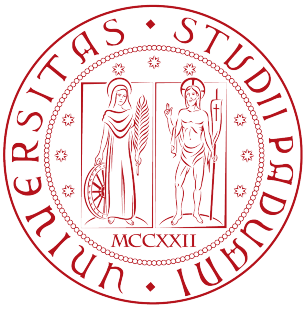
Università degli studi di Padova
Dipartimento di Scienze Statistiche

Corso di Laurea Magistrale in
Scienze Statistiche



RELAZIONE FINALE

Topic Modeling, dietro le quinte: modelli grafici diretti e indiretti

Relatore prof. Livio Finos
Dipartimento di Scienze Statistiche

Laureando Ferraccioli Federico
Matricola N. 1092677

Anno Accademico 2015/2016

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

Nel corso degli anni la conoscenza collettiva sta subendo una trasformazione, digitalizzandosi in forma di notizie, blog, pagine web, articoli scientifici, libri, immagini, suoni, video e social network, e diventa sempre più difficile districarsi in questo oceano di informazioni alla ricerca degli argomenti di nostro interesse. Sono necessari nuovi strumenti informatici che ci aiutino ad organizzare, cercare e comprendere questa enorme quantità di informazioni. Ogni giorno digitiamo parole in un motore di ricerca nella speranza di trovare articoli e informazioni coerenti, e magari attraverso un *link* ampliare la nostra conoscenza. Questo tipo di ricerca attraverso parole chiave è certamente potente, ma sembra manchi ancora qualcosa... Immaginiamo di poter cercare ed esplorare questi documenti attraverso i temi di cui trattano: potremmo generalizzare o rendere la nostra ricerca più specifica con l'aiuto di macro e micro argomenti; potremmo addirittura esaminare il cambiamento di questi argomenti nel tempo e analizzare le relazioni esistenti tra loro. Invece di trovare documenti legati esclusivamente a qualche parola chiave, potremmo decidere per prima cosa l'argomento che ci interessa, e quindi cercare gli articoli che ne trattano. Consideriamo ad esempio la storia completa del settimanale *L'Espresso*: i macro argomenti potrebbero corrispondere alle sezioni della rivista quali politica, economia, sport; addentrandoci nella ricerca i micro argomenti potrebbero corrispondere ad argomenti di nostro interesse, quali ad esempio l'economia nazionale o estera con particolare attenzione alla crisi monetaria. Potremmo a questo punto pensare di navigare nel tempo e scoprire i cambiamenti avvenuti in questo tema, delineando l'andamento economico degli ultimi 50 anni. La struttura e le relazioni tematiche dei documenti dovrebbero essere il nuovo metodo per gestire ed analizzare le informazioni, ma purtroppo non è così; con l'aumento esponenziale di testi ed articoli disponibili *on-line* non abbiamo le capacità umane di leggere, studiare e catalogare questa collezione in modo da poterne usufruire come appena descritto.

Negli ultimi anni ricercatori e statistici hanno iniziato a definire e sviluppare una nuova gamma di strumenti. I *topic model* sono modelli probabilistici che attraverso l'analisi delle parole caratterizzanti i testi, individuano gli argomenti trattati, le loro connessioni e come cambiano nel tempo; questo tipo di modelli non necessita di nessuna annotazione manuale, i temi emergono direttamente dall'analisi permettendo di archiviare e classificare ciò che sarebbe umanamente impossibile.

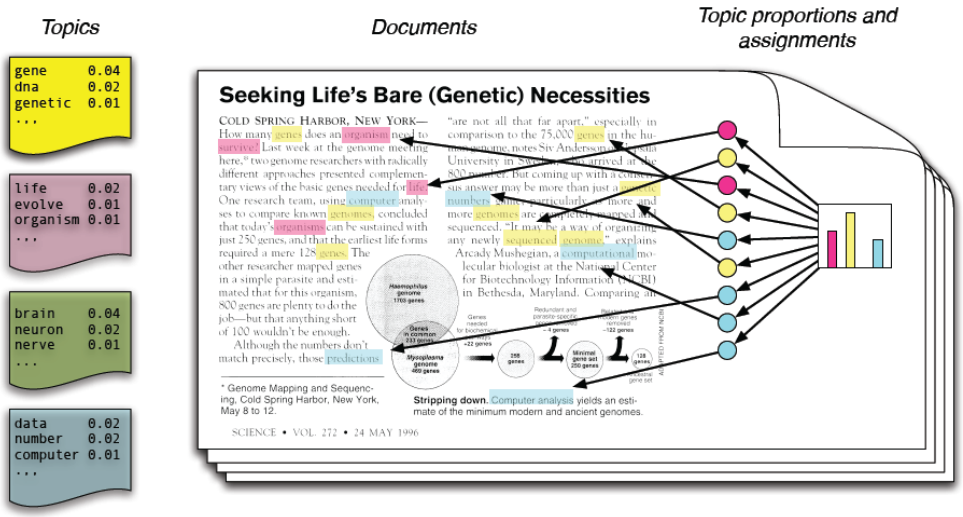


Figura 1: Cos'è un topic model

In questa tesi verranno presentati e confrontati alcuni tra i modelli più utilizzati in questo ambito di ricerca, nello specifico *Latent Dirichlet Allocation* e *Replicated Softmax*. Particolare attenzione sarà rivolta alla struttura probabilistica e specialmente alla modellazione grafica sottostante, definendo le dipendenze statistiche tra le variabili in gioco e intervenendo con opportune modifiche laddove sono presenti criticità. Verranno inoltre analizzate in dettaglio numerose procedure di stima, allo scopo di proporre una visione d'insieme, anche se minima, di questo fondamentale settore della ricerca statistica.

Capitolo 1

Analisi preliminari

1.1 Dataset NIPS

I dati utilizzati in questa tesi comprendono i testi degli articoli presentati al NIPS (Conference and Workshop on Neural Information Processing Systems) nel 2015; si tratta del più importante meeting riguardante machine learning e computational neuroscience. Gli argomenti trattati, oltre a quelli già accennati, comprendono scienze cognitive, psicologia, computer vision, linguistica (statistica) e teoria dell'informazione.

Il dataset è composto da 403 articoli di diversa lunghezza, di cui sono noti il titolo, il sommario e il testo completo; in particolare i testi sono composti da più di 57.000 termini diversi. Si tratta di un dataset accessibile pubblicato da Hamner, (2015), scaricabile dal sito `https://www.kaggle.com/benhamner/nips-2015-papers`.

1.2 Preprocessamento

Prima di procedere alla modellazione dei testi è necessaria una fase di preparazione e pulizia dei dati; le operazioni più importanti sono lo *stemming* e la rimozione delle *stopword* e dei simboli poco comuni. Con la parola *stemming* si intende quella procedura di riduzione della forma flessa di una parola alla sua forma radice; è una fase molto delicata, in quanto si rischia di ridurre due parole con significati completamente diversi alla stessa radice, ed è necessario dunque scegliere accuratamente l'algoritmo da utilizzare. Per mantenere la chiarezza

del significato si è deciso di non attuare questa procedura nelle analisi descrittive; più avanti nel testo sarà specificato quando verrà utilizzata. La seconda operazione riguarda l'eliminazione delle *stopword*, cioè quell'insieme di termini utilizzati comunemente nel linguaggio e presenti in tutti i testi, quali articoli, congiunzioni, pronomi *etc...* Questi non aggiungono nessuna informazione riguardo l'argomento trattato in un testo, anzi porterebbero notevoli problemi di stima vista l'alta frequenza con cui vengono utilizzati; per ultimo, dato che i testi sono di ambito scientifico, sono stati eliminati i simboli matematici delle formule presenti e i numeri.

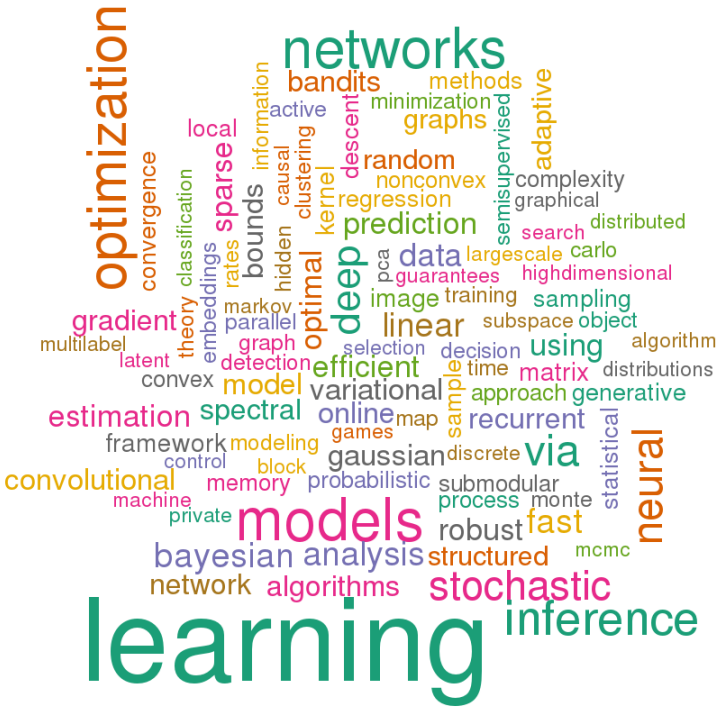


Figura 1.1: Visualizzazione grafica delle parole più importanti ("NIPS 2015 Papers" Kaggle Competition)

1.3 Analisi descrittive

Una prima caratterizzazione dei documenti presi in esame è data dalla lunghezza: è possibile definirla come numero di parole componenti il testo o numero di caratteri totali. In figura 1.2 sono rappresentati gli istogrammi corrispondenti, rispettivamente per il numero di parole e per il numero di caratteri: trattandosi di pubblicazioni scientifiche, si nota immediatamente come le lunghezze siano molto simili, in particolare tra i 3000 e i 4500 termini; discorso analogo per il numero di caratteri, compreso tra i 30000 e i 40000.

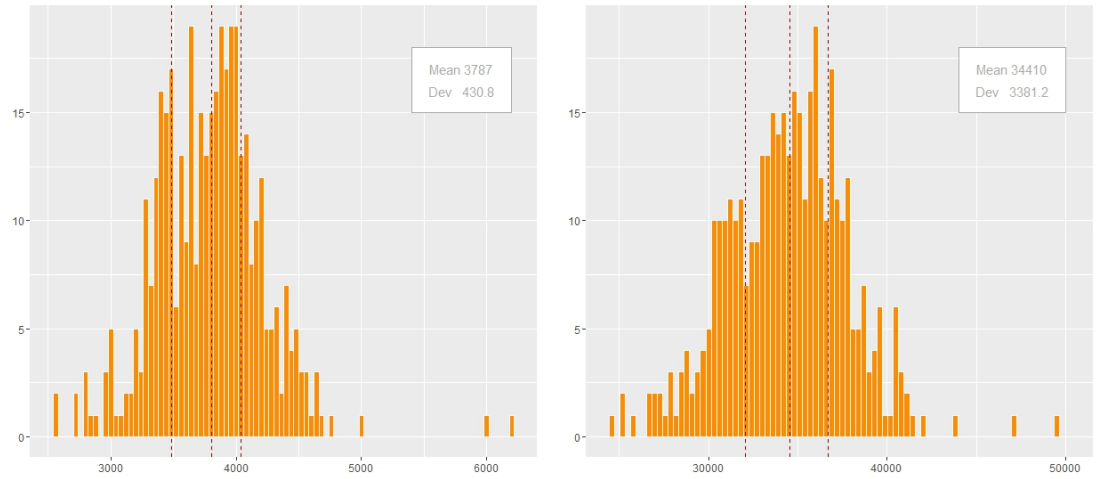


Figura 1.2: L'istogramma di sinistra è relativo ai conteggi delle parole per ogni documento; l'istogramma di destra ai conteggi del numero di caratteri.

Un approccio semplice per avere una panoramica degli argomenti trattati negli articoli consiste nel conteggio delle parole e dei bigrammi (composizione di due termini) più frequenti. La *wordcloud* in figura 1.1 mostra alcune tra le parole più frequenti, con la dimensione del carattere proporzionale alla frequenza. Si nota immediatamente la predominanza di argomenti quali *reti neurali* e *deep learning*; negli ultimi anni l'ascesa della potenza di calcolo delle GPU ha permesso di riportare in luce questo tipo di modelli, fondamentali nei rami di ricerca riguardanti il riconoscimento automatico di immagini, testi o più generalmente segnali digitali. Altro argomento molto in voga negli ultimi anni è la statistica bayesiana, si notano infatti termini quali *bayesian analysis*, *variational inference* e *markov models*. In tabella 1.1 (a fine capitolo) sono presentati per esteso i conteggi delle 100 parole e dei 50 bigrammi più frequenti; le considerazioni riguardo

gli argomenti trattati sono le stesse fatte per la *wordcloud*, con l'aggiunta di parole quali *abstract*, *arxiv preprint* e *conference machine* che ovviamente sono molto frequenti essendo articoli scientifici (eliminate nella *wordcloud* perchè poco interessanti).

Uno strumento fondamentale per qualsiasi analisi testuale è la *Document Term Matrix*, in cui ogni riga corrisponde ad un testo, ed ogni colonna ad una parola nel vocabolario. Sia N il numero di documenti, M il numero di parole del dizionario (il totale delle parole uniche presenti in tutti i testi): la matrice avrà dimensioni $M \times N$, e le celle w_{ij} saranno i conteggi corrispondenti alle volte in cui la parola j è comparsa nel documento i. La figura 1.3 è una rappresentazione grafica della *Term Document Matrix* (la matrice trasposta della *Document Term Matrix* appena descritta), in cui l'intensità del colore delle celle è proporzionale alla frequenza delle parole. Data la diversità nelle frequenze, si è scelto di utilizzare la trasformata logaritmica; inoltre, per ovvie limitazioni di spazio, sono stati selezionati solo i termini che sembravano di maggiore interesse.

1.4 Riduzione della dimensionalità

I dati testuali a disposizione, nonostante le operazioni di pulizia, risultano ancora difficilmente interpretabili. La difficoltà più grande risiede nella dimensionalità: il numero di termini utilizzati è decisamente troppo vasto per poterne carpire qualche informazione. In questo capitolo verranno presentate e confrontate due tecniche di riduzione della dimensionalità, allo scopo di individuare eventuali *cluster* di termini utilizzati frequentemente insieme.

1.4.1 Componenti principali

Un metodo frequentemente utilizzato è quello delle componenti principali: una trasformazione ortogonale lineare del sistema di coordinate basata sulla varianza delle osservazioni. Più formalmente, le componenti principali di un insieme di dati in $\mathbb{R}^p$ forniscono una sequenza dei migliori approssimatori lineari, per tutti i ranghi $q \leq p$ [J. Friedman et al., 2001]. Sia $\mathbf{x}_1, \ldots, \mathbf{x}_N$ un insieme di unità, il modello lineare di rango q che le rappresenta è definito

$$f(\lambda) = \mu + \mathbf{V}_q \lambda \tag{1.1}$$

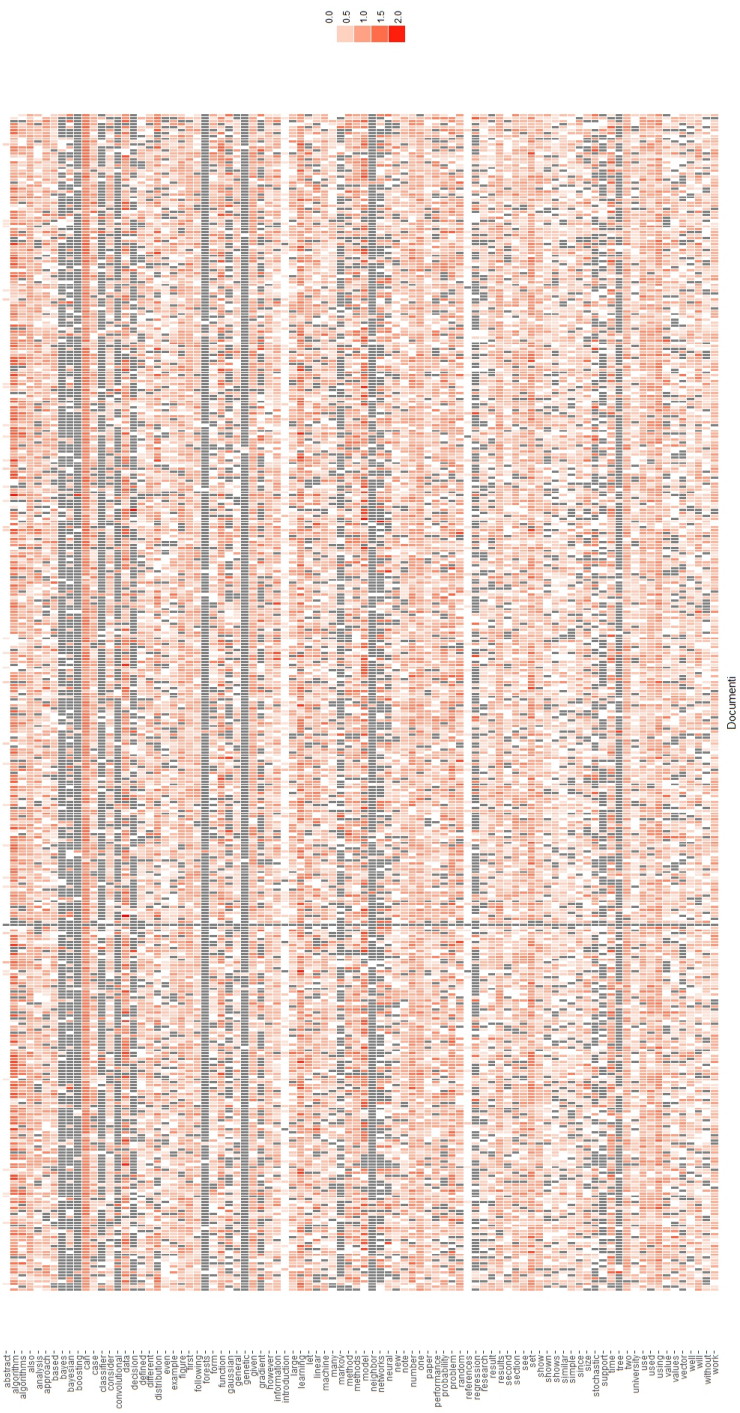


Figura 1.3: Visualizzazione grafica di una porzione di Term Document Matrix: ogni riga corrisponde ad un termine del vocabolario, ogni colonna un documento. La gradazione di colore è proporzionale alla frequenza (il grigio corrisponde ad un conteggio pari a 0). Le parole sono state selezionate tra le più frequenti e le più significative dal punto di vista statistico.

con μ vettore di posizione in $\mathbb{R}^p$, $\mathbf{V}_q$ matrice $p \times q$ a colonne ortogonali e λ vettore q-dimensionale di parametri. La matrice $\mathbf{V}_q$ è ottenibile da

$$\min_{\mathbf{V}_q} \sum_{i=1}^{N} ||(x_i - \bar{x}) - \mathbf{V}_q \mathbf{V}_q^T (x_i - \bar{x})||^2 \tag{1.2}$$

(per convenienza si può assumere $\bar{x} = 0$). La matrice $\mathbf{H}_q = \mathbf{V}_q \mathbf{V}_q^T$ è una matrice di proiezione $p \times p$, che mappa ogni elemento $\mathbf{x}_i$ nel sottospazio delle colonne di $\mathbf{V}_q$, definendo la proiezione $\mathbf{H}_q x_i$. La stessa soluzione è ottenibile attraverso la scomposizione in valori singolari di $\mathbf{X}$:

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^T. \tag{1.3}$$

Per ogni rango q, la soluzione per $\mathbf{V}_q$ consiste nelle prime q colonne di $\mathbf{V}$; le colonne di $\mathbf{UD}$ sono definite componenti principali di $\mathbf{X}$.

In figura 1.5 sono rappresentate le prime due componenti principali; si nota immediatamente come le distanze nel sottospazio generato collassino nell'origine, mappando la maggior parte delle unità all'interno di una circonferenza. Non è possibile intravedere nessun *cluster* di termini; solo alcune delle parole, tra le più frequenti, sono rappresentate come punti molto lontani dall'origine.

1.4.2 t-SNE

Un modello più efficiente per la riduzione della dimenzionalità è il t-SNE (*t-distributed stochastic neighbor embedding*), ideato da Maaten e G. Hinton, (2008); lo scopo è sempre quello di definire un omeomorfismo $g : \mathbb{R}^n \longrightarrow \mathbb{R}^m$, con $m \ll n$ (in particolare viene spesso utilizzato $m = 2, 3$).
Questa tecnica non-lineare consiste in due fasi:

- Si definisce una distribuzione sulle coppie di unità n-dimensionali, assegnando probabilità alta alle unità più simili, e viceversa probabilità bassa alle unità più diverse;
- Si definisce in modo analogo una distribuzione sui punti dello spazio m-dimensionale, e si minimizza la distanza di Kullback-Leibler tra le due distribuzioni.

Entrando più in dettaglio, siano $\mathbf{x}_1, \ldots, \mathbf{x}_N$ un insieme di unità N dimensionali. La somiglianza tra x_i e x_j è definita come la probabilità condizionata,

$p_{j|i}$, che x_i abbia x_j come punto più vicino, sapendo che x_j è stato estratto con probabilità proporzionale alla densità di una variabile casuale Gaussiana centrata in x_i. Per punti vicini, la probabilità $p_{j|i}$ è relativamente alta, mentre per punti distanti la probabilità è infinitesimale (assumendo di avere valori di σ_i ragionevoli). Matematicamente, la probabilità condizionata $p_{j|i}$ è definita come

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)} \tag{1.4}$$

dove σ_i è la varianza della variabile casuale Gaussiana centrata in x_i. È possibile dunque specificare la funzione di probabilità congiunta di tutto lo spazio come

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N} \tag{1.5}$$

(ponendo $p_{i|i} = 0$). Equivalentemente, siano $\mathbf{y}_1, \ldots, \mathbf{y}_M$ un insieme di unità M-dimensionali; si definisce la somiglianza tra le unità y_i e y_j come probabilità congiunta

$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k\neq l}(1 + \|\mathbf{y}_k - \mathbf{y}_l\|^2)^{-1}} \tag{1.6}$$

Importante notare la differenza nella definizione della probabilità nei due spazi: la prima fa riferimento ad una densità gaussiana, la seconda da una t di Student con un grado di libertà. Questa particolare scelta permette di evitare il cosiddetto *crowding*, cioè la tendenza dei punti dello spazio m-dimensionale ad accorparsi in una sfera; le code più pensanti della t di Student permettono di avere più repulsione tra punti maggiormente distanti nello spazio n-dimensionale. La scelta è inoltre giustificata dal fatto che la t di Student può essere scritta come mistura infinita di densità Gaussiane, con il vantaggio computazionale dato dalla mancanza dell'esponenziale.

Avendo definito le due funzioni di somiglianza nei due spazi, è necessario che queste ultime definiscano le stesse distribuzioni di probabilità. Una scelta naturale per misurare la somiglianza in ambito probabilistico è la distanza di Kullback-Leibler. La funzione da minimizzare, rispetto ai punti y_i, risulta dunque

$$C(\mathcal{E}) = KL(P||Q) = \sum_{i\neq j} p_{ij}\log\frac{p_{ij}}{q_{ij}} \tag{1.7}$$

con $\mathcal{E}$ la mappatura definita dall'algoritmo. Il problema di ottimizzazione è non triviale e gli algoritmi standard di discesa del gradiente non sono efficienti. Una possibile soluzione utilizza l'approssimazione di Barnes-Hut [Van Der Maaten, 2014], un'approssimazione del gradiente definita con una procedura iterativa basata su alberi. Partendo dalla funzione di perdita (1.7), è possibile suddividerne il gradiente in due parti:

$$\frac{\partial C}{\partial \mathbf{y}_i} = 4(F_{attr} - F_{rep}) = 4\left(\sum_{j \neq i} p_{ij} q_{ij} Z(\mathbf{y}_i - \mathbf{y}_j) - \sum_{j \neq i} q_{ij}^2 Z(\mathbf{y}_i - \mathbf{y}_j)\right) \tag{1.8}$$

con $z = \sum_{i \neq j}(1 + ||\mathbf{y}_k - \mathbf{y}_l||^2)^{-1}$. La parte sinistra della somma, F_{attr} è computazionalmente efficiente, è sufficiente definire una soglia e porre $p_{j|i} = 0$ se la distanza tra i punti è sufficientemente grande; il calcolo della parte destra della somma, F_{rep} rimane invece nell'ordine di $\mathcal{O}(N^2)$. L'approssimazione di Barnes-Hut si basa sulla seguente considerazione: siano $\mathbf{y}_i$, $\mathbf{y}_j$ e $\mathbf{y}_k$ tali che $||\mathbf{y}_i - \mathbf{y}_j|| \approx ||\mathbf{y}_i - \mathbf{y}_k|| \gg ||\mathbf{y}_j - \mathbf{y}_k||$, allora il contributo di $\mathbf{y}_j$ e $\mathbf{y}_k$ a F_{rep} sarà approssimativamente uguale. Con questa ipotesi è possibile definire la procedura seguente:

1. Si costruisce un *quadtree* sulla mappatura corrente;
2. Si attua una ricerca (di tipo *depth-first*) nell'albero;
3. Per ogni nodo si decide se la cella corrispondente può essere definita come un'approssimazione ragionevole del contributo a F_{rep} di tutti i punti all'interno della cella.

Un *quadtree* è una struttura ad albero nella quale ogni nodo rappresenta una cella rettangolare con particolare centro, altezza e larghezza. I nodi non foglia hanno quattro nodi figli, corrispondenti a quattro celle che suddividono la stessa in quattro parti; i nodi foglia rappresentano le celle che contengono al più un punto (figura 1.4).

L'approssimazione di F_{rep} è definita come $-N_{cell} q_{i,cell}^2 Z(\mathbf{y}_i - \mathbf{y}_{cell})$, con N_{cell} il numero di punti all'interno della cella, $\mathbf{y}_{cell}$ il centro di massa della cella e $q_{i,cell}^2 Z = (1 + ||\mathbf{y}_i - \mathbf{y}_{cell}||^2)^{-1}$. La condizione per decidere se la cella è una buona approssimazione si basa sulla seguente disuguaglianza:

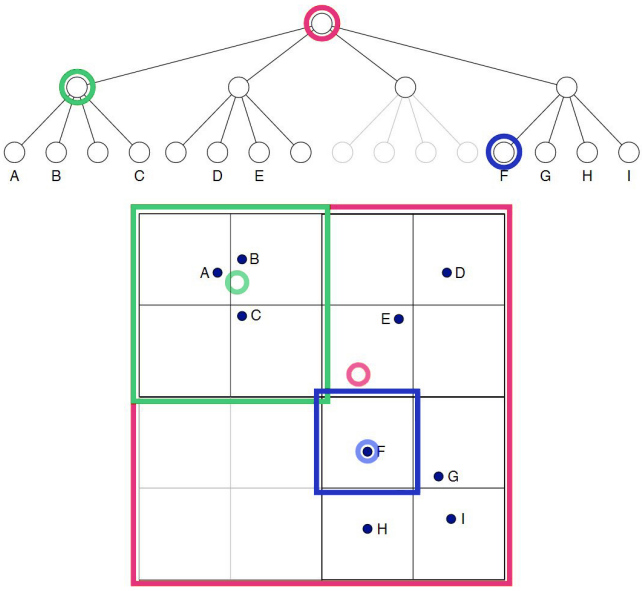


Figura 1.4: La struttura ad albero di un *quadtree*: ogni nodo ha quattro nodi figli, che corrispondono alla suddivisione in quattro celle dello stesso. In basso è rappresentata la ripartizione dello spazio indotta dall'albero: i colori sono utilizzati in modo da sottolineare la corrispondenza tra le due rappresentazioni. Ogni nodo nel grafo corrisponde ad una cella, e nodi più profondi corrispondono a celle più piccole. Le circonferenze disegnate all'interno delle celle corrispondono ai centri di massa, i punti blu corrispondono alle osservazioni; si nota immediatamente che i nodi foglia contengono una sola osservazione. La parte opaca dell'albero corrisponde ad una non ancora creata, non sono infatti presenti punti in quella porzione di spazio (quadrante in basso a sinistra).

$$\frac{r_{cell}}{||\mathbf{y}_i - \mathbf{y}_{cell}||^2} < \theta \tag{1.9}$$

dove r_{cell} rappresenta la lunghezza della diagonale della cella presa in considerazione e θ il parametro che regola la soglia (e di conseguenza il compromesso tra velocità e accuratezza dell'algoritmo). Nel caso in cui $\theta = 0$ vengono calcolate le distanze per tutte le coppie e l'approssimazione di Barnes-Hut si riduce al calcolo standard del gradiente del t-SNE.

In figura 1.6 si nota immediadiatamente un notevole miglioramento rispetto alle componenti principali [1]. Il problema di *crowding* che affligge i metodi linea-

[1] Le analisi sono state svolte in ambiente `R` utilizzando i pacchetti `prcomp` per l'analisi delle componenti principali e `Rtsne` per t-SNE.

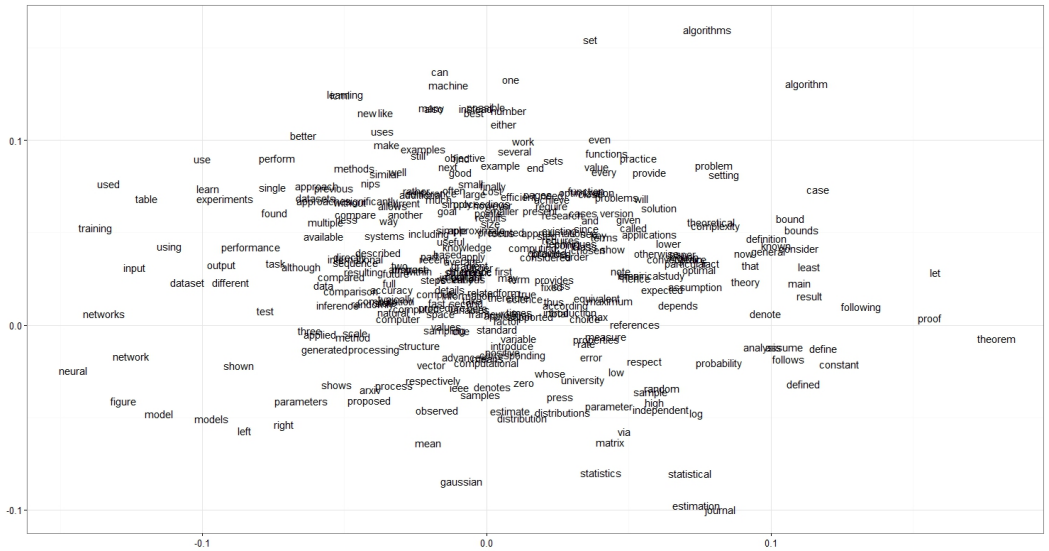


Figura 1.5: Mappatura definita dalle prime due componenti principali

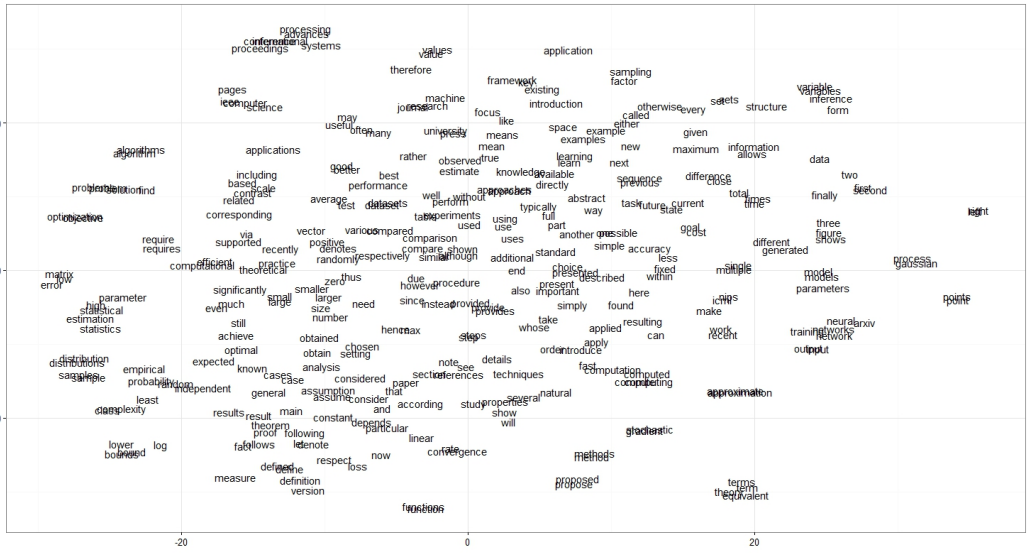


Figura 1.6: Mappatura definita dall'algoritmo t-SNE

(Per chiarezza grafica sono, stati rappresentati solo i 300 termini più frequenti)

ri di riduzione della dimensionalità non è più presente. È possibile notare raggruppamenti di parole significativi: *stochastic* e *gradient* sono molto vicini (algoritmo *stochastic gradient descent* utilizzato nel seguito della tesi), i termini *future*, *goal* e *current*, *state* che denotano possibili sviluppi rispetto agli attuali modelli; anche *international*, *processing* e *conference* sono posizionati insieme (il nome della conferenza, NIPS). Questi rappresentano solo alcuni esempi, ma già da qui sono identificabili possibili macro argomenti trattati nei testi a disposizione, caratterizzati ognuno da un particolare vocabolario di termini.

Tabella 1.1: Parole e bigrammi più frequenti ordinati per conteggio

	Parole	Conteggi		Parole	Conteggi		Bigrammi	Conteggi
1	abstract	402	51	order	333	1	machine learning	1450
2	using	401	52	second	332	2	neural networks	787
3	references	399	53	probability	331	3	lower bound	564
4	set	398	54	shown	330	4	low rank	520
5	given	397	55	result	329	5	gradient descent	475
6	introduction	397	56	values	329	6	high dimensional	461
7	results	394	57	simple	328	7	neural network	427
8	learning	393	58	distribution	328	8	stochastic gradient	427
9	number	392	59	method	327	9	large scale	423
10	use	386	60	high	326	10	log likelihood	413
11	based	386	61	shows	324	11	upper bound	411
12	used	386	62	small	323	12	monte carlo	388
13	work	379	63	possible	322	13	arxiv preprint	387
14	following	377	64	log	319	14	log log	382
15	large	372	65	models	318	15	variational inference	359
16	information	370	66	standard	316	16	preprint arxiv	358
17	case	369	67	follows	316	17	state art	357
18	function	369	68	parameters	315	18	conference machine	349
19	different	369	69	journal	315	19	objective function	333
20	problem	367	70	experiments	314	20	artificial intelligence	323
21	section	366	71	proposed	314	21	optimization problem	318
22	non	366	72	step	310	22	training set	314
23	algorithm	365	73	define	309	23	sample complexity	309
24	data	365	74	systems	308	24	learning research	304
25	approach	364	75	previous	308	25	logistic regression	298
26	paper	362	76	problems	308	26	journal machine	296
27	random	361	77	assume	307	27	matrix completion	294
28	note	361	78	setting	307	28	convergence rate	290
29	time	360	79	related	306	29	data sets	286
30	machine	357	80	efficient	306	30	real world	280
31	similar	357	81	best	306	31	convex optimization	278
32	consider	353	82	corresponding	304	32	loss function	270
33	example	351	83	neural	304	33	random variables	268
34	methods	351	84	known	303	34	ieee transactions	267
35	does	350	85	particular	303	35	training data	264
36	analysis	349	86	parameter	301	36	learning algorithm	264
37	model	349	87	terms	301	37	markov chain	262
38	performance	349	88	space	300	38	ground truth	261
39	figure	348	89	better	298	39	active learning	251
40	algorithms	347	90	pages	298	40	step size	250
41	general	347	91	error	298	41	worst case	248
42	new	341	92	fixed	298	42	lower bounds	240
43	university	339	93	single	297	43	data points	240
44	research	338	94	vector	297	44	computer science	239
45	defined	337	95	denote	296	45	related work	233
46	let	336	96	provide	295	46	test set	231
47	size	335	97	applied	295	47	figure shows	227
48	form	334	98	point	294	48	high probability	225
49	linear	334	99	theory	293	49	graphical models	223
50	value	333	100	obtain	290	50	data set	220

Capitolo 2

Latent Dirichlet Allocation (LDA)

In questo capitolo verranno introdotti i concetti di Bayesian Network e di modello gerarchico. Lo scopo è quello di affrontare il problema della modellazione di dati testuali con un approccio bayesiano, presentando in particolare quelle che risultano essere le tecniche più diffuse in questo contesto.

2.1 Bayesian Network

Una rete bayesiana (Bayesian Network) è un modello probabilistico che rappresenta un insieme di variabili e le corrispondenti dipendenze condizionali attraverso un grafo aciclico diretto (DAG). Più formalmente, i nodi (vertici) del grafo corrispondono alle variabili casuali, che possono essere quantità osservabili, variabili latenti o parametri ignoti; gli archi corrispondono alle dipendenze condizionate, in particolare nodi che non sono connessi rappresentano variabili casuali condizionatamente indipendenti le une dalle altre [Koller e N. Friedman, 2009]. Ad ogni nodo è associata una funzione di probabilità avente come dominio i valori delle variabili dei nodi "padri" (i nodi collegati a v da un singolo arco diretto) e come codominio le probabilità delle variabili rappresentate dal nodo.

2.1.1 Fattorizzazione

Una delle possibili definizioni formali per una rete bayesiana è quella basata sul concetto di fattorizzazione. Sia $G = (V, E)$ un grafo diretto aciclico con vertici V e archi E; dato un insieme di variabili casuali $X = (X_v)_{v \in V}$, sia $P(X = x)$

la probabilità di una particolare configurazione x in X.
Si può definire X una rete bayesiana rispetto a G se la probabilità congiunta può essere scritta come prodotto delle probabilità marginali, condizionatamente ai valori assunti dalle variabili dei nodi "padri":

$$p(x) = \prod_{v \in V} p(x_v | x_{\mathrm{pa}(v)}) \tag{2.1}$$

dove pa(v) l'insieme dei nodi padri di v. Quindi, dato un insieme di variabili casuali, la probabilità congiunta può essere derivata dalle probabilità condizionate utilizzando la regola a catena:

$$\mathrm{P}(X_1 = x_1, \ldots, X_n = x_n) = \prod_{v=1}^{n} \mathrm{P}\left(X_v = x_v \mid X_{v+1} = x_{v+1}, \ldots, X_n = x_n\right) \tag{2.2}$$

In questo modo la proprietà di Markov è assicurata localmente: ogni variabile è condizionatamente indipendente rispetto alle variabili dei nodi non figli, dati i valori delle variabili dei nodi padri.

2.2 Modelli gerarchici

È possibile sfruttare il concetto di rete bayesiana per specificare modelli gerarchici specifici per i dati testuali. L'assunto di base per questi modelli è che i testi siano composti da una mistura di argomenti, *topic*, aventi una possibile correlazione tra loro; ognuno di questi topic è una distribuzione multinomiale sulle parole, queste ultime raggruppate in un vocabolario definito in precedenza sulla base dei testi analizzati: le parole con probabilità più alta forniscono un'idea dei temi trattati nella collezione di documenti. Un topic model è dunque un modello per la generazione di documenti: per generare un nuovo testo si estrae un topic, e successivamente un termine dalla distribuzione sul vocabolario corrispondente; il processo va iterato per tutta la lunghezza del documento. Ovviamente il processo può essere invertito attraverso tecniche statistiche, allo scopo di fare inferenza sull'insieme di topic che ha generato il documento. Sono stati proposti svariati modelli per l'analisi dell'informazione contenuta nei documenti e del significato delle parole; questi hanno in comune un presupposto fondamentale, un documento è una mistura di topic, come accennato in precedenza, e si differenziano per assunzioni statistiche.

Sia $\Pr(z)$ la distribuzione di probabilità sui topic in un particolare documento, e $\Pr(w|z)$ la distribuzione di probabilità di una parola dato un particolare topic z. Di conseguenza $\Pr(z_i = j)$ sarà la probabilità che il j-esimo topic sia estratto per la i-esima parola e $\Pr(w_i|z_i = j)$ la probabilità della parola w_i sotto il topic j. Il modello definisce la seguente distribuzione sulle parole in un documento:

$$\Pr(w_i) = \sum_{j=1} \Pr(w|z = j) \Pr(z = j) \tag{2.3}$$

2.2.1 Interpretazione geometrica

Questo tipo di modelli ha un'elegante interpretazione geometrica. Dato un vocabolario contenente W parole distinte, esso definisce uno spazio W dimensionale dove ogni asse corrisponde alla probabilità di osservare una specifica parola. Il simplesso $W - 1$ dimensionale identificato rappresenta tutte le distribuzioni di probabilità sulle parole. In figura 2.1 la regione ombreggiata corrisponde al simplesso bidimensionale che rappresenta tutte le distribuzioni di probabilità sulle tre parole. Come distribuzione di probabilità sulle parole, ogni documento può essere identificato da un punto sul simplesso; allo stesso modo, ogni topic può essere identificato da un punto sul simplesso. Ogni documento che viene generato dal modello è una combinazione convessa dei T topic che non solo identifica tutte le distribuzioni di parole come punti sul simplesso $W - 1$ dimensionale, ma anche come punti del simplesso $T - 1$ dimensionale generato dai topic.

2.3 Latent Dirichlet Allocation

Partendo dagli assunti di base appena presentati, è possibile definire il modello LDA, *Latent Dirichlet Allocation* [Blei et al., 2003], tra i più comunemente utilizzati in questo ambito: esso rappresenta ogni documento come una mistura di topic, ove ogni topic è una distribuzione multinomiale sulle parole del vocabolario (figura 2.2). Il processo generatore per un documento è il seguente:

1. Si estrae $\theta_i \sim \text{Dirichlet}(\alpha)$, dove $i \in \{1, \ldots, M\}$
2. Si estrae $\varphi_k \sim \text{Dirichlet}(\beta)$, dove $k \in \{1, \ldots, K\}$
3. Per ogni valore i , j della parola, dove $j \in \{1, \ldots, N_i\}$, e $i \in \{1, \ldots, M\}$

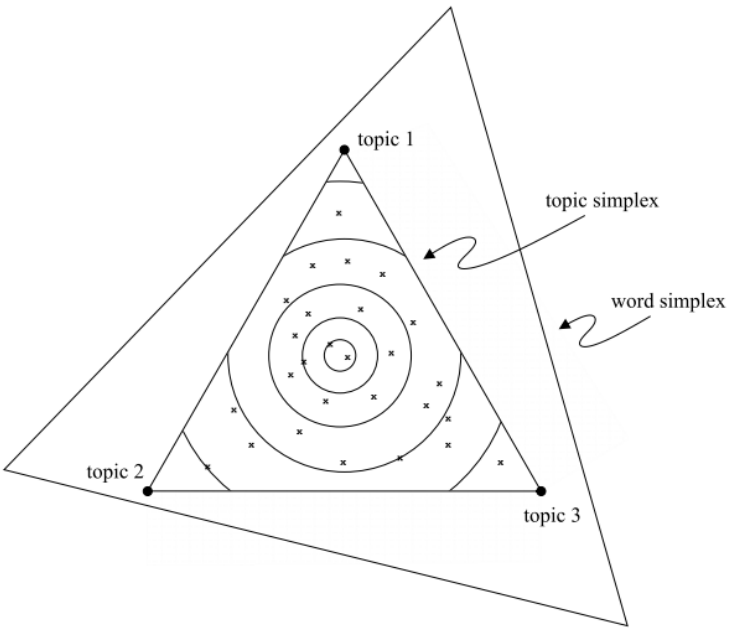


Figura 2.1: Interpretazione geometrica del topic model. Ogni documento è una mistura di topic, quindi un punto del simplesso definito dai topic (*topic simplex*); ogni topic a sua volta è un punto del simplesso definito dalle parole del vocabolario (*word simplex*).

(a) Si estrae un topic da $z_{i,j} \sim \text{Multinomiale}(\theta_i)$

(b) Si estrae una parola da $w_{i,j} \sim \text{Multinomiale}(\varphi_{z_{i,j}})$

Essendo coniugata naturale alla Multinomiale, la distribuzione di Dirichlet è molto conveniente come *a priori*, e ne risulta una notevole semplificazione per quanto riguarda i problemi di inferenza. La densità di una Dirichlet di dimensione K per il vettore di probabilità $p = (p_1, \ldots, p_K)$ di una Multinomiale è definita come:

$$Dir(\alpha_1, \ldots, \alpha_K) = \frac{\Gamma(\sum_j \alpha_j)}{\prod_j \Gamma(\alpha_j)} \prod_{j=1}^{K} p_j^{\alpha_j - 1} \tag{2.4}$$

I parametri di questa distribuzione sono specificati da $\alpha_1, \ldots, \alpha_K$; ogni α_j può essere interpretato come un conteggio a priori per il numero di estrazioni di un dato topic in un documento, prima di aver osservato qualsiasi parola del documento. È conveniente utilizzare una Dirichlet simmetrica con un singolo iperparametro $\alpha_1 = \cdots = \alpha_K = \alpha$; questa scelta porta ad avere una distribuzione *a priori* sui topic sufficientemente liscia, con parametro di lisciamento α.

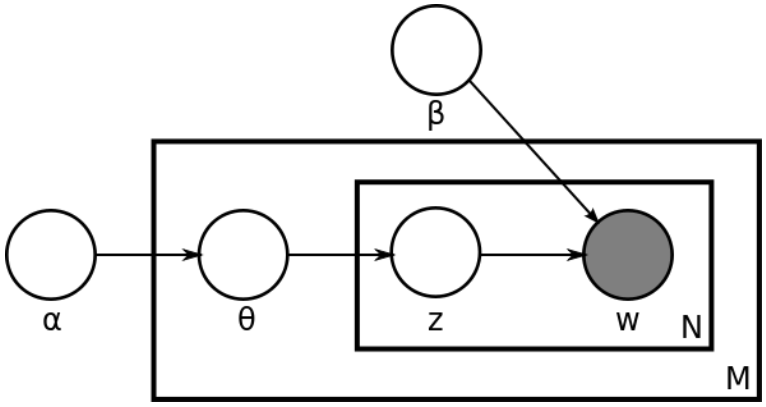


Figura 2.2: Modello grafico per il modello LDA

Il vantaggio riguarda anche la scambiabilità, assicurata dal singolo parametro, assunzione alla base dei modelli gerarchici.

L'interpretazione geometrica basata sul concetto di simplesso, accennata in precedenza, è particolarmente conveniente per esprimere le possibili ditribuzioni di probabilità: per ogni punto del simplesso $p = (p_1, \ldots, p_K)$, si ha $\sum_j p_j = 1$. La *a priori* sulla distribuzione dei topic può essere interpretata come forza sul livello di combinazione degli stessi: più α aumenta più i topic saranno lontani dagli angoli del simplesso. Particolare attenzione va posta quando $\alpha < 1$: in questo caso la distribuzione non presenterà una singola moda all'interno del simplesso, ma forzerà maggiore massa di probabilità sugli angoli; questa proprietà risulta notevolmente utile perchè direttamente legata al problema della sparsità delle parole. Dato l'alto numero di parametri della *a priori* sulle parole, è necessario ridurne la dimensionalità concentrando gran parte della densità su un insieme ridotto di queste, e lasciare probabilità quasi nulla sulla rimanente parte del dominio. Questo è possibile scegliendo accuratamente il paremetro β della distribuzione Dirichlet di φ.

Il modello finale che ne risulta è il seguente:

$$\Pr(\boldsymbol{W}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{\varphi}; \alpha, \beta) = \prod_{i=1}^{K} \Pr(\varphi_i; \beta) \prod_{j=1}^{M} \Pr(\theta_j; \alpha) \prod_{t=1}^{N} \Pr(Z_{j,t}|\theta_j) \Pr(W_{j,t}|\varphi_{Z_{j,t}}) \tag{2.5}$$

2.4 Metodi di stima

Il modello LDA è stato per ora presentato solo dal punto di vista teorico, è necessario a questo punto passare alle procedure di stima dei parametri. Verranno presentate tre varianti che utilizzano approcci basati sia su estensioni dell'algoritmo EM (Variational Bayes), sia su simulazioni MCMC quale il Gibbs Sampler; lo scopo è di confrontare le metodologie e soprattutto dare una panoramica più ampia possibile per quanto riguarda la stima di modelli gerarchici quale quello presentato.

2.4.1 Collapsed Gibbs Sampling

Le variabili di interesse riguardano la distribuzione delle parole nei topic, φ, e la distribuzione dei topic θ per ogni documento. Dato l'alto numero di parametri c'è il rischio che l'algoritmo rimanga "intrappolato" in massimi locali della distribuzione *a posteriori*. Una possibile soluzione è quella di non fare inferenza su φ e θ, bensì stimare la distribuzione *a posteriori* di z (le assegnazioni delle parole ai topic), condizionandosi alle parole osservate w. In particolare, è conveniente utilizzare un Gibbs Sampler per generare dalla *a posteriori* appena definita, marginalizzando rispetto a φ e θ.

Come già accennato nei precedenti paragrafi, la distribuzione di Dirichlet è coniugata naturale alla Multinomiale. Risulta dunque immediato derivare le distribuzioni *full conditional* per z [Griffiths e Steyvers, 2002]:

$$\Pr(z_i = j | \mathbf{z}_{-i}, w_i, d_i, \cdot) \propto \frac{C^{WT}_{w_i j} + \beta}{\sum_{w=1}^{W} C^{WT}_{wj} + W\beta} \frac{C^{DT}_{d_i j} + \alpha}{\sum_{t=1}^{T} C^{DT}_{d_i t} + T\alpha}$$

dove C^{WT} e C^{DT} sono matrici di conteggi di dimensioni $W \times T$ e $D \times T$ rispettivamente; C^{WT} contiene il numero di volte che la parola w è assegnata al topic j, senza includere il passo corrente i, mentre C^{DT} contiene il numero di volte che il topic j è assegnato al documento d, senza includere il passo corrente i. La parte sinistra dell'equazione rappresenta la probabilità della parola w sul topic j, la parte destra la probabilità del topic j sul documento d. Ogni volta che una parola è assegnata al topic j, la probabilità di assegnare altre parole specifiche a questo topic aumenta. Allo stesso tempo, se il topic j è usato più volte nello stesso documento, aumenta la probabilità che le parole del documento vengano

assegnate ad esso. Quindi le parole sono assegnate ai topic più verosimili come ai topic predominanti in un documento.

L'algoritmo fornisce stime dirette di z per ogni parola; rimane comunque di interesse conoscere le stime per φ e θ. Queste possono essere ottenute come segue:

$$\varphi_i'^{(d)} = \frac{C_{ij}^{WT} + \beta}{\sum_{k=1}^{W} C_{kj}^{WT} + W\beta} \qquad \theta_j'^{(d)} = \frac{C_{dj}^{DT} + \alpha}{\sum_{k=1}^{T} C_{dk}^{DT} + T\alpha}$$

I valori corrispondono rispettivamente alle distribuzioni di probabilità predettive di estrarre una nuova parola i dal topic j, e di estrarre un nuovo topic j nel documento d; in particolare sono le medie a posteriori di queste quantità condizionate ad un particolare valore di z.

2.4.2 Mean Field Variational Bayes

Il *Gibbs sampler* appena definito fornisce soluzioni analitiche semplici ed eleganti per le stime dei parametri, ma risulta computazionalmente inefficiente in presenza di un alto numero di documenti; il problema è dovuto alla catena di Markov utilizzata per generare dalla distribuzione a posteriori, che non consente l'utilizzo del calcolo parallelo. Per superare questa limitazione, è possibile fare inferenza sulla distribuzione *a posteriori* sfruttando un algoritmo di tipo *Variational Bayes* (VB); in questo tipo di metodi la vera distribuzione *a posteriori* è approssimata con una più trattabile $q(\mathbf{z}|\theta, \varphi)$, detta distribuzione *variazionale* [M. Hoffman et al., 2010].

Sia $p(\mathbf{w}|\alpha, \beta)$ la vera distribuzione *a posteriori* del modello, intrattabile dal punto di vista inferenziale; sfruttando la disuguaglianza di Jensen (e la concavità della funzione logaritmo), è possibile definire un limite inferiore per la verosimiglianza:

$$\begin{aligned}
\log p(x) &= \log \int p(x, z, \beta)\, \mathrm{d}z\mathrm{d}\beta \\
&= \log \int p(x, z, \beta) \frac{q(z, \beta)}{q(z, \beta)}\, \mathrm{d}z\mathrm{d}\beta \\
&= \log \mathbb{E}_q \left(\frac{p(x, z, \beta)}{q(z, \beta)} \right) \\
&\geq \mathbb{E}_q \left(p(x, z, \beta) \right) - \mathbb{E}_q \left(q(z, \beta) \right)
\end{aligned}$$

Massimizzare il limite inferiore appena presentato equivale a minimizzare la distanza di Kullback-Leibler tra la $q(\boldsymbol{z}, \theta, \varphi)$ e la vera distribuzione *a posteriori* del modello, $p(\boldsymbol{z}, \theta, \varphi|\boldsymbol{w}, \alpha, \beta)$, come dimostrato in [M. D. Hoffman et al., 2013]. L'uguaglianza è ricavabile direttamente dalla seguente equazione:

$$\begin{aligned} \mathrm{KL}(q(z,\beta)||p(z,\beta|x)) &= \mathbb{E}_q\left(q(z,\beta)\right) - \mathbb{E}_q\left(p(z,\beta|x)\right) \\ &= \mathbb{E}_q\left(q(z,\beta)\right) - \mathbb{E}_q\left(p(x,z,\beta)\right) + \log p(x) \\ &= -\mathcal{L}(q) + cost \end{aligned}$$

dove il termine costante deriva dall'indipenzanda rispetto a $q(z, \beta)$. Nel caso del modello LDA si ottiene:

$$\log p(\boldsymbol{w}|\alpha,\beta) \geq \mathcal{L}(\boldsymbol{w},\boldsymbol{\phi},\gamma,\lambda) = \mathbb{E}_q\left(p(\boldsymbol{w},\boldsymbol{z},\theta,\varphi|\alpha,\beta)\right) - \mathbb{E}_q\left(q(\boldsymbol{z},\theta,\varphi)\right) \quad (2.6)$$

Una scelta usuale consiste nel definire la distribuzione $q(\boldsymbol{z}, \theta, \varphi)$ in modo tale che sia fattorizzabile rispetto alle variabili latenti; l'indipendenza condizionata permette la scomposizione dei valori attesi. La distribuzione sarà dunque della forma:

$$q(\boldsymbol{z},\theta,\varphi) = q(\boldsymbol{z}|\boldsymbol{\phi})q(\theta|\gamma)q(\varphi|\lambda) \quad (2.7)$$

Se le distribuzioni sono scelte in modo tale da appartenere alla stessa famiglia esponenziale delle corrispettive distribuzioni esatte, ne risulta una notevole semplificazione analitica nel calcolo del limite inferiore. Le distribuzioni sono dunque definite come segue:

$$q(z_{di} = k) = \boldsymbol{\phi}_{dw_{di}k}, \quad q(\theta_d) \sim \mathrm{Dir}(\theta_d;\gamma_d), \quad q(\beta_k) \sim \mathrm{Dir}(\beta_k;\lambda_k) \quad (2.8)$$

dove la distribuzione *a posteriori* di ogni topic sul vocabolario è una Multinomiale parametrizzata da $\boldsymbol{\phi}$. A questo punto è possibile, con un algoritmo di tipo *coordinate ascent*, la massimizzazione del limite inferiore. Gli aggiornamenti dei parametri, analogamente alla procedura Expectation-Maximization (EM), avvengono in modo alternato: durante la fase di Expectation si stimano γ e $\boldsymbol{\phi}$, tenendo λ fissato; durante la fase di Maximization si stima λ fissando $\boldsymbol{\phi}$. Più formalmente, le formule di aggiornamento dei parametri sono le seguenti:

$$\begin{aligned}\phi_{dwk} &\propto \exp(\mathbb{E}_q(\log\theta_{dk}) + \mathbb{E}_q(\log\varphi_{kw}))\\ \gamma_{dk} &= \alpha + \sum_w n_{dw}\phi_{dwk}\\ \lambda_{kw} &= \beta + \sum_d n_{dw}\phi_{dwk}\end{aligned} \tag{2.9}$$

I valori attesi rispetto a $\log\theta$ e $\log\varphi$ sono calcolabili come segue:

$$\begin{aligned}\mathbb{E}_q(\log\theta_{dk}) &= \Psi(\gamma_{dk}) - \Psi\left(\sum_{i=1}^{K}\gamma_{di}\right)\\ \mathbb{E}_q(\log\varphi_{kw}) &= \Psi(\lambda_{kw}) - \Psi\left(\sum_{i=1}^{W}\lambda_{ki}\right)\end{aligned} \tag{2.10}$$

dove Ψ indica la funzione *Digamma*. Il metodo variazionale appena presentato, anche se analiticamente più difficile rispetto al *Gibbs-Sampling*, comporta notevoli vantaggi: non è necessario attendere la convergenza della catena, di conseguenza necessita di molte meno iterazioni per ottenere una stima; a meno dell'inizializzazione casuale dei pesi, l'algoritmo è deterministico; infine permette l'aggiornamento dei parametri anche con l'arrivo di nuove osservazioni, senza la necessità di far ripartire l'algoritmo (aggiornamento *online*).

2.4.3 Mini-batch Variational Bayes

Nonostante i vantaggi appena discussi, il metodo risulta non del tutto efficiente nel caso in cui il numero di osservazioni (documenti) sia molto elevato: ad ogni iterazione infatti è necessario calcolare il gradiente su tutti i dati. É possibile però sfruttare la scomposizione di $\mathcal{L}$, definita come somma di contributi di ogni singolo documento:

$$\mathcal{L}(n,\lambda) = \sum_d \ell(n_d, \gamma(n_d,\lambda), \phi(n_d,\lambda), \lambda) \tag{2.11}$$

dove $\gamma(n_d,\lambda)$ e $\phi(n_d,\lambda)$ sono le stime dei parametri ottenute nella fase di Expectation. A questo punto, invece di utilizzare tutti i documenti ad ogni iterazione, è possibile suddividere le osservazioni in sottocampioni, detti *mini-batch* (più avanti nella tesi verranno presentati più ampiamente). Definendo con S

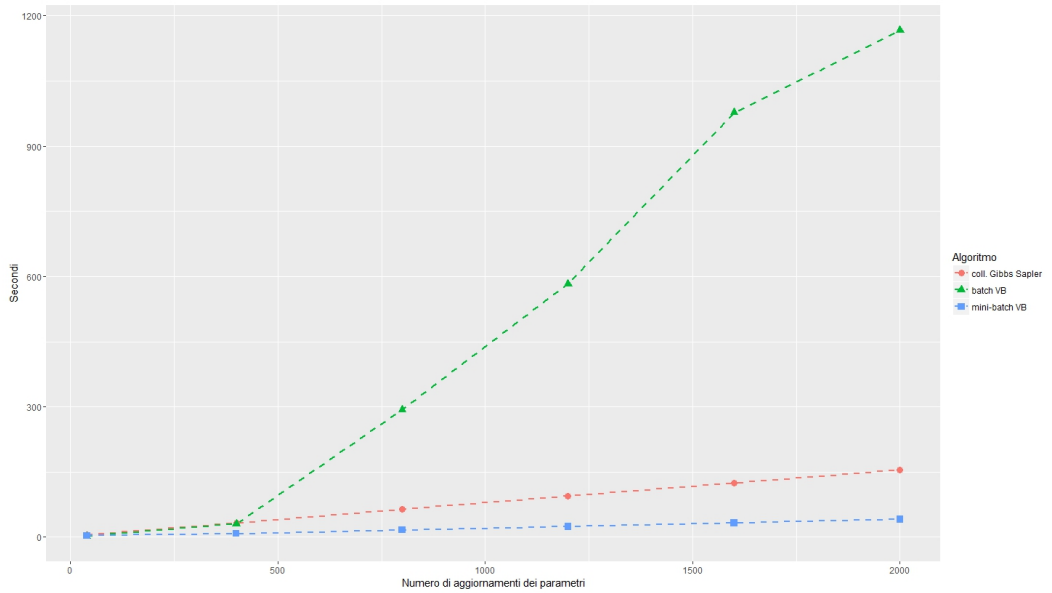


Figura 2.3: Confronto in termini di tempo computazionale per i tre algoritmi presentati; i tempi non sono stati calcolati all'interno dei cicli dell'algoritmo, ma corrispondono ai tempi totali per ognuna delle combinazioni. Si osserva un netto miglioramento dell'algoritmo *Variational Bayes* con l'utilizzo di *mini-batch*; il *Gibbs Sampler* rimane più lento di quest'ultimo, anche se non di molto nel caso preso in esame.

il numero di osservazioni per ogni *mini-batch*, l'unica modifica da apportare all'algoritmo riguarda il passo di Maximization, definito come segue:

$$\begin{aligned} \tilde{\lambda}_t &= \eta + \frac{D}{S} \sum_s n_{ts} \phi_{ts} \\ \lambda &= (1 - \rho_t)\lambda + \rho_t \tilde{\lambda}_t \end{aligned} \tag{2.12}$$

dove t il numero di iterazioni. L'aggiornamento è fatto in due passi: il primo è equivalente a quello già presentato, a cui è stato aggiunto il peso $\frac{D}{S}$; nel secondo passo avviene l'effettivo aggiornamento delle stime, utilizzando una media mobile esponenziale con peso $\rho_t = (\tau_0 + t)^{-\kappa}$, dove $\kappa \in (0.5, 1]$ è il parametro di *smoothing* e $\tau_0 \geq 0$ limita l'influenza delle stime ottenute nelle prime iterazioni.

Per valutare quanto appena affermato, i tre algoritmi sono stati confrontati calcolando il tempo di esecuzione necessario per diversi valori di aggiornamenti dei parametri; in figura 2.3 sono rappresentati i risultati ottenuti. In linea con quanto discusso in precedenza, la modifica *mini-batch* comporta un netto mi-

glioramento rispetto all'algoritmo standard, in particolare con l'aumentare del numero totale di aggiornamenti. Il *Gibbs Sampler*, valutato nella versione *collapsed*, nonostante il numero di documenti non sia troppo elevato rimane peggiore dell'algoritmo *Variational Bayes*. Va sottolineato che, come accennato in precedenza, il *Gibbs Sampler* necessita di molte più iterazioni per raggiungere la convergenza alla distribuzione di equilibrio, a differenza degli altri due presi in esame [1].

[1] I confronti sono stati fatti utilizzando i pacchetti `lda` e `sklearn`, rispettivamente per il *Gibbs Sampler* e per le due varianti del *Variational Bayes*, entrambe implementati in `Python`.

Capitolo 3

Replicated softmax

Nel capitolo precedente si è presentato un modello di tipo Bayesiano gerarchico, LDA, per modellare la struttura dei dati testuali caratterizzati da misture di argomenti. Mantenendo l'idea della modellazione basata su grafi, è possibile rilassare l'ipotesi gerarchica eliminando la struttura causale dettata dal grafo diretto. Verranno introdotti dunque i concetti di Markov Random Field e di Restricted Boltzman Machine, sviluppando i risultati inferenziali inizialmente con un approccio frequentista e successivamente con una modifica di tipo Bayesiano.

3.1 Markov random field

In ambito fisico e in teoria della probabilità, un Markov Random Field (MRF), o Markov Network, è un insieme di variabili casuali che godono della proprietà di Markov descritta da un grafo indiretto. Un Markov Random Field può essere facilmente paragonato ad un Bayesian Network (presentato nel capitolo 2), per il modo in cui vengono espresse e rappresentate le dipendenze tra le variabili; la differenza cruciale risiede nel fatto che una rete bayesiana è un grafo diretto e aciclico, mentre un MRF è un grafo indiretto e può essere ciclico. Queste caratteristiche portano a vantaggi diversi per i due modelli: un MRF può rappresentare dipendenze cicliche tra variabili casuali, mentre una rete bayesiana può rappresentare dipendenze indotte [Koller e N. Friedman, 2009]. Per quanto riguarda il grafo che caratterizza un MRF, può essere di dimensione finita o infinita.

Un punto fondamentale per parametrizzare un MRF è la rappresentazione indiretta della dipendenza tra variabili, che a differenza di una rete bayesiana non permette di definire le probabilità dei nodi condizionandosi ai nodi padri; si introduce allora una parametrizzazione basata sul concetto di fattorizzazione. Nel seguito verranno definiti alcuni concetti fondamentali per proseguire nella trattazione.

3.1.1 Distribuzione di Gibbs

La distribuzione di Gibbs (o distribuzione di Boltzmann) è un concetto derivante dalla fisica statistica, ed in particolare è una distribuzione definita su un campo che assegna probabilità a determinate configurazioni degli stati (variabili). Nel caso in cui la probabilità congiunta delle variabili casuali sia strettamente positiva, è possibile definire la distribuzione di Gibbs nel modo seguente:

$$P(X = x) = \frac{1}{Z(\beta)} \exp(-\beta E(x)) \tag{3.1}$$

dove $E(x)$ è una funzione definita dallo spazio degli stati ai numeri reali, chiamata *energia* della configurazione delle variabili x (termine derivante dall'ambito fisico), β è un parametro libero mentre $Z(\beta)$ è la costante di normalizzazione. In particolare, un MRF (con densità strettamente positiva) può essere scritto come un modello log-lineare con funzione di densità totale come segue:

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_k w_k^\top f_k(x_{\{k\}})\right) \tag{3.2}$$

con il prodotto vettoriale definito come:

$$w_k^\top f_k(x_{\{k\}}) = \sum_{i=1}^{N_k} w_{k,i} \cdot f_{k,i}(x_{\{k\}}) \tag{3.3}$$

e la costante di normalizzazione:

$$Z = \sum_{x \in \mathcal{X}} \exp\left(\sum_k w_k^\top f_k(x_{\{k\}})\right) \tag{3.4}$$

dove $\mathcal{X}$ denota l'insieme di tutti i possibili valori assumibili da tutte le variabili causali della rete. Generalmente, le funzioni $f_{k,i}$ sono definite come indicatrici della configurazione *clique*, cioè $f_{k,i}(x_{\{k\}}) = 1$ se $x_{\{k\}}$ corrisponde alla i-esima possibile delle k configurazioni clique, e 0 altrimenti.

In ambito matematico ed in particolare in teoria dei grafi, una *clique* è un sottoinsieme di vertici di un grafo indiretto tale che il sottografo indotto da questi ultimi è completo: ogni coppia di vertici distinti della *clique* è adiacente, cioè esiste una connessione tra ogni coppia di variabili del sottoinsieme.

3.1.2 Fattorizzazione *clique*

Come è già stato accennato, perchè il grafo sia un MRF deve essere soddisfatta la proprietà di Markov; la verifica di questa proprietà risulta essere difficile per un grafo arbitrario. La fattorizzazione *clique* permette di definire una particolare classe di modelli che godono di questa proprietà.

Sia $G = (V, E)$ un grafo indiretto con vertici V e archi E; dato un insieme di variabili casuali $X = (X_v)_{v \in V}$, sia $P(X = x)$ la probabilità di una particolare configurazione x in X. Dato che X è un insieme di eventi, è possibile definire la probabilità di x tramite la funzione di densità congiunta di X_v.
Se quest'ultima può essere fattorizzata sulle clique di G:

$$P(X = x) = \prod_{C \in \mathrm{cl}(G)} \phi_C(x_C) \tag{3.5}$$

allora X forma un Markov Random Field su G (cl(G) rappresenta l'insieme di clique di G). Questo tipo di fattorizzazione è immediatamente comparabile ai modelli a classi latenti, utilizzati più avanti nella trattazione.

La specificazione come modello log-lineare (3.1.1) è possibile solo nel caso in cui nessuno degli elementi di $\mathcal{X}$ (insieme di tutti i possibili valori assunti dalle variabili casuali della rete) ha probabilità nulla; questo permette di trattare più facilmente le matrici che definiscono il grafo con proprietà dell'algebra lineare. Il modello log-lineare è specialmente conveniente dal punto di vista interpretativo; inoltre la log-verosimiglianza cambiata di segno è una funzione convessa. Nonostante questo, spesso il problema di minimizzazione risulta

computazionalmente oneroso, e sono dunque necessarie tecniche più avanzate che verranno presentate più avanti nella trattazione.

3.2 Restricted Boltzmann Machine

Un caso particolare di Markov Random Field, il cui modello grafico corrisponde a quello definito dall'analisi fattoriale, è il Restricted Boltzmann Machine. Il modello RBM standard consiste in due strati di variabili casuali binarie, uno strato osservabile v_i ed uno nascosto h_j. Alle variabili è associata una matrice di pesi $W_{m\times n} = (w_{i,j})$ che ne descrive la relazione; sono inoltre definiti due vettori di pesi (*offset*) per le unità nascoste e quelle osservabili, rispettivamente a_i e b_j. È dunque possibile definire la funzione di energia nel modo seguente:

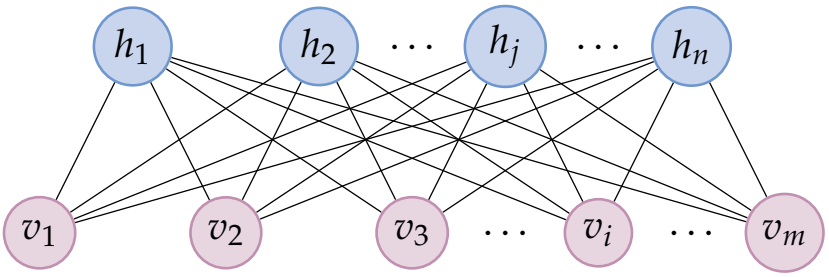


Figura 3.1: Modello grafico per una Restricted Boltzmann Machine

$$E(v,h) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j v_i h_j w_{i,j} \tag{3.6}$$

o equivalentemente in notazione matriciale:

$$E(v,h) = -a^{\mathrm{T}} v - b^{\mathrm{T}} h - v^{\mathrm{T}} W h \tag{3.7}$$

Come definito in precedenza, utilizzando la distribuzione di Gibbs è immediato caratterizzare la funzione di probabilità:

$$P(v,h) = \frac{1}{Z} e^{-E(v,h)} \tag{3.8}$$

con $Z = \sum_{v,h} exp(-E(v,h))$ costante di normalizzazione (necessaria per assicurare che la distribuzione di probabilità sommi a 1). In modo del tutto analogo

è possibile calcolare le probabilità marginali per le unità osservabili sommando su tutte le possibili configurazioni dello strato nascosto:

$$P(v) = \frac{1}{Z} \sum_{h} e^{-E(v,h)} \tag{3.9}$$

Data la particolare struttura a grafo bipartito della RBM, in cui non sono presenti connessioni tra unità appartenenti allo stesso strato, le variabili dello strato nascosto sono tra loro indipendenti condizionatamente a quelle dello strato osservabile, e viceversa le variabili dello strato osservabili sono mutualmente indipendenti condizionatamente a quelle dello strato nascosto. Di conseguenza la probabilità condizionata di una certa configurazione delle unità visibili v, data una certa configurazione delle unità h, e viceversa, sono le seguenti:

$$P(h|v) = \prod_{j=1}^{n} P(h_j|v) \qquad P(v|h) = \prod_{i=1}^{m} P(v_i|h) \tag{3.10}$$

Le probabilità per le singole variabili sono invece:

$$\begin{aligned} P(h_j = 1|v) &= \sigma\left(b_j + \sum_{i=1}^{m} w_{i,j} v_i\right) \\ P(v_i = 1|h) &= \sigma\left(a_i + \sum_{j=1}^{n} w_{i,j} h_j\right) \end{aligned} \tag{3.11}$$

dove $\sigma(x) = \frac{1}{1+exp(-x)}$ è la funzione logistica.

Nel caso del *topic modeling*, in cui le parole possono essere presenti più di una volta nello stesso documento, lo strato visibile è composto da variabili Multinomiali, mentre lo strato nascosto rimane composto da variabili Bernoulli. La funzione di probabilità risulta dunque essere la seguente:

$$P(v_i^k = 1|h) = \frac{\exp(a_i^k + \Sigma_j W_{ij}^k h_j)}{\Sigma_{k=1}^{K} \exp(a_i^k + \Sigma_j W_{ij}^k h_j)} \tag{3.12}$$

dove K rappresenta il numero di possibili valori.

3.3 Replicated Softmax

I modelli finora descritti rappresentano solo le fondamenta teoriche, è necessario ora concentrarsi sull'applicazione al topic modeling: verrà presentato in particolare il modello Replicated Softmax, introdotto da G. E. Hinton e Salakhutdinov, (2009).

Sia $v \in \{1, \dots, K\}^D$, con K numerosità del dizionario e D numerosità di un singolo documento, e sia $h \in \{0,1\}^F$ il vettore nascosto di topic. Si definisce dunque $V_{K\times D}$ la matrice di osservazioni binarie, tale che $v_i^k = 1$ se la i-esima unità assume il k-esimo valore; in altre parole, ogni colonna della matrice corrisponde ad una parola del documento, ed è composta da tutti zeri a meno di un uno sulla riga corrispondente alla parola nel dizionario. È possibile a questo punto identificare per ogni documento la seguente funzione di *energia*:

$$E(V,h) = -\sum_{i=1}^{D}\sum_{j=1}^{F}\sum_{k=1}^{K} W_{ij}^k h_j v_i^k - \sum_{i=1}^{D}\sum_{k=1}^{K} v_i^k b_i^k - \sum_{j=1}^{F} h_j a_j \tag{3.13}$$

dove $\{W, a, b\}$ rappresenta l'insieme dei parametri del modello. Facendo uso della misura di Gibbs è possibile definire la funzione di probabilità del modello come segue:

$$\begin{aligned} P(V) &= \frac{1}{\mathcal{Z}}\sum_h exp(-E(V,h)) \\ \mathcal{Z} &= \sum_V\sum_h exp(-E(V,h)) \end{aligned} \tag{3.14}$$

Le distribuzioni condizionate per i due strati v e h sono:

$$\begin{aligned} p(v_i^k = 1|h) &= \frac{exp(b_i^k + \sum_{j=1}^F h_j W_{ij}^k)}{\sum_{q=1}^K exp(b_i^q + \sum_{j=1}^F h_j W_{ij}^q)} \\ p(h_j = 1|V) &= \sigma\left(a_j + \sum_{i=1}^{D}\sum_{k=1}^{K} v_i^k W_{ij}^k\right) \end{aligned} \tag{3.15}$$

A questo punto sorge il problema di definire il modello per l'intero corpus di documenti. Si suppone dunque di creare una RBM per ognuno di questi, con tante unità dello strato osservabile quante sono le parole del documento.

Assumendo di poter ignorare l'ordine di queste ultime, tutte le unità possono condividere lo stesso insieme di pesi che le connettono allo strato nascosto. Considerando un documento contenente D parole, si può definire:

$$E(V,h) = -\sum_{j=1}^{F}\sum_{k=1}^{K} W_j^k h_j \hat{v}^k - \sum_{k=1}^{K} \hat{v}^k b^k - D\sum_{j=1}^{F} h_j a_j \tag{3.16}$$

dove $\hat{v}^k = \sum_{i=1}^{D} v_i^k$ denota il conteggio per la k-esima parola. Interessante osservare che il termine di offset per le unità nascoste è riscalato per la lunghezza del documento; questo è cruciale e permette di gestire adeguatamente la presenza di documenti di diversa lunghezza.

3.4 Penalized Replicated Softmax

La specificazione del modello Replicated Softmax è analiticamente semplice ma comporta un'estrema flessibilità per quanto riguarda le dipendenze tra parole e *topic*. Il problema fondamentale risiede nel numero di parametri, ed in particolare nella matrice di pesi $[W_{ij}]_{F\times K}$, soprattutto in presenza di un vocabolario di termini non ristretto. È naturale inoltre aspettarsi che parole poco o per nulla utilizzate in un dato contesto non diano informazione sullo stesso; si pensi ad esempio all'argomento *atletica* e alle parole *verosimiglianza* o *boosting*.

Prendendo le mosse da queste considerazioni e data la particolare specificazione del modello, è possibile con un approccio Bayesiano apportare una modifica al modello: si può pensare di introdurre una distribuzione *a priori* sui parametri, in modo tale da ridurne la variabilità nelle stime. Indicando con $P(\theta)$ una generica distribuzione sui parametri, è possibile definire la funzione di densità congiunta e quindi la verosimiglianza del modello come segue:

$$-\log P(x,\theta) = -\log P(x|\theta) - \log P(\theta) \tag{3.17}$$

con $\log P(x|\theta)$ corrispondente alla verosimiglianza del modello. La scelta di $P(\theta)$ risulta a questo punto cruciale: è necessario tenere in considerazione il dominio dei pesi W_{ij}, corrispondente all'insieme $\mathbb{R}$, e la specificazione del modello come distribuzione di Gibbs. Due possibili alternative risultano essere una ditribuzione *a priori* Laplace o Gaussiana [J. Friedman et al., 2001]: la prima, for-

zando alcuni dei parametri a 0, comporta una selezione delle variabili ma risulta poco trattabile; la seconda non elimina del tutto le variabili ma comprime i valori dei parametri verso lo 0 e non comporta problemi dal punto di vista analitico. Data la forma funzionale della densità congiunta, entrambe le distribuzioni *a priori* possono essere viste come vincoli sulla funzione di perdita: rispettivamente una penalizzazione di tipo $L1$ per la distribuzione Laplace, della forma $\lambda \sum_i |\theta_i|$, e $L2$ per la distribuzione Gaussiana, definita invece $\lambda \sum_i \theta_i^2$. In figura 3.2 è riportata l'interpretazione geometrica per queste due possibili scelte: è immediato notare la differenza nella forma della frontiera, che costringe alcuni parametri a 0 nel caso di penalizzazione $L1$, mentre ne riduce solamente il valore nel caso $L2$.

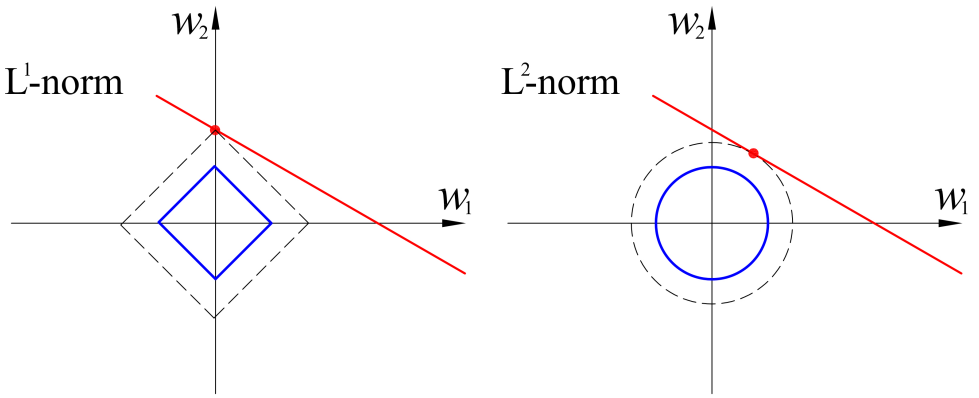


Figura 3.2: Interpretazione geometrica delle penalizzazioni $L1$ a sinistra e $L2$ a destra; la linea rossa corrisponde alla funzione di perdita da minimizzare. Si nota come la norma di tipo $L1$ faccia si che alcuni dei parametri vengano posti a 0; la norma $L2$ invece comprime i parametri verso l'origine, penalizzando la somma $\lambda \sum_i \theta_i^2$ e di conseguenza valori elevati.

Considerando quanto appena detto, è possibile derivare la funzione di densità congiunta come prodotto tra $P(V|W)$ e la distribuzione a priori sui parametri $P(W)$, che in questa tesi è stata scelta come densità Gaussiana. Se si assume che i pesi W_{ij} siano tali che $W_{ij} \sim \mathcal{N}(0, \tau^2)$ indipendenti, la funzione di densità che ne risulta è la seguente:

$$P(V,W) = \frac{1}{\mathcal{Z}} exp\Bigg(- \sum_{i=1}^{D}\sum_{j=1}^{F}\sum_{k=1}^{K} W_{ij}^k h_j v_i^k - \sum_{i=1}^{D}\sum_{k=1}^{K} v_i^k b_i^k - \sum_{j=1}^{F} h_j a_j - \lambda \sum_{i=1}^{D}\sum_{j=1}^{F} W_{ij}^2 \Bigg) \tag{3.18}$$

con $\lambda = \frac{1}{\tau^2}$. Il parametro λ è direttamente legato alla varianza della distribuzione *a priori*: valori alti comportano una distribuzione molto concentrata in 0, e dunque una penalizzazione elevata; al contrario valori bassi comportano una distribuzione più piatta, con un conseguente rilassamento della penalizzazione. Da un punto di vista Bayesiano, la stima dei parametri può essere vista come moda *a posteriori*; nel caso di distribuzione Gaussiana, preso in considerazione in questo modello, la moda corrisponde alla media *a posteriori*. È interessante a questo punto avere un primo riscontro, per capire se la modifica apportata al modello comporti un effettivo miglioramento delle stime; una trattazione più approfondita delle procedure di stima sarà presentata nel prossimo capitolo.

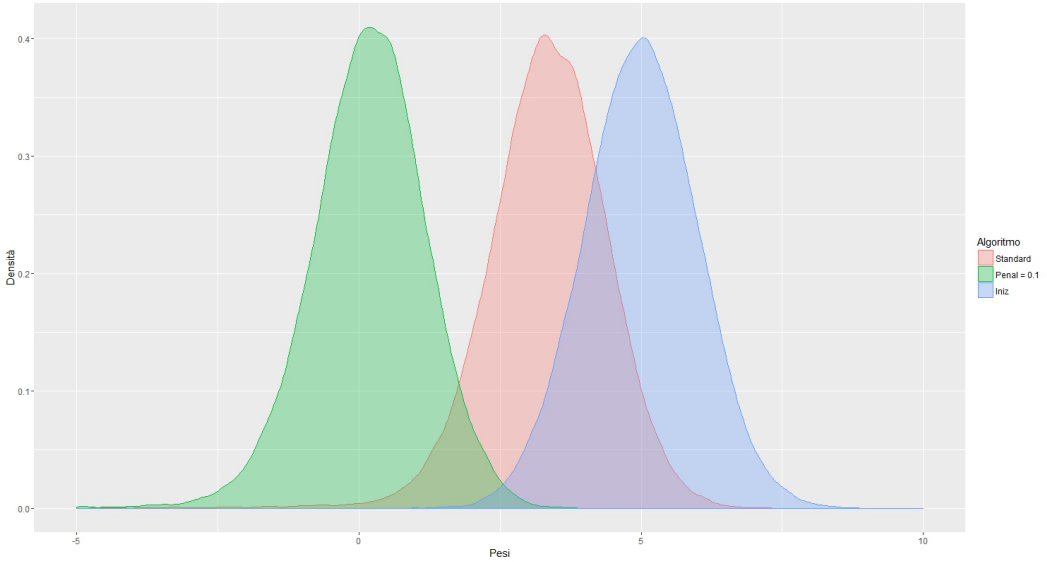


Figura 3.3: Confronto tra le densità a posteriori per i pesi W_{ij}, inizializzati da una distribuzione Normale a media 5: anche qui la *a priori* come penalizzazione influenza pesantemente le stime dei parametri, riducendone la variabilità e comprimendole verso lo 0. Interessante far notare come il modello penalizzato, a parità di numero di iterazioni, sia arrivato a convergenza a differenza del modello standard.

In figura 3.3 sono state confrontate le densità *a posteriori* nelle due situazioni presentate (con e senza distribuzione *a priori*): l'inizializzazione dei pesi è stata fatta da una distribuzione Normale $\mathcal{N}(5, 100)$, collocata nella parte più a destra del grafico. É possibile notare come la *a priori* comporti una notevole riduzione del valore dei parametri, che a parità di numero di iterazioni risultano essere centrati in 0; il modello standard sembra portare comunque ad una riduzione del valore dei parametri, ma molto più debole (distribuzione a media 3). Oltre

a comprimere i parametri, la distribuzione *a priori* migliora anche la velocità di convergenza: in entrambi i casi sono state effettuate 8000 iterazioni e, a differenza di quello presentato, il modello standard non è arrivato a convergenza. Per un'analisi più approfondita, si è deciso di valutare l'effetto della *a priori* su media, varianza e distribuzione dei pesi: in figura 4.2 sono presentati i risultati. I due grafici in alto corrispondono ai valori di media e varianza, valutati per tutta la durata della procedura di stima, per valori del parametro di regolazione $\lambda \in (0, 0.1, 0.9)$. Nonostante l'inizializzazione dei pesi sia stata fatta questa volta da una Normale $\mathcal{N}(0, 100)$, i modelli con *a priori* sembrano essere più robusti, mantenendo la media dei parametri costantemente a 0 e apportando una riduzione della varianza delle stime; il modello standard invece risulta avere una media che tende ad aumentare anche dopo 8000 iterazioni, oltre ad avere una variabilità più elevata. Per quanto riguarda i due grafici nella parte inferiore, a destra sono rappresentate le distribuzioni dei pesi stimati; il grafico di sinistra corrisponde invece ai *trace plot* del parametro di regolazione: interessante notarne l'andamento, soprattutto per quello riguardante $\lambda = 0.9$, che mostra come la penalizzazione oscilli tra valori elevati inizialmente, andando poi a stabilizzarsi con l'avanzare delle iterazioni.

In questo capitolo ci si è focalizzati sulla parte teorica dei modelli, nel capitolo seguente verranno trattate in dettaglio le procedure di stima e le possibilità di ottimizzazione degli algoritmi, con particolare attenzione all'implementazione della penalizzazione.

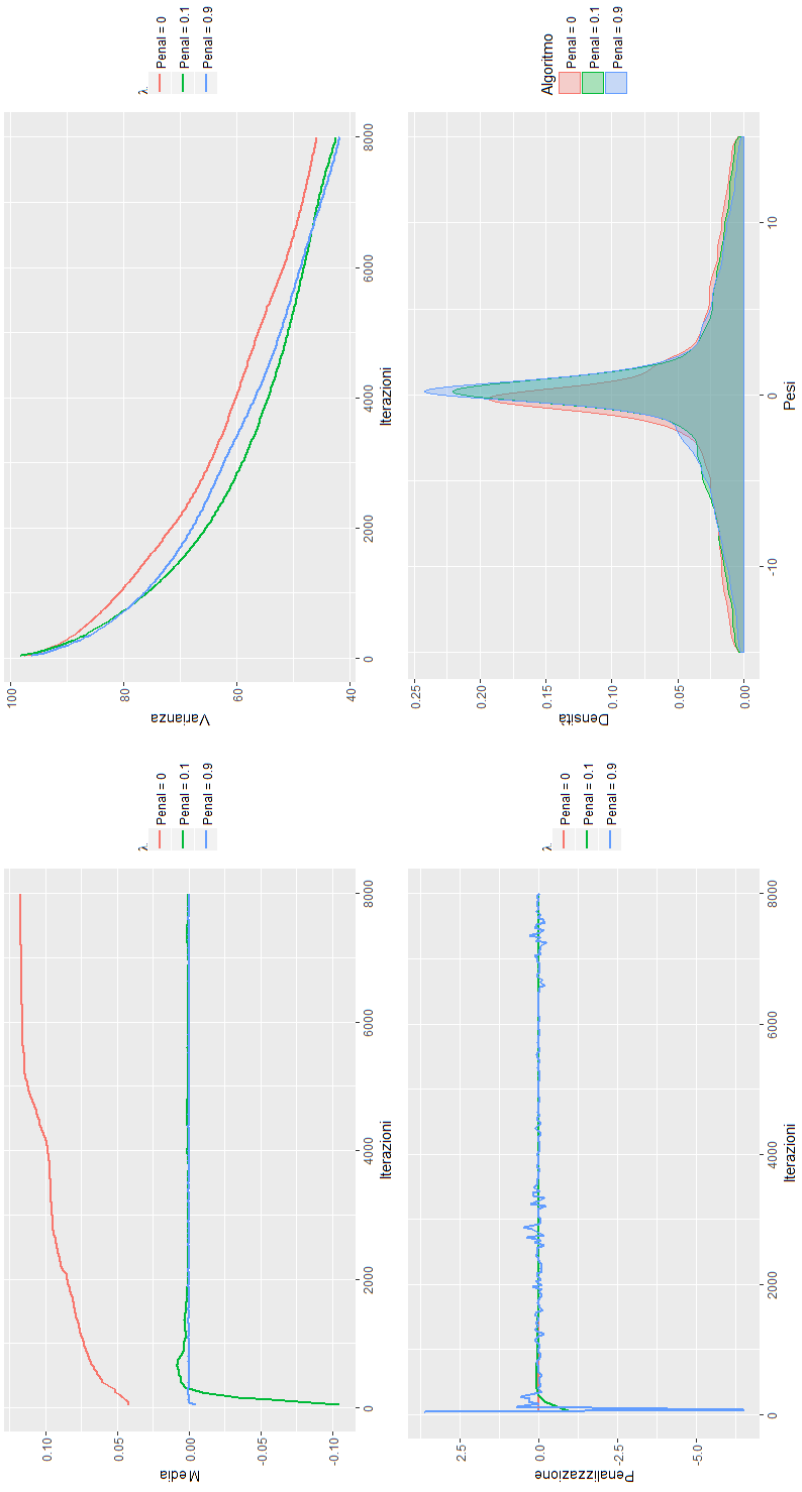


Figura 3.4: I due grafici in alto mostrano i *trace plot* di media e varianza dei pesi; i due grafici in basso rappresentano, a destra la densità dei pesi stimati, a sinistra i valori di penalizzazione valutati per tutta la durata dell'algoritmo. Si nota come nel complesso la modifica apportata al modello risulta migliorare sia la robustezza sia la velocità di convergenza delle stime.

Capitolo 4

Procedura di stima e applicazione ai dati

Il modello Replicated Softmax è stato finora presentato solo dal punto di vista teorico. In questo capitolo verrà introdotta un'approssimazione della funzione di log-verosimiglianza necessaria per la stima dei parametri. Si passerà dunque alla spiegazione dettagliata delle modifiche apportate all'algoritmo di stima per ottenere stime soddisfacenti.

4.1 Contrastive Divergence

La stima tramite massima verosimiglianza dei parametri W, dato un insieme di osservazioni, può essere fatta tramite un algoritmo di tipo *gradient descent* [Carreira-Perpinan e G. Hinton, 2005]. L'aggiornamento dei parametri avviene secondo la formula:

$$W^{(k+1)} = W^{(k)} - \alpha \left. \frac{\partial L(W;x)}{\partial W} \right|_{W}^{(k)} \tag{4.1}$$

con L funzione di perdita opportuna, in questo caso la funzione di log-verosimiglianza (le formule per l'aggiornameto dei parametri a e b sono del tutto analoghe, non vengono presentate per non appesantire la trattazione). Dato un corpus di N documenti $\{V_n\}_{n=1}^{N}$, è immediato verificare che la derivata della log-verosimiglianza rispetto a ciascuno dei parametri W prende la forma seguente:

$$\frac{1}{N}\sum_{n=1}^{N}\frac{\partial \log P(V_n)}{\partial W_j^k} = \mathbb{E}_{P_{data}}[\hat{v}^k h_j] - \mathbb{E}_{P_{Model}}[\hat{v}^k h_j] \tag{4.2}$$

dove $\mathbb{E}_{P_{data}}[\cdot]$ denota il valore atteso rispetto alla distribuzione dei dati:

$$P_{data}(h, V) = p(h|V)P_{data}(V)$$

e $P_{data}(V)$ la distribuzione empirica, e $\mathbb{E}_{P_{Model}}[\cdot]$ il valore atteso rispetto alla distribuzione definita dal modello. Il problema sorge nel calcolo di $\mathbb{E}_{P_{Model}}[\cdot]$, esponenziale dell'ordine di $min\{D, F\}$, cioè il numero di unità visibili. Per superare questa difficoltà, è possibile utilizzare un metodo chiamato *Contrastive Divergence*: invece di seguire il gradiente della funzione appena descritta, si approssima P_{Model} con P_T, la distribuzione limite di un Gibbs Sampler dopo T iterazioni (inizializzato sui dati). L'aggiornamento dei parametri avviene dunque come segue:

$$\Delta W_j^k = \alpha\left(\mathbb{E}_{P_{data}}[\hat{v}^k h_j] - \mathbb{E}_{P_T}[\hat{v}^k h_j]\right) \tag{4.3}$$

dove α è il *learning rate*. La particolare struttura bipartita della RBM permette un Gibbs Sampler abbastanza efficiente, che alterna estrazioni indipendenti dallo strato nascosto condizionandosi alle unità dello strato visibile, e viceversa.

L'approssimazione considerata nell'algoritmo Contrastive Divergence è direttamente collegata alla minimizzazione della distanza di Kullback-Leibler. Si può verificare che la stima di massima verosimiglianza minimizza la seguente distanza:

$$\mathrm{KL}(p_0||p_\infty) = \sum_{\mathrm{x}} p_0(\mathrm{x})\log\frac{p_0(\mathrm{x})}{p(\mathrm{x};\mathrm{W})} \tag{4.4}$$

Come specificato precedentemente, la difficoltà è data dal denominatore, cioè il calcolo della distribuzione del modello. L'algoritmo Contrastive Diverge approssima questa funzione, seguendo il gradiente della differenza di due distanze:

$$\mathrm{CD_T} = \mathrm{KL}(p_0||p_\infty) - \mathrm{KL}(p_T||p_\infty) \tag{4.5}$$

Per $T \to \infty$ le stime convergono a quelle di massima verosimiglianza.

4.2 Algoritmo di discesa del gradiente

L'algoritmo di discesa del gradiente (*batch gradient descent*), presentato nel paragrafo precedente, è una tecnica di ottimizzazione numerica per la ricerca di massimi e minimi di una funzione a più variabili. Più formalmente, si suppone di dover risolvere il problema di ottimizzazione non vincolata in $\mathbb{R}^n$

$$\min_{\theta} L(\theta), \quad \theta \in \mathbb{R}^n \tag{4.6}$$

con $L(\theta)$ una certa funzione di perdita. L'algoritmo si basa sull'osservazione che, dato un punto $\bar{\theta}$, la direzione di massima discesa corrisponde a quella determinata dall'opposto del suo gradiente in quel punto, $-\nabla L(\bar{\theta})$. Questa scelta garantisce che la soluzione tenda ad un punto di minimo (*locale*) di $L(\theta)$. Dato un punto iniziale θ_0, la procedura iterativa è definita come

$$\theta_{k+1} = \theta_k - \alpha \nabla L(\theta_k) \tag{4.7}$$

con $\alpha \in \mathbb{R}^+$ parametro che regola la lunghezza del passo di discesa, e di conseguenza la velocità di convergenza dell'algoritmo. É immediato verificare che l'algoritmo corrisponde alla formula di aggiornamento dei pesi W presentata nel paragrafo precedente.

4.2.1 Stochastic Gradient Descent

Nell'algoritmo di discesa del gradiente standard appena presentato, il calcolo del gradiente viene fatto su tutte le osservazioni disponibili per ogni singola iterazione; questo comporta, nel caso di *dataset* caratterizzati da un'alta numerosità, un notevole costo computazionale. Una possibile soluzione consiste nel calcolare il gradiente utilizzando una singola osservazione per ogni iterazione: la tecnica è definita *stochastic gradient descent*. La formula di aggiornamento dei parametri diventa

$$\theta_{k+1} = \theta_k - \alpha \nabla L(\theta_k; x^{(i)}) \tag{4.8}$$

dove $x^{(i)}$ rappresenta la i-esima osservazione di una permutazione dei dati (la permutazione è fatta all'inizio dell'algoritmo per assicurare che l'ordinamento sia casuale). L'utilizzo di questa procedura permette di evitare il calcolo del gradiente più volte per osservazioni simili, portando ad un netto aumento del-

la velocità computazionale; l'algoritmo permette inoltre un'esplorazione molto più ampia dello spazio parametrico, migliorando la ricerca di punti di minimo. Lo svantaggio è dato dall'alta variabilità delle stime data dai frequenti aggiornamenti dei parametri, che rende difficile la convergenza; nei paragrafi seguenti saranno presentate possibili soluzioni per contrastare questo problema.

4.2.2 Mini-batch Gradient Descent

Nonostante il miglioramento per quanto riguarda il costo computazionale, in presenza di molte migliaia di osservazioni è comunque necessario il calcolo del gradiente per ognuna di esse [G. Hinton, 2010]. Un'alternativa che sfrutta i vantaggi dati da entrambe le tecniche è quella del *mini-batch gradient descent*: invece di utilizzare un'osservazione alla volta, viene effettuata una permutazione inziale e successivamente si suddivide il *dataset* in campioni di numerosità fissata m. L'aggiornamento dei parametri diventa

$$\theta_{k+1} = \theta_k - \alpha \nabla L(\theta_k; x^{(i:i+m)}) \tag{4.9}$$

dove $x^{(i:i+m)}$ indica l'm-esimo sottocampione dei dati permutati. Con questa modifica in primo luogo si riduce la varianza delle stime dei parametri, con una conseguente maggiore stabilità nella convergenza; in secondo luogo, è possibile sfruttare al massimo le capacità del calcolo parallelo per il calcolo del gradiente di ogni *mini-batch*. Scelte comuni per il parametro m vanno da 10 a 100 (m deve essere divisore del numero di osservazioni a disposizione); in questa tesi si è scelto un valore di 10.

4.3 Ottimizzazione dell'algoritmo

La convergenza ad un punto di minimo può essere difficoltosa in punti caratterizzati da alta curvatura: il rischio è che le stime dei parametri oscillino nella direzione perpendicolare a quella del punto di minimo, rallentando di molto la velocità di convergenza, come mostrato nella parte sinistra in figura 4.1. Un metodo frequentemente utilizzato per velocizzare la velocità di convergenza è quello definito *momentum*: l'idea è molto semplice, consiste nell'aggiungere ad ogni passo di aggiornamento dei parametri una frazione dell'aggiornamento precedente [G. Hinton, 2010]. Più formalmente, la stima avviene sfruttando una media mobile esponenziale:

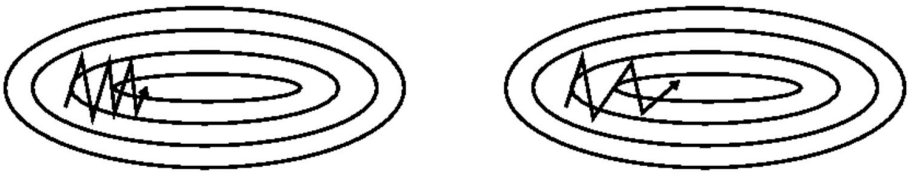


Figura 4.1: A sinistra, i passi dell'algoritmo *gradient descent* in un punto ad alta curvatura: le stime tendono ad oscillare nella direzione perpendicolare a quella del punto di minimo. A destra, i passi dello stesso algoritmo con l'aggiunta del *momentum*: la correzione permette di annullare parte dell'oscillazione e velocizzare la convergenza verso il punto di minimo.

$$\begin{aligned} v_k &= \gamma v_{k-1} + \nabla L(\theta_k; x^{(i)}) \\ \theta_{k+1} &= \theta_k - \alpha v_k \end{aligned} \tag{4.10}$$

L'aggiunta del termine v_{k-1} permette all'algoritmo di muoversi nella direzione perpendicolare rispetto a quella definita da $-\nabla L$: questa correzione porta ad un rallentamento delle oscillazioni nella direzione perpendicolare a quella del punto di minimo e ad un aumento della velocità di convergenza (figura 4.1, parte destra).

4.4 Penalizzazione

Come specificato nel capitolo precedente, al modello è stato aggiunto un termine di penalizzazione per la matrice di pesi W. Il problema di ottimizzazione della funzione di perdita diventa dunque di minimizzazione vincolata, risolvibile sempre grazie al metodo *gradient descent*. In letteratura, ed in particolare facendo riferimento a modelli basati su RBM e reti neurali, i metodi di penalizzazione per i pesi sono noti come *weight decay* [Bengio, 2012]; modelli di questo tipo permettono grande flessibilità a scapito però dell'elevato numero di parametri, con un evidente rischio di sovradattamento che non deve essere sottovalutato.

Mantenendo le notazioni finora utilizzate, data una funzione di perdita L e un insieme di parametri da stimare θ, l'algoritmo di aggiornamento che considera anche la penalizzazione diventa il seguente:

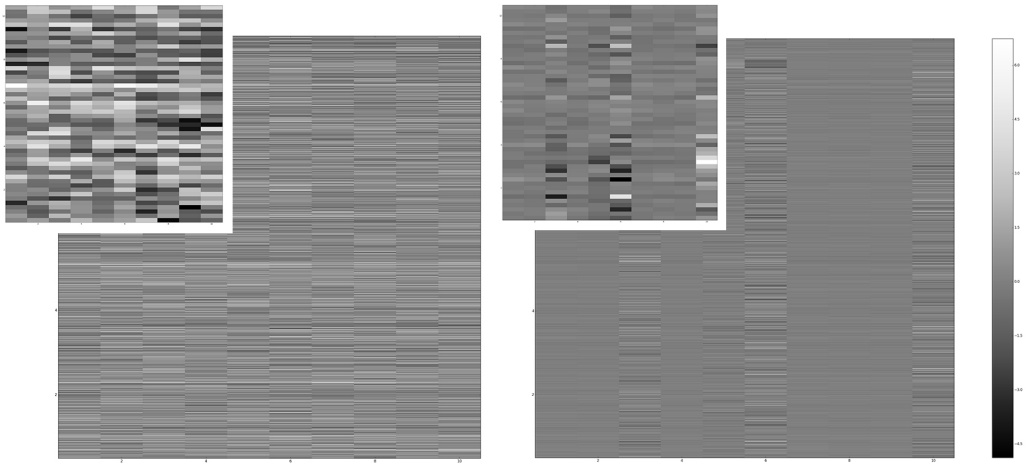


Figura 4.2: Heatmap delle matrici dei pesi W. A sinistra, la matrice dei pesi ottenuta dall'algoritmo standard; a destra, quella ottenuta aggiungengo la penalizzazione. I toni cromatici vanno dal nero (valori negativi) al bianco (valori positivi). Si nota immediatamente che nella matrice di destra la maggior parte dei pesi ha valori prossimi a 0 (in grigio), a differenza di quella di sinistra in cui la variabilità delle stime è molto più alta.

$$v_k = \gamma v_{k-1} + \nabla L(\theta_k; x^{(i:i+m)}) - \lambda \frac{m}{N} \sum_j \theta_{k_j}$$
$$\theta_{k+1} = \theta_k - \alpha v_k \tag{4.11}$$

dove la somma si intende su tutti i parametri. Il termine $\nabla L(\theta_k; x^{(i:i+m)})$ corrisponde alla derivata, rispetto ai parametri, dell'approssimazione della verosimiglianza del modello. Come descritto nel capitolo precedente, alla funzione verosimiglianza è stata aggiunta una *a priori* del tipo $\lambda \sum_i \theta_i^2$ come penalizzazione per i parametri. Derivando anche questo termine rispetto ai parametri si ottiene, nel modello in questione, $\frac{1}{2\tau^2} \sum_{i=1}^{D} \sum_{j=1}^{F} W_{ij}$. Il parametro di regolazione si può dunque definire come $\lambda = \frac{1}{2\tau^2}$. È utile a questo punto ricordare che è possibile scrivere $L(\theta)$ come somma di contributi per ogni documento; il modello RSM definisce infatti una Restricted Boltzman Machine per ognuno di questi, caratterizzata da una matrice di pesi W_{ij} condivisa. La procedura di stima, come già analizzato, comporta l'utilizzo di *mini-batch*, sotto-campioni di osservazioni di dimensione m: è necessario perciò aggiungere la costante moltiplicativa $\frac{m}{N}$, per assicurare la non distorsione delle stime. Come già accennato, si è scelto di utilizzare una penalizzazione di tipo L2: da un punto di vista analitico compor-

ta notevoli vantaggi in quanto è derivabile in tutto il dominio; da un punto di vista teorico, eliminare del tutto l'informazione apportata da certi termini (come succederebbe con la penalizzazione L1) sembra una scelta azzardata, soprattutto considerando che il lessico di tutti gli articoli è di carattere scientifico.

In figura 4.2 sono rappresentate graficamente le matrici dei pesi, dette anche *heatmap*, con tonalità in scala di grigio proporzionali ai valori assunti (dal nero, $-\infty$, al bianco, $+\infty$): la matrice di sinistra corrisponde alle stime del modello standard, la matrice di destra alle stime del modello con penalizzazione presentato; nell'angolo in alto a sinistra di entrambe le *heatmap* sono rappresentati i corrispondenti ingrandimenti. Si nota immediatamente come la variabilità delle stime del modello standard sia molto più elevata, segno che la penalizzazione sta effettivamente comprimendo verso lo 0 i pesi. È fondamentale a questo punto la scelta del parametro di regolazione λ (un valore di λ pari a 0 corrisponde al modello standard). In figura 4.3 sono riportati i confronti in termini di *perplexity* per vari valori del parametro λ, su un totale di 40.000 iterazioni per ognuno di essi; interessante notare come valori tra 0.01 e 0.1 permettano all'algoritmo di continuare a migliorare anche dopo un numero così elevato di iterazioni, oltre ad apportare un netto calo della *perplexity*. La penalizzazione sui pesi permette di limitare le zone dello spazio in cui l'algoritmo "ricerca" le stime: questo comporta un rallentamento iniziale della convergenza permettendo però un miglioramento in termini di verosimiglianza [1].

4.5 Confronto dei modelli

Ora che i modelli sono stati analizzati sia dal punto di vista teorico che dal punto di vista delle procedure di stima, è interessante un confronto per capire se il loro comportamento è simile o, al contrario, uno prevale sull'altro; trattandosi di analisi non supervisionate risulta però difficile definire una misura della bontà del modello. Nella letteratura riguardante i dati testuali ed in particolare i *topic model* si fa riferimento spesso alla cosiddetta *perplexity*: possibile definire questa misura come il reciproco della media geometrica della verosimiglianza calcolata per ogni parola. Più formalmente, dato un modello, la *perplexity* si calcola come segue:

[1] I confronti sono stati fatti utilizzando una funzione definita appositamente per il modello Replicated Softmax con penalizzazione, implementata in `Python` e presentata in *Appendice*.

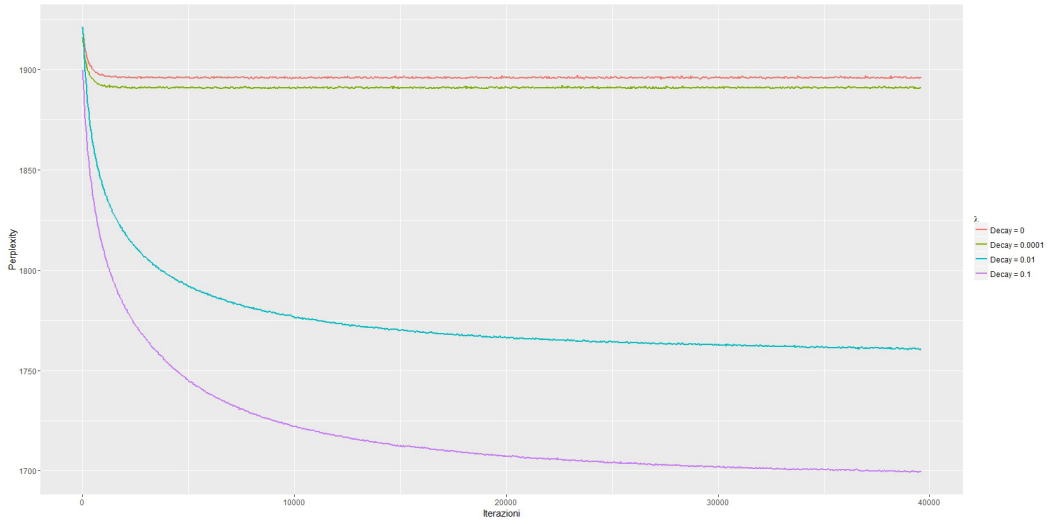


Figura 4.3: Confronto in termini di *perplexity* per vari valori del parametro di penalizzazione; si nota come un aumento della penalizzazione porti ad un evidente miglioramento nell'algoritmo di discesa del gradiente. In particolare valori di λ tra 0.01 e 0.1, anche dopo 40.000 iterazioni, continuano a far decrescere il valore di *perplexity* (la decrescita è lenta a causa del parametro di *learning rate* che è molto basso, $\alpha = 0.0001$)

$$\mathcal{P}(W|M) = \exp\left(-\frac{\sum_{m=1}^{M} \log p(w_m|M)}{\sum_{m=1}^{M} N_m}\right) \tag{4.12}$$

dove M indica il modello stimato, mentre w_m il vettore di parole del documento m-esimo. Con questa specificazione, un valore basso di *perplexity* indica una migliore rappresentazione delle parole del documento attraverso i *topic* stimati. Il calcolo esatto di $\log p(w_m|M)$ risulta però intrattabile per entrambi i modelli presentati; seguendo quanto riportato in M. Hoffman et al., (2010), è stato utilizzato per il modello LDA il seguente limite superiore:

$$\mathcal{P}_{LDA} \leq \exp\left(-\frac{\sum_{m=1}^{M} \mathbb{E}_q(\log p(w_m, \theta_m, z_m|\alpha, \beta)) - \mathbb{E}_q(\log q(\theta_m, z_m))}{\sum_{m=1}^{M} N_m}\right) \tag{4.13}$$

mentre per il modello RSM è stata utilizzata:

$$\mathcal{P}_{RSM} \leq \exp\left(-\frac{\sum_{m=1}^{M} \mathbb{E}_{P_{data}}(\log p(v_m, W, h)) - \mathbb{E}_{P_T}(\log p(v_m, W, h))}{\sum_{m=1}^{M} N_m}\right) \tag{4.14}$$

Entrambe possono essere viste come la distanza di Kullback-Leibler tra la funzione di probabilità esatta e l'approssimazione definita dal modello.

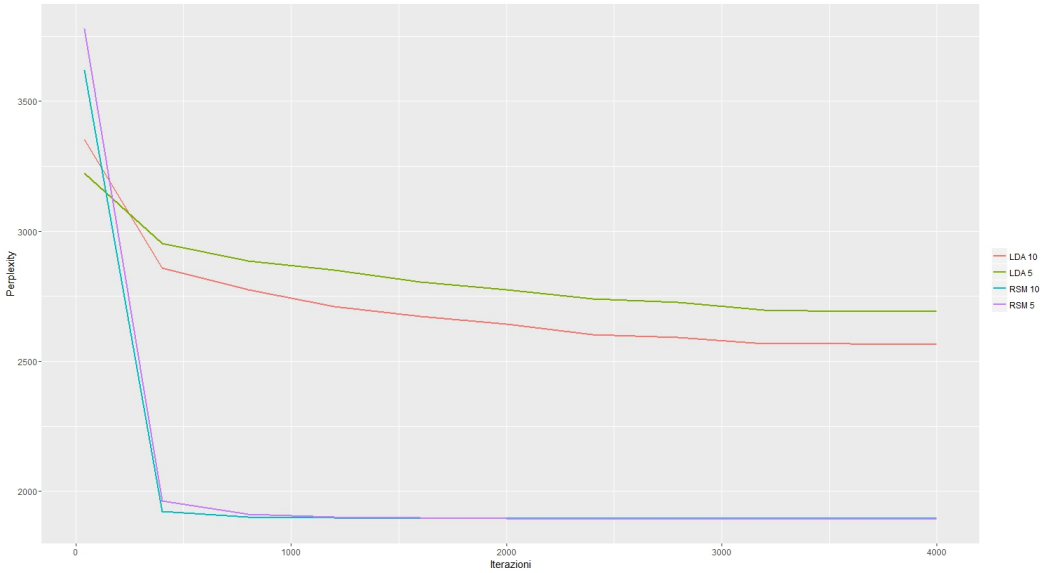


Figura 4.4: Confronto in termini di *perplexity* per i due modelli presentati; in particolare il confronto è stato fatto per $k = 5, 10$ (con k numero di topic). In entrambi i casi il modello *Replicated Softmax* con le modifiche apportate risulta decisamente migliore.

In figura 4.4 è stata valutata la *perplexity* per entrambi i modelli con numero di *topic* k pari a 5 e 10; da questo confronto il modello RSM penalizzato risulta essere decisamente migliore rispetto al modello LDA (il parametro di regolazione utilizzato è $\lambda = 0.1$). Nonostante la *perplexity* non sia una misura assoluta di valutazione dei modelli, considerando nel complesso tutti i risultati ottenuti finora, la modifica al modello RSM risulta essere del tutto soddisfacente. Oltre ad un miglioramento della velocità di convergenza, la distribuzione *a priori* riduce drasticamente il rischio di sovradattamento dovuto all'elevato numero di parametri e ne comporta una compressione verso lo 0, riducendone la varianza.

Conclusioni

La parola *topic* pervade l'intera tesi, trattata come fattore latente e come definizione linguistica; i modelli presentati hanno la presunzione di poter addirittura estrarre questi *topic* da un insieme di documenti, ma nonostante questo rimangono ancora un'entità astratta. È giunto il momento di dare forma a questi argomenti, trattati negli articoli presentati alla conferenza NIPS 2015. In tabella 4.1 sono presentate le parole classificate come le più influenti per ognuno dei dieci *topic* identificati dal modello RSM penalizzato (con valore $\lambda = 0.1$):

Tabella 4.1: Topic estratti

Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8	Topic 9	Topic 10
neural	matrix	risk	gpu	magnitude	approach	state	regret	causal	gradient
computer	sampling	vertex	results	source	set	hessian	recovery	clustering	layer
predict	generated	queries	learning	dynamical	gradient	note	subspace	kernels	deep
vs	multiple	submodular	stochastic	black	showed	logistic	pca	walk	learning
terms	variational	vertices	batch	average	input	multi	principal	cluster	model
task	modeling	classifier	gradient	identifies	data	output	kf	clusters	space
classification	sampler	label	operator	d2	address	network	projection	graphs	inference
variable	also	lemma	total	statistics	units	found	lasso	topic	equation
conditioned	recognition	graph	proposed	architecture	error	recurrent	eigenvalues	rnn	demonstrate
natural	parameters	margin	two	mcmc	cost	algorithm	kernel	edges	also
embedding	list	graphs	subject	evaluation	lower	improves	rank	topics	following
including	networks	metric	bound	gaussian	model	motivates	inducing	human	given
give	synthetic	game	regularization	functional	sparse	data	sparse	communities	time
probabilistic	samples	items	also	points	dataset	based	matrices	cut	result
supervised	employ	clustering	processing	consider	relative	language	proximal	detection	top
marginal	le	causal	spaces	dynamics	framework	min	tensor	parent	neurons
answers	online	item	output	dimensionality	bengio	experiments	rn	ordering	factors
search	review	classifiers	viewed	hierarchical	entropy	observation	matrix	spatial	methods
popular	weight	recovery	questions	topic	see	code	completion	graph	interest
network	representation	theorem	hessian	region	networks	empirical	covariance	fn	workshop

Il numero di *topic*, pari a 10 in questo caso, è stato scelto sulla base della valutazione in termini di *perplexity*. L'interpretazione rimane oggettiva, ma si è comunque cercato di identificare i temi trattati evidenziando alcune delle parole più caratteristiche: gli argomenti spaziano dalla riduzione della dimensionalità al *clustering* su grafi, dal più generico *deep learning* alla minimizzazione di funzioni *submodulari* per variabili binarie.

Lo scopo iniziale di queste analisi però non era la semplice estrazione delle parole identificative di un argomento, ma la classificazione dei documenti in base ai temi trattati, nella speranza di migliorarne l'archiviazione, la ricerca e l'esplorazione. È fondamentale avere dunque uno strumento intuitivo per scoprire quali dei *topic* estratti caratterizzano un testo, a prescindere dall'utilizzo del modello LDA o RSM. Il punto focale risiede nella specificazione, entrambi infatti considerano un documento come una mistura di *topic*; partendo da questi presupposti si è deciso di presentare graficamente le probabilità di mistura, e quindi le proporzioni di temi trattati, per ognuno dei testi presi in considerazione. In figura 4.5 per chiarezza grafica sono presentati solo 20 documenti: ognuno di essi è associato ad una riga, mentre i cerchi in colonna rappresentano le probabilità di mistura. Focalizzandosi sull'area del cerchio, proporzionale alla probabilità di aver osservato quel *topic* in quel documento, è immediato identificarne i temi più o meno trattati.

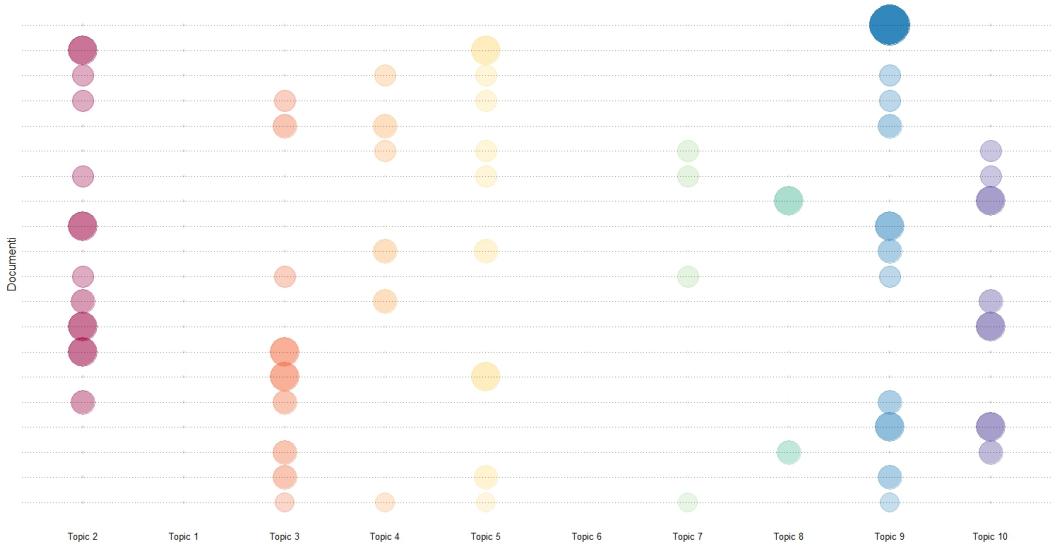


Figura 4.5: Rappresentazione grafica delle probabilità di mistura dei *topic* all'interno dei documenti: ogni riga corrisponde ad un documento, mentre l'area dei cerchi è proporzionale alla probabilità.

Oltre al desiderio di estrarre informazioni da un insieme di dati testuali, elencando un insieme di termini non del tutto casuali, lo scopo della tesi è soprattutto quello di esplorare, analizzare e contribuire, dal punto di vista statistico, a questo ramo della ricerca riguardante il *topic modeling*. L'interesse prin-

cipale è rivolto al confronto, sia teorico che inferenziale, di due tipi di modelli grafici probabilistici: il Bayesian Network e il Markov Random Field. Particolare attenzione è stata dedicata all'estensione del modello Replicated Softmax, a cui è stata aggiunta una distribuzione *a priori* sui parametri: lo scopo è quello di limitare problemi di identificabilità dovuti alla numerosità dei parametri e di rendere più robuste le stime. Restano sicuramente aperte molte strade ancora da esplorare: una penalizzazione di tipo $L1$ che consentirebbe anche una fase di selezione delle variabili, l'estensione del modello Replicated Softmax aggiungendo più strati nascosti, allo scopo di individuare relazioni più complesse tra parole e *topic*, e non ultimo l'implementazione più efficiente degli algoritmi per aumentarne la scalabilità. Nel complesso, l'insieme di metodologie presentate è volutamente ampio, spaziando dalle semplici analisi descrittive alla più complessa modellazione grafica, con uno sguardo a tratti Bayesiano a tratti frequentista, e solo in parte informatico.

Appendice A

Codici

Codice 1: Pre-processamento

```
import pandas as pd
import nltk
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import CountVectorizer
from nltk.tokenize import RegexpTokenizer
import string

papers = pd.read_csv("Indirizzo del file", encoding = "utf-8")

# Creazione della lista di stopwords

en_stop = list(stopwords.words('english'))
letters = list(string.ascii_lowercase)
numbers = list(map(str, range(0, 2020)))

# Stopwords aggiuntive

strangechar = ['xi', 'x1', 'tr', 'exp', 'gp', 'ep', 'p0', 'eq', 's0', 'xt', '
    xn', 'lp', 'xk', 'k2', 'log', 'use', 'yi', '00', '000', 'z1', 'abs', 'et'
    , 'al']
en_stop = list(set(en_stop + letters + numbers + strangechar))

# Conteggio delle parole
countvec = CountVectorizer(encoding='utf-8', lowercase='TRUE', stop_words =
    en_stop, min_df = 20)
```

Codice 2: Modello online LDA con mini-batch

```
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
from sklearn.decomposition import NMF, LatentDirichletAllocation

# Creazione della Document Term Matrix
tf = countvec.fit_transform(papers['PaperText'])

# Modello online LDA con mini-batch
# NB: nel caso si voglia utilizzare l'algoritmo standard Variational Bayes,
    basta modificare learning_method='batch'
lda = LatentDirichletAllocation(n_topics=5, max_iter=i, learning_method='
    online', learning_offset=50., random_state=0)
lda.fit(tf)

# Definizione della funzione per estrarre i topic
def print_top_words(model, feature_names, n_top_words):
        for topic_idx, topic in enumerate(model.components_):
                print("Topic #%d:" % topic_idx)
                print(" ".join([feature_names[i]
                        for i in topic.argsort()[:-n_top_words - 1:-1]]))
        print()

# Estrazione dei topic
print("\nTopics in LDA model:")
tfidf_feature_names = countvec.get_feature_names()
print_top_words(lda, tfidf_feature_names, n_top_words)
```

Codice 3: Algoritmo RSM con penalizzazione

```
import scipy as sp
import numpy as np

def train(data, units, epochs = 10, iter = 10, lr = 0.0001, weightinit = 0.1,
     momentum = 0.9, btsz = 10, decay = 0.0001):
        """
        Replicated Softmax con penalizzazione
        @param data: Document Term Matrix
        @param units: Numero di topic
        @param epochs: Numero di iterazioni
        @param lr: Learning Rate
        @param weightinit: Fattore di scala per i pesi iniziali
        @param momentum: Momentum
        @param btsz: Grandezza delle mini-batch
        """
        dictsize = data.shape[1]
```

```
        # Inizializzazione casuale dei pesi
        w_vh = weightinit * np.random.randn(dictsize, units)
        w_v = weightinit * np.random.randn(dictsize)
        w_h = np.zeros((units))
        # Inizializzazione degli aggiornamenti dei pesi
        wu_vh = np.zeros((dictsize, units))
        wu_v = np.zeros((dictsize))
        wu_h = np.zeros((units))
        delta = lr/btsz
        batches = data.shape[0]/btsz
        print "updates per epoch: %s | total updates: %s" % (batches, batches
            *epochs)
        words = np.sum(data)
        for epoch in xrange(epochs):
                lik = 0
                # Permutazione dei dati
                np.random.shuffle(data)
                # Ciclo per le mini-batch
                for b in xrange(batches):
                        start = b * btsz
                        v1 = data[start : start+btsz]
                        D = v1.sum(axis=1)
                        # Strato nascosto
                        h1 = sigmoid((np.dot(v1, w_vh) + np.outer(D, w_h)))
                        v2 = v1; h2 = h1
                        for i in xrange(iter):
                                (v2,h2,z) = cdn (v2,h2,w_vh,w_v,w_h,D)
                                if i == 0:
                                        lik += z
                        # Aggiornamento dei pesi
                        penal = ((btsz**2)/float(data.shape[0]))*decay*w_vh.
                            sum()
                        wu_vh = wu_vh * momentum + np.dot(v1.T, h1) - np.dot(
                            v2.T, h2) - penal
                        wu_v = wu_v * momentum + v1.sum(axis=0) - v2.sum(axis
                            =0)
                        wu_h = wu_h * momentum + h1.sum(axis=0) - h2.sum(axis
                            =0)

                        w_vh += wu_vh * delta
                        w_v += wu_v * delta
                        w_h += wu_h * delta
                # Lower bound della perplexity
                ppl = np.exp (- lik / words)
```

```
                print "Epoch[%2d] : PPL = %.02f [iter=%d]" % (epoch, ppl,iter
                    )
        return {"w_vh" : w_vh,
                "w_v"  : w_v,
                "w_h"  : w_h,
                "rate"   : lr,
                "iter"   : iter,
                "batch"  : btsz,
                "epoch"  : epochs,
                "init"   : weightinit,
                "moment" : momentum,
                "ppl"    : ppl
                }

def cdn (v1,h1,w_vh,w_v,w_h,D):
        """
        Algoritmo Contrastive Divergence
        """
        lik = 0
        btsz = v1.shape[0]
        # Strato visibile
        v2 = np.dot(h1, w_vh.T) + w_v
        tmp = np.exp(v2)
        sum = tmp.sum(axis=1)
        sum = sum.reshape((btsz,1))
        v2pdf = tmp / sum
        # log verosimiglianza
        lik += np.nansum(v1 * np.log(v2pdf))
        # Estrazione dalla multinomiale
        v2 *= 0
        for i in xrange(btsz):
                v2[i] = np.random.multinomial(D[i],v2pdf[i],size=1)
        # Strato nascosto
        h2 = sigmoid(np.dot(v2, w_vh) + np.outer(D, w_h))
        return (v2,h2,lik)

def sigmoid(X):
        """
        Funzione sigmoide
        """
        return (1 + sp.tanh(X/2))/2
```

Codice 4: Modello RSM

```
import pandas as pd
```

```
# Creazione della DTM

DTM = pd.DataFrame(countvec.fit_transform(papers['PaperText']).toarray(),
    columns = countvec.get_feature_names())
X = DTM.as_matrix()

# Stima
mod = train(X, 10, epochs=500, decay = 0.1, weightinit=0.2)
```

Codice 5: Estrazione dei topic

```
import numpy as np

#Ciclo per l'estrazione dei topic
for x in range(0, 10):
        print("Topic %d" % (x + 1))
        print(DTM.columns.values[sorted(range(len(mod['w_vh'][:, x])), key=
            lambda i:mod['w_vh'][:, x][i], reverse = True)[0:20]])
```

Bibliografia

Bengio, Yoshua (2012). «Practical recommendations for gradient-based training of deep architectures». In: *Neural Networks: Tricks of the Trade*. Springer, pp. 437–478.

Blei, D. et al. (2004). «Hierarchical topic models and the nested Chinese restaurant process». In: *Advances in Neural Information Processing Systems 16, Cambridge, MA, MIT Press*.

Blei et al. (2003). «Latent Dirichlet Allocation». In: *Journal of Machine Learning Research, 3, 993-1022*.

Carreira-Perpinan, Miguel A e Geoffrey Hinton (2005). «On Contrastive Divergence Learning.» In: *AISTATS*. Vol. 10. Citeseer, pp. 33–40.

David, Mimno, Li Wei e McCallum Andrew (2007). «Mixtures of Hierarchical Topics with Pachinko Allocation». In: *Proceedings of the* 24^{th} *International Conference on Machine Learning*.

Friedman, Jerome, Trevor Hastie e Robert Tibshirani (2001). *The elements of statistical learning*. Vol. 1. Springer series in statistics Springer, Berlin, da guardare.

Geman, Stuart e Donald Geman (1984). «Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images». In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 6.6, pp. 721–741. ISSN: 0162-8828. DOI: http://doi.ieeecomputersociety.org/10.1109/TPAMI.1984.4767596.

Griffiths, Thomas L e Mark Steyvers (2002). «A probabilistic approach to semantic representation». In: *Proceedings of the 24th annual conference of the cognitive science society*. Citeseer, pp. 381–386.

Hamner, Ben (2015). «NIPS 2015 Papers». In: Explore and analyze this year's NIPS papers. Kaggle. URL: https://www.kaggle.com/benhamner/nips-2015-papers.

Hinton, Geoffrey (2010). «A practical guide to training restricted Boltzmann machines». In: *Momentum* 9.1, p. 926.

Hinton, Geoffrey E e Ruslan R Salakhutdinov (2009). «Replicated softmax: an undirected topic model». In: *Advances in neural information processing systems*, pp. 1607–1614.

Hoffman, Matthew D et al. (2013). «Stochastic variational inference.» In: *Journal of Machine Learning Research* 14.1, pp. 1303–1347.

Hoffman, Matthew, Francis R Bach e David M Blei (2010). «Online learning for latent dirichlet allocation». In: *advances in neural information processing systems*, pp. 856–864.

Koller, Daphne e Nir Friedman (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

Landauer, T. et al. (2004). «Latent Semantic Analysis: A Road to Meaning». In: *Laurence Erlbaum, Probabilistic Topic Models*.

Lim, Kar Wai, Changyou Chen e Wray Buntine (2013). «Twitter-Network Topic Model: A Full Bayesian Treatment for Social Network and Text Modeling». In: *NIPS2013 Topic Model workshop*.

Maaten, Laurens van der e Geoffrey Hinton (2008). «Visualizing data using t-SNE». In: *Journal of Machine Learning Research* 9.Nov, pp. 2579–2605.

Tierney, L. (1994). «Markov Chains for Exploring Posterior Distributions». In: *Annals of Statististics*.

Van Der Maaten, Laurens (2014). «Accelerating t-SNE using tree-based algorithms.» In: *Journal of machine learning research* 15.1, pp. 3221–3245.

Wei, Li e Blei Davidand McCallum Andrew (2007). «Nonparametric Bayes Pachinko Allocation». In: *CoRR*.

Wei, Li e McCallum Andrew (2007). «Pachinko Allocation: DAG-Structured Mixture Models of Topic Correlations». In: *Proceedings of the* 23^{rd} *International Conference on Machine Learning*.