ISYE 6740, Summer 2024, Homework 3

100 points

Prof. Yao Xie

1. Implementing EM for MNIST dataset. [30 points]

Implement the EM algorithm for fitting a Gaussian mixture model for the MNIST handwritten digits dataset. For this question, we reduce the dataset to be only two cases, of digits "2" and "6" only. Thus, you will fit GMM with $C = 2$. Use the data file data.mat or data.dat. True label of the data are also provided in label.mat and label.dat.

The matrix images is of size 784-by-1990, i.e., there are 1990 images in total, and each column of the matrix corresponds to one image of size 28-by-28 pixels (the image is vectorized; the original image can be recovered by mapping the vector into a matrix).

First use PCA to reduce the dimensionality of the data before applying to EM. We will put all "6" and "2" digits together, to project the original data into 4-dimensional vectors.

Now implement EM algorithm for the projected data (with 4-dimensions).
(In this question, we use the same set of data from the provided data files for training and testing)

1. (10 points) Implement EM algorithm yourself. Use the following initialization

   - initialization for mean: random Gaussian vector with zero mean
   - initialization for covariance: generate two Gaussian random matrix of size n-by-n: S_1 and S_2, and initialize the covariance matrix for the two components are $\Sigma_1 = S_1 S_1^T + I_n$, and $\Sigma_2 = S_2 S_2^T + I_n$, where I_n is an identity matrix of size n-by-n.

   Plot the log-likelihood function versus the number of iterations to show your algorithm is converging.

2. (10 points) Report the fitted GMM model when EM has terminated in your algorithms as follows:

   - The numerical weights for each component
   - The mean of each component by mapping it back to the original space and reformatting the vectors into 28-by-28 matrices. These should be displayed as images, ideally corresponding to a kind of "average" of the images.
   - Two 4-by-4 covariance matrices by visualizing their intensities (i.e. a gray-scaled image or heatmap.)

3. (10 points) Use the τ_k^i to infer the labels of the images, and compare with the true labels. Report the mis-classification rate (1 - Accuracy) for digits "2" and "6" respectively. Perform K-means clustering with $K = 2$ (you may call a package or use the code from your previous homework). Find out the mis-classification rate for digits "2" and "6" respectively, and compare with GMM. Which one achieves the better performance?

2. Optimization (20 points).

Consider a simplified logistic regression problem. Given m training samples (x^i, y^i), $i = 1, \ldots, m$. The data $x^i \in \mathbb{R}$, and $y^i \in \{0, 1\}$. To fit a logistic regression model for classification, we solve the following optimization problem, where $\theta \in \mathbb{R}$ is a parameter we aim to find:

$$\max_{\theta} \ell(\theta), \tag{1}$$

where the log-likelihood function

$$\ell(\theta) = \sum_{i=1}^{m} \left\{ -\log(1 + \exp\{-\theta^T x^i\}) + (y^i - 1)\theta^T x^i \right\}.$$

1. (5 points) Show step-by-step mathematical derivation for the gradient of the cost function $\ell(\theta)$ in (1).

2. (5 points) Write a pseudo-code for performing **gradient descent** to find the optimizer θ^*. This is essentially what the training procedure does.

3. (5 points) Write the pseudo-code for performing the **stochastic gradient descent** algorithm to solve the training of logistic regression problem (1). Please explain the difference between gradient descent and stochastic gradient descent for training logistic regression.

4. (5 points) We will **show that the training problem in basic logistic regression problem is concave.** Derive the Hessian matrix of $\ell(\theta)$ and based on this, show the training problem (1) is concave. Explain why the problem can be solved efficiently and gradient descent will achieve a unique global optimizer, as we discussed in class.

3. Bayes Classifier for spam filtering (30 points)

In this problem, we will use the Bayes Classifier algorithm to fit a spam filter by hand. This will enhance your understanding to the Bayes classifier and build intuition. This question does not involve any programming but only derivation and hand calculation. Tools can be used (Python, Excel, Etc.) but all calculations and derivations must still be provided.

Spam filters are used in all email services to classify received emails as "Spam" or "Not Spam". A simple approach involves maintaining a vocabulary of words that commonly occur in "Spam" emails and classifying an email as "Spam" if the number of words from the dictionary that are present in the email is over a certain threshold. We are given the vocabulary consists of 15 words

$V = \{\text{secret, offer, low, price, valued, customer, today, dollar, million, sports, is, for, play, healthy, pizza}\}$.

We will use V_i to represent the ith word in V. As our training dataset, we are also given 3 example spam messages,

- million dollar offer for today
- secret offer today
- secret is secret

and 4 example non-spam messages

- low price for valued customer today
- play secret sports today
- sports is healthy
- low price pizza today

Recall that the Naive Bayes classifier assumes the probability of an input depends on its input feature. The feature for each sample is defined as $x^{(i)} = [x_1^{(i)}, x_2^{(i)}, \ldots, x_d^{(i)}]^T$, $i = 1, \ldots, m$ and the class of the ith sample is $y^{(i)}$. In our case the length of the input vector is $d = 15$, which is equal to the number of words in the vocabulary V. Each entry $x_j^{(i)}$ is equal to the number of times word V_j occurs in the i-th message.

1. (5 points) Calculate class prior $\mathbb{P}(y = 0)$ and $\mathbb{P}(y = 1)$ from the training data, where $y = 0$ corresponds to spam messages, and $y = 1$ corresponds to non-spam messages. Note that these class prior essentially corresponds to the frequency of each class in the training sample. Write down the feature vectors for each spam and non-spam messages.

2. (15 points) Assuming the keywords follow a multinomial distribution, the likelihood of a sentence with its feature vector x given a class c is given by

$$\mathbb{P}(x|y = c) = \frac{n!}{x_1! \cdots x_d!} \prod_{k=1}^{d} \theta_{c,k}^{x_k}, \quad c = \{0, 1\}$$

where $n = x_1 + \cdots x_d$, $0 \leq \theta_{c,k} \leq 1$ is the probability of word k appearing in class c, which satisfies

$$\sum_{k=1}^{d} \theta_{c,k} = 1, \quad c = \{0, 1\}.$$

Given this, the complete log-likelihood function for our training data is given by

$$\ell(\theta_{0,1}, \ldots, \theta_{0,d}, \theta_{1,1}, \ldots, \theta_{1,d}) = \sum_{i=1}^{m} \sum_{k=1}^{d} x_k^{(i)} \log \theta_{y^{(i)},k}$$

(In this example, $m = 7$.) Calculate the maximum likelihood estimates of $\theta_{0,1}$, $\theta_{0,7}$, $\theta_{1,1}$, $\theta_{1,15}$ by maximizing the log-likelihood function above.
(Hint: We are solving a constrained maximization problem and you will need to introduce Lagrangian multipliers and consider the Lagrangian function.)

3. (10 points) Given a test message "today is secret", using the Naive Bayes classier that you have trained in Part (a)-(b), calculate the posterior and decide whether it is spam or not spam. Derivations must be shown in your report.

4. Comparing classifiers: Divorce classification/prediction (20 points)

In lectures, we learn different classifiers. This question is compare them on two datasets. Python users, please feel free to use Scikit-learn, which is a commonly-used and powerful Python library with various machine learning tools. But you can also use other similar libraries in other languages of your choice to perform the tasks.

This dataset is about participants who completed the personal information form and a divorce predictors scale. The data is a modified version of the publicly available at `https://archive.ics.uci.edu/dataset/539/divorce+predictors+data+set` (by injecting noise so you will not get the exactly same results as on UCI website). The dataset **marriage.csv** is contained in the homework folder. There are 170 participants and 54 attributes (or predictor variables) that are all real-valued. The last column of the CSV file is label y (1 means "divorce", 0 means "no divorce"). Each column is for one feature (predictor variable), and each row is a sample (participant). A detailed explanation for each feature (predictor variable) can be found at the website link above. Our goal is to build a classifier using training data, such that given a test sample, we can classify (or essentially predict) whether its label is 0 ("no divorce") or 1 ("divorce").

We are going to compare the following classifiers (**Naive Bayes, Logistic Regression, and KNN**). Use the first 80% data for training and the remaining 20% for testing. If you use scikit-learn you can use train_test_split to split the dataset.

Remark: Please note that, here, for Naive Bayes, this means that we have to estimate the variance for each individual feature from training data. When estimating the variance, if the variance is zero to close to zero (meaning that there is very little variability in the feature), you can set the variance to be a small number, e.g., $\epsilon = 10^{-3}$. We do not want to have

include zero or nearly variance in Naive Bayes. This tip holds for both Part One and Part Two of this question.

1. (10 points) Report testing accuracy for each of the three classifiers. Comment on their performance: which performs the best and make a guess why they perform the best in this setting.

2. (10 points) Now perform PCA to project the data into two-dimensional space. Build the classifiers (**Naive Bayes, Logistic Regression, and KNN**) using the two-dimensional PCA results. Plot the data points and decision boundary of each classifier in the two-dimensional space. Comment on the difference between the decision boundary for the three classifiers. Please clearly represent the data points with different labels using different colors.