CSCI567 2017 Homework Assignment 4

Due date and time for the algorithmic part Oct. 29th (Sunday), 11:59pm *(No late day!)*

Due date and time for the programming part Nov. 7th (Tuesday), 11:59pm

Starting point Your repository will have a directory 'homework4/'. Please do not change the name of this repository or the names of any files/folders we have added to it. Please perform a `git pull` to retrieve these files. You will find within 'homework4/':

- Three python scripts `kmeans.py`, `gmm.py`, and `nb_spam.py`, which you will be amending by adding codes for questions in Sect. 6, 7, and 8, respectively.
- Four scripts `kmeans.sh`, `gmm.sh`, `nb1.sh`, and `nb2.sh`. You will use them to perform training and testing, and generate the output files.
- Datasets: `hw4_circle.json`, `hw4_blob.json`, and the folder '`enron/`' containing the email dataset.

Submission instructions The following will constitute your submission (in 'homework4/'):

- A PDF report named `firstname_lastname_USCID.pdf`, which contains your solutions for questions in Sect. 1, Sect. 2, Sect. 3, and Sect. 4. For your written report, your answers must be typeset with LaTeX and generated as a PDF file. No handwritten submission will be permitted. Note that this part is due on Oct. 29th (Sunday), 11:59pm *(No late day!)*.
- The python scripts `kmeans.py`, `gmm.py`, and `nb_spam.py`, amended with the codes you added for Sect. 6, 7, and 8, respectively. Be sure to commit your changes!
- Four .json files, `kmeans.json`, `gmm.json`, `nb1.json`, and `nb2.json`, which will be the outputs of `kmeans.sh`, `gmm.sh`, `nb1.sh`, and `nb2.sh`, respectively.

Note that if your submission doesn't include the .json files mentioned above, we will assume that you do not do the corresponding parts of the homework.

Collaboration You may discuss with your classmates. However, you need to write your own solutions and submit separately. Also in your written report, you need to list with whom you have discussed for each problem. Please consult the syllabus for what is and is not acceptable collaboration.

Algorithmic component

1 Generative models

[Recommended maximum time spent: 1 hour]

Q1.1 Let $X \in \mathbb{R}$ be a random variable. We assume that it is uniformly-distributed on some unknown interval $(0, \theta]$, where $\theta > 0$. In particular,

$$P(X = x|\theta) = \begin{cases} \frac{1}{\theta} & \text{, if } x \in (0, \theta], \\ 0 & \text{, otherwise,} \end{cases} \tag{1}$$

$$= \frac{1}{\theta}\mathbf{1}[0 < x \leq \theta], \tag{2}$$

where $\mathbf{1}$ is an indicator function that outputs 1 when the condition is true, and 0 otherwise.

Suppose $x_1, x_2, \ldots, x_\mathsf{N}$ are drawn i.i.d. from this distribution. Write down the likelihood of the observations $P(x_1, \ldots, x_\mathsf{N})$. What is the maximum likelihood (ML) estimate of θ? Give a sentence or two explaining why your equation corresponds to the maximum likelihood estimate.

What to submit: The equation for the likelihood (at most two lines), the final equation of ML estimate for θ (one line), and fewer-than-two sentences of explanation.

Q1.2 Now suppose X is distributed according to a **mixture** of two uniform distributions: one on some unknown interval $(0, \theta_1]$ and the other on $(0, \theta_2]$, where $0 < \theta_1 \leq \theta_2$. In particular,

$$P(X = x) = \omega_1 U(X = x|\theta_1) + \omega_2 U(X = x|\theta_2), \tag{3}$$

where U is the uniform distribution defined as in Eq. (1) or Eq. (2), and ω_1, ω_2 are mixture weights such that

$$\omega_1 \geq 0, \omega_2 \geq 0, \text{ and } \omega_1 + \omega_2 = 1. \tag{4}$$

Suppose $x_1, x_2, \ldots, x_\mathsf{N}$ are drawn i.i.d. from this mixture of uniform distributions. We will use an EM algorithm to derive the ML estimates of the parameters in this model. Answer the following three questions.

- First, what is the form of $P(k|x_n, \theta_1, \theta_2, \omega_1, \omega_2)$, where $k \in \{1, 2\}$ indicates the corresponding mixture component? You answer should be explicit. You may include the indicator function as in Eq. (2) and the U function as in Eq. (3).
- Second, what is the form of the expected complete-data log-likelihood of the observations $\{x_1, \ldots, x_\mathsf{N}\}$, given that the parameters from the last EM iteration are $\{\theta_1^{\text{OLD}}, \theta_2^{\text{OLD}}, \omega_1^{\text{OLD}} > 0, \omega_2^{\text{OLD}} > 0\}$, where $\theta_2^{\text{OLD}} \geq \max\{x_1, \ldots, x_\mathsf{N}\} \geq \theta_1^{\text{OLD}} \geq \min\{x_1, \ldots, x_\mathsf{N}\}$? You answer should be explicit. You may include the indicator function as in Eq. (2) and the U function as in Eq. (3).

  Hint: Please refer to page 34/44 and other related pages of lecture 14.

- Third, what are the forms of the M-step updates for θ_1 and θ_2? You may use $P_{\text{OLD}}(k|x_n) = P(k|x_n, \theta_1^{\text{OLD}}, \theta_2^{\text{OLD}}, \omega_1^{\text{OLD}}, \omega_2^{\text{OLD}})$, where $k \in \{1, 2\}$, to simplify your derivation/answer if needed. You can view $\log 0 = -\infty$ and $0 \times \log 0 = 0$ in this question.

What to submit: The form of $P(k|x_n, \theta_1, \theta_2, \omega_1, \omega_2)$ (at most two lines), the form of the expected complete-data log-likelihood (at most five lines), and the forms of the M-step update (at most ten lines).

2 Mixture density models

[Recommended maximum time spent: 1 hour]

Consider a density model given by a mixture distribution

$$P(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k P(\boldsymbol{x}|k),$$

$$\text{where } \pi_k \geq 0 \ \forall k \in [K] \text{ and } \sum_{k=1}^{K} \pi_k = 1,$$

and suppose that we partition the vector $\boldsymbol{x}$ into two parts so that $\boldsymbol{x} = [\boldsymbol{x}_a^T, \boldsymbol{x}_b^T]^T$. Show that the conditional density $P(\boldsymbol{x}_b|\boldsymbol{x}_a)$ is itself a mixture distribution. That is,

$$P(\boldsymbol{x}_b|\boldsymbol{x}_a) = \sum_{k=1}^{K} \lambda_k P(\boldsymbol{x}_b|\boldsymbol{x}_a, k),$$

$$\text{where } \lambda_k \geq 0 \ \forall k \in [K] \text{ and } \sum_{k=1}^{K} \lambda_k = 1.$$

Q2.1 Find an expression for the mixing coefficients λ_k of the component densities in terms of π_k and $P(\boldsymbol{x}_a|k)$. Do not forget to verify if your answer obeys the constraint on λ_k mentioned above.

Hint: a) λ_k is a function of $\boldsymbol{x}_a$ instead of a constant. b) You may consider Bayes rule for derivation.

What to submit: No more than ten lines of derivation that leads to an expression for the mixing coefficients λ_k.

3 The connection between GMM and K-means

[Recommended maximum time spent: 1 hour]

Consider a Gaussian mixture model (GMM) in which all components have (diagonal) covariance $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}$ and the K-means algorithm introduced in lecture 14.

Q3.1 In the case where both the GMM and the K-means algorithm have K components and the parameters π_k are pre-defined to be nonzero $\forall k \in [K]$, show that in the limit $\sigma \to 0$, **maximizing** the following expected complete-data log likelihood w.r.t. $\{\boldsymbol{\mu}_k\}_{k=1}^K$ for the GMM model

$$\sum_n^N \sum_k^K \gamma(z_{nk}) \log p(\boldsymbol{x}_n, z_n = k) = \sum_n^N \sum_k^K \gamma(z_{nk})[\log \pi_k + \log \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \sigma^2 \boldsymbol{I})],$$

$$\text{where} \quad \gamma(z_{nk}) = \frac{\pi_k \exp(-||\boldsymbol{x}_n - \boldsymbol{\mu}_k||^2/2\sigma^2)}{\sum_j \pi_j \exp(-||\boldsymbol{x}_n - \boldsymbol{\mu}_j||^2/2\sigma^2)}$$

is equivalent (up to a scaling or constant factor) to **minimizing** the distortion measure J w.r.t. $\{\boldsymbol{\mu}_k\}_{k=1}^K$ for the K-means algorithm given by

$$J = \sum_k^K \sum_n^N r_{nk} ||\boldsymbol{x}_n - \boldsymbol{\mu}_k||_2^2,$$

$$\text{where} \quad r_{nk} = \begin{cases} 1, & \text{if } k = \arg\min_{k'} ||\boldsymbol{x}_n - \boldsymbol{\mu}_{k'}||_2^2, \\ 0, & \text{otherwise.} \end{cases}$$

Hint: Start by showing that $\gamma(z_{nk}) \to r_{nk}$ as $\sigma \to 0$. Note that for this question the only set of parameters to learn for the GMM are $\{\boldsymbol{\mu}_k\}_{k=1}^K$. Any term independent of $\{\boldsymbol{\mu}_k\}_{k=1}^K$ can be treated as a constant.

What to submit: Derivation and reasoning with less than ten lines.

4 Naive Bayes

[Recommended maximum time spent: 1 hour]

Recall the naive Bayes model we have seen in class. Given a random variable $X \in \mathbb{R}^{\mathsf{D}}$ and a dependent class variable $Y \in [\mathsf{C}]$, the joint distribution of features X and class Y is defined as

$$P(X = \boldsymbol{x}, Y = c) = P(Y = c) P(X = \boldsymbol{x} | Y = c) \tag{5}$$

$$= P(Y = c) \prod_{d=1}^{\mathsf{D}} P(X_d = x_d | Y = c) \tag{6}$$

In this problem, we consider a naive Bayes model where each feature x_d of each class c is modeled as a (different) Gaussian. That is,

$$P(X_d = x_d | Y = c; \boldsymbol{\mu}, \boldsymbol{\sigma}) = \frac{1}{\sqrt{2\pi\sigma_{cd}^2}} \exp\left(-\frac{(x_d - \mu_{cd})^2}{2\sigma_{cd}^2}\right), \tag{7}$$

where μ_{cd} and σ_{cd} are the mean and the standard deviation, respectively.

Moreover, we model Y as a multinomial distribution with parameter $\boldsymbol{\pi}$ (Note: this $\boldsymbol{\pi}$ is a parameter, whereas π in Eqn. (7) is a constant). That is,

$$P(Y = c; \boldsymbol{\pi}) = \pi_c \geq 0 \ \ \forall c \in [\mathsf{C}], \text{ and } \sum_c^{\mathsf{C}} \pi_c = 1. \tag{8}$$

Q4.0 (Warm-up) What are the parameters to be learned in this model?

What to submit: Nothing.

Q4.1 Given the dataset $\{(\boldsymbol{x}_n \in \mathbb{R}^{\mathsf{D}}, y_n \in [\mathsf{C}])\}_{n=1}^{\mathsf{N}}$, assumed to be drawn i.i.d. from this model, write down explicitly the expression for the joint log-likelihood.

What to submit: The expression for the joint log-likelihood (no more than ten lines). Parameters to be learned should all be in the expression.

Q4.2 Using the objective in **Q4.1**, derive the maximum likelihood (ML) estimates of the parameters. You should be able to compute these estimates from the training data $\{(\boldsymbol{x}_n \in \mathbb{R}^{\mathsf{D}}, y_n \in [\mathsf{C}])\}_{n=1}^{\mathsf{N}}$.

What to submit: The final equation of ML estimate as well as your short (no more than five lines) derivation for *each* parameter.

Programming component

5 High-level descriptions

5.1 Datasets

For Sect. 6 and 7, we will use 2-dimensional synthetic data `hw4_circle.json` and `hw4_blob.json` to perform clustering and density estimation. For Sect. 8, we will use the email dataset released during the 2001 Enron investigation for (binary) spam classification. See the text of the corresponding sections for more details.

5.2 Tasks

You will be asked to implement the K-means algorithm for clustering (Sect. 6), the Gaussian mixture model for density estimation (Sect. 7), and the naive Bayes model for binary classification (Sect. 8). Specifically, you will

- For Sect. 6: finish implementing the function `kmeans_clustering` and related functions in `kmeans.py`. Run the script `kmeans.sh`, which will output `kmeans.json`. Add, commit, and push `kmeans.json` and `kmeans.py`.
- For Sect. 7: finish implementing the function `gmm_clustering` in `gmm.py`. Run the script `gmm.sh`, which will output `gmm.json`. Add, commit, and push `gmm.json` and `gmm.py`.
- For Sect. 8: finish implementing the following three functions—`nb_train`, `nb_predict`, and `nb_train_smoothing`—in `nb_spam.py`. Run the scripts `nb1.sh` and `nb2.sh`, which will output `nb1.json` and `nb2.json`, respectively. Add, commit, and push `nb_spam.py`, `nb1.json`, and `nb2.json`.

As in the previous homework, you are not responsible for loading/pre-processing data; we have done that for you. For specific instructions, please refer to text in Sect. 6, 7, 8, and the instructions in their corresponding scripts.

5.3 Cautions

Please do not import packages that are not listed in the provided code. Follow the instructions in each section strictly to code up your solutions. **Do not change the output format**. **Do not modify the code unless we instruct you to do so**. A homework solution that does not match the provided setup, such as format, name, initializations, etc., **will not** be graded. It is your responsibility to **make sure that your code runs with the provided commands and scripts on the VM**. Finally, make sure that you **git add, commit, and push all the required files**, including your code and generated output files.

6 K-means

As introduced in the class, the K-means algorithm tries to minimize the following distortion measure (or objective function):

$$J = \sum_{k}^{K} \sum_{n}^{N} r_{nk} ||\boldsymbol{x}_n - \boldsymbol{\mu}_k||_2^2,$$

where r_{nk} is an indicator variable:

$$r_{nk} = \begin{cases} 1, & \text{if } k = \arg\min_{k'} ||\boldsymbol{x}_n - \boldsymbol{\mu}_{k'}||_2^2, \\ 0, & \text{otherwise.} \end{cases}$$

and $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k\}$ are the cluster centers, each with the same dimensionality of data points.

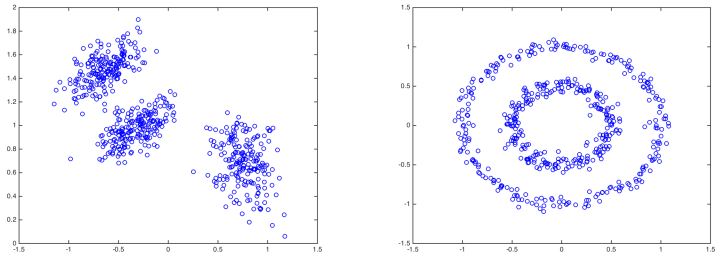

Figure 1: `hw4_blob` dataset (left) and `hw4_circle` dataset (right)

Q6.1 Implement K-means using random initialization for cluster centers. Run the algorithm on 2 two-dimensional synthetic datasets `hw4_circle.json` and `hw4_blob.json` (see Fig. 1 for illustration) with different values of $K \in \{2, 3, 5\}$. The algorithm should run until none of the cluster assignments are changed.

What to do and submit: Finish the function `kmeans_clustering` and related functions in `kmeans.py`. *Please do not change the random seeds set in our template code.* Detailed implementation instructions are provided in the python script. Then, run `kmeans.sh`, which will generate a `kmeans.json` file containing cluster center and the cluster assigned to each data point with all values of K. Add, commit, and push the modified `kmeans.py` and `kmeans.json` before the due date.

Q6.2 You are encouraged to plot the clustering assignments by different colors and markers. Does the algorithm work well in separating the two circles in the `hw4_circle.json` dataset and why?

What to submit: Nothing.

7 Gaussian Mixture Models

Q7.1 In this problem, you will implement an EM algorithm to fit a Gaussian mixture model on the `hw4_blob.json` dataset.

What to do and submit: Finish the implementation of the function `gmm_clustering` in `gmm.py`. Detailed implementation instructions are provided in the python script. Then, run `gmm.sh`. It will run 5 rounds of your EM algorithm with the number of components $K = 3$ and generate a `gmm.json` file containing the mean and co-variance matrices of all the three Gaussian components. Add, commit, and push the modified `gmm.py` and `gmm.json` before the due date.

Q7.2 You are encouraged to plot the most likely clustering assignments by different colors and markers. Are the results from those 5 rounds the same? Why or why not?

What to submit: Nothing.

8 Naive Bayes

We will build a (binary) spam classifier using a naive Bayes model that classifies each document as "Spam" (1) or "Ham" (0).

Dataset We will use the dataset drawn from emails released during the 2001 Enron investigation.

In what follows, we will describe this dataset in detail. *However, note that we have provided a python function that handles reading these files for you.*

In the directory `enron`, you will find

- Word ID to word mapping in `ind_to_vocab.txt`. Specifically, each line consists of word ID and its corresponding actual word (space-separated). There are 158,963 words in the vocabulary.
- 20,229 training emails in `train_features.txt` and their labels in `train_labels.txt`.
- 6,743 validation emails in `val_features.txt` and their labels in `val_labels.txt`.

```
0
word_id_0,1 frequency_word_id_0,1
word_id_0,2 frequency_word_id_0,2
.
.
word_id_0,D0 frequency_word_id_0,D0
#
1
word_id_1,1 frequency_word_id_1,1
word_id_1,2 frequency_word_id_1,2
.
.
word_id_1,D2 frequency_word_id_1,D1
#
.
.
.
#
N
word_id_N,1 frequency_word_id_N,1
word_id_N,2 frequency_word_id_N,2
.
.
word_id_N,DN frequency_word_id_N,DN
#
```

Figure 2: Format of `*_features.txt`

- 6,744 sampled test emails in `sampled_test_features.txt` and their labels in `sampled_test_labels.txt`.

The input format of `*_features.txt` is shown in Fig. 2. Each of these features files consists of multiple documents separated by `#`. Each document starts with its ID `n` followed by `Dn` lines, where each line consists of word id and its frequency (space-separated). For example, the word `word_id_3,5` appears in document `3` for `frequency_word_id_3,5` times. The labels of these documents are in `*_labels.txt`, in which each line consists of document ID and its label (0 or 1).

Q8.1 Understand Data Format In `nb_spam.py`, we have provided the function `load_docs` which reads documents (words and their corresponding frequencies) for you. This function takes in two file names (features and labels), and outputs two lists. The first list contains documents, each of which is a list of tuples of word and its frequency in the document. The second contains labels of those documents. Your task is to read this function to understand the format of the data you will be working with.

What to do and submit: Nothing.

Q8.2 Train and Predict Please finish the implementation of the functions—`nb_train` and `nb_predict`—in `nb_spam.py`:

- `nb_train` estimates the parameters of the naive Bayes model, including the probability distribution of class being Ham or Spam *a priori* as well as the class-specific word probability distributions.
- `nb_predict` uses the estimates from `nb_train` to predict which class each new document belongs to.

Please follow the following implementation details. First, use the log-likelihood instead of the likelihood to avoid any numerical underflow issues. Moreover, if your model decides that a document has an equal chance of being Ham (0) or Spam (1), it should predict a Spam (1).

Finally, we will have to take care of unseen words. *Unseen words* are defined as words that do not appear in the **training** documents. In this question, we will ignore all unseen words during prediction. In addition, if you encounter computing log of a zero, replace the output with -1e50. This scenario could happen when a word appears in the training documents that belong to one class but it does not appear in those that belong to the other class.

What to do and submit: Finish the implementation of the functions `nb_train` and `nb_predict` in `nb_spam.py`. Then, run script `nb1.sh`, which will generate a `nb1.json` file containing training, validation, and test accuracies. You should see that all accuracies are above 97%. Add, commit, and push the modified `nb_spam.py` and `nb1.json` before the due date.

Q8.3 Train with smoothing We now take another approach to dealing with unseen words: a smoothing technique. Let $\alpha > 0$ be a continuous-valued smoothing parameter. For each word in the vocabulary in each class, you will assume that you have seen it α times before even seeing any training data. That is, you should add α to its *count* before you train your model. This assumption will affect class-specific word distribution estimates. (*Hint*: you should see that α (and adjusted counts) affects both the numerator and the denominator in the formula of your parameter estimates).

Please finish the implementation of the function `nb_train_smoothing` in `nb_spam.py` with the assumption described above. `nb_train_smoothing` estimates the parameters of the naive Bayes model, similar to `nb_train`. Implementation details in the previous question apply to this question. Try $\alpha \in \{$1e-8, 1e-7, 1e-6, 1e-5, 1e-4$\}$.

What to do and submit: Finish the implementation of the function `nb_train_smoothing` in `nb_spam.py`. Then, run script `nb2.sh`, which will generate a `nb2.json` file containing training, validation, and test accuracies for all values of α. You should see that, for the best α, all accuracies are above 98%. Add, commit, and push the modified `nb_spam.py` and `nb2.json` before the due date.