CUED - Engineering Tripos Part IIB 2023-2024 **Module Coursework**

Module		Title of report	

Date submitted:	Assessment for this module is ☐ 100% / ☐ 25% coursework of which this assignment forms _____ %

UNDERGRADUATE and POST GRADUATE STUDENTS		
Candidate number:		☐ Undergraduate ☐ Post graduate

Feedback to the student

☐ **See also comments in the text**

		Very good	**Good**	Needs improvmt
CONTENT	**Completeness, quantity of content:** Has the report covered all aspects of the lab? Has the analysis been carried out thoroughly?			
	Correctness, quality of content Is the data correct? Is the analysis of the data correct? Are the conclusions correct?			
	Depth of understanding, quality of discussion Does the report show a good technical understanding? Have all the relevant conclusions been drawn?			
	Comments:			
PRESENTATION	**Attention to detail, typesetting and typographical errors** Is the report free of typographical errors? Are the figures/tables/references presented professionally?			
	Comments:			

Marker: Date:

4F13 Probabilistic Machine Learning
Coursework 3 - Latent Dirichlet Allocation

Candaidate Number: 5663A

1 Exercise A

For a total number of M unique words in D documents, a discrete categorical distribution where each word is drawn from is $\boldsymbol{\beta} \in \mathbb{R}^M$. The maximum-likelihood solution for $\boldsymbol{\beta}$, as derived during the lecture, is given by

$$\beta_i^{ml} = \frac{c_i}{\lambda} = \frac{c_i}{\sum_{n=1}^{M} c_n} = \frac{c_i}{C} \tag{1}$$

where C is the total number of words and c_i is the count of word i. The probability of observing a new vector $\boldsymbol{k}$ with N unique words and J total words is:

$$p(\boldsymbol{k}|\boldsymbol{\beta}, J) = \frac{J!}{\prod_{i=1}^{N} k_i!} \prod_{i=1}^{N} \beta_n^{k_i} \tag{2}$$

In equation 2, β_n takes the value from $\boldsymbol{\beta}$ if the word that k_i represents is in the training set. Otherwise, $\beta_n = 0$. Figure 1 shows the top 20 words from set A by empirical frequencies. FThe word "bush" occurs 3833 times and is most likely. Therefore, the highest log-probability for a test set is $log(3833/271898) = -4.262$ when there is only a single word "bush". The lowest probability occurs when there is at least a word not in the training set. The test set will have zero probability ($-\infty$ log-probability). New words that the model has not seen will break down the testing due to a single zero β_n from equation 2, and the model does not specialise, which is impractical.

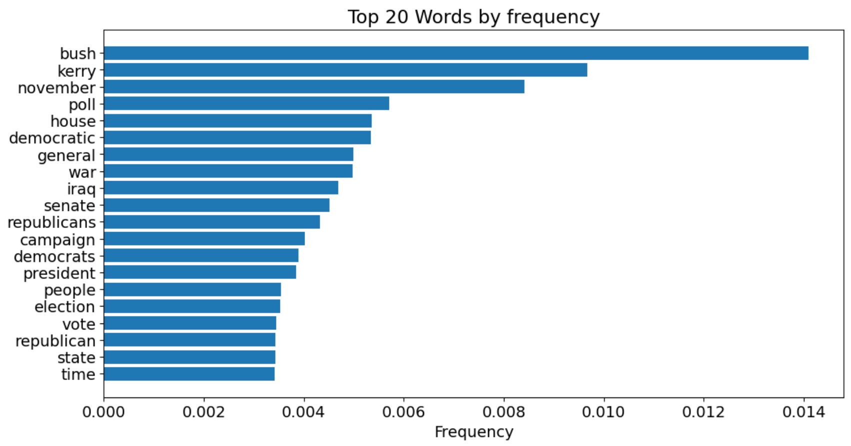


Figure 1: Top 20 words by frequency from training set A

2 Exercise B

We incorporate a symmetric Dirichlet prior with a concentration parameter α for words. The posterior over $\boldsymbol{\beta}$ becomes:

$$p(\boldsymbol{\beta}|\boldsymbol{\alpha}) \propto \frac{\Gamma(\sum_{i=1}^{M} \alpha_i)}{\prod_{i=1}^{M} \Gamma(\alpha_i)} \prod_{i=1}^{M} \beta_i^{\alpha_i - 1} \times \prod_{i=1}^{M} \beta_n^{c_i} \propto \prod_{i=1}^{M} \beta_i^{\alpha + c_i - 1} \quad \text{,where all } \alpha_i\text{s are equal} \tag{3}$$

This is also a Dirichlet distribution whose parameters depend on word counts. The predictive probability of a word $w_i = i$ is its expected value from the Dirichlet distribution:

$$p(w_i = i|\boldsymbol{\beta}, \boldsymbol{\alpha}) = \frac{\alpha_i + c_i}{C + \sum_{j=1}^{M} \alpha_j} = \frac{\alpha + c_i}{C + \alpha M} \tag{4}$$

Setting equation 1 $\leq$ equation 4, we have:

$$\frac{\alpha + c_i}{C + \alpha M} \leq \frac{c_i}{C} \implies c_i \geq \frac{C}{M} \tag{5}$$

Equation 5 implies that in the test set, words with counts larger than C/M will experience a decrease in predictive probability from the prior Dirichlet distribution compared to the ML solution, and words with counts smaller than C/M will have a higher probability.

This intuitively makes sense. The introduction of the prior reduces overfitting. By adding pseudo-counts α to words, new words would still have some non-zero probability of $\frac{\alpha}{C+\alpha M}$. For words from the training set, the effect of large c_i for common words is reduced by α, while the effect of smaller c_i for rare words is strengthened. The effect is less obvious when α is small (i.e. a weak prior). Conversely, consider when $\alpha = \infty$, all words would share the same probability regardless of their actual counts c_i.

3 Exercise C

Using the Bayesian model, the categorical and multinomial distribution functions for a sequence of words $\boldsymbol{w} \in \mathbb{R}^J$ with N unique words are:

$$\text{Categorical: } p(\boldsymbol{w}|\boldsymbol{\beta}, \boldsymbol{\alpha}) = \prod_{i=1}^{J} p(w_i = i|\boldsymbol{\beta}, \boldsymbol{\alpha})$$

$$\text{Multinomial: } p(\boldsymbol{w}|\boldsymbol{\beta}, \boldsymbol{\alpha}) = \frac{J!}{\prod_{i=1}^{N} c_i!} \prod_{i=1}^{J} p(w_i = i|\boldsymbol{\beta}, \boldsymbol{\alpha})$$

Obviously, the order of words in a document matters. Ssimply doing word counts using multinomial distribution can distort the original meaning of the texts. Therefore, the categorical distribution function is used to compute the log-likelihhod of a test document, whose expression is:

$$\begin{aligned} log[p(\boldsymbol{w}|\boldsymbol{\beta}, \boldsymbol{\alpha})] &= log[\prod_{i=1}^{J} p(w_i = i|\boldsymbol{\beta}, \boldsymbol{\alpha})] \\ &= log(\prod_{i=1}^{N} \beta_i^{c_i}) \text{ ,where } \beta_i \text{ is the probability of word i} \\ &= \sum_{i=1}^{N} c_i log \beta_i \end{aligned}$$

Using the above expression, the log-probability of Document 2001 is -3688.6 for $\alpha = 1$ (also used for thereafter calculations in this section).

The per-word perplexity measures how well a language model predicts each individual word in a sequence:

$$perplexity = exp(\frac{-log[p(\boldsymbol{w}|\boldsymbol{\beta}, \boldsymbol{\alpha})]}{J}) \tag{6}$$

From equation 6, the perplexity for a uniform multinomial of size N is $exp(-[log(1/N)]^J/J)$ (i.e. the size of unique words N). This is also the largest possible perplexity when the model is the most uncertain about its prediction.

Document 2001 has 440 words and its perplexity is 4373.1. For the entire test set B, the perplexity is 2684.0. In general, for each individual document, if there are more common words from the training set, the perplexity will be lower because of its higher log-probability. The perplexity also depends on the value of α. Consider for $\alpha = \infty$, the posterior becomes a uniform multinomial distribution, and the perplexity just becomes the size of the vocabularies.

Listing 1: exercise C: Calculating the posterior

```
prior_counts, word_counts = alpha*np.ones(num_words+1), np.zeros(num_words+1)
prior_counts[0] = 0
for row in A:
    word_counts[row[1]] += row[2]
posterior_counts = word_counts + prior_counts
posterior_prob = posterior_counts/np.sum(posterior_counts)
```

4 Exercise D

K distinct categories are introduced to the documents. The new discrete latent variable $z_d \in 1, ...K$ is defined as the assignment of category of document d, and $\boldsymbol{\theta}$ is a categorical distribution over K categories. In the remaining of the report, we assign a symmetric Dirichlet prior to $\boldsymbol{\theta}$ with concentration $\alpha = 10$ and to each $\boldsymbol{\beta_k}$ with concentration $\gamma = 0.1$.The posterior of the mixing proportion is:

$$p(\theta_k|z, \alpha) \propto Dir(c_k + \alpha) = \frac{c_k + \alpha}{\alpha K + N} \quad \text{, where N is the total number of documents} \tag{7}$$

In **bmm.py**, we introduced matrices *mixing_proportions* and *mixing_proportions_diff* that keep track of the normalised *sk_docs* and the differences (for checking convergence) at each Gibbs iteration. The results are plotted in figure 2.

The parameters were randomly initialised several times. Interestingly, the posterior always converges in a way that a few categories dominate. However, the dominant categories and their proportions can change across initialisations, suggesting multiple local optimal points. It is hard to decide the exact answer due to the high dimensionality. However, in all cases, the Gibbs sampler would converge to stationary distributions within 10 iterations as shown in figures 2b and 2d.

Listing 2: exercise D: Getting mixing proportions

```
    mixing_proportions = np.zeros((K, num_iters_gibbs))
    # Following codes are within each Gibbs iteration
    proportion = sk_docs / np.sum(sk_docs)
    if iter != 0:
        diff = proportion.flatten() - mixing_proportions[:, iter-1]
        mixing_proportions_diff[:, iter-1] = diff
    mixing_proportions[:, iter] = proportion.flatten()
```

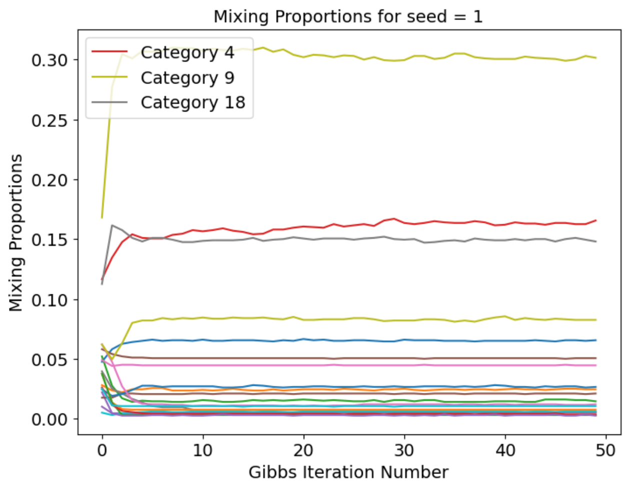
5 Exercise E

We now use LDA such that every document has its own category distribution $\boldsymbol{\theta_d}$ and each word from document d can be drawn from different categories. This model intuitively makes more sense because each document should not be limited to a single category. The posterior that document d belongs to topic k is:

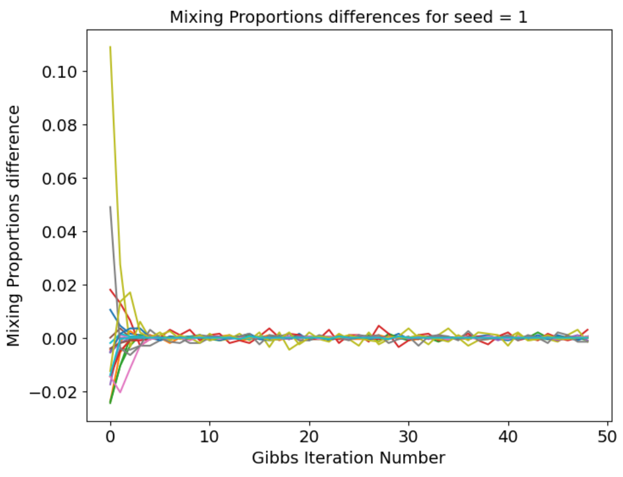
$$p(\boldsymbol{\theta_{d,k}}|\alpha, \boldsymbol{z}) = \frac{\alpha + c_{k,d}}{K\alpha + N_d} \tag{8}$$

Where $c_{k,d}$ is the number of words in document d belonging to topic k, and N_d is the total number of words in d. We could also calculate the posterior of all the documents as a whole, by replacing $c_{k,d}$ with c_k (i.e. total number of words from topic k in all documents) and N_d with N (i.e. total word counts from all documents).

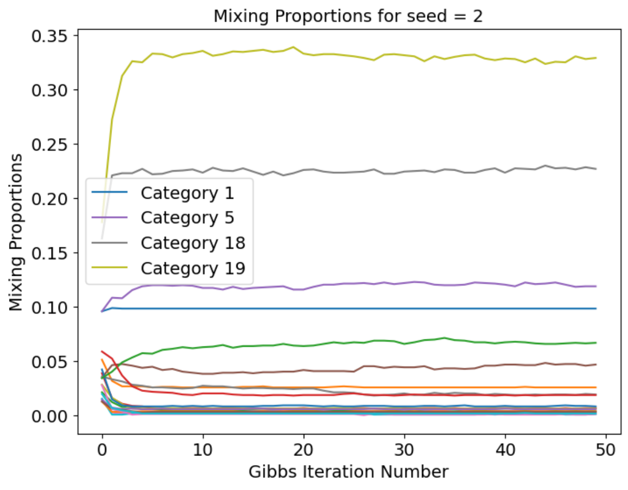
The model was run for 50 iterations under different initialisations. It takes longer to converge to a stationary distribution than the BMM as expected because we introduced new latent variables for topics over words. However, it appears that the distribution does not settle down after 50 iterations (figure 3), and more Gibbs sweeps are necessary for a stationary posterior over all documents. Some topics keep having their proportions growing (which also happened for the BMM) because of the "rich getting richer" phenomenon in Gibbs sampling. The more likely topic is also more likely drawn for future samples, making it harder for the distribution to stablise.



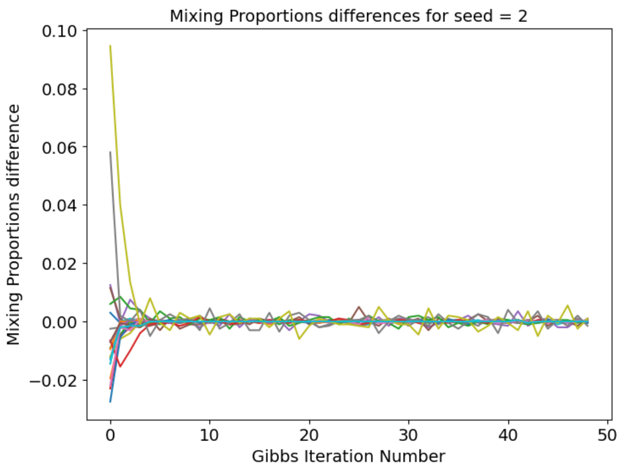
(a) posterior for seed = 1, perplexity = 2092



(b) posterior difference for seed = 1



(c) posterior for seed = 2, , perplexity = 2151



(d) posterior difference for seed = 2

Figure 2: Mixing proportions and differences between each Gibbs iteration for BMM (different initialisations)

The testing perplexity for the LDA model is 2071, which is lower than 2092 for BMM and 4373 for the Bayesian model, suggesting that the LDA is more superior. Ideally, the result will be even better after convergence.

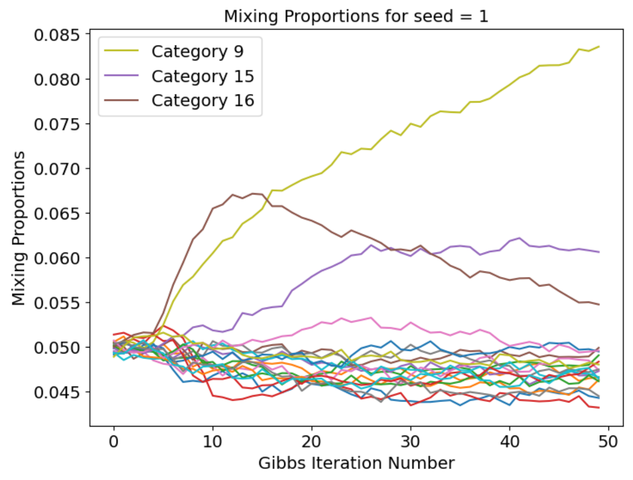
The word entropy(unit: bits/word) to each topic measures the uncertainty related to the prediction of each word under a specific topic. It is easily implemented by using the **entropy** method from the *scipy.stats* library:

Listing 3: exercise E: Calculating Entropies

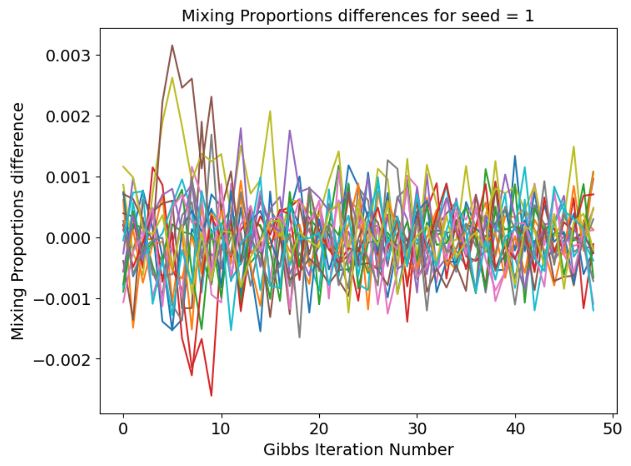
```
word_entropies = np.zeros((K, num_gibbs_iters))
for k in range(K):
    word_entropies[k, iter] = entropy(swk[:, k])
```

The entropies over iterations are shown in figure 4. The entropy decreases over iterations because the model becomes less uncertain in its predictions with more samplings. However, this effect decays and saturation will eventually be reached once the model sees its limitations after about 40 iterations. Meanwhile, the entropy also varies across different topics, meaning that some topics will have certain words more likely to occur, and the distribution is more concentrated.

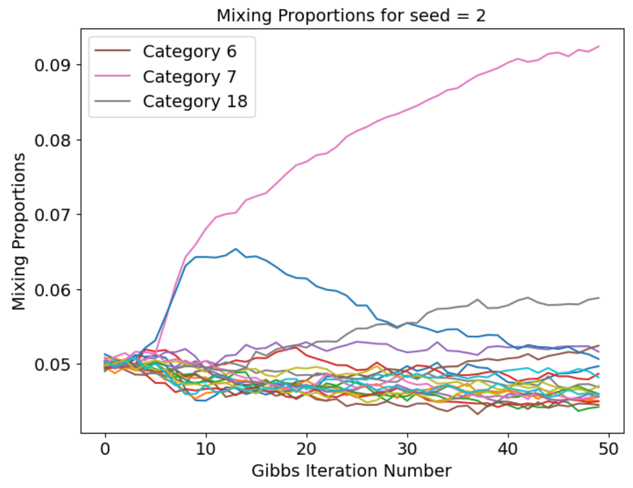
WORD COUNT: 998



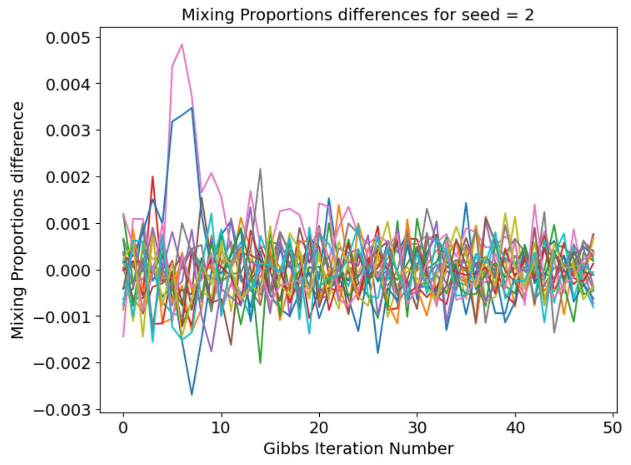
(a) posterior for seed = 1, test perplexity = 2071



(b) posterior difference for seed = 1

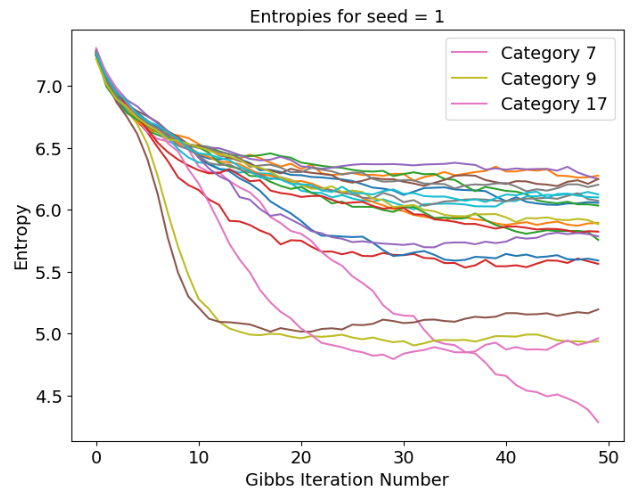


(c) posterior for seed = 2, , perplexity = 2072

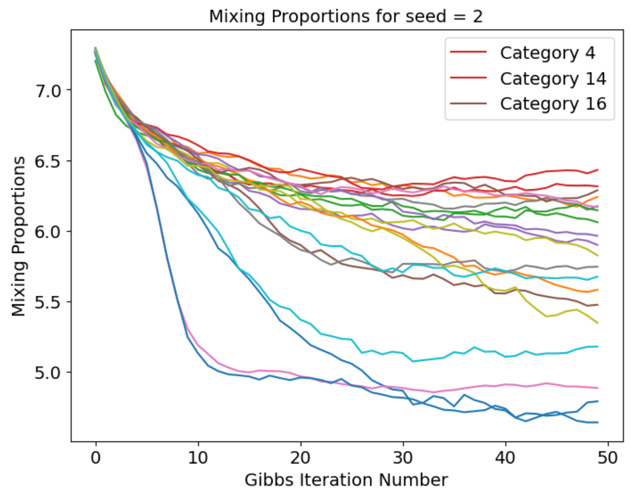


(d) posterior difference for seed = 2

Figure 3: Mixing proportions and differences between each Gibbs iteration for LDA (different initialisations)



(a) Seed = 1



(b) Seed = 2

Figure 4: Word entropies of each topic for LDA under different seeds)