CS6730 : Assignment 3

Instructor and TAs

Release: 11th April 2019; **Due: 18th April 2019, 11.59pm**

- Submit to **GradeScope a single LaTeX-generated pdf file** containing your solutions. Please type your answers in the solutions blocks in the source LaTeX file of this assignment.
- All necessary files discussed here can be found in moodle in a .zip file.

---

1. (10 points) [DIRICHLET TO YOUR RESCUE] Because of the upcoming end-semester exams, you are not able to follow up long, detailed articles from T5E (The Fifth Estate), the newsletter that provides you with regular updates of news from both IITM campus and around Chennai. You only want to have a glance at important topics and words in an article to save time. You decided to develop a bot called Dirichlet() that can summarize you the topics present in a T5E article, along with the important words in varying proportions, describing that topic. Note that Dirichlet() uses Latent Dirichlet Allocation (LDA) for topic modelling, and to do that, we need to do collapsed Gibbs sampling equations for LDA with conditional probabilities:

$$\phi_k \sim \mathrm{Dirichlet}(\beta) \tag{1}$$

$$\theta_i \sim \mathrm{Dirichlet}(\alpha) \tag{2}$$

$$z_{ij}|\theta_i \sim \mathrm{Discrete}(\theta_i) \tag{3}$$

$$d_{ji}|z_{ji}, \phi_{z_{ji}} \sim \mathrm{Discrete}(\phi_{z_{ji}}) \tag{4}$$

Here j is the index for words ($d_i = \{d_{1i}, ..., d_{Ni}\}$), i is the index for documents and k is the index for topics. Also, we use the following notation: $N_{wki} = \sum_w N_{wki}$ and $N_{wk} = \sum_i N_{wki}$. We use superscript (-ji) (e.g. N_{wki}^{-ji}) to indicate the corresponding word d_{ji} in document i is not counted in N_{wki}.
.5.5.5

   (a) (1.5 points) Write down P(d|z,β) and P(z|α) using their conditional probabilities. (Hint: Integrate out ϕ and θ, respectively)

   (b) (1 point) Exact probabilistic inference on p(z|d) is infeasible. Explain the reason why the exact inference is infeasible.

   (c) (5 points) Since exact inference is infeasible, we will use approximate inference. In particular, in this problem, we are interested in collapsed Gibbs sampling (It is called collapsed Gibbs sampling since ϕ and θ are integrated out in the inference procedure). Prove the following LDA collapsed Gibbs sampling equation:

$$p(z_{ji} = k|z/z_j, d, \alpha, \beta) \propto (N_{ki}^{(-ji)} + \alpha_k)\frac{N_{wk}^{(-ji)} + \beta_w}{N_k^{(-ji)} + \sum_w \beta_w}$$

where w = d_{ji}.
(Hint : $\Gamma(x+1) = x\Gamma(x)$)
.5.5.5

(d) (2.5 points) Note that θ_i (document-topic proportion) and ϕ_k (topic-word distribution) can be represented by using only z_{ji} (topic assignment for each word d_{ji} in document i). Let $\tilde{z} \in \{1, ..., K\}$ be a new topic assignment drawn from $p(\tilde{z}|\{z_{ji}\}_{j=1}^{N_i}, \alpha)$. Write down $\theta_{ik} := p(\tilde{z} = k|\{z_{ji}\}_{j=1}^{N_i}, \alpha)$. Similarly, $\tilde{w}$ be a token drawn from $p(\tilde{w}|\tilde{z}, \{z_{ji}, w_{ji}\}_{j=1}^{N_i}, \beta)$, write down ϕ_{kw} = $p(\tilde{w}|\tilde{z}, \{z_{ji}, w_{ji}\}_{j=1}^{N_i}, \beta)$, write down $\phi_{kw} := p(\tilde{w} = w|\tilde{z} = z, \{z_{ji}, w_{ji}\}_{j=1}^{N_i}, \beta)$ where w indexes the vocabulary. Together θ_{ik} and ϕ_{kw} fully specify the generative process described earlier. You don't need to show the derivation, but you are welcome to check out the Wikipedia page on Dirichlet-multinomial distribution and give it a shot.

2. (20 points) [DIRICHLET IN ACTION!] Particularly, Dirichlet() has downloaded some important articles (documents) from T5E, and stored them in a file "iitm_train_tiny.csv". First line of this file has "body_text", following which, there will be a number of lines. Each of the following lines will contain one document. Dirichlet() makes use of the documents to train itself. It first performs a **preprocessing step**, details of which can be found at the end of this PDF. Some of the preprocessed words can be like:

```
 Showing first 5 words of preprocessed document.....

 Document 1:
institute sport secretary soapbox hold

 Document 2:
groundwater level chennai go average
```

Dirichlet() creates a model object using a bag of words corpus. The output of the model learned using "iitm_train_tiny.csv" (and fixing number of latent topics =2) should be as follows:

```
Topics found:
Topic: 0 Word: 0.011*"institute" + 0.010*"sport" + 0.009*"believe" +
0.009*"vijaya" + 0.009*"bhaskar" + 0.008*"soapbox" + 0.008*"facilities" +
0.008*"candidates" + 0.008*"right" + 0.007*"practice"
Topic: 1 Word: 0.012*"level" + 0.012*"meter" + 0.011*"groundwater" +
0.009*"go" + 0.009*"nagar" + 0.009*"average" + 0.009*"place" +
0.008*"triplicane" + 0.008*"july" + 0.008*"chennai"
```

As observed, the first topic reflects "sports", which consists of words like "institute", "sport" with varying weightage, possibly indicating that the article is about soapbox for institute elections, for post of sports-sec. The second topic is about "water", possibly indicating an article about the ongoing water scarcity.

Now, given a test file "iitm_test_tiny.csv" in the same format as the train file, Dirichlet() has to classify the documents present there, and summarize them to you. For, example, "iitm_test_tiny.csv" has two documents, and the output of Dirichlet() on them is expected to be as follows:

```
 Testing the documents.....

Scores for document 1:
Score: 0.979525625706    Topic: 0.011*"institute" + 0.010*"sport" +
0.009*"believe" + 0.009*"vijaya" + 0.009*"bhaskar"
Score: 0.0204743854702   Topic: 0.012*"level" + 0.012*"meter" +
0.011*"groundwater" + 0.009*"go" + 0.009*"nagar"

Scores for document 2:
Score: 0.792369842529     Topic: 0.012*"level" + 0.012*"meter" +
0.011*"groundwater" + 0.009*"go" + 0.009*"nagar"
Score: 0.207630231977     Topic: 0.011*"institute" + 0.010*"sport" +
0.009*"believe" + 0.009*"vijaya" + 0.009*"bhaskar"
```

The first document is on "sports", and hence gets a higher score for the first topic, while the second documents gets a higher score for the topic "water". During testing, you may only print the top 5 words for each topic (eg. we did not print the words from "soapbox", and "average" onwards for the first and second topics respectively).

Now, it's time to put Dirichlet() to some real use, and help you out. "iitm_train_tiny.csv" and "iitm_test_tiny.csv" were used to illustrate the problem to you, and the expected forms of outputs. You can find a larger version: "iitm_train.csv". Train Dirichlet() on it and do the following (for all parts follow the output forms discussed earlier):

(a) (2 points) Report the topics (along with constituting words) found using "iitm_train.csv". Keep number of topics=2 for your implementation.

(b) (4 points) Use the model learned by Dirichlet() to classify the documents given in "iitm_test.csv", and report the output (for number of topics=2). You may report only top 5 words for each topic.

(c) (4 points) Now, go through the test documents in "iitm_test.csv". Have a qualitative overview of the content therein and the topics present. What do you interpret of the quantitative results? Explain briefly.

(d) (2 points) Report the topics (along with constituting words) found using "iitm_train.csv". Keep number of topics=7 for your implementation.

(e) (4 points) Use the model learned by Dirichlet() to classify the documents given in "iitm_test.csv", and report the output (for number of topics=7). You may report only top 5 words for each topic.

(f) (4 points) What difference did you find when number of topics was changed to 7? And why? Explain briefly.

3. (10 points) [DIRICHLET AGAIN!] Well, now that Dirichlet() is able to keep you updated with news from around insti, can it help you out with staying updated with news headlines from around the world? Let's put Dirichlet() to some bigger test! Train your bot on the larger "worldnews_train.csv", and do similar tasks:

   (a) (4 points) Report the topics (along with constituting words) found using "worldnews_train.csv". Keep number of topics=5 for your implementation.

   (b) (4 points) Use the model learned by Dirichlet() to classify the documents given in "worldnews_test.csv", and report the output (for number of topics=5). You may report only top 5 words for each topic.

   (c) (2 points) Want to explain something?

4. (10 points) [UNVEIL DIRICHLET!] Dirichlet() being so handy could be of use to us as well. Please provide its source code here, so that we can understand its working in details. Dirichlet() is available as part of standard libraries in popular languages, which we are not interested in! We expect it to be a self implemented code, not plagiarised from the web. Also, make sure to get it well-documented (else we won't evaluate your code), so that the evaluators may understand the same without hassle.

Steps of preprocessing:

1. **Tokenization**: i) Split the text into sentences and the sentences into words. ii) Lowercase the words, and iii) Remove punctuation.
2. Remove all **stopwords**.
3. Remove all words that have fewer than 3 characters.
4. **Lemmatize** the document (words in third person are changed to first person, and verbs in past and future tenses are changed into present).

NOTE: Conventional topic models like LDA suffer from a severe sparsity problem when facing extremely short texts such as social media posts (tweets for eg.). The family of Dirichlet multinomial mixture (DMM) can handle the sparsity problem. However, they are still very sensitive to ordinary and noisy words, resulting in inaccurate topic representations at the document level. You can refer to the recent AAAI 2019 paper (if you are interested in learning more about topic modelling): Dirichlet Multinomial Mixture with Variational Manifold Regularization: Topic Modeling over Short Texts, by Li et. al.