VARIATIONAL EM ALGORITHM FOR LDA

HONGFEI LI

CONTENTS

I implemented LDA model with Variational EM algorithm in python. I followed Blei's algorithm exactly. And Blei's C implementation of this algorithm is heavily referenced.

SIMULATION

DATA:

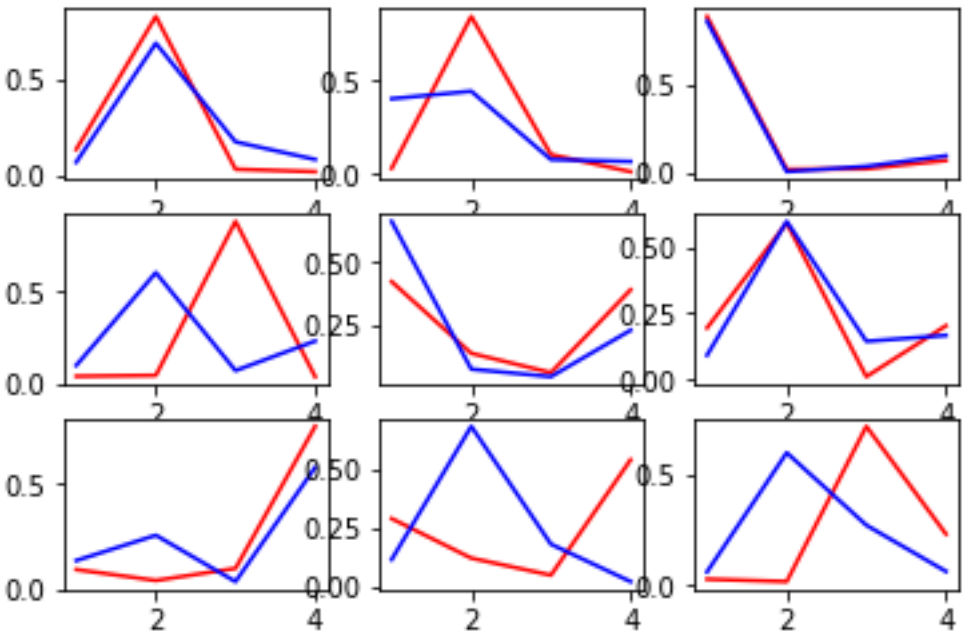
- 10 documents, each contains 20 words
- 4 topics
- Vocabulary size: 80

RUNTIME:

Converged after 17 iterations, 0.5 seconds

Topics distribution for first 9 documents:

Red: inferred distribution of topics Blue: True distribution of topics



REAL DATA: GENERATIVE MODEL

DATA: REUTERS-21578

I used the famous dataset *Reuters-21578 "ApteMod" Corpus* (imported from nltk). It contains contains 10,788 news documents totaling 1.3 million words with 90 topics. Unfortunately, my algorithm is too slow to train with so many documents. So I randomly sampled a small amount of documents from this dataset and split them into 80% training documents and 20% testing documents.

Preprocessing:

- Removed stop words
- Stemmed tokens
- Ignored new words that appear in test documents but not in our training vocabulary

RUNTIME: SLOW

I found that my algorithm runs exponentially slower while the training data size and number of topics increases. With more documents to train, more topics to learn, each EM iteration takes longer to run, and it takes more iterations to converge.

I compared the runtime of my LDA and a python LDA package, which uses collapsed Gibbs sampling instead of variational inference. The package beat me by a huge margin.

Number of Documents	My LDA	Package LDA
16	69 seconds (Converged)	6.5 seconds (2000 iteration)
40	7.85 mins (200 iteration)	7.1 seconds (2000 iteration)
80	2 Hours (1000 iteration)	9.16 seconds (2000 iteration)

My algorithm runs very slow with a lot of topics, possibly due to my inefficient implementation. I followed Blei's algorithm exactly. It takes way too long when trained with hundreds of documents. Although I am aware that parallel computing will improve the speed drastically, I haven't learnt how to code it.

TRAINING:

I chose the topic that has highest probability as the “inferred topic” of each document and printed out the top ten most likely words of that topic. Table 1 shows part of the inference of the training documents. It seemed that, the model did distinguish documents that belong to different categories. But the model also split the category ‘acq’, (means corporate mergers/acquisitions), into smaller clusters. Roughly speaking, the model successfully summarized each topic.

Table 1: Trained on 40 documents, here are the first ten

True Label	Index of Inferred Topic	Top ten words of the topic
'acq'	17	['pct', 'lt', 'unit', 'group', 'stake', 'said', 'dlr', 'airleas', 'may', 'price']
'acq'	17	['pct', 'lt', 'unit', 'group', 'stake', 'said', 'dlr', 'airleas', 'may', 'price']
'earn'	18	['vs', 'ct', 'net', 'shr', 'loss', 'mln', 'profit', 'dlr', 'rev', 'lt']
'acq'	3	['lt', 'share', 'ltd', 'said', 'corp', 'san', 'miguel', 'compani', 'buy', 'busch']
'acq'	16	['pct', 'march', 'februari', 'year', 'said', 'compar', 'china', 'consum', 'bank', 'institut']
'interest'	0	['pct', 'rate', 'polici', 'pledg', 'balladur', 'accord', 'respect', 'louvr', 'monetari', 'urg']
'potato'	5	['futur', 'market', 'potato', 'trade', 'physic', 'lpfa', 'stg', 'said', 'pmb', 'price']
'cotton'	10	['mln', 'cotton', 'bale', 'year', 'said', 'pakistan', 'export', 'fiscal', 'last', 'interest']
'earn'	18	['vs', 'ct', 'net', 'shr', 'loss', 'mln', 'profit', 'dlr', 'rev', 'lt']
'money-fx', money-supply'	14	['rate', 'polici', 'said', 'money', 'bundesbank', 'bank', 'suppli', 'mueller', 'monetari', 'interest']
…….	…….	…….

TESTING:

The model was trained on 40 documents. I fed 10 new documents into it. Table 2 shows the inferred topic of these test documents. Looking at the top ten words of inferred topics, I think the model captured the topic of most document.

Table 2: (Trained on 40 documents,) Test on 10 documents

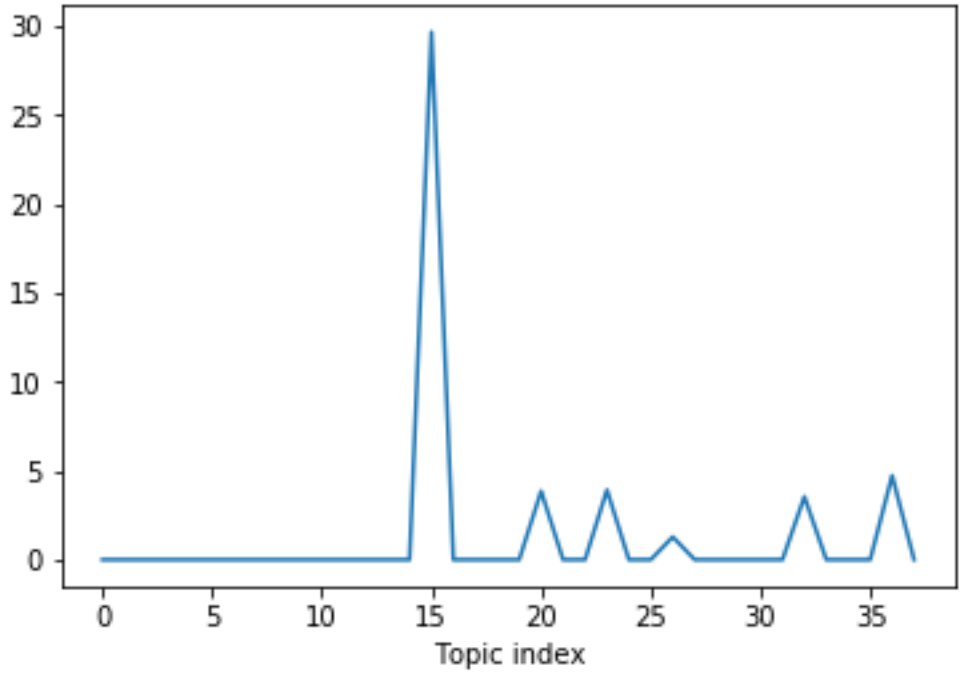
True Label	Index of Inferred Topic	Top ten words of the topic
'bop', 'trade'	1	['japan', 'trade', 'japanes', 'dlr', 'said', 'part', 'smart', 'last', 'made', 'state']
'earn'	18	['vs', 'ct', 'net', 'shr', 'loss', 'mln', 'profit', 'dlr', 'rev', 'lt']
'grain', 'wheat'	10	['mln', 'cotton', 'bale', 'year', 'said', 'pakistan', 'export', 'fiscal', 'last', 'interest']
'acq'	2	['said', 'harper', 'analyst', 'row', 'share', 'week', 'sharehold', 'storag', 'next', 'belli']
'money-fx'	14	['rate', 'polici', 'said', 'money', 'bundesbank', 'bank', 'suppli', 'mueller', 'monetari', 'interest']
'earn'	18	['vs', 'ct', 'net', 'shr', 'loss', 'mln', 'profit', 'dlr', 'rev', 'lt']
'cpi'	16	['pct', 'march', 'februari', 'year', 'said', 'compar', 'china', 'consum', 'bank', 'institut']
'interest', 'money-fx'	14	['rate', 'polici', 'said', 'money', 'bundesbank', 'bank', 'suppli', 'mueller', 'monetari', 'interest']
'carcass', 'l-cattle', 'livestock', 'sugar'	3	['lt', 'share', 'ltd', 'said', 'corp', 'san', 'miguel', 'compani', 'buy', 'busch']
'earn'	6	['vs', 'mln', 'one', 'dlr', 'bonu', 'issu', 'dividend', 'share', 'propos', 'dresdner']

I trained another model on 80 training documents, and increased the iteration limit to 1000. It hit the limit before meeting the convergence threshold (1e-4). I test it on 20 new documents. Table 3 shows the result. I think the top words make sense.

Table 3: (Trained on 80 documents) Test on 20 documents		
True Label	Index of Inferred Topic	Top ten words of the topic
[['money-fx']	2	['japan', 'trade', 'dlr', 'japanes', 'part', 'said', 'smart', 'made', 'state', 'last']
['earn']	34	['vs', 'net', 'ct', 'mln', 'shr', 'loss', 'dlr', 'year', 'profit', 'oper']
['gnp']	12	['said', 'dollar', 'yen', 'nakason', 'nation', 'japan', 'minist', 'pari', 'trade', 'bag']
['grain''rice' 'wheat']	15	['credit', 'nil', 'export', 'mln', 'meal', 'soybean', 'wheat', 'import', 'program', 'product']
['ship']	5	['prefer', 'cargo', 'american', 'repeal', 'lugar', 'costli', 'law', 'senat', 'farmer', 'ship']
['earn']	34	['vs', 'net', 'ct', 'mln', 'shr', 'loss', 'dlr', 'year', 'profit', 'oper']
['earn']	35	['lt', 'ct', 'march', 'april', 'dunhil', 'dividend', 'said', 'stock', 'capit', 'total']
['grain']	37	['futur', 'market', 'potato', 'trade', 'physic', 'stg', 'pmb', 'lpfa', 'said', 'harri']
['earn']	34	['vs', 'net', 'ct', 'mln', 'shr', 'loss', 'dlr', 'year', 'profit', 'oper']
['earn']	34	['vs', 'net', 'ct', 'mln', 'shr', 'loss', 'dlr', 'year', 'profit', 'oper']
['earn']	34	['vs', 'net', 'ct', 'mln', 'shr', 'loss', 'dlr', 'year', 'profit', 'oper']
['acq']	6	['bank', 'atico', 'research', 'cottrel', 'said', 'close', 'dlr', 'intercontinent', 'halcyon', 'acquisit']
['earn']	35	['lt', 'ct', 'march', 'april', 'dunhil', 'dividend', 'said', 'stock', 'capit', 'total']
['earn']	34	['vs', 'net', 'ct', 'mln', 'shr', 'loss', 'dlr', 'year', 'profit', 'oper']
['acq']	35	['lt', 'ct', 'march', 'april', 'dunhil', 'dividend', 'said', 'stock', 'capit', 'total']
['grain']	22	['billion', 'mark', 'bank', 'profit', 'said', 'earn', 'group', 'oper', 'deutsch', 'christian']
['earn']	34	['vs', 'net', 'ct', 'mln', 'shr', 'loss', 'dlr', 'year', 'profit', 'oper']
['earn']	34	['vs', 'net', 'ct', 'mln', 'shr', 'loss', 'dlr', 'year', 'profit', 'oper']
['acq']	16	['said', 'caledonian', 'buy', 'lt', 'pct', 'china', 'bank', 'own', 'price', 'seedman']
['earn']]	15	['credit', 'nil', 'export', 'mln', 'meal', 'soybean', 'wheat', 'import', 'program', 'product']

ISSUES: PROBLEM WITH ‘ACQ’ AND MULTI-LABEL CASES

1. ‘earn’ and ‘acq’ are two big topics in Reuters-21578. The model did a good job on capturing ‘earn’, despite splitting some of ‘earn’ into a new group. However, the model performed bad when it comes to ‘acq’. I think a possible reason is overfitting. I set the number of topics in my model equal to the number of unique true labels, which is too much for a highly unbalanced dataset like this.
2. It is hard to get the multi-label cases right. For example, the inferred topic distribution for the document ['grain' 'rice' 'wheat'] looks like the following graph. The model doesn’t think this document has 3 labels—it’s either 1 or 5. The model will not recover the labels of the documents, but design its own latent variables.



REAL DATA: THREE-CLASSES CLASSIFICATION

Here I use the LDA as a feature reduction method.

DATA:

Documents randomly sampled from Reuters-21578.

LABEL:

Regrouped into three classes: ‘earn’, ‘acq’, ‘others’

CLASSIFIER:

One-versus-all SVM

RESULT:

Train Size	Test Size	Feature Space	Runtime(min)	Accuracy
800	200	10	9.5	75%
800	200	20	19.5	74.5%
2400	600	10	22.7	74%
7769	3019	10	82.2	76%

I did several experiments and found that the accuracy was always around 75%.

Look closer at the last experiment, I realize that it is consistent with the previous generative model. 1/6 of ‘earn’ is classified as ‘others’. And ‘acq’ is split into many groups. I think it is because of the intrinsic diversity of the ‘acq’ group.

Count		True label			
		‘earn’	‘acq’	‘others’	Sum
pred	‘earn’	891	158	14	1063
	‘acq’	8	201	3	212
	‘others’	188	357	1199	1744
	Sum	1087	716	1216	3019

Fraction		True label			
		‘earn’	‘acq’	‘others’	Sum
pred	‘earn’	0.295	0.052	0.005	0.352
	‘acq’	0.003	0.067	0.001	0.070
	‘others’	0.062	0.118	0.397	0.578
	Sum	0.360	0.237	0.403	1.000