4F13 Coursework 3: Latent Dirichlet Allocation

Word Count: 999

December 1, 2022

1 Problem a)

Listing 1 The code for calculating the maximum likelihood multinomial over words

```
# Calculate the total number of unique words in A and B and the number of
documents in A
W= np.max([np.max(A[:, 1]), np.max(B[:, 1])])
D_A = np.max(A[:, 0])
# Caluate the MLE estimate of the multinomial distribution parameter
beta_MLE_multinomial = []
for i in range(1,W+1):
    beta_MLE_multinomial.append(np.sum(A[:,2][A[:,1]==i])/np.sum(A[:,2]))

beta_MLE_multinomial = np.array(beta_MLE_multinomial)
```

Listing 2 The code for the histogram of top 20 largest probabilities

```
# Find the top 20 highest MLE estimate together with their word id
top_20_id = np.flip(np.argsort(beta_MLE_multinomial))[:20]
top_20_prob = np.flip(np.sort(beta_MLE_multinomial))[:20]
# barplot of the top 20 probabilities
plt.barh(np.concatenate(V[np.flip(top_20_id)][:,0]),np.flip(top_20_prob))
plt.title('The 20 largest MLE estimated probability items of mutinomial
                                        model')
plt.xlabel('MLE estimated probabilities')
plt.ylabel('Word')
```

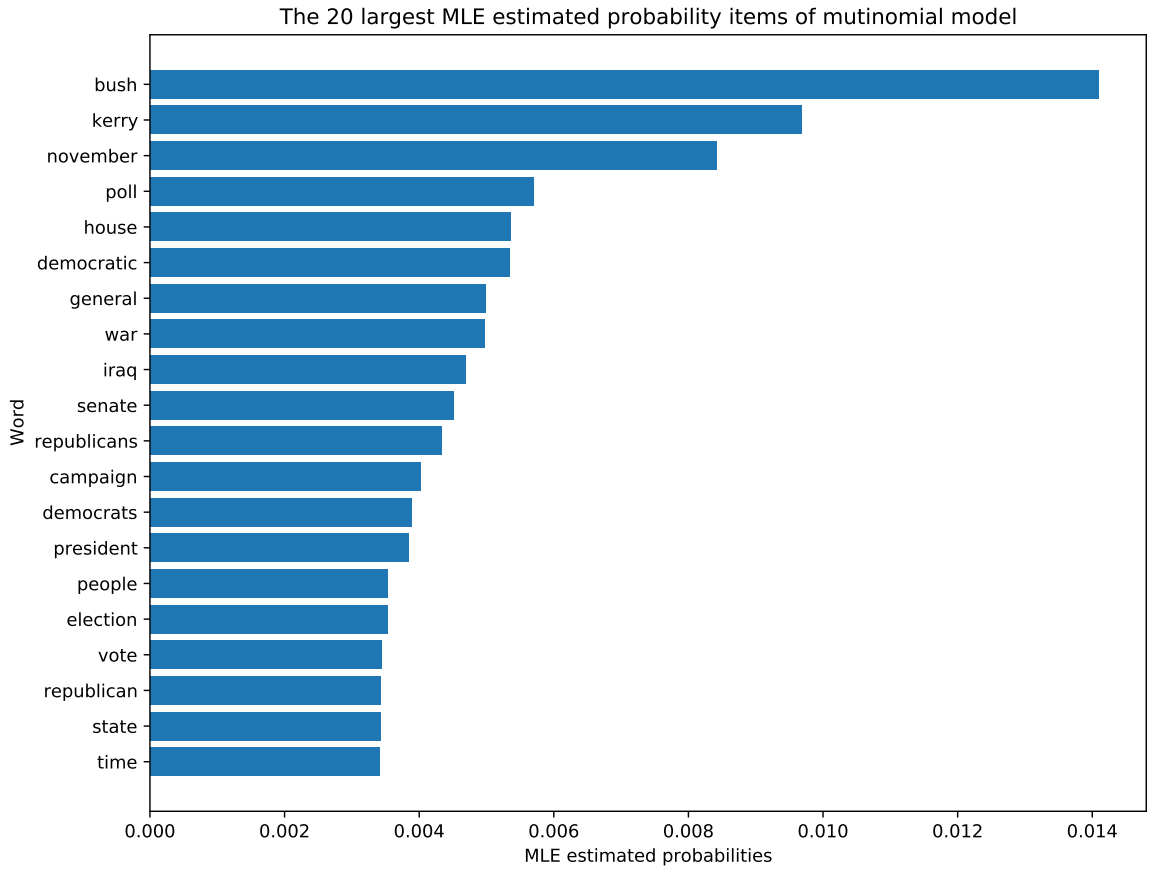

Figure 1: Barplot of the top 20 highest MLE estimated probabilities of the multinomial model

Assume we have D documents from a vocabulary of M unique words for the test set. Let $\mathrm{N_d}$ be the number of words in document d, $\mathrm{w_{nd}}$ be the n-th word in document d and $\mathrm{w_{nd}} \sim \mathrm{Cat}(\hat{\boldsymbol{\beta}})$, where $\hat{\boldsymbol{\beta}}$ is the MLE as returned above. If we let $\mathrm{c_m}$ be the total number of vocabulary m in all D documents, the log probability of the test set is:

$$\log(p(\{\mathrm{w_{nd}}\}|\hat{\boldsymbol{\beta}})) = \sum_{\mathrm{m}=1}^{\mathrm{M}} \mathrm{c_m} \log(\hat{\beta}_{\mathrm{m}}) \tag{1.1}$$

The highest log probability is achieved when the test data set has only one document with one word which is 'bush' (it has the largest $\hat{\beta}_m$ of 0.0140972). The highest log probability is about -4.26. Moreover, the lowest log probability is $-\infty$, which is obtained when there are words that haven't appeared in the train data set appearing in the test set (In this case, there must be one $\hat{\beta}_m = 0$ making $\log(\hat{\beta}_{\mathrm{m}}) = -\infty$).

The above implies that using MLE of the multinomial model does not work well if there are rare words in the documents as the MLE of the $\hat{\beta}_{\mathrm{m}}$ of these rare words would be very small or even become 0 if these words does not appear in training documents, which would make the test log probability small (or even $-\infty$) if these rare words appear in the test documents.

2 Problem b)

Given a Dir(α) prior on the word probabilities $\boldsymbol{\beta}$, the posterior distribution of $\boldsymbol{\beta}$ is:

$$
\begin{aligned}
p(\boldsymbol{\beta}|\alpha, \mathcal{D}) &\propto (\prod_{\mathrm{m}=1}^{\mathrm{M}} \beta_{\mathrm{m}}^{\alpha-1})(\prod_{\mathrm{m}=1}^{\mathrm{M}} \beta_{\mathrm{m}}^{\mathrm{c_m}}) \\
&\propto \prod_{\mathrm{m}=1}^{\mathrm{M}} \beta_{\mathrm{m}}^{\mathrm{c_m}+\alpha-1} \\
&\propto \mathrm{Dir}(\alpha + \mathbf{c}), \quad \mathbf{c}_{\mathrm{m}} = \mathrm{c}_{\mathrm{m}} \text{ is the total number of vocabulary m in all the training documents.}
\end{aligned}
$$

So $p(\boldsymbol{\beta}|\alpha, \mathcal{D}) \sim \mathrm{Dir}(\alpha + \mathbf{c})$.

Listing 3 The code for Bayesian inference using a symmetric Dirichlet prior (together with the code for finding the number of word counts of each word in the training documents)

```
# Number of word counts in the training document for each of the unique
                                        words
C = []
for i in range(1,W+1):
    C.append(np.sum(A[:,2][A[:,1]==i]))
C = np.array(C)

# posterior beta of symmetric Dirichlet prior with alpha = 0.1
alpha_small = 0.1
alpha_small_posterior = alpha_small + C

# posterior beta of symmetric Dirichlet prior with alpha = 10
alpha_large = 10
alpha_large_posterior = alpha_large + C
```

The two expressions of predictive word probabilities are:

MLE:

$$p(\mathrm{w}_{\mathrm{next}} = \mathrm{m}|\hat{\boldsymbol{\beta}}) = \hat{\beta}_{\mathrm{m}}, \quad \text{where } \hat{\beta}_{\mathrm{m}} = \hat{\boldsymbol{\beta}}_{\mathrm{m}} \text{ is the MLE of the probability of vocabulary m.}$$

Bayesian:

$$
\begin{aligned}
p(\mathrm{w}_{\mathrm{next}} = \mathrm{m}|\alpha, \mathcal{D}) &= \int p(\mathrm{w}_{\mathrm{next}} = \mathrm{m}|\boldsymbol{\beta}) p(\boldsymbol{\beta}|\alpha, \mathcal{D}) d\boldsymbol{\beta} \\
&= \int \beta_{\mathrm{m}} p(\boldsymbol{\beta}|\alpha, \mathcal{D}) d\boldsymbol{\beta} \\
&= \mathrm{E}_{p(\boldsymbol{\beta}|\alpha,\mathcal{D})}(\beta_{\mathrm{m}}) \\
&= \frac{\alpha + \mathrm{c}_{\mathrm{m}}}{\sum_{\mathrm{m}'=1}^{\mathrm{M}} \alpha + \mathrm{c}_{\mathrm{m}'}}
\end{aligned}
$$

The MLE approach use a point estimate of the word probability vector $\boldsymbol{\beta}$ for model fitting and the predictive word probability is simply the MLE $\hat{\boldsymbol{\beta}}$, which performs well for common words but performs badly for rare words (the test log probability would be $-\infty$ if words not appearing in training documents appear in test documents).

The Bayesian model, instead, gives a posterior probability distribution over the word probabilities $\boldsymbol{\beta}$, combining information from the prior and the likelihood. The predictive word probability in Bayesian model is obtained by averaging over all possible $\boldsymbol{\beta}$. For rare words, as they have small c_m [1], both small and large α would in some sense affect their posterior and predictive probabilities. If α is much larger than c_m, than their posterior and predictive probabilities would be dominated by the prior. However, the posterior and predictive probabilities of Bayesian model would not be 0 even if the word has never appeared in the training documents (due to the Dirichlet prior). For common words with larger c_m, the likelihood of the data would dominate the posterior and predictive probabilities for small α and for large α the Dirichlet prior and the likelihood of the data would have approximately the same contributions to the posterior and the predictive probabilities.

3 Problem c)

Categorical distribution assumes that the words in documents are observed in some order while multinomial distribution does not care about the order of these words. Therefore, as the words in these documents are actually observed in order [2], the categorical distribution function is more appropriate for calculating the log probability[3] (Also, the categorical and multinomial distribution only differ by the combinatorial factor and the combinatorial factor would not affect the inference results of the model, thus we can simply ignore it to make the calculation of log probability easier). For test document with ID 2001 ($\alpha = 1$), it gives a log probability of -3688.62117:

Listing 4 The code for log probability of test document 2001

```
# Number of word counts in the training document for each unique word.
C = []
for i in range(1,W+1):
    C.append(np.sum(A[:,2][A[:,1]==i]))
C = np.array(C)
# log probability for doc ID 2001
doc_2001 = B[B[:,0]==2001]
w_2001 = doc_2001[:,1]
c_2001 = doc_2001[:,2]
lp_2001 = np.dot(c_2001, (np.log(1+C)-np.log(np.sum(1+C)))[w_2001-1])
```

The per-word perplexity of a document is given by:

perplexity $= \exp\left(-\frac{l}{N}\right)$, l is the log joint probability over all words, N is the total number of words.

Using $\alpha = 1$, the perplexity for the test document with ID 2001 is 4373.11099:

Listing 5 The code for perplexity of test document 2001

```
# perplexity for doc ID 2001 with alpha = 1
np_2001 = np.sum(c_2001)
perplexity_2001 = np.exp(-lp_2001/np_2001)
```

[1] As the likelihood does not provide much information about these words

[2] The bag of words is only for convenience of calculation here

[3] To calculate the log probability, I use formula (1.1) together with the predictive probability for Bayesian model derived in b)

Similarly, the per-word perplexity over all documents in B is calculated and summarised in a barplot ($\alpha = 1$):

Listing 6 The code for perplexity of B and the barplot of all perplexities of documents in B

```
# perplexity per words for all documents in B with alpha = 1
perplexity_B = []
for i in np.unique(B[:,0]):
    w_i = B[B[:,0]==i,1]
    c_i = B[B[:,0]==i,2]
    lp_i = np.dot((np.log(1+C)-np.log(np.sum(1+C)))[w_i-1],c_i)
    np_i = np.sum(c_i)
    perplexity_B.append(np.exp(-lp_i/np_i))
# Barplot for documents in B
plt.figure(figsize=(18, 12), dpi=150)
plt.bar(np.unique(B[:,0]), np.array(perplexity_B))
plt.xlabel("document ID in B")
plt.ylabel("perplexity")
```

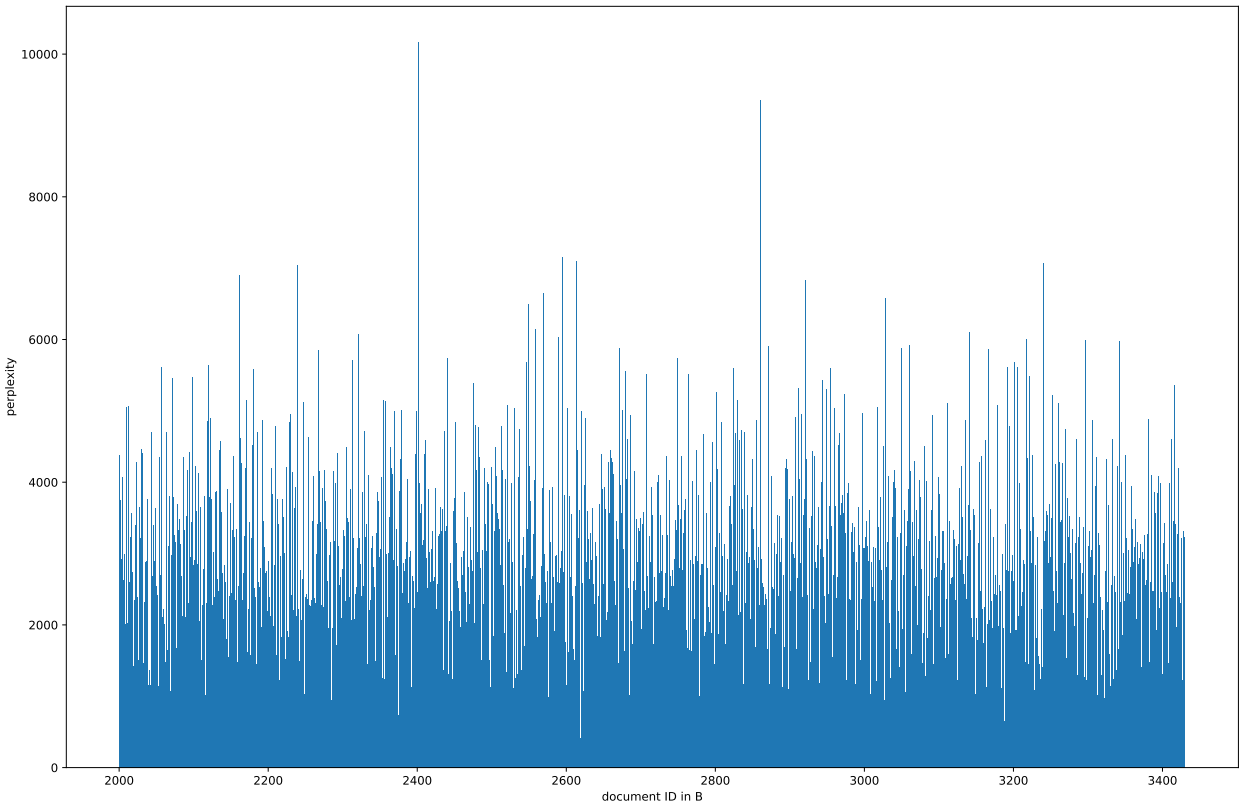

Figure 2: Barplot of the perplexities for all documents in B

Here, the Bayesian model assumes that all words in different documents are sampled independently from the same Cat($\boldsymbol{\beta}$) distribution such that the posterior distribution of $\boldsymbol{\beta}$ reflects the aggregation of all documents, which means that the model does not specialise for each docu-

ment (each document may have different word distributions[4]), which explains the differences in perplexities for different documents.

For a uniform multinomial model with M unique vocabulary and N words, the perplexity is:

$$\begin{aligned}\text{perplexity} &= \exp\left(-\frac{\log((\frac{1}{\mathrm{M}})^{\mathrm{N}})}{\mathrm{N}}\right)\\ &= \exp\left(-\log\left(\frac{1}{\mathrm{M}}\right)\right)\\ &= \exp\left(\log(\mathrm{M})\right)\\ &= \mathrm{M}\end{aligned}$$

So the perplexity for a uniform multinomial model is always 6906 here (there are 6906 unique vocabulary).

4 Problem d)

The posterior distribution of the mixture proportion $\boldsymbol{\theta}$ is:

$$\begin{aligned}p(\boldsymbol{\theta}|\mathbf{z},\alpha) &\propto p(\boldsymbol{\theta}|\alpha)p(\mathbf{z}|\boldsymbol{\theta})\\ &\propto \mathrm{Dir}(\alpha+\mathbf{c}),\ \text{Where } \mathbf{c}_{\mathrm{k}}=\mathrm{c}_{\mathrm{k}} \text{ is the number of documents assign to mixture (topic) k.}\end{aligned}$$

To visualise the evolution of mixture proportions, we can plot the posterior mean of $p(\boldsymbol{\theta}|\mathbf{z},\alpha)$ (which I consider being a better representation of $p(\boldsymbol{\theta}|\mathbf{z},\alpha)$ than simply use samples drawn from $p(\boldsymbol{\theta}|\mathbf{z},\alpha)$) for all the 20 topic components over the 50 iterations:

$$\mathrm{E}(\theta_{\mathrm{k}}|\mathbf{z},\alpha)=\frac{\alpha+\mathrm{c}_{\mathrm{k}}}{\sum_{j=1}^{\mathrm{K}}\alpha+\mathrm{c}_{\mathrm{j}}},\ \text{Where } \mathbf{z}_{\mathrm{d}}=\mathrm{z}_{\mathrm{d}} \text{ is the topic assiginment of document d.}$$

An additional `mixture_proportions` variable is added to BMM():

[4] As the type of vocabulary and the counts of each vocabulary in different documents are different

Listing 7 Adding `mixture_proportions` to BMM function

```
mixing_proportions = np.zeros((num_iters_gibbs,K))
# The next line is done in each iteration of the Gibbs sampler in BMM and
# mixing_proportions is added as an output of BMM()
mixing_proportions[iter] = ((sk_docs.copy()+alpha)/np.sum(sk_docs.copy()+
                                        alpha)).flatten()
```

With a random seed of 1, $\alpha = 1$, $\gamma = 0.1$ and K $= 20$ (20 topics), the evolution of mixture components is plotted as:

Listing 8 Plot of the evolution of mixture components with random seed 1, $\alpha = 10$ and $\gamma = 0.1$

```
np.random.seed(1)
# load data
data = sio.loadmat('kos_doc_data.mat')
A = np.array(data['A'])
B = data['B']
V = data['V']
K = 20  # number of clusters
alpha = 1   # parameter of the Dirichlet over mixture components
gamma = .1  # parameter of the Dirichlet over words
perplexity, swk, mixing_proportions = BMM(A, B, K, alpha, gamma)
print(perplexity)
label = []
for i in range(1,K+1):
    label.append("Component (Topic) "+str(i))
fig, ax = plt.subplots(figsize=(10,8))
for i in range(20):
    ax.plot(np.linspace(1,50,50),mixing_proportions[:,i],label=label[i])

ax.set_xlabel("number of iteration")
ax.set_ylabel("mixing propotions")
ax.legend(bbox_to_anchor=(1.04, 1), loc="upper left")
```

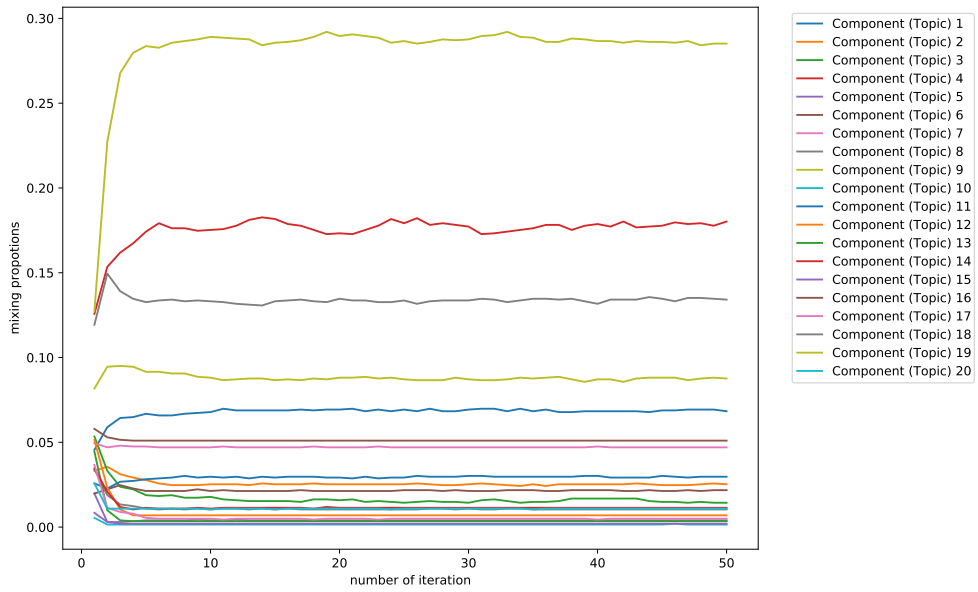

Figure 3: Plot of the evolution of mixture components with random seed 1, $\alpha = 1$, $\gamma = 0.1$ and K=20

Then, I change the random seed to 2 and 5 respectively and then rerun the Gibbs sampler twice:

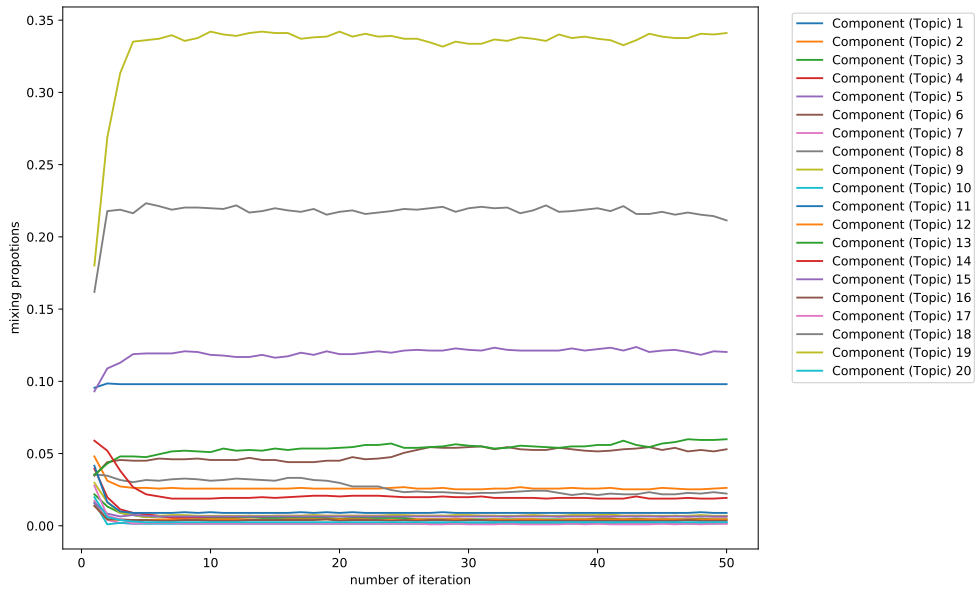

Figure 4: Plot of the evolution of mixture components with random seed 2, $\alpha = 1$, $\gamma = 0.1$ and K=20

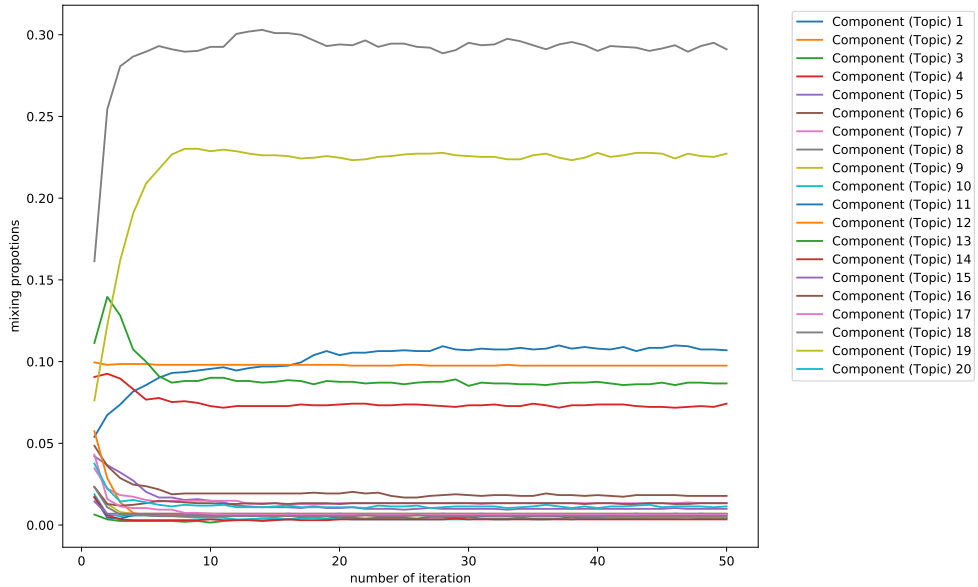

Figure 5: Plot of the evolution of mixture components with random seed 5, $\alpha = 1$, $\gamma = 0.1$ and K=20

Under different random seeds, all the 20 mixture proportions on Figure 3, 4 and 5 converge to fixed values after the 10th iteration, but the same component under different seeds converges to different values. The reason is that the posterior distribution is multimodal with multiple peaks in its density function and thus under different seeds the Gibbs sampler will reach different modes (local optima) of the posterior distribution and get stuck there, which reflects the "rich get richer" property of the Gibbs sampler. Moreover, from Figure 3, 4 and 5, we can also see that under different seeds the modes the Gibbs sampler converge to are non-symmetric (If two modes are symmetric, it means that they are invariant to permutations of the topic assignments. Converging to symmetric modes would not affect the prediction). All of the above indicates that Gibbs sampler does not explore all regions of the posterior and each time

the Gibbs sampler only reaches (and gets stuck in) a local optima of the multimodal posterior distribution.

5 Problem e)

Topic Posteriors and Perplexity

The topic posterior of document d, $\boldsymbol{\theta}_{\mathrm{d}}$, is:

$$p(\boldsymbol{\theta}_{\mathrm{d}}|\{\mathrm{z_{nd}}\},\alpha) \propto p(\{\mathrm{z_{nd}}\}_{\mathrm{n=1}}^{\mathrm{N_d}}|\boldsymbol{\theta}_{\mathrm{d}})p(\boldsymbol{\theta}_{\mathrm{d}}|\alpha)$$
$$\propto \mathrm{Dir}(\alpha+\mathbf{c}^{\mathrm{d}}),\ \ \mathbf{c}_{\mathrm{k}}^{\mathrm{d}}=\mathrm{c}_{\mathrm{k}}^{\mathrm{d}} \text{ is the number of words in document d that is assigned to topic k.}$$

Still, we can plot the posterior mean of the topic posterior $\boldsymbol{\theta}_{\mathrm{d}}$ for the 50 iterations (the same reason as in d)):

$$\mathrm{E}(\theta_{\mathrm{d,k}}|\{\mathrm{z_{nd}}\},\alpha)=\frac{\alpha+\mathrm{c}_{\mathrm{k}}^{\mathrm{d}}}{\sum_{\mathrm{k'=1}}^{\mathrm{K}}\alpha+\mathrm{c}_{\mathrm{k'}}^{\mathrm{d}}}$$

I add an additional `theta_all` variable in LDA():

Listing 9 Modify LDA() by adding `theta_all` to store all the mean of topic posteriors

```
theta_all = np.zeros((D,num_gibbs_iters,K))
# The next line is done for all iterations of Gibbs sampler in LDA and
# theta_all is added as an output of LDA()
theta_all[d,iter,:]=((skd[:,d].copy()+alpha)/np.sum(skd[:,d].copy()+alpha)).
                                                    flatten()
```

I run LDA() with K = 20, $\alpha = 1$ and $\gamma = 0.1$ and plot the topic posteriors for document 1 and 5 together with the average topic posterior for all the 20 documents:

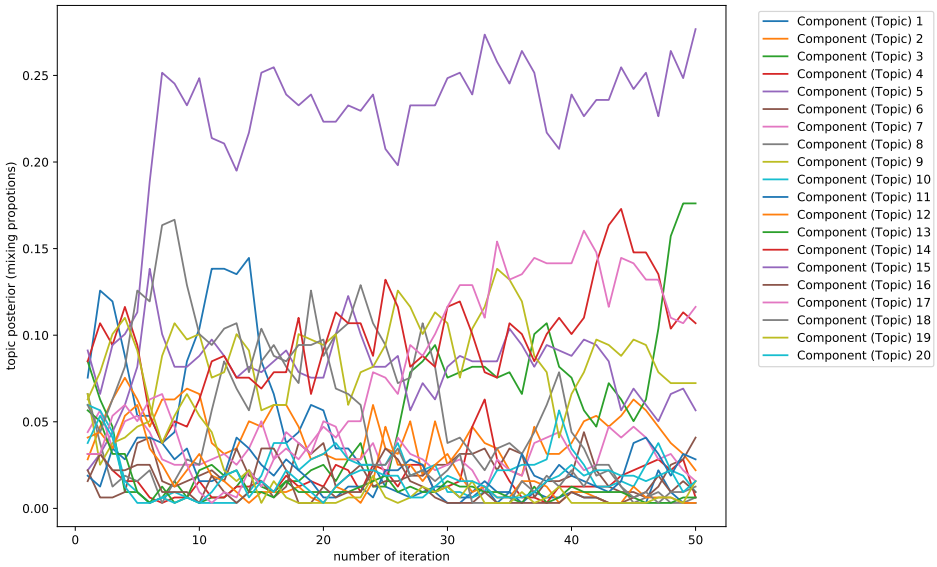

Figure 6: Plot of the evolution of topic posteriors for document 1 with random seed 0, $\alpha = 1$, $\gamma = 0.1$ and K=20

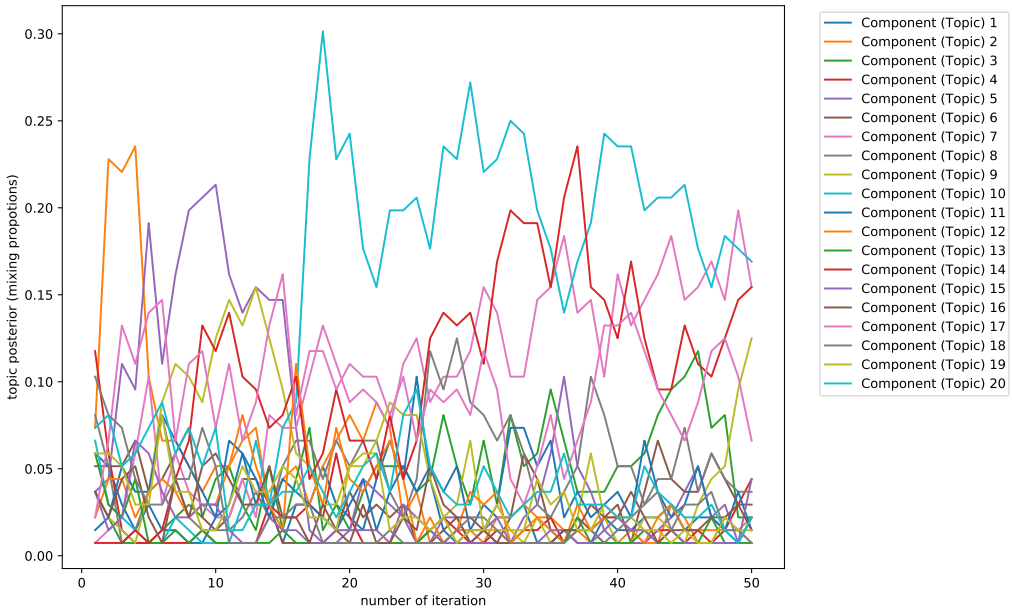

Figure 7: Plot of the evolution of topic posteriors for document 5 with random seed 0, $\alpha = 1$, $\gamma = 0.1$ and K=20

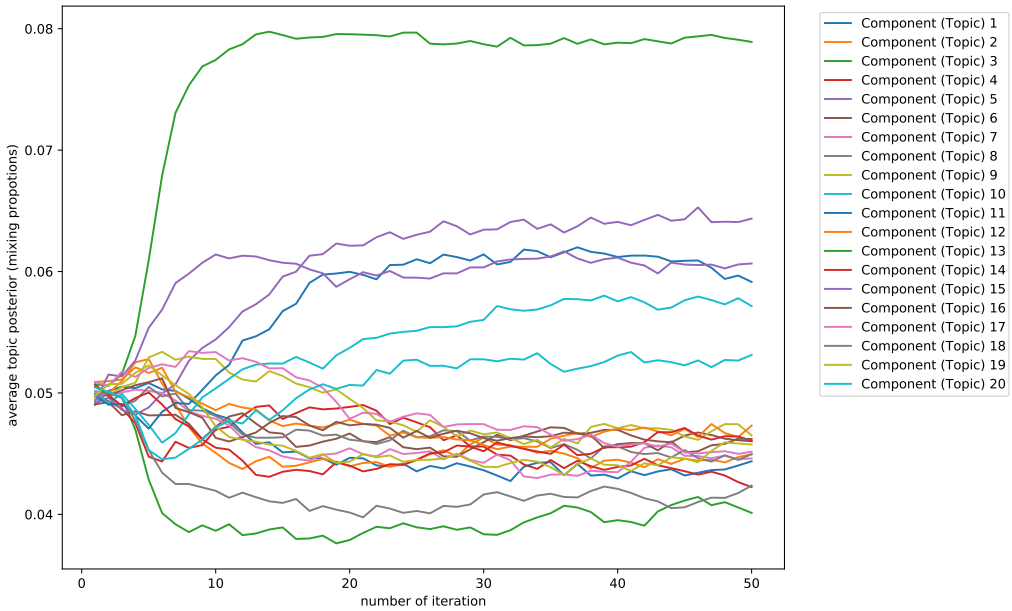

Figure 8: Plot of the evolution of average topic posteriors for all 2000 documents with random seed 0, $\alpha = 1$, $\gamma = 0.1$ and K=20

From Figure 6, 7 and 8, the topic posteriors for document 1 and 5 are fairly unstable and fluctuated over the 50 iterations while the average topic posterior are much more stable. It seems that 50 iterations are inadequate for these topic posterior to converge to fixed values.

Table 1 below summarises the perplexities of Bayesian multinomial model, BMM with seeds=1, 2 and 5, and LDA. Here the perplexity for Bayesian multinomial model is calculated by treating B as a single document and then calculate its per-word perplexity [5]:

[5] i.e. averaging the negative joint log probabilities for all words in B and take the exponential

	Bayesian Multinomial Model	BMM with seed=1	BMM with seed=2	BMM with seed=5	LDA (50 iterations)
perplexity	2683.98407	2091.97682	2147.2610	2107.45510	1680.00019

Table 1: Table for perplexities of different models

From Table 1, The perplexity for LDA after 50 iteration is the smallest among the three models, which means that the performance of LDA is the best among the three (The lower the perplexity is, the more certain the model would be when it predicts a sample. Thus lower perplexity means better generalisation.).

From Figure 6, 7, the topic posteriors for document 1 and 5 do not converge to fixed values in 50 iterations, which means that the model is still not sure what are the underlying topic distributions responsible for document 1 and 5 and this indicates that 50 iterations are inadequate for the convergence of Gibbs sampler.

Word Entropy

To calculate word entropy, we need the posterior distribution of $\boldsymbol{\beta}_{\mathrm{k}}$[6]:

$$
\begin{aligned}
p(\boldsymbol{\beta}_{\mathrm{k}}|\{\mathrm{w_{nd}}\},\gamma) &\propto p(\{\mathrm{w_{nd}}\}|\boldsymbol{\beta}_{\mathrm{k}})p(\boldsymbol{\beta}_{\mathrm{k}}|\gamma) \\
&\propto \mathrm{Dir}(\gamma+\mathbf{c}^{\mathrm{k}}), \quad \mathbf{c}^{\mathrm{k}}_{\mathrm{m}} = \mathrm{c}^{\mathrm{k}}_{\mathrm{m}} \text{ is the number of times that the unique vocabulary m is assigned} \\
&\qquad\qquad \text{to topic k in all documents.}
\end{aligned}
$$

Then, the word entropy for each topic is:

$$
\begin{aligned}
\text{Entropy} &= -\sum_{\mathrm{m}=1}^{\mathrm{M}} \frac{\gamma+\mathrm{c}^{\mathrm{k}}_{\mathrm{m}}}{\sum_{\mathrm{m}'=1}^{\mathrm{M}}\gamma+\mathrm{c}^{\mathrm{k}}_{\mathrm{m}'}} \log_{\mathrm{e}}\left(\frac{\gamma+\mathrm{c}^{\mathrm{k}}_{\mathrm{m}}}{\sum_{\mathrm{m}'=1}^{\mathrm{M}}\gamma+\mathrm{c}^{\mathrm{k}}_{\mathrm{m}'}}\right) \\
\text{Where } &\frac{\gamma+\mathrm{c}^{\mathrm{k}}_{\mathrm{m}}}{\sum_{\mathrm{m}'=1}^{\mathrm{M}}\gamma+\mathrm{c}^{\mathrm{k}}_{\mathrm{m}'}} = \mathrm{E}(\beta_{\mathrm{k,m}}|\{\mathrm{w_{nd}}\},\gamma)
\end{aligned}
$$

As I use the natural logarithm for calculating entropy, the units of the entropy is natural units (or nat). Here is code I add to LDA():

Listing 10 Modify LDA() by adding `entropy` to store all calculated word entropy

```
entropy = []
# The next line is done for each iteration of the Gibbs sampling in LDA
beta_all=(swk.copy()+gamma)/np.sum(swk.copy()+gamma,axis=0)
entropy.append(-np.sum(beta_all.copy()* np.log(beta_all.copy()),axis=0))
```

Here is the plotted word entropy:

[6] $\boldsymbol{\beta}_{\mathrm{k}}$ is the parameters of a categorical distribution1 over the M vocabulary words for topic k

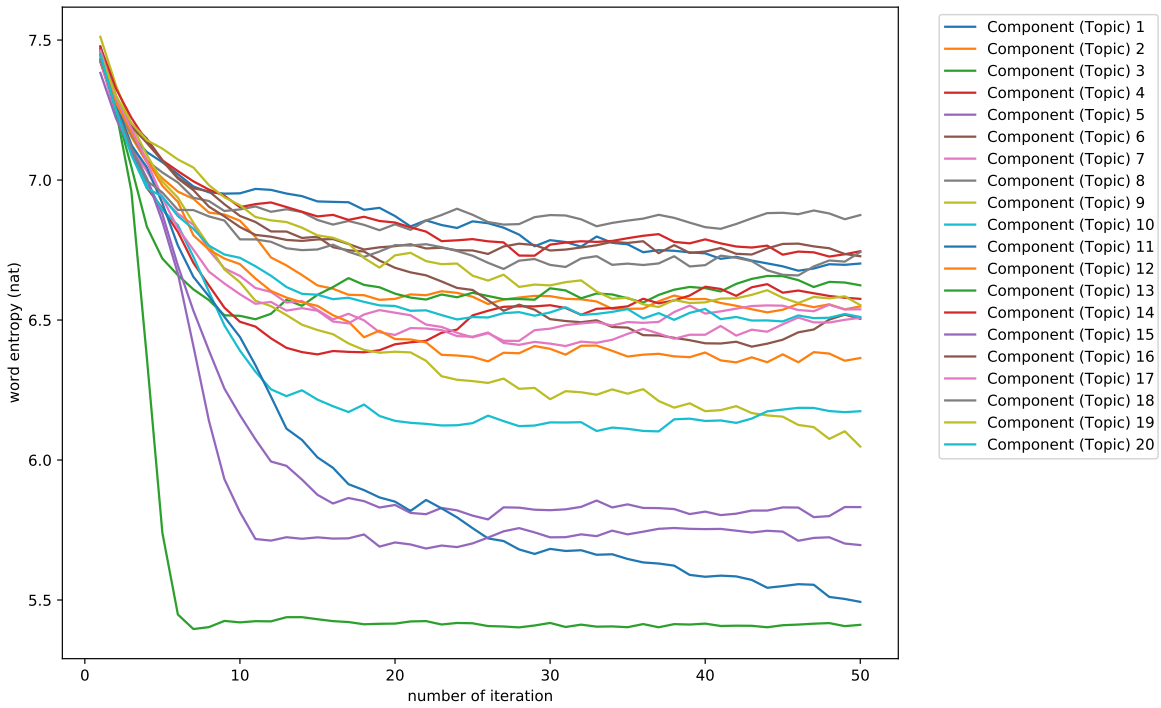

Figure 9: Plot of the evolution of word entropy for 20 topics with random seed 0, $\alpha = 1$, $\gamma = 0.1$ and K=20

The word entropy has overall decreased with more and more iterations, which means that over the 50 iterations each topic are becoming more and more certain about which words are more representative of that topic and that for each topic larger probabilities are assigned to those words that are are more representative of that topic (i.e. these topic posterior distributions are getting more focused or concentrated). From Figure 9, the word entropy of topic 13, 11, 15 and 5 drops faster than others. Specifically, the word entropy for topic 13 dropped the fastest and has the lowest word entropy over the 50 iterations, which means that the words representing topic 13 are the most informative ones.