UNIVERSITY OF CAMBRIDGE

MPhil in Machine Learning and, Machine Intelligence 2018-2019

Module Coursework Feedback

Module Title: Probabilistic Machine Learning

Module Code: 4F13

Candidate Number: F606F

Coursework Number: 3

I confirm that this piece of work is my own unaided effort and adheres to the Department of Engineering's guidelines on plagiarism. √

Date Marked: Click here to enter a date. Marker's Name(s): Click here to enter text.

Marker's Comments:

This piece of work has been completed to the following standard *(Please circle as appropriate)*:

	Distinction			**Pass**			**Fail (C+ - marginal fail)**		
Overall assessment (circle grade)	**Outstanding**	**A+**	**A**	**A-**	**B+**	**B**	**C+**	**C**	**Unsatisfactory**
Guideline mark (%)	90-100	80-89	75-79	70-74	65-69	60-64	**55-59**	**50-54**	**0-49**
Penalties	**10% of mark for each day, or part day, late (Sunday excluded).**								

The assignment grades are given **for information only**; results are provisional and are subject to confirmation at the Final Examiners Meeting and by the Department of Engineering Degree Committee.

4F13 Probabilistic Machine Learning: Coursework 3: Latent Dirichlet Allocation (979 words)

a.

The maximum likelihood multinomial of a word i is given by π_i,

$$\pi_i = \frac{k_i}{\sum_j k_j}$$

, where k_i the count of the word i within training data A.

Regardless of the document ID, the words with the same index from different documents are grouped together and their numbers of occurrence are added up to give the total appearance of that distinct word in dataset A. Adding up all word counts in the 3rd column of dataset A give the total number of words.

The maximum likelihood multinomial for each distinct word is then obtained.

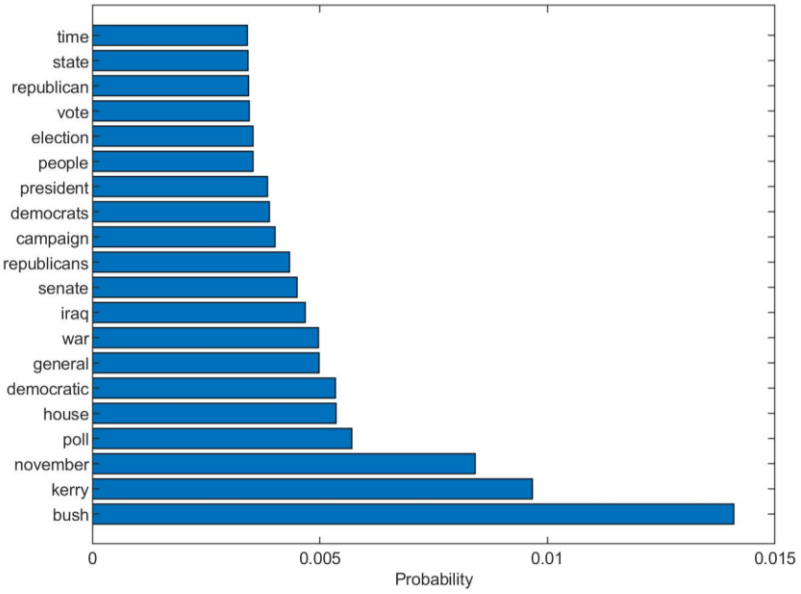

Figure 1: A histogram of 20 largest probability items.

The test set probability is given by the following expression,

$$p(\underline{k}|\underline{\pi}, n) = \frac{n!}{\prod_j k_j\,!}\prod_{i=1}\pi_i^{k_i}$$

, where n the total number of words in dataset B, k_i the number of occurrences of a distinct word i in dataset B and π_i the maximum likelihood multinomial obtained by training set A for each distinct word i.

Based on this model, the training set A which does not contain the word has no count on that word. The probability of that word occurring is zero. Therefore, the test set probability is zero.

It implies if some words do not occur in the training set, the maximum likelihood multinomial model fails to estimate their probabilities of occurrences. It cannot properly estimate the probability of a test set which contains these missing words.

b.

By the maximum likelihood multinomial model, the predictive word probability is

$$p(x = i|D_A) = \frac{c_i}{\sum_j c_j}$$

, where x the word, i the distinct word index, D_A the training set A and c_i the count of the distinct word i.

By the symmetric Dirichlet prior model with a concentration parameter $\alpha = 0.1$, the predictive probability corresponds to the mean of the posterior Dirichlet distribution which is given by,

$$p(x = i|D_A) = \frac{\alpha + c_i}{\sum_j (\alpha + c_j)}$$

Table 1: Predictive probabilities for some words based on maximum likelihood multinomial and dirichlet prior model.

Category	Word	Maximum likelihood multinomial (MLM)	Dirichlet prior model
		Predictive Probability	
Common	Bush	0.0141	0.0141
	Kerry	0.0097	0.0097
	November	0.0084	0.0084
Rare	Bennett	3.6778×10^{-6}	4.0350×10^{-6}
	Argument		
	Accent		
Does not exist in Dataset A	Amphibians	0	0.3669×10^{-6}
	Amdt		
	Alhusainy		

In Table 1, the predictive word probabilities for these two types of inference for the common words look similar. However, the probabilities for rare words (including those non-existing words in training set A) are different, since the MLM model only counts words that exist in the dataset, but the Dirichlet model predicts by the mean of the posterior distribution. So, those words that never exist in the data A have zero probabilities in the MLM model but non-zero probabilities in the Dirichlet model. The probabilities for all words are increased in the Dirichlet prior model.

c.

Using the Bayesian model with the symmetric Dirichlet prior, the log probability without and with combinatorial factor for the test document with ID 2001 are -3691 and -1691 respectively.

The multinomial without combinatorial factor is used. It is because the document is an article but not a vocabulary list that can draw any word from data and put them together without concerning grammar. One sequence of words is concerned. Then the model without the factor is a better estimate in predicting words from article than the one with the factor which ignores the word sequencing completely.

Per-word perplexity is given by,

$$exp\left(-\frac{l}{n}\right) = exp\left(-\frac{\sum_i log\left(\pi_i^{k_i}\right)}{n}\right) = exp\left(-\frac{\sum_i k_i \, log(\pi_i)}{n}\right)$$

For the log multinomial probability without combinatorial factor, the per-word perplexity for the document ID 2001 is 4399. The per-word perplexity over all documents in B is 2697.

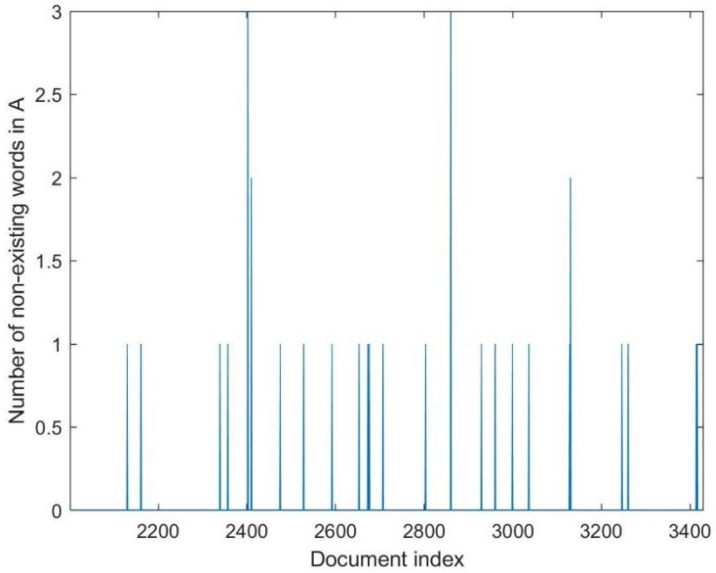

Figure 2: The number of words which each document in B contains but A does not contain.

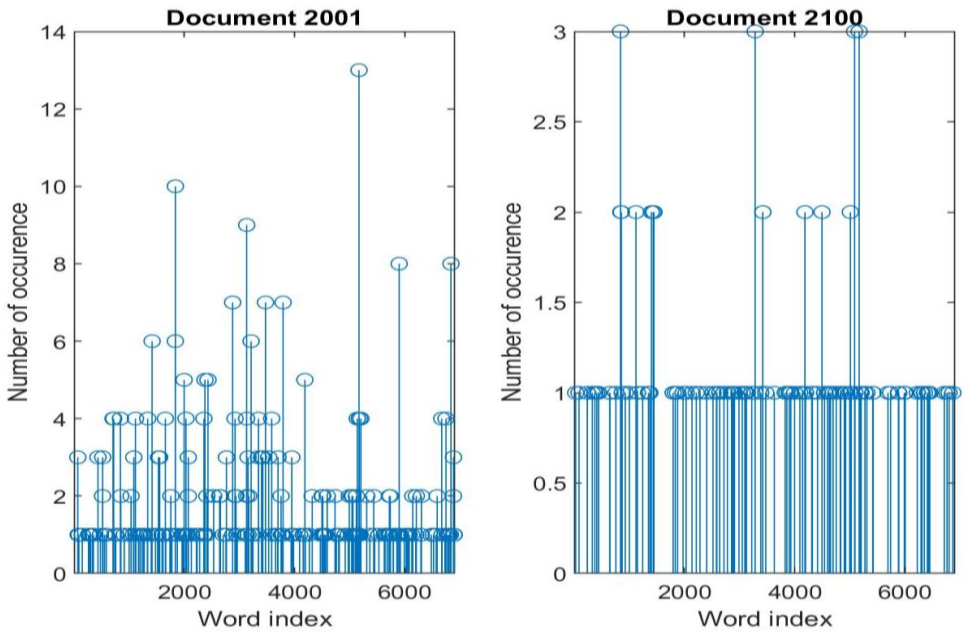

Figure 3: The word occurrence distributions of document 2001 (left) and 2100 (right).

In Figure 2, it shows different documents in B have different number of words which do not exist in training set A. This is important since those words which have small probabilities close to zero lead to higher contribution to the perplexity. In Figure 3, it shows different documents have different occurrences for each word.

So, the perplexities are different for different documents since they have different word distributions.

For a uniform multinomial, those 6906 distinct words have the equal probability of $\frac{1}{6906} \approx 1.448 \text{ x } 10^{-4}$. The per-word perplexity becomes $exp\left(-\frac{nlog\left(\frac{1}{6906}\right)}{n}\right) = 6906$ which is greater than that computed by the symmetric Dirichlet prior model. It indicates that, compared with the Dirichlet prior model, the uniform model performs badly in predicting samples.

d.

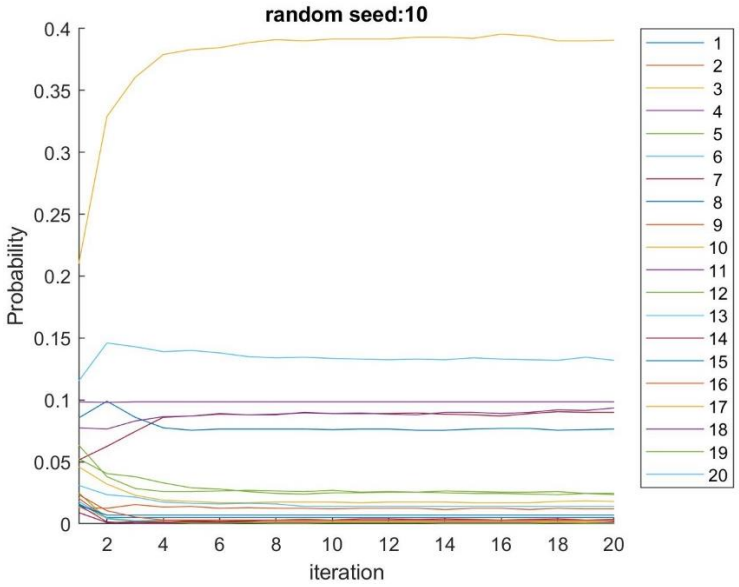

Figure 4: The evolution of the mixing proportions as a function of iteration with random seed 10.

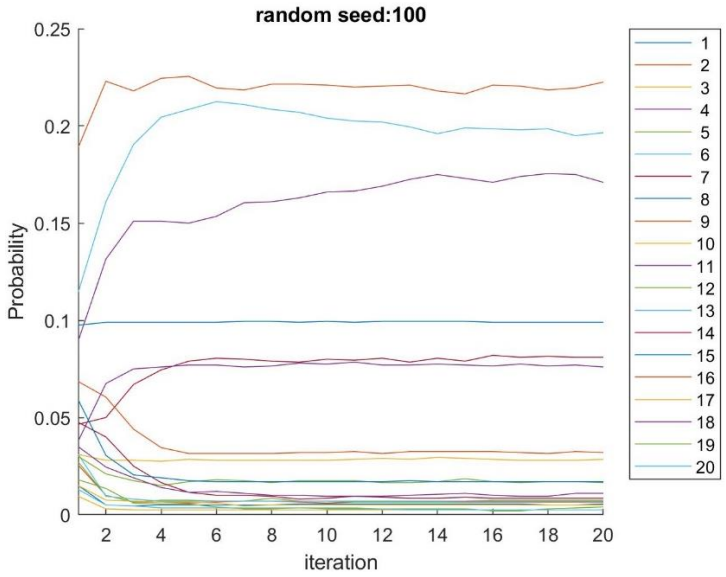

Figure 5: The evolution of the mixing proportions as a function of iteration with random seed 100.

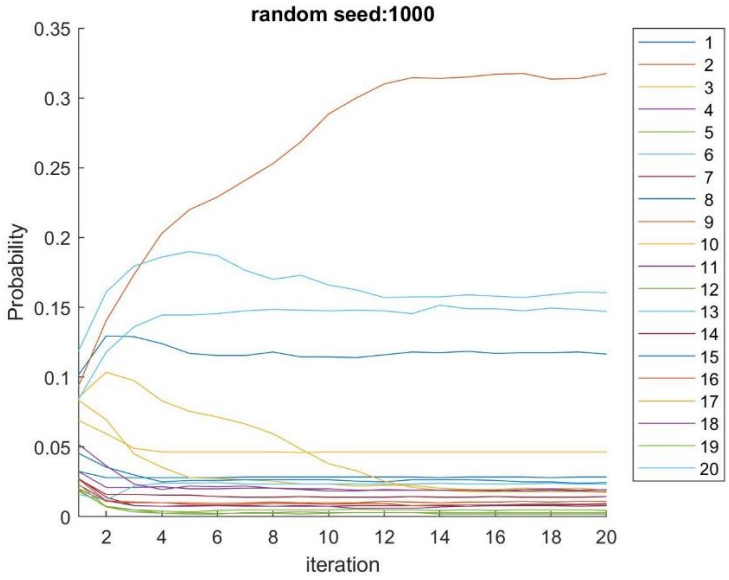

Figure 6: The evolution of the mixing proportions as a function of iteration with random seed 1000.

The following command is substituted into the ‘iter’ for-loop,

```
docs_mixpro(:,iter) = sk_docs/2000;
```

It extracts the mixing proportions of all mixing components as a function of iteration.

The Gibbs sampler cannot converge since slopes are found to be greater or less than 0 for the mixing proportions.

With different random seeds, the documents are distributed to different mixture components randomly. From Figures 4, 5 and 6, different topic indices are arranged to take the top rank according to their mixing proportions. It indicates the algorithm has no idea about what topic each index represents.

The mixing proportion separations between topics are not identical for different seeds, for example, the samplings with random seed 10 (Figure 4) have the greatest separation between the 1^{st} and 2^{nd} rank mixture components in terms of mixing proportion, meanwhile, the ones with random seed 100 (Figure 5) have the smaller separation. It shows the sampler does not explore the stationary distribution for the mixture components.

e.

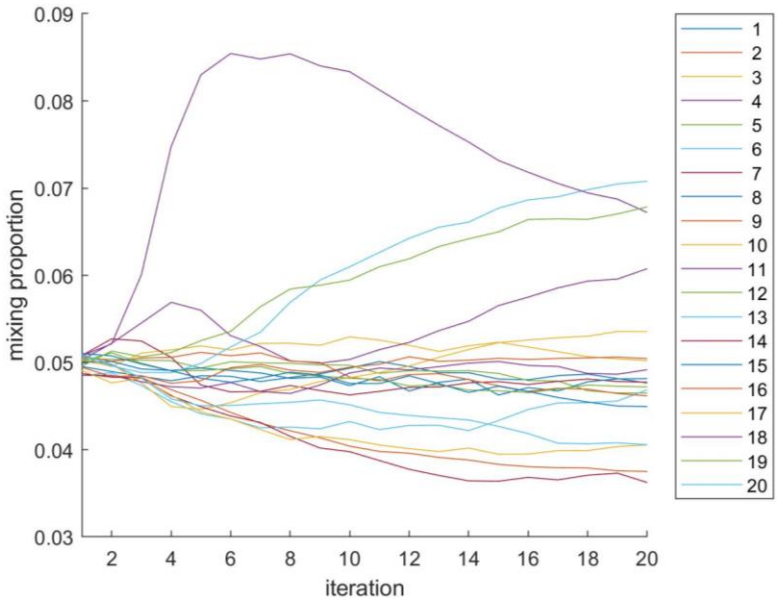

Figure 7: The evolution of topic posteriors as a function of Gibbs sweeps, up to 20 iterations.

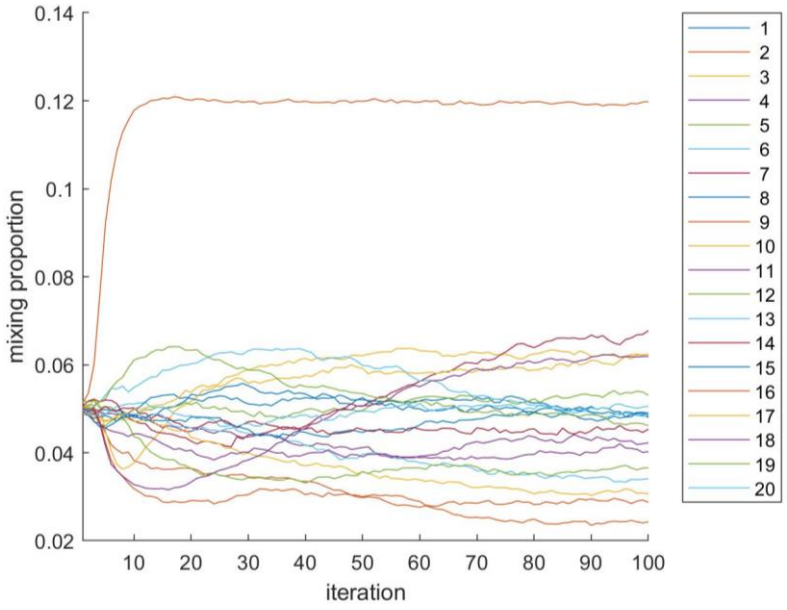

Figure 8: The evolution of topic posteriors as a function of Gibbs sweeps, up to 100 iterations.

The following command is substituted into the first 'iter' for-loop,

```
topic_mixpro(:,iter) = sk/sum(sk);
```

It extracts the topic posteriors as a function of iteration.

The Gibbs sampler is not able to converge within 20 sweeps, since it is observed that the posterior distributions of all topics are still not stable. Especially for the topic which was the top posterior at 6th iteration in Figure 7, its slope is significantly lower than 0.

After 20 Gibbs sweeps, the per-word perplexity calculated is 1716. It is lower than 2697, the one calculated in part c. It is possible that 'lda.m' successfully groups words together according to topics and limits the ways that the words can be used. Therefore, it gives a lower perplexity, which indicates it is a better model at predicting the sample.

From Figure 8, the perplexity is 1617 at 100th sweep, and the posteriors do not stabilize after 100 iterations. Especially for the topic which just reaches the 2nd top posterior at 100th iteration, its posterior is still growing upwards.

It may indicate that 20 sweeps are not adequate for the algorithm to converge since the perplexity is still decreasing, which means there are still some improvements allowed.

The word entropy for each of the topics is given by S,

$$S = -\sum_i p_i \, log(p_i)$$

, where p_i the probability of a distinct word i.

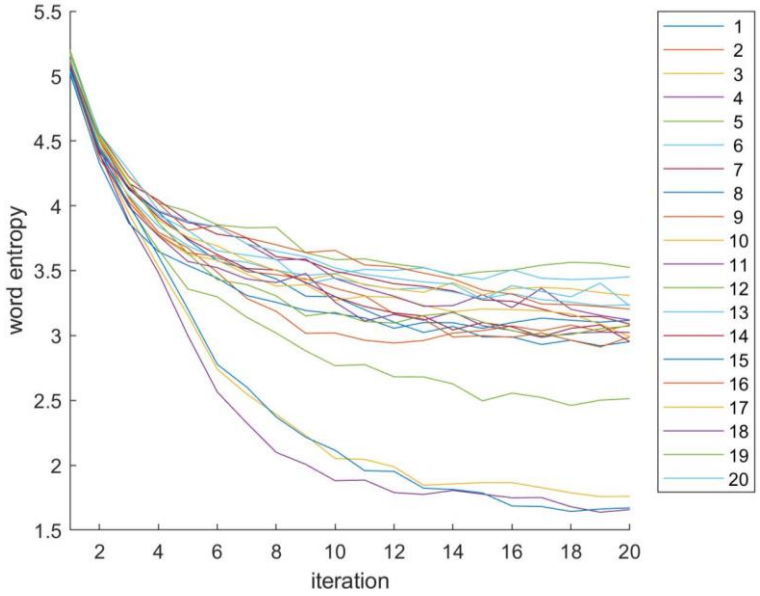

Figure 9: Word entropy for each of the topics as a function of Gibbs sweeps.

The 'lda.m' is run once, and the variable 'b' is stored as 'store_b' since it contains the probabilities of all distinct words. The script is then modified by putting the following commands into the first 'iter' for-loop, and re-run again.

```
for j = 1:20 %topic
    topic_s = swk(:,j);
    en = 0;
    for wo = 1:6906 %word
        if topic_s(wo) == 0
          en = en;
        else
          en = en - store_b(wo)*log(store_b(wo));
        end
    end
    entropy(j,iter) = en;
end
```

The 'swk' variable stores the record of which distinct word i is assigned to the distinct topic j. Based on this record, for each topic, a new entropy variable 'en' = 0 is initialized, and for each distinct word which is assigned to the topic, the entropy value of the corresponding distinct word is subtracted from 'en'. The final value of 'en' is assigned back as an element (representing the overall entropy of topic j in a given iteration 'iter') to the 'entropy' matrix which stores all topics' entropies for every iteration.

In Figure 9, the entropies for all topics decrease and begin to converge as the slopes are less steep. This implies that the general probabilities of the distinct words rise, the distinct words carry less information (less possibilities to occur in any document and topic) because of the correct topic grouping. The word distribution becomes less chaotic.