)$ e vediamo come il metodo del gradiente funzioni con più di due variabili, anche se di questa situazione non potremo avere una visualizzazione. Cosi come ci spostiamo da un punto all'altro nella valle, aggiornando le due variabili della nostra posizione, ora aggiorniamo le variabili della funzione dei costi, ovvero w i pesi e b i bias. Le regola per aggiornare queste componenti sarà:

$$w_k \to w'_k = w_k - \eta \frac{\partial C}{\partial w_k} \tag{1}$$

$$b_l \to b'_l = b_l - \eta \frac{\partial C}{\partial b_l} \tag{2}$$

dove k è il (FARSI AIUTARE A CAPIRE BENE k ED L). Ripetendo questa regola arriveremo a trovare il minimo della funzione. (SPIEGARE IL MINIBATCH E IL FATTO CHE È STOCASTICO IL GRADIENTE)
Ora che abbiamo spiegato come funziona il gradiente dobbiamo andare più a fondo a

spiegare come si rapportano fra loro i neuroni collegati, come si attivano e come propagano il proprio segnale. Ogni neurone j^{th} è collegato (stiamo sempre parlando del caso delle reti feed forward) ai neuroni dello strato precedente dall'arco di peso C (cioè dal neurone k in $l-1$ al neurone j in l)e la sua attivazione viene determinata da una computazione fatta sui segnali in ingresso, più precisamente è la somma delle attivazioni dei neuroni dello stato precedente a lui collegati, moltiplicati per il peso del loro arco verso j^{th} più una quantità b_j specifica del neurone, questà somma prende il nome di z_j^l.

$$a_j^l = \sigma\left(z_j^l\right) = \sigma\left(\sum_k w_{kj}^l a_k^{l-1} + b_j^l\right) \tag{3}$$

Esiste una quantità a_j^l per ogni neurone j nello strato l, possiamo annotare come un vettore questi a output per tutti gli n neuroni di quello strato $a^l = \left[a_1^l, a_2^l, \cdots, a_j^l, \cdots, a_n^l\right]$. Vediamo ora come di strato in strato i neuroni si attivano fra loro e per questo adottiamo una notazione matriciale di quanto abbiamo appena visto: denotiamo allora una matrice w^l dei pesi per ogni layer della rete, dove le sue entry sono w_{kj}^l i vari pesi che legano i neuroni fra lo strato $l-1$ ed l e denotiamo, allo stesso modo di a^l anche il vettore dei *bias* b^l per il livello l e riscriviamo la formula 3 come:

$$a^l = \sigma\left(w^l a^{l-1} + b^l\right) \tag{4}$$

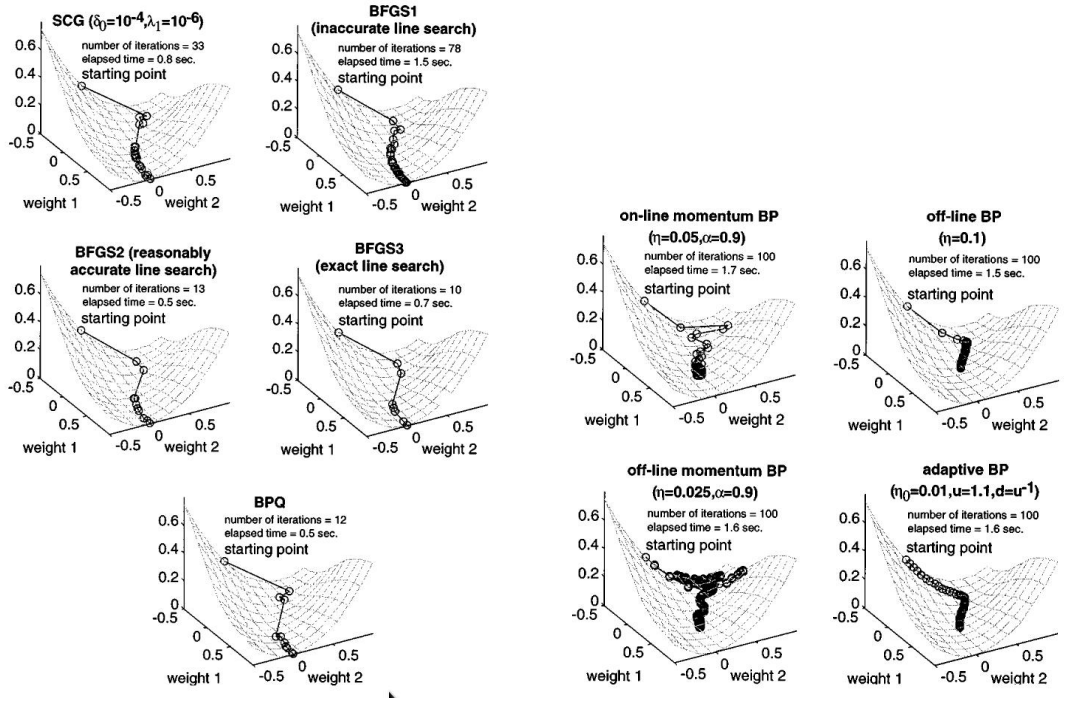
.
Introdurremo ora il concetto di errore, per lo specifico neurone e le quattro equazioni fondamentali per il backpropagation.
Un neurone j nel livello l si attiva in base all'output della sua funzione di attivazione che ha per argomento z_j^l . Backpropagation si propone di modificare nelle sue iterazioni i pesi

I miglioramenti

Migliorare backpropagation attraverso diversi approcci sulla discesa del gradiente

L'algoritmo di backpropagation si basa sull'ottimizzazione di una funzione d'errore tra l'output della rete e il risultato desiderato. La funzione in questione ha come variabili i pesi associati ai collegamenti tra i vari neuroni della rete e i loro bias. Nella sua versione più semplice la funzione cerca il proprio minimo seguendo la direzione del proprio gradiente. Venendo alle variazioni su questo metodo; furono proposti il *metodo dei minimi quadrati* (in inglese OLS: Ordinary Least Squares) e il *metodo detto di quasi-newton* (che è una variazione sul canonico metodo di Newton) che risultano essere però troppo lenti da computare, in special modo nelle reti molto ampie e ad entrare nello specifico il problema per i metodi di quasi-newton è che lo spazio in memoria richiesto per salvare una approssimazione dell'inversa dell'Hessiana cresce quadraticamente rispetto al numero dei pesi su cui viene effettuato il calcolo [2]. Altre tecniche come *BFGS* o il *gradiente coniugato* possono essere in alcuni casi delle valide alternative. Più genericamente il problema del migliorare l'ottimizzazione dell'errore trova soluzione quando si migliora la capacità di computare la più utile lunghezza di passo possibile (step-lenght) ovvero

quanto distante una iterazione deve essere da quella che la precede avendo seguito la direzione del gradiente. Esistono per questo algoritmi del secondo ordine e del primo ordine, che possiamo vedere a confronto nelle due immagini in fig.1.1.



(a) *Traiettorie algoritmi del primo ordine* (b) *Traiettorie algoritmi del secondo ordine*

Figura 4: Confronto tra tipologie diverse di algoritmi

Si può vedere come l'andamento della traiettoria conduca al minimo in numero inferiori di passi utilizzando algoritmi del secondo ordine.

Per velocizzare BP è stato anche introdotto il momento , che ha lo scopo di ridurre le oscillazioni della traiettoria dovute ad una non proficua scelta della lunghezza del passo per le iterazioni e fu proprio nel 1988 che Robert A. Jacobs, propose quattro euristiche per il miglioramento del tasso di convergenza del BP.

Nel 1988 la ricerca aveva già riconosciuto la bontà della procedura BP e si investigavano quegli algoritmi di ricerca dell'ottimo per la funzione di minimo che computassero la discesa del gradiente soltanto localmente (ovvero rimanendo in intorni molto vicini al punto di iterazione). Questo cosidetto *locality constraint* veniva motivato dal fatto che costituiva una buona metafora tecnologica della controparte biologica , le reti neurali, a cui si ispirava e in secondo luogo dall'ipotesi che questi algoritmi *locali* fossero più adatti ad essere processati in parallelo. Fra le quattro euristiche di Jacobs vi era l'adattamento dei tassi d'apprendimento nel tempo e si spiega che ogni peso della rete dovrebbe avere il proprio tasso d'apprendimento specifico. Le implementazioni di queste euristiche venivano individuate nel *momento* e nella regola d'apprendimento *delta-bar-delta*.

low complexity NNs

Alcuni altri indirizzi di miglioramento dell'efficacia delle reti sfruttano il BP per modellare delle reti meno complesse (*low complexity*) che possano essere utili a scopi meno sofistica-

ti. Ad esempio, nelle reti convuluzionali, dove il processo di riconoscimento degli oggetti ha luogo, gli algoritmi vengono eseguiti su costese GPU che dissipano grandi quantità di energia e un tale scenario non è adatto a scopi dove il livello di dettaglio nel riconoscere gli oggetti non è così elevato. Applicazioni più popolari e frequenti come il riconoscimento facciale nei dispositivi mobili deve per forza di cose girare in locale sui processori embedded che animano gli smartphone, per questo i ricercatori sfruttano varianti del BP per creare reti convuluzionali a bassa complessità. Queste modellano problemi molto meno *demanding* e possono quindi essere eseguite più velocemente anche sui dispositivi meno prestanti [3].

Bibliografia

[1] . History of neural networks. https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/History/index.html, 2010. Online; accessed 28 Febraury 2019.

[2] Kazumi Saito and Ryohei Nakano. Partial bfgs update and efficient step-length calculation for three-layer neural networks. *Neural Computation*, 9(1):123–141, 1997.

[3] Subarna Tripathi, Gokce Dane, Byeongkeun Kang, Vasudev Bhaskaran, and Truong Nguyen. Lcdet: Low-complexity fully-convolutional neural networks for object detection in embedded systems. 05 2017.

[4] Wikipedia contributors. Types of artificial neural networks — Wikipedia, the free encyclopedia, 2019. [Online; accessed 1-March-2019].