UNIVERSITÁ DEGLI STUDI "ROMA TRE"
Dipartimento di Matematica e Fisica
Corso di Laurea Magistrale in Scienze Computazionali

Tesi di Laurea Magistrale

Ricerca della tolopologia ottimale di un sistema di deep learning per identificazioni di oggetti architettonici

Candidato
Dèsirèe Adiutori

Relatore
Prof. Alberto Paoluzzi

Anno Accademico 2017/2018
Luglio 2018

Indice

Introduzione

Dall'invenzione dei computer, l'uomo fa sempre più affidamento sulle macchine per risolvere problemi complessi di calcolo. Con l'aumentare delle prestazioni dei computer, man mano si sono sviluppati algoritmi di calcolo sempre più efficienti. Nel 1959 l'ingegnere del MIT, Arthur Samuel coniò il termine *"machine learning"*, descrivendo l'apprendimento automatico come un "campo di studio che dà ai computer la possibilità di apprendere senza essere programmati esplicitamente per farlo".[1]
Definiamo l'apprendimento automatico come un insieme di metodi in grado di rilevare automaticamente i modelli tramite dei dati e quindi utilizzare i modelli scoperti per prevedere i dati futuri o per eseguire altri tipi di processi decisionali in condizioni di incertezza. L'insegnamento alla macchina è, pertanto, imprescindibile dai dati. Generalmente, più dati si passano alla macchina, più può imparare. Per questo motivo con l'avvento di Internet, dagli anni '90 ad oggi il tema del "machine learning" è diventato sempre più attuale, la mole di dati reperibile dal web è cospicuo e ha permesso che questo campo sia esponenzialmente progredito.
Il *"deep learning"* è un tipo particolare di machine learning, che riguarda l'emulazione di come gli esseri umani apprendono. Esso affronta i problemi del machine learning, rappresentando il mondo come una gerarchia di concetti annidati: ogni concetto è definito in relazione a concetti più semplici e le rappresentazioni astratte vengono calcolate in termini di concetti meno astratti. Il Deep Learning implica l'utilizzo di reti neurali artificiali (*deep artificial neural networks*), algoritmi e sistemi computazionali, ispirati al cervello umano, per affrontare i problemi del Machine Learning.
L'analogia di Shehzad Noor Taus Priyo può aiutare a capire meglio cosa siano le reti neurali:
"Immaginiamole come una serie di porte da oltrepassare, dove l'input è l'uomo che le deve oltrepassare e ogni volta che lo fa cambia qualcosa nel suo comportamento finchè, all'ultima porta oltrepassata, l'uomo è diventato una persona del tutto differente, rappresentando l'output di questo processo."[2]
Questa tesi si focalizza su un problema particolare di algoritmo di machine learning: la Classificazione (*Classification*), in particolar modo della classificazione di immagini. L'obiettivo principale è trovare un'architettura ottimale per l'algoritmo che identifica le immagini di oggetti architettonici, per fare questo bisogna trovare la giusta topologia, la giusta profondità e la giusta larghezza di ogni livello della rete neurale.

Capitolo 1

Algoritmi di apprendimento

Gli algoritmi di machine learning sono solitamente divisi in tre tipi principali:

- Supervised learning (apprendimento supervisionato)
- Unsupervised learning (apprendimento non supervisionato)
- Reinforcement learning (apprendimento per rinforzo)

Quali usare? Perchè sceglierne uno piuttosto che un altro?
La scelta dell'algoritmo da utilizzare dipende dal tipo di dati di cui si dispone. Ma la scelta finale va fatta solo esclusivamente dopo aver testato l'algoritmo, e si sceglie in base a quello più performante: un insieme di ipotesi che funziona bene in un dominio, potrebbe funzionare male in un altro.
Teorema del No Free Lunch [3, Wolper,1996]
Non esiste una definizione universale di algoritmo "migliore".

1.1 Costruzione di un algoritmo di apprendimento

Per costruire un algoritmo di apprendimento bisogna avere:

- processi(*task*), compiti che l'algoritmo deve eseguire;
- misuratori di rendimento, rilevatori delle caratteristiche dei processi;
- esperienze, quantità di dati dal quale imparare.

I processi di apprendimento automatico descrivono come il sistema dovrebbe elaborare un esempio.

Definizione 1 *Un **esempio** è una raccolta di caratteristiche che sono state misurate quantitativamente da alcuni oggetti o eventi elaborati.*

Di solito, un esempio viene rappresentato da un vettore $x \in \mathbb{R}^n$, dove ogni elemento x_i rappresenta una caratteristica. Dato un processo si cerca di capire quale sia la caratteristica principale, sulla quale si deve misurare il suo rendimento. Infine, dobbiamo dare all'algoritmo un'esperienza sulla quale apprendere, che è quella che lo classificherà in uno dei tre tipi principali. Questa esperienza l'apprende dai *dataset*: una collezione di esempi.
I dataset possono essere di vari tipi:

- insieme di addestramento (*training set*)
- insieme di prova (*test set*)
- insieme di validazione (*validation set*)

L' **insieme di addestramento** è una parte dell'insieme di dati che vengono utilizzati per addestrare un sistema di apprendimento supervisionato. Da questo insieme, l'algoritmo deve costruire una funzione che capisca, dai parametri, quali caratteristiche descrivono le varie categorie.
L' **insieme di prova** è un insieme di dati che, con l'insieme di addestramento, forma una partizione del dataset di partenza. Questi nuovi dati vengono utilizzati per valutare l'apprendimento dell'algoritmo "addestrato".
L' **insieme di validazione** è usato in maniera analoga all'insieme di prova, ma dei dati inseriti per testare l'algoritmo già si conosce la risposta (una parte di essi può far parte dell'insieme di addestramento) e da questa si valuta se l'output ottenuto è ottimale o meno.
Questi tre insiemi possono essere usati anche tutti e tre contemporaneamente. La scelta della cardinalità dei vari sottoinsiemi non è universale e dipende dal tipo di problema che viene affrontato.
Vediamo ora come valutare l'efficienza di un algoritmo:

Definizione 2 *L'**errore di allenamento**(training error) è una misura di errore che si può calcolare sul set di allenamento. Indica quanto l'algortimo sta apprendendo.*

Definizione 3 *La **generalizzazione** è la capacità di un algoritmo di essere ottimale in seguito ad un input proveniente dall' insieme di prova.*

Definizione 4 *L'**errore di generalizzazione**(generalization error) è una misura di errore che si può calcolare sull'insieme di prova. Verifica se l'algoritmo ha imparato o solo memorizzato. Esso viene detto anche errore di test (test error).*

Ipotizziamo che tutti gli esempi siano eventi indipendenti e che tuti gli insiemi, in cui partizioniamo l'insieme di dati, hanno la stessa distribuzione di probabilità uniforme.

Definizione 5 *Una **funzione di perdita** (loss function) $L(y, \hat{y})$ è una funzione che misura la distanza (o l'errore) tra i valori di output previsto $\hat{y}$ e i valori effettivi y.*

Si possono usare varie misure, per esempio nel caso dell'errore quadratico medio(MSE):

$$L(y, \hat{y})_{train} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_{(train)} - y_{(train)})_i^2 \tag{1.1}$$

La nostra funzione di predizione dipenderà da dei parametri, rappresentati da un vettore w, lo scopo è di minimizzare l'errore di allenamento variando w. In base al tipo di apprendimento e al problema da affrontare verranno usati vari algoritmi per risolvere problemi di minimizzazione libera. Ovvero gli algoritmi per risolvere il problema:

$$f(x^*) = \min_{x \in \mathbb{R}^n} f(x), \qquad f \in C^2$$

Ad esempio verrà spesso utilizzato il metodo di discesa del gradiente.[**?**]
La struttura generale di un metodo di discesa iterativo di minimizzazione è:

$$x_{k+1} = x_k + \beta_k d_k \tag{1.2}$$

dove x_0 è assegnato, $\beta_k \in \mathbb{R}^+$ è il passo e $d_k \in \mathbb{R}^n$ è la direzione lungo la quale ci si muove, che essendo una direzione di discesa sarà $(d_k, \nabla f(x_k)) < 0$. Sia il passo che la direzione vanno scelti opportunamente ad ogni passo, in modo che $f(x_{k+1}) < f(x_k)$.
Il passo viene scelto in modo che si abbia:

$$f(x_k + \beta_k d_k) = \min_{\beta} \{f(x_k + \beta d_k)\} \qquad (\textit{strategia di ricerca esatta})$$

Mentre la scelta della direzione é $d_k = -\nabla f(x_k)$. La derivata direzionale di f nella direzione d_k vale

$$\frac{\partial f}{\partial d_k}(x_k) = \frac{(d_k, \nabla f(x_k))}{\| d_k \|} = - \| \nabla f(x_k) \|$$

e per la disuguaglianza di Cauchy-Schwartz si ha anche:

$$\frac{| (d_k, \nabla f(x_k)) |}{\| d_k \|} \leq \frac{\| d_k \| \| \nabla f(x_k) \|}{\| d_k \|} = \| \nabla f(x_k) \|$$

che mostra come la direzione di ricerca sia quella in cui la derivata direzionale di f è negativa e di modulo massimo.
Le condizioni di arresto del metodo sono: $\| x_{k+1} - xk \| \leq m$, $\| \nabla f(x_{k+1}) \| \leq m\prime$ oppure $k > k_{max}$, dove m e $m\prime$ sono soglie date e k_{max} il numero massimo di iterazioni da effettuare.
I risultati ottenuti sono garantiti dai seguenti teoremi di convergenza:

Teorema 1 *Sia $f(x) \in C^1$, strettamente convessa sull'insieme $\Sigma_0 = \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$, e la successione $\{x_k\}$ sia generata tramite l'algoritmo 1.2. Si supponga*

1. *che l'insieme Σ_0 sia compatto;*
2. *che le direzioni d_k siano t.c. $\frac{(d_k, \nabla f(x_k))}{\|d_k \quad \| \|\nabla f(x_k)\|} \leq -\cos\theta$ per $k \in I$, con I insieme illimitato di indici;*
3. *che per $k \in I$, β_k sia ottenuto tramite ricerca esatta.*

Allora la successione $\{x_k\}$ converge all'unico punto x^* di minimo per f.

Teorema 2 *Sia $f(x) \in C^2$, strettamente convessa sull'insieme (che si suppone compatto) $\Sigma_0 = \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$, e la successione $\{x_k\}$ sia generata tramite l'algoritmo 1.2, con $d_k = -\nabla f(x_k)$.*
Allora, se i passi β_k sono determinati tramite ricerca esatta, la successione x_k converge all'unico punto x^* di minimo per f.

Minimizzare l'errore di allenamento non necessariamente comporta l'ottimizzazione di apprendimento dell'algoritmo, potrebbe verificarsi il fenomeno di adattamento insufficiente (*underfitting*),ovvero non si hanno abbastanza dati per creare un modello di predizione accurato . Bisogna quindi valutare anche altri fattori: analizzare l'insieme di prova.
Ricordandoci dell'eq.1.1, calcoliamo l'errore di prova:

$$L(y, \hat{y})_{test} = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_{(test)} - y_{(test)})_i^2 \tag{1.3}$$

Vorremmo che ,con i parametri trovati per minimizzare l'errore di allenamento, anche questo errore sia minimo (l'ottimalità è 0). Ma come detto in precedenza non sempre questo accade, vorremmo quindi che il divario tra i due errori sia minimo. In caso contrario si verifica il fenomeno di adattamento eccessivo (*overfitting*) del modello all'insieme di dati che descrive, tramite un eccessivo numero di parametri. Il modello quindi non sarà generalizzabile ad un nuovo insieme di dati.

Consideriamo il valore atteso dell'errore di prova, calcolato prendendo una coppia di punti (X, Y)dall'insieme di prova:

$$\mathbb{E}[L(y, \hat{y})_{test}] = \mathbb{E}[(Y - \hat{y}(X))^2] \tag{1.4}$$

e definiamo la funzione dell'output effettivo come:

$$y(X) = \mathbb{E}(Y|X)$$

la quale avrà sicuramente un errore, dovuto da qualche interferenza che chiameremo: distorzione stimata (*estimation bias*).
Ma con diversi insiemi di allenamento, possiamo costruire diverse funzioni $\hat{y}$, e anche questo è un'altra fonte di errore: la varianza stimata (*estimation variance*). Possiamo quindi scrivere l'output come:

$$Y = y(X) + \epsilon$$

con ϵ indipendente da X tale che $\mathbb{E}[X] = 0$ e $Var(X) = \sigma^2$.
Possiamo quindi riscrivere l'equazione 1.4 come:

$$\begin{aligned} \mathbb{E}[L(y, \hat{y})_{test}] &= \mathbb{E}[(Y - \hat{y}(X))^2 | X = x] \\ &= \mathbb{E}[(Y - y(x))^2 | X = x] + \mathbb{E}[(y(x) - \hat{y}(x))^2 | X = x] \\ &= \sigma^2 + \mathbb{E}[(y(x) - \hat{y}(x))^2] \end{aligned} \tag{1.5}$$

dove σ^2 è chiamato errore Bayes e

$$\begin{aligned} \mathbb{E}[(y(x) - \hat{y}(x))^2] &= (\mathbb{E}[\hat{y}(x)] - y(x))^2 + \mathbb{E}[(\hat{y}(x) - \mathbb{E}[\hat{y}(x)])^2] \\ &= Bias(\hat{y}(x))^2 + Var(\hat{y}(x)) \end{aligned} \tag{1.6}$$

Si ottiene così il compromesso distorzione-varianza (*bias-variance tradeoff*):

$$\mathbb{E}[L(y, \hat{y})_{test}] = \sigma^2 + Bias(\hat{y}(x))^2 + Var(\hat{y}(x)) \tag{1.7}$$

Se la distorzione ha valori alti e la varianza bassi avremo un fenomeno di adattamento insufficiente, mentre se la distorzione ha valori bassi e la varianza alti avremo un adatteamento eccessivo.[**?**]

1.2 Apprendimento supervisionato

Gli algoritmi di apprendimento supervisionato vengono utilizzati per risolvere problemi di classificazione e di regressione. Si parla di apprendimento supervisionato quando il dataset che si utilizza contiene delle variabili, una

delle quali è un'etichetta. Dato un vettore di input $x = (x_1, \cdots, x_n)$, ogni x_i è un vettore d-dimensionale di numeri rappresentanti una caratteristica, da questi dati si costruisce l'insieme di addestramento di cardinalità N: $D = \{(x_i, y_i)\}_{i=1}^{N}$, dove $y = (y_1, \cdots, y_m)$ è l'output dei risultati desiderati e y_i è l'etichetta. Lo scopo è di apprendere una regola generale che colleghi i dati in ingresso con quelli in uscita, in modo che l'algoritmo apprenda a classificare un esempio completamente nuovo, non contenente l'etichetta.
Se y_i è di tipo testuale si parla di classificazione, quando invece è di tipo numerico si parla di regressione. Se indichiamo con C il numero delle classi a cui può appartenere l'output: $y \in \{1, ..., C\}$, se $C = 2$ la classificazione sarà binaria (in questo caso spesso $y \in \{0, 1\}$); se $C > 2$ sarà multiclasse.
Vediamo ora nel dettaglio gli algoritmi di questo tipo che ci interessano.

1.2.1 Classificazione

La Classificazione viene usata quando è necessario decidere a quale categoria appartiene un determinato dato.Per esempio, data una foto capire a quale categoria appartiene, in questo caso capire a quale tipo di monumento appartiene.
Questo tipo di algoritmo deve specificare a quale delle k categorie appartiene un input. Crea una funzione $f : \mathbb{R}^n \to \{1, \cdots, k\}$, quando $y = f(x)$, il modello assegna l'input descritto dal vettore x ad una categoria identificata dal codice numerico y. Esistono altre varianti dell'attività di classificazione, ad esempio, dove f genera una distribuzione di probabilità su classi.

1.2.2 Regressione

La Regressione prevede il valore futuro di un dato avendo noto il suo valore attuale. Un esempio è la previsione della quotazione delle valute o delle azioni di una società. Nel marketing viene utilizzato per prevedere il tasso di risposta di una campagna sulla base di un dato profilo di clienti; nell'ambito commerciale per stimare come varia il fatturato dell'azienda al mutare della strategia.

Regressione lineare

Preso un vettore $x \in \mathbb{R}^n$ in input, l'algoritmo cerca di prevedere l'output: $y \in \mathbb{R}$. Dove $y = f(x)$, con f una funzione lineare.
Sia $\hat{y}$ il valore di output che l'algoritmo prevede. Definiamo l'output come:

$$\hat{y} = w^\top x$$

dove $w \in \mathbb{R}^n$ è un vettore di parametri e $w^\top$ il suo trasposto.

1.3 Apprendimento non supervisionato

Gli algoritmi di apprendimento non supervisionato vengono utilizzati per risolvere problemi di raggruppamento. All'algoritmo viene passato solo l'input: $D = \{x_i\}_{i=1}^N$ e cerca una relazione tra i dati per capire se e come essi siano collegati tra di loro. Non contenendo alcuna informazione preimpostata, l'algoritmo è chiamato a creare "nuova conoscenza" (*knowledge discovery*). A differenza del caso supervisionato, questo apprendimento non ha una classificazione o un risultato finale con il quale determinare se il risultato è attendibile, ma generalizza le caratteristiche dei dati e in base ad esse attribuisce ad un input un output: serve generalmente ad estrarre informazioni non ancora note.

1.4 Apprendimento per rinforzo

Gli algoritmi di apprendimento per rinforzo vengono utilizzati per risolvere problemi di regressione. Lo scopo di questo algoritmo è di realizzare un sistema in grado di apprendere ed adattarsi ai cambiamenti dell'ambiente in cui si trovano, attraverso la distribuzione di una "ricompensa" detta rinforzo, data dalla valutazione delle prestazioni. Questi algoritmi sono costruiti sull'idea che i risultati corretti dovrebbero essere ricordati, per mezzo di un segnale di rinforzo, in modo che diventino più probabili e quindi più facilmente riottenuti nelle volte future; viceversa se il rusultato è errato , il segnale sarà una penalità, ovvero si avrà una probabilità più bassa legata a quel determinato output.[4]

Capitolo 2

Reti neurali

Bibliografia

[1] Arthur Samuel. https://www.ibm.com/developerworks/community/blogs/jfp/entry/What_Is_Machine_Learning?lang=en, 1959.

[2] Shehzad Noor Taus Priyo. https://towardsdatascience.com/intro-to-deep-learning-d5caceedcf85, 2017.

[3] David H. Wolper and William G. Macready. No free lunch theorems for optimization, 1996.

[4] Frank L Lewis. *Reinforcement learning and approximate dynamic programming for feedback control*. Wiley, 2013.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[6] K.P. Murphy. *Machine Learning: A Probabilistic Perspective*. Adaptive computation and machine learning. MIT Press, 2012.