ISYE 6740, Fall 2025, Prof. Yao Xie
Homework 4

100 points

Student Name Here

Provided Data and Submission Requirements:

- All results, explanations, and math must be included in your PDF submission. Handwritten work is not accepted per the syllabus. Screenshots of work is also not accepted.
- If applicable, questions marked with **GS** must be submitted to Gradescope. Failure to pass gradescope tests will result in penalties to the points for that question.

This assignment does not have any gradescope requirements
Assignment Data:

- Q4: Comparing Classifiers: Divorce Classification / Prediction [marriage.csv]

1. Optimization (25 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. (10 points) We will **prove 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.

2. *Fair* Gaussian mixture model (25 points).

This question builds on EM for GMM model estimation, as well as optimization lectures; the purpose is to practice constrained optimization. Recall the EM step in the estimation of Gaussian mixture models. Now suppose that in addition to all the original setup, we require a *fairness* constraint: the weights of the Gaussian components cannot be smaller than a pre-specified constant c. We will derive a modified EM algorithm for the Gaussian model, with two components $K = 2$, when the weights for each component are required to be greater than c: $\pi_1 \geq c$, and $\pi_2 \geq c$ (with $c < 0.5$).

Show the modified EM algorithm, as well as the complete mathematical derivations.

Hint: The only thing you will need to change is that in the M-step, we have to solve a constrained optimization problem with the original constraint $\pi_1 + \pi_2 = 1$, as well as $\pi_1 > c$ and $\pi_2 > c$. You will need to introduce additional Lagrangian multipliers for the new constraints and discuss the cases when the constraints are active (or not active) using the KKT condition (in particular, the complementarity condition).

Expectation: The expectation step is not modified by the extra constraints.

$$\tau_k^i = \frac{\pi_k \mathcal{N}(x^i|\mu_k, \Sigma_k)}{\sum_{k'=1\ldots K} \pi_{k'} \mathcal{N}(x^i|\mu_k, \Sigma_k)}$$

Maximization: The maximization step incorporates the constraint terms. Without the extra terms, the objective function is:

$$f(\theta) = \sum_{i=1}^{m}\sum_{k=1}^{K} \tau_k^i(\log(\pi_k) - \frac{1}{2}(x^i - \mu_k)^T\Sigma_k^{-1}(x^i - \mu_k) - \frac{1}{2}\log(|\Sigma_k|) - \frac{n}{2}\log(2\pi))$$

With the initial constraint Lagrangian term:

$$f(\theta) = \sum_{i=1}^{m}\sum_{k=1}^{K} \tau_k^i(\log(\pi_k) - \frac{1}{2}(x^i - \mu_k)^T\Sigma_k^{-1}(x^i - \mu_k) - \frac{1}{2}\log(|\Sigma_k|) - \frac{n}{2}\log(2\pi)) + \lambda(1 - \sum_{i=1}^{K}\pi_k)$$

So, we need to add additional Lagrangian terms to enforce the constraint. I will make these λ_1 and λ_2 while designating the initial constraint λ_0. Additionally, I will expand the π summations since $K = 2$.

$$f(\theta) = \sum_{i=1}^{m}\sum_{k=1}^{K} \tau_k^i(\log(\pi_k) - \frac{1}{2}(x^i - \mu_k)^T\Sigma_k^{-1}(x^i - \mu_k) - \frac{1}{2}\log(|\Sigma_k|) - \frac{n}{2}\log(2\pi)) \\ + \lambda_0(1 - \pi_1 - \pi_2) + \lambda_1(c - \pi_1) + \lambda_2(c - \pi_2)$$

As far as the maximization goes, we perform the usual steps: take the partial derivatives and set equal to zero. We can see that the extra constraints didn't impact the μ_k or Σ_k

terms, so their values are unchanged from what we already know:

$$\mu_k = \frac{\sum_{i=1}^{k} \tau_k^i x^i}{\sum_{i=1}^{k} \tau_k^i} \quad \Sigma_k = \frac{\sum_{i=1}^{k} \tau_k^i (x^i - \mu_k)(x^i - \mu_k)^T}{\sum_{i=1}^{k} \tau_k^i}$$

The constraints impact the maximization of π_k. As seen in the normal EM derivation, the terms unrelated to π_k drop out. We will take derivatives with respect to π_1 and π_2.

$$0 = \frac{\partial L}{\partial \pi_k} = \frac{\partial}{\partial \pi_k} \lambda_0(1 - \pi_1 - \pi_2) + \lambda_1(c - \pi_1) + \lambda_2(c - \pi_2) + \sum_{i=1}^{m} (\tau_k^i \log \pi_1) + (\tau_k^i \log \pi_2)$$

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 in your report.

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

$V = \{$free, cheese, pizza, get, fast, now, order, available, slice, offer, not, good, pepperoni, tastes, crust$\}$.

We will use V_i to represent the ith word in V. Use the training set provided in the table below.

Spam	**Non Spam**
free cheese pizza get fast now	good pepperoni cheese pizza now available
order available pizza slice fast	pepperoni pizza tastes good
pizza offer not good	pizza crust tastes not good
	cheese slice now pizza

Table 1: Spam and Non-Spam Messages

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.

Spam	**Non Spam**
111111000000000	011001010001100
001010111000000	001000000001110
001000000111000	001000000011011
	011001001000000

Table 2: Spam and Non-Spam Vectors

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 $\theta_{0,1}$, $\theta_{0,5}$, $\theta_{1,2}$, $\theta_{1,15}$ 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.)

The constraint $\sum_{k=1}^{d} \theta_{c,k} = 1, \quad c = \{0, 1\}$ is actually two constraints. Using the bag of words example, this constraint says that "when pulling words from the spam bag, the sum of the probabilities of pulling each word in this class is 1^1." More simply, within each bag, all the word probabilities in that bag must sum to 1. The two constraints look like:

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

To maximize this function, we use the Lagrangian to incorporate the constraints directly into the function.

$$max_\theta \ell(\theta_{c,k}) = \sum_{i=1}^{m}\sum_{k=1}^{d} x_k^{(i)} \log \theta_{y^{(i)},k} - \lambda_0(1 - \sum_{k=1}^{d}\theta_{0,k}) - \lambda_1(1 - \sum_{k=1}^{d}\theta_{1,k})$$

The Lagrangian multipliers λ_c acts as punishment terms for exceeding the constraints. Since $1-\sum_{k=1}^{d}\theta_{c,k}$ is distributed to lambda, exceeding the constraints $\sum_{k=1}^{d}\theta_{c,k}$ results in a negative term in the function. If the parameters exceed the constraints, the Lagrangian multiplier reduces the objective function, and those parameters are unlikely to be the maximized parameters.

To continue, we can make this easier by separating the classes. We can split the $\sum_{i=1}^{m}$ term based on which vectors belong to each class.

$$\begin{aligned} max_\theta \ell(\theta_{c,k}) = [\sum_{i:y^i=0}\sum_{k=1}^{d} x_k^{(i)} \log \theta_{0,k} - \lambda_0(1 - \sum_{k=1}^{d}\theta_{0,k})]+ \\ [\sum_{i:y^i=1}\sum_{k=1}^{d} x_k^{(i)} \log \theta_{1,k} - \lambda_1(1 - \sum_{k=1}^{d}\theta_{1,k})] \end{aligned}$$

3. (10 points) Given a test message "cheese pizza available", using the Naive Bayes classifier that you have trained in Part (a)-(b), to 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.