DS5230/DS4420 Unsupervised Machine Learning and Data Mining – Fall 2018 – Homework 4

Submission Instructions

- It is recommended that you complete this exercises in **Python 3** and submit your solutions as a **Jupyter notebook**.
- You may use any other language, as long as you **include a README** with simple, clear instructions on how to run (and if necessary compile) your code.
- Please upload all files (code, README, written answers, etc.) to **blackboard** in a single **zip file** named $\{firstname\}\_\{lastname\}\_DS5230/DS4220\_HW4.zip$.

Exercise 1 : Gaussian Mixture Model (70pts)

In the expectation maximization for Gaussian Mixture Model, we iterate between E-step and M-step:

E-step

$$\gamma_{nk} := p(z_n = k|x_n, \theta) \propto p(x_n|z_n = k, \theta)p(z_n = k) \propto \frac{p(x_n|z_n = k, \theta)p(z_n = k)}{\sum_{k=1}^{K} p(x_n|z_n = k, \theta)p(z_n = k)} \tag{1}$$

M-step

$$N_k = \sum_{n=1}^{N} \gamma_{nk} \tag{2}$$

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk} x_n \tag{3}$$

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_{nk}(x_n - \mu_k)(x_n - \mu_k)^T \tag{4}$$

a. (10pts) Explain how the cluster assignment is difference between cluster assignment (in K-means) and E-step (in GMM).

b. (10pts) Explain how the M-step in GMM is different from the centriods update step in K-means.

c. (50pts) Now implement Expectation Maximization and apply it on 3 datasets, `dataset1.txt`, `dataset2.txt`, and `dataset3.txt`. Each dataset contains two columns of features and a third column with labels. To evaluate your clustering results, consider 3 different metrics:

   Note that you may use functions from scikit-learn to compute metrics, but you need to implement GMM algorithm from scratch and are NOT allowed to directly call function from scikit-learn.

   - The normalized mutual information (NMI)

   $$\mathrm{NMI}(Y;Z) = \frac{\mathrm{I}(Y;Z)}{\sqrt{H(Y)H(Y)}}$$

     scikit-learn: `scklearn.metrics.normalized_mutual_info_score`
   - The Calinski-Harabaz (CH) index $\mathrm{tr}(S_B)/\mathrm{tr}(S_W)$.

     scikit-learn: `sklearn.metrics.calinski_harabaz_score`
   - The Silhouette coefficient (SC) (*see attached note*).

     scikit-learn: `sklearn.metrics.calinski_harabaz_score`

   As a simplifying assumption, you may take the covariance matrix for each cluster to be diagonal. Once again, your implementation should perform multiple restarts and accept initial values for the mean and variance as inputs. Initialize the mean for each of your clusters by sampling from a Gaussian distribution centered on a random point in your data. Initialize the variance along each dimension to a random fraction of the total variance in the data.

   For each dataset, plot the log likelihood, the CH index, the SC, and the NMI as a function of $K = 2, 3, 4, 5$. Also include the scatter plots that you made for k-means, using $z_n = \arg\max_k \gamma_{nk}$ to determine the cluster assignments.

Hint: When implementing your E-step, you need to be careful about numerical underflow and overflow when calculating

$$\gamma_{nk} = p(x_n, z_n = k \,|\, \theta) / \sum_l p(x_n, z_n = l \,|\, \theta).$$

A common trick is to first calculate the log probabilities

$$\omega_{nk} = \log p(x_n, z_n = k \,|\, \theta), \qquad \omega_n^* = \max_k \omega_{nk}.$$

Before exponentiation you can now first subtract ω_n^*, which is equivalent to dividing both the numerator and denominator by $\exp \omega_n^*$

$$\gamma_{nk} = \exp(\omega_{nk} - \omega_n^*) / \sum_l \exp(\omega_{nl} - \omega_n^*).$$

Explain in your code comments why this helps prevent underflow/overflow.

Exercise 3 : Learning Title Topics using Latent Dirichlet Allocation (40pts)

In this section you will apply Latents Dirichlet Allocation (LDA) to the academic publication titles dataset (`publications.txt`), and compare the results with what you got when doing Principle Component Analysis (PCA). By comparison, hopefully you could see the similarity between them in the sense that LDA can also be regarded as some form of matrix factortization.

For PCA and LDA, you may use packages from scikit-learn and don't need to implement them from scratch. are going to use scikit-learn to implement both algorithms, including converting to texts to word count vectors.

1. (0pt) Firstly you need to clean up the raw dataset. I wrote a script `prep.py` which does tokenization, stemming, filtering non-alphabets-numeric , numeric replacement, stop words removing. When it finishes, the texts are written into a file called `titles_prep.txt`, so that you only need to do this preprocessing once and can import that file as the actual dataset.

   In this question **we only need to run the script** `prep.txt`. You could either import the function in notebook or directly execute the python script in terminal. The purpose is to get you familiar with the basic text preprocessing if you are not. For the following questions please load `titles_prep.txt` as the dataset.

2. (20pts) Apply LDA to the same dataset.

Step 1. Load the dataset and convert each title to word count vector using the the function `CountVectorizer` in scikit-learn. Set the parameter `min_df` to be 800, then you vocabulary size will be 1023. (you can also try other methods to truncate the vocabulary and get a size roughly around 1000).

Step 2. Fit the word count vectors to lda and set the number of topics as **10, 20, 50**, repsectively. For each experiment, print out the top 10 words for each topic in a descending order. Explain the trend and change as you change the number of principle components.

3. (5pts) Now apply PCA to the datset.

   Step 1. Simiarly you need to convert the texts to word count vectors.

   Step 2.By setting the number of principle components to **10, 20, 50**, also list the top 10 words for each component (topic) in a descending order. Explain the trend and change as you change the number of principle components.
   To be more specific, you normalize a component eigenvector, and sort the absolute value of each element in it. Then you can retrive the top words with largest values using your vovabulary.

4. (5pts) Based on the results, do you think LDA outperforms pca in topic modeling task? Explain your reason for why or why not.