Interazione Naturale - Analisi Teorica

Alessandro Costella

Università degli Studi di Milano

1 Introduzione

Le Reti Neurali Ricorrenti rappresentano la più utilizzata soluzione nel campo dell'apprendimento automatico per modellare sequenze nel tempo. La loro caratteristica, ossia la riapplicazione ricorsiva dei pesi al segnale in ingresso, permette di tenere traccia di dipendenze a lungo termine e mantenere informazioni sull'ordinamento. Per questo motivo si sono rivelate di grande efficacia in problemi di analisi del linguaggio, generazione musicale ed elaborazione di video. Proprio in quest'ultima categoria si colloca l'esperimento di Kuehne, Richard e Gall riportato in "A Hybrid RNN-HMM Approach for Weakly Supervised Temporal Action Segmentation" pubblicato nell'Aprile dell'anno corrente su IEEE. Gli autori propongono nel loro studio un'architettura ricorrente che apprende come riconoscere azioni e sottoazioni in una sequenza video trattandole come un Hidden Markov Model. Lo scopo del loro lavoro è creare un framework per l'etichettatura di dati, utile all'apprendimento superivisionato, tramite apprendimento debolmente supervisionato allo scopo di sgravare un operatore umano dal lento lavoro di annotazione manuale di video frame per frame. Il presente testo tratterà principalmente l'analisi dei due strumenti utilizzati, rispettivamente quello tecnologico e matematico, e gli accorgimenti specifici dell'architettura dell'esperimento.

2 Il Modello Matematico

2.1 Processi Markoviani

L'assunzione fatta dagli autori dell'esperimento è che la sequenza di azioni e sottoazioni all'interno del video possa essere rappresentata efficacemente da un Modello Markoviano Nascosto (o HMM, "Hidden Markov Model). Un Modello Markoviano è, in generale, un modello che rappresenta un sistema in cambiamento tramite un processo stocastico che gode della proprietà di Markovianità, definita come segue: data una qualsiasi serie di stati del processo, consecutivi nel tempo, $(x_1, t_1), (x_2, t_2), ..., (x_n, t_n)$ di probabilità $P(i) = P(x_1, t_1)$ con $i = 1, ..., n$ è detto Markoviano un processo in cui

$$P(k \mid k-1, k-2, ..., 1) = (P(k \mid k-1) \,\forall\, k \neq 1 \tag{1}$$

Si noti che la formula completa per la probabilità di un determinato stato k è data da

$$P(k) = P(k \mid k-1 ..., 1) P(k-1 \mid k-2, ..., 1) ... P(2 \mid 1) P(1) \tag{2}$$

che possiamo, applicando l'equazione (1), ridurre infine alla forma

$$P(k) = P(k \mid k-1) P(k-1 \mid k-2) ... P(2 \mid 1) P(1) \tag{3}$$

2.2 Modelli Markoviani Nascosti

Nella simulazione dell'evoluzione di un sistema esso può essere dunque rappresentato come un Processo Markoviano. Tuttavia non è sempre è dato osservare o conoscere lo stato del sistema: si fa dunque uso di un HMM. E' detto Modello Markoviano Nascosto un modello definito come segue.

Siano X_n e Y_n processi stocastici a tempo discreto. Si definisce HMM la coppia formata da (X_n, Y_n) se:

- X_n è un Processo Markoviano i cui stati non sono osservabili. Questi prendono il nome di *stati nascosti*.
- Si ha che $P(Y \in S \mid X_n = x_n, ..., X_1 = x_1) = P(Y \in S \mid X_n = x_n)$.

con S set arbitrario e misurabile. La probabilità $P(Y \in S \mid X_n = x_n)$ è detta *probabilità di output*. Si palesa dalla definizione come l'obiettivo di un simile modello sia quello di massimizzare la verosimiglianza di uno stato nascosto dato l'output che esso ha prodotto $P(X_n \mid Y_n \in S)$.

Nell'esperimento i ricercatori trattano dunque le azioni compiute nel video, che coincidono con le etichette che devono essere riconosciute e assegnate dall'architettura, come stati nascosti X_n e i frame come output Y_n prodotti dagli stati nascosti. Nell'esperimento il numero di stati nascosti è, insieme al loro ordine, noto a priori. Gli autori procedono inoltre a una suddivisione delle azioni in sottoazioni secondo un modello gerarchico, assegnando un Modello Markoviano Nascosto a ognuna delle azioni presenti nel video. L'assunzione di markovianità si traduce all'atto pratico in un vincolo di classificazione: data l'azione (o la sottazione) $a(t) = a_i(t)$ assegnata al frame t, è possibile assegnare al frame $t+1$ solamente la stessa azione (o sottoazione) $a(t+1) = a_i(t+1)$ o la successiva $a(t+1) = a_{i+1}(t+1)$.

2.3 Probabilità delle etichette

I dati in input sono forniti come un insieme di tuple della forma $(\mathbf{x}_1^T, \mathbf{a}_1^N)$ dove $\mathbf{x}_1^T$ denota il vettore di T *features* frame per frame di un video contenente, appunto, un numero T di frames e il vettore $\mathbf{a}_1^N$ una sequenza ordinata di N azioni che avvengono nel video. Non è presente, nei dati in questa forma, alcuna indicazione circa l'inizio, il termine o la durata di ogni azione. Questo per permettere, in caso di successo, di ottenere un'architettura che richieda a un operatore umano il solo sforzo di inserire in input le azioni compiute nell'ordine in cui sono state compiute senza dover osservare il video un frame alla volta per annotare tali informazioni con precisione.

Il compito che spetta alla macchina è quindi quello di identificare una segmentazione che mappi ognuno dei T frames in un'etichetta $a_n(t)$ che rappresenta l'azione in esso compiuta. La mappa è dunque della forma

$$n(t) : \ 1, ..., T \longmapsto 1, ..., N. \tag{4}$$

All'inizio dell'addestramento tale segmentazione è completamente uniforme, suddivide cioè la sequenza di T frames in N segmenti di lunghezza $\frac{T}{N}$. Si noti che, a causa della struttura gerarchica, questo avviene anche all'interno di ogni segmento per dividerlo a sua volta in sottoazioni. La verosimiglianza iniziale del modello è dunque

$$p(\mathbf{x}_1^T \mid \mathbf{a}_1^N) := \Pi_{t=1}^T p(x_t | a_{n(t)}) \tag{5}$$

dove $p(x_t|a_{n(t)})$ denota la probabilità che il frame x_t sia stato generato dall'azione $a_{n(t)}$. Tale processo si applica identicamente alla sequenza di sottoazioni che compongono ciascuna delle azioni. Applicando la regola precedentemente formulata per

markovianità riguardo all'assegnazione delle etichette alle sottoazioni associate a ogni frame e dall'equazione (5) è dunque possibile affermare che

$$p(\mathbf{x}_1^T|\mathbf{s}_1^T) = \Pi_{t=1}^T P(x_t|s(t))p(s(t)|s(t-1)) \tag{6}$$

dove *s(t)* denota la sottoazione associata al frame t-esimo.

3 Apprendimento tramite sequenze di input

Il problema della modellazione di sequenze nel ltempo si è rivelato annoso nel campo dell'apprendimento automatico tramite reti neurali. Esso ha infatti portato alla luce diversi limiti dell'approccio "feed-forward" classico in caso di necessità di trattare dati interrelati nel tempo. In particolare, nel campo dell'analisi del linguaggio e di sequenze video, colmare simmultaneamente le seguenti necessità ha messo alla prova la ricerca:

- La necessità di gestire sequenze input di lunghezza variabile. Le reti classiche, infatti, sono genericamente progettate per apprendere tramite vettori input di lunghezza fissa.
- Tener traccia delle dipendenze a lungo termine. Tecniche come la "bag of words" appartenente al campo dell'analisi del linguaggio, nate per risolvere il problema della lunghezza variabile, non riescono a preservare l'interdipendenza tra le diverse parti dell'input.
- Gestire l'ordine degli input. L'ordine è un'altra informazione perduta nell'applicazione delle tecniche appena menzionate, eppure è un'informazione fondamentale per inferire il significato di una frase o il contenuto di una sequenza di frames video.
- Condivisione dei parametri attraverso la sequenza. Alcune tecniche proposte per la risoluzione delle problematiche sovracitate devono ricorrere a vincoli molto forti, come la conservazione delle posizioni delle "parti" di informazione nella sequenza di input in maniera ferrea e immutabile (si pensi ad esempio all'ordine soggetto-predicato-oggetto in una frase da elaborare). Questo implica però che, una volta appresi determinati parametri relativi ad essa (ad esempio all'analisi del soggetto), la rete non è in grado di elaborare correttamente alcun altro tipo di informazione nella stessa posizione, poiché i parametri che la riguardano sono stati appresi in una posizione differente.

3.1 Reti Neurali Ricorrenti

La soluzione a questa convergenza di necessità nasce con le Reti Neurali Ricorrenti (o Recurrent Neural Networks, RNN). La particolarità di tali reti profonde è nell'input di ciascun nodo: esso non riceve infatti solo il dato, unidirezionalmente diretto verso l'output come in una classica rete feed-forward, ma anche lo "storico", tramite lo stato precedente del nodo stesso. Pertanto, in ogni momento t, lo stato del nodo può essere espresso come

$$h_t = f_W(h_{t-1}, x_t) \tag{7}$$

L'applicazione ricorrente dello stato precedente permette dunque di tenere in memoria la storia pregressa del nodo. Si noti che la funzione f_w e i parametri $\mathbf{W}$ del nodo sono gli stessi ad ogni iterazione per permettere di conservare l'informazione, evitano così di "confinarla" a una specifica posizione all'interno della sequenza. La rete

sta perciò a ogni iterazione, compiendo in realtà due operazioni differenti: il cacolo dell'output e l'aggiornamento dello stato interno h_t, che verrà utilizzato nell'iterazione successiva per calcolare h_{t+1}.

Nello specifico la funzione di aggiornamento di h_t prende la forma

$$h_t = \sigma(\mathbf{W_{hh}} h_{t-1} + \mathbf{W_{xh}} x_t) \tag{8}$$

con x_t t-esimo elemento della sequenza di input, $\mathbf{W_{hh}}$ e $\mathbf{W_{xh}}$ le matrici dei pesi e σ funzione di attivazione, spesso tangente iperbolica. Dopo l'aggiornamento dello stato interno, l'output $\hat{y}_t$ viene calcolato con un'opportuna matrice di pesi $\mathbf{W_{hy}}$

$$\hat{y}_t = \mathbf{W_{hy}} h_t \tag{9}$$

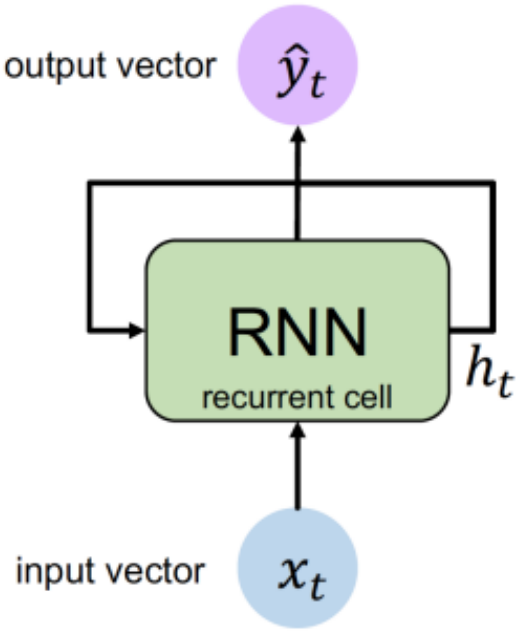

Fig 1. Schema "a loop" di una Rete Neurale Ricorrente.

3.2 Back Propagation Through Time

Il processo di apprendimento lungo la sequenza richiede di calcolare una funzione di perdita che possa essere funzionale lungo l'intera sequenza e non solo allo stato attuale della rete. Essa pertanto viene calcolata a ogni singolo passo della sequenza:

$$L_t = L(h_t, x_t). \tag{10}$$

Le perdite così calcolate vengono poi aggregate in una funzione di perdita totale $L = f(L_1, ..., L_T)$ a sequenza terminata. La perdita totale viene dunque propagata a tutti gli step della rete e ogni perdita L_t viene propagata a tutti gli stati precedenti secondo lo schema mostrato in figura (2).

Tale propagazione presenta però una difficoltà: come visivamente intuibile da figura (2), il calcolo dei gradienti di stati via via più vicini allo stato iniziale h_0 implica applicazioni ripetute della matrice dei pesi $\mathbf{W_{hh}}$. Ciò può portare a due differenti degenerazioni del gradiente:

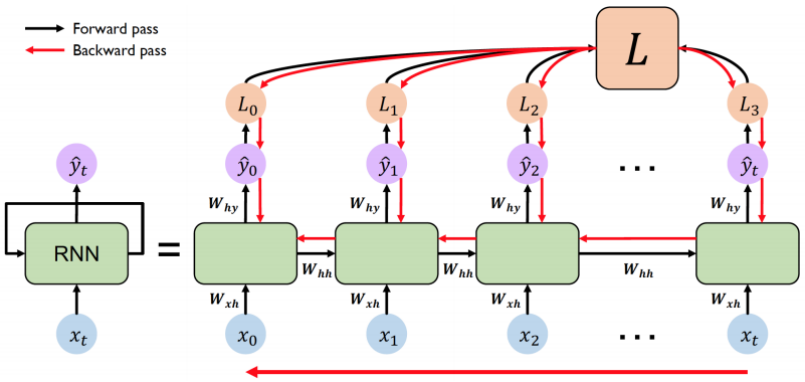

Fig 2. Propagazione delle funzioni di perdita lungo gli step della rete.

- Se la matrice presenta molti valori $w_{ij} > 1$ il gradiente tende a esplodere verso valori sempre più grandi, rendendo rapidamente impraticabile la minimizzazione della funzione di perdita. Tale scenario prende il nome di "exploding gradient". E' possibile contenere tale eventualità tramite una riscalatura del gradiente quando i valori dello stesso superano una determinata soglia.

- Se la matrice presenta molti valori $w_{ij} < 1$ il valore del gradiente tende rapidamente a 0, rendendo ancora una volta impraticabile l'ottimizzazione. Tale scenario prende il nome di "vanishing gradient" e causa una perdita delle relazioni di dipendenza a lungo termine. La ricerca di una soluzione al problema ha portato a diversi tipi di architetture e accorgimenti, di cui esempio notevole sono le reti ricorrenti di tipo Long-Short Term Memory.

3.3 Reti Long-Short Term Memory

Vengono descritte qui in generale le reti LSTM in quanto l'architettura GRU di cui Kuehne, Richard e Gall hanno, nell'esperimento, fatto uso è nel loro lavoro definita come "una versione semplificata delle LSTM che mostra prestazioni paragonabili alla classica" ispirata al lavoro di Jozefowicz et al. in [4]. Si è deciso quindi, vista l'importanza e la frequenza d'uso di questo tipo di architettura, di dedicare un paragrafo all'analisi della loro versione più completa.

Le reti ricorrenti Long-Short Term Memory sono, come già accennato, una possibile soluzione al problema del vanishing gradient, e il *de facto* standard nella costruzione di reti ricorrenti robuste. La particolarità delle reti LSTM è duplice. Innanzitutto ogni nodo presenta, oltre al già disusso stato h_t, un secondo stato C_t, detto "cell state", di aggiornamento più semplice e descritto in seguito. Seconda specifica è quella di inserire, nella struttura dei singoli nodi della rete ricorrente, dei *gates* che filtrano il loro ingresso in differenti maniere prima di propagarlo. Si distinguono quattro differenti tipi di operazione all'interno della "cell":

- Un gate *forget* che si occupa di far "scordare" al cell state c_t le informazioni che è opportuno dimenticare in quanto non più informativamente rilevanti all'interno dellla sequenza. Tale gate può anche resettare del tutto il cell state, se necessario. La forma matematica del gate è

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{11}$$

  con σ funzione sigmoidale. L'output è un vettore della stessa lunghezza di C_t di valori compresi tra 0 ("forget") e 1 ("remember"). Tale vettore viene moltiplicato

per il precedente stato c_{t-1}, causando la conservazione degli elementi che si trovino nelle stesse posizioni dei valori unitari.

- Un gate *input* il cui compito è di stabilire quali nuove informazioni vadano memorizzate nel cell state. Tale gate è composto di due parti, di cui una rappresenta l'elemento "candidato" a venir inserito nel cell state al passo t e l'altra un vettore analogo a quello presente nel *forget* gate per stabilire quali valori dello stato aggiornare con il candidato. Matematicamente, il candidato ha forma

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{12}$$

mentre l'updater si presenta analogo al gate precedente

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{13}$$

con σ ancora una volta funzione sigmoidale e output e funzioni analoghe.

- Un gate *update* detto anche gate di *store* che unisce tutte le informazioni fin qui prodotte

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{14}$$

Com'è visibile dall'equazione (14) è questo gate a produrre l'effettivo "cell state" dello step t-esimo che verrà utilizzato come input nell'iterazione successiva.

- Un gate *output* con la funzione di produrre l'effettivo output h_t relativo al passo t-esimo. Esso è una versione filtrata dello stato C_t e viene generato dal gate in due parti: viene prima selezionata tramite una sigmoide quali parti del cell state riportare in input, dopodiché viene applicata una tangente iperbolica per riportare tali valori nel range tra -1 e 1

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{15}$$

$$h_t = o_t * tanh(C_t) \tag{16}$$

Scorporando dunque lo stato interno h_t dal cell state C_t il problema della sparizione del gradiente viene evitato: come si osserva in (14), il passaggio da C_{t-1} a C_t consiste solamente di operazioni puntuali di moltiplicazione e addizione. Non vi sono moltiplicatori matriciali né funzioni di attivazione, proprie dello stato interno, pertanto la propagazione all'indietro non risente di rimpicciolimenti ricorrenti nel calcolo del gradiente.

4 Specificità dell'esperimento

L'uso che i ricercatori hanno fatto degli strumenti sopra descritti è, come già accennato, gerarchico: una prima Rete Ricorrente più "grezza" è utilizzata per dare alla sequenza video un'ipotetica partizione in base alla lista ordinata delle azioni fornite in input, seguita da un addestramento del modello markoviano nascosto per allineare i frames con le singole sottoazioni. Dopodiché ognuna di tali azioni viene suddivisa in sottoazioni tramite l'uso di Reti GRU (la sovracitata versione semplificata di una rete LSTM) più "a grana fine" e l'HMM utilizzato per "rilocalizzare" le sottoazioni secondo un'ipotesi di inizio e fine. Durante tale processo, tuttavia, gli autori hanno osservato una tendenza del sistema a focalizzarsi solo su alcuni stati (ossia alcune sottoazioni) per tempi molto lunghi, riservando a molte altre solo uno o due frame. Da queste osservazioni nasce dunque l'ultima particolarità nell'architettura dell'esperimento, il *length prior*.

4.1 Length Prior

Si tratta di un fattore di regolarizzazione della lunghezza degli stati finalizzato a "equalizzare" il più possibile tali lunghezze. L'obiettivo di tale modulo è, come sopra accennato, quello di evitare che il predittore assegni solo pochi frame alla maggior parte delle sottoazioni e a pochissime altre la maggior parte dei frame del segmento di video che compongono l'azione che le aggrega. Ciò viene ottenuto con una funzione in grado di preimare la permanenza in uno stato quando la sua lunghezza è inferiore alla media e punirla man mano che la sua lunghezza aumenta. La lunghezza di uno stato è definita ricorsivamente: se la sottoazione $s_t \neq s_{t-1}$ allora la lunghezza è $l_t(s_t) = 1$, altrimenti

$$l_t(s_t) = l_{t-1}(s_t) + 1 \tag{17}$$

Definita la media di ogni stato come

$$len(s) = \frac{numero\,di\,frames\,allineati\,a\,s}{numero\,di\,istanze\,di\,s} \tag{18}$$

il length prior è dunque una funzione che decade allontanandosi da tale media. I tre autori propongono un esempio gaussiano

$$\tilde{p}(l_t(s_t)|s_t) = e^{-\frac{(l_t(s_t)-\mu)^2}{\sigma^2}} \tag{19}$$

con media nulla e deviazione standard $\sigma = len(s_t)$.

5 Conclusione

L'utilizzo di una struttura gerarchica con una rete neurale "a grana grossa" e altre più "raffinate" per raffinare il lavoro della prima rete si dimostra molto funzionale a sperimentazioni su differenti tipi di apprendimento e modifiche mirate alla struttura. Gli autori ne hanno fatto uso per osservare le differenze tra apprendimento debolmente supervisionato a diversi gradi di "ricchezza" dei dati e per identificare con precisione differenti aspetti del problema trattato e della struttura, permettendo l'utilizzo di soluzioni mirate come l'implementazione del length prior. L'architettura ha permesso di affiancare risultati prestazionali molto diversi tra loro a fronte di piccole differenze nella struttura dei dati in input, evidenziandone la rilevanza e identificando "stepping stones" fondamentali nella risoluzione del problema di automatizzazione del lavoro di etichettatura di azioni in sequenze video, un campo di innumerevoli applicazioni e fondamentale importanza per la ricerca presente e futura.

References

1. Koehne, Richard e Gall, "A Hybrid RNN-HMM Approach for Weakly Supervised Temporal Action Segmentation".

2. R. Lanzarotti, "Modelli Deep per sequenze temporali".

3. G. Boccignone, "Processi stocastici (3): Processi Markoviani" dal corso di Modelli di Computazione Affettiva.

4. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures"