ISYE 6740, Spring 2025, Homework 4

100 points

Prof. Yao Xie

Provided Data:

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

1. 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).

$$\frac{d}{d\theta} \log(1 + \exp\{-\theta^T x^i\}) = \frac{-\exp\{-\theta^T x^i\} \cdot (-x^i)}{1 + \exp\{-\theta^T x^i\}}$$

$$\frac{d}{d\theta} \left(-\log(1 + \exp\{-\theta^T x^i\}) \right) = \frac{x^i \exp\{-\theta^T x^i\}}{1 + \exp\{-\theta^T x^i\}} = x^i \left(1 - \sigma(\theta^T x^i)\right)$$

$$\frac{d}{d\theta} \left((y^i - 1)\theta^T x^i \right) = (y^i - 1)x^i$$

$$\frac{d}{d\theta}\ell(\theta) = \sum_{i=1}^{m} \left[x^i(1 - \sigma(\theta^T x^i)) + (y^i - 1)x^i \right] = \sum_{i=1}^{m} x^i \left[(1 - \sigma(\theta^T x^i)) + (y^i - 1) \right]$$

$$= \boxed{\sum_{i=1}^{m} x^i \left[y^i - \sigma(\theta^T x^i) \right]}$$

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

Algorithm 1 Gradient Descent

$\theta \leftarrow 0$
for $t = 1$ to T **do**
  $\nabla \ell(\theta) \leftarrow \sum_{i=1}^{m} x^i (y^i - \sigma(\theta^T x^i))$
  $\theta \leftarrow \theta + \alpha \nabla \ell(\theta)$
end for
return θ

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.

Algorithm 2 Stochastic Gradient Descent

$\theta \leftarrow 0$
for $t = 1$ to T **do**
  for $i = 1$ to m **do**
    $\nabla \ell_i(\theta) \leftarrow x^i (y^i - \sigma(\theta^T x^i))$
    $\theta \leftarrow \theta + \alpha \nabla \ell_i(\theta)$
  end for
end for
return θ

Gradient Descent computes the gradient using all training samples before updating θ. It provides a more stable update but can be slow for large datasets. Stochastic Gradient Descent, on the other hand, updates θ after computing the gradient for each individual sample. It is faster but has more variance, leading to noisier updates.

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.

$$H(\theta) = \nabla^2 \ell(\theta) = -\sum_{i=1}^{m} x^i (1 - \sigma(\theta^T x^i)) \sigma(\theta^T x^i) x^{iT}$$

The term $(1 - \sigma(\theta^T x^i))\sigma(\theta^T x^i)$ is always non-negative for any input x^i, meaning that $H(\theta)$ has nonpositive eigenvalues and thus is a negative semi-definite matrix. Since the Hessian is negative semi-definite, the function $\ell(\theta)$ is concave.

Because $\ell(\theta)$ is concave, the optimization problem has a unique global maximum. This ensures that gradient ascent will efficiently converge to the optimal solution without getting trapped in local optima.

2. *Fair* Gaussian mixture model (15 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).

The goal of the M-step is to maximize the expected complete-data log-likelihood:

$$Q(\pi_1, \pi_2) = \sum_{i=1}^{n} \left[\gamma_i^{(1)} \log \pi_1 + \gamma_i^{(2)} \log \pi_2 \right],$$
$$\pi_1 + \pi_2 = 1, \quad \pi_1 \geq c, \quad \pi_2 \geq c$$

$$\mathcal{L}(\pi_1, \pi_2, \lambda, \alpha_1, \alpha_2) = \sum_{i=1}^{n} \left[\gamma_i^{(1)} \log \pi_1 + \gamma_i^{(2)} \log \pi_2 \right] + \lambda(\pi_1 + \pi_2 - 1) + \alpha_1(c - \pi_1) + \alpha_2(c - \pi_2),$$

$\alpha_1, \alpha_2 \geq 0$ are the KKT multipliers for the constraints $\pi_1 \geq c$ and $\pi_2 \geq c$

$$\frac{\partial \mathcal{L}}{\partial \pi_1} = \frac{\gamma^{(1)}}{\pi_1} + \lambda - \alpha_1 = 0$$
$$\frac{\partial \mathcal{L}}{\partial \pi_2} = \frac{\gamma^{(2)}}{\pi_2} + \lambda - \alpha_2 = 0$$

$$\pi_1 + \pi_2 = 1$$

When both constraints are inactive ($\alpha_1 = 0, \alpha_2 = 0$):

$$\frac{\gamma^{(1)}}{\pi_1} = \frac{\gamma^{(2)}}{\pi_2}$$

$$\Rightarrow \pi_1 = \frac{\gamma^{(1)}}{n}, \quad \pi_2 = \frac{\gamma^{(2)}}{n}$$

If both values satisfy $\pi_1, \pi_2 \geq c$, then this is the optimal solution.

The other case is when one constraint is active:
If $\pi_1 < c$, then we set $\pi_1 = c$ and solve for π_2:

$$\pi_2 = 1 - c$$

Similarly, if $\pi_2 < c$, then we set $\pi_2 = c$ and solve for π_1:

$$\pi_1 = 1 - c$$

If both constraints are active, then the only possible solution is:

$$\pi_1 = \pi_2 = c$$

Thus, the final solution for π_1 and π_2 is:

$$\pi_1^* = \max\left(c, \frac{\gamma^{(1)}}{n}\right), \quad \pi_2^* = 1 - \pi_1^*$$

3. Bayes Classifier for sentiment analysis (35 points)

In this problem, we will use the Bayes Classifier algorithm to fit a sentiment classifier by hand. This will enhance your understanding of 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.

Sentiment classifiers are commonly used in natural language processing to determine whether a given text expresses a positive or negative sentiment. A simple approach involves maintaining a vocabulary of words that commonly occur in positive or negative reviews and classifying a review as "Positive" if the number of words from the dictionary that are present in the review is over a certain threshold.

We are given a vocabulary consisting of 22 words:

V={I, love, this, movie, such, an, amazing, experience, best, day, of, my, life, is, terrible, hate, product, worst, service, ever, a, bad}

We will use V_i to represent the ith word in V. As our training dataset, we are given the following 7 example messages:

Positive Messages	**Negative Messages**
I love this movie	This movie is terrible
Such an amazing experience	I hate this product
Best day of my life	Worst service ever
	Such a bad experience

Table 1: Example Positive and Negative Sentences

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 positive messages, and $y = 1$ corresponds to negative 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 positive and negative message.

Positive messages (y = 0):

- "I love this movie"
- "Such an amazing experience"
- "Best day of my life"

→ Total count: 3 messages

Negative messages (y = 1):

- "This movie is terrible"
- "I hate this product"
- "Worst service ever"
- "Such a bad experience"

→ Total count: 4 messages

$$P(y = 0) = \frac{\text{Number of Positive Messages}}{\text{Total Messages}} = \frac{3}{7}$$

$$P(y = 1) = \frac{\text{Number of Negative Messages}}{\text{Total Messages}} = \frac{4}{7}$$

Feature Vectors for Positive Messages (y = 0):

(a) "I love this movie"

$$[1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$$

(b) "Such an amazing experience"

$$[0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$$

(c) "Best day of my life"

$$[0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$$

Feature Vectors for Negative Messages (y = 1):

(a) "This movie is terrible"

$$[0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0]$$

(b) "I hate this product"

$$[1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0]$$

(c) "Worst service ever"

$$[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0]$$

(d) "Such a bad experience"

$$[0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1]$$

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 all the following thetas 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.

Theta	**MLE**
$\theta_{0,1}$	0.0769
$\theta_{0,6}$	0.0769
$\theta_{1,2}$	0
$\theta_{1,15}$	0.0667

Table 2: Table of Theta values and their corresponding MLEs

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

Positive Class $y = 0$:

$$sum[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0] = 13$$

Negative Class $y = 1$:

$$sum[1, 0, 2, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1] = 15$$

$$\theta_{0,k} = \frac{\text{word count in positive class}}{13}$$

$$\theta_{1,k} = \frac{\text{word count in negative class}}{15}$$

For the given words:

1. $\theta_{0,1}$ ("I" in Positive Class):

$$\theta_{0,1} = \frac{1}{13}$$

2. $\theta_{0,6}$ ("an" in Positive Class):

$$\theta_{0,6} = \frac{1}{13}$$

3. $\theta_{1,2}$ ("love" in Negative Class):

$$\theta_{1,2} = \frac{0}{15} = 0$$

4. $\theta_{1,15}$ ("terrible" in Negative Class):

$$\theta_{1,15} = \frac{1}{15}$$

3. (15 points) Given the following test messages, using the Naive Bayes classifier that you have trained in parts 1-2, calculate the posteriors and decide whether these messages are positive or negative. Derivations must be shown in your report. When reviewing your decisions, please consider how they correlate with your understanding of the message, and if there are differences mention why you think that may happen.

Test Messages	**Log P(Positive)**	**Log P(Negative)**	**Decision**
This product is a bad movie	$-\infty$	-16.1149	**Negative**
Best movie ever, amazing experience	-13.6718	$-\infty$	**Positive**
This service is amazing	$-\infty$	$-\infty$	**Inconclusive**

Table 3: Test Messages with Sentiment Probabilities and Final Decision

$$P(x|y=c) = \prod_{k=1}^{d} \theta_{c,k}^{x_k}$$

$$\Rightarrow \log P(x|y=c) = \log P(y=c) + \sum_{k=1}^{d} x_k \log \theta_{c,k}$$

Test Message 1: "This product is a bad movie"

$$[0,0,1,1,0,0,0,0,0,0,0,0,0,0,1,0,1,1,0,0,0,1,0]$$

For $y = 0$ (Positive Class):

$$\log P(y=0) + \sum x_k \log \theta_{0,k}$$

$$= \log \frac{3}{7} + \log \theta_{0,3} + \log \theta_{0,4} + \log \theta_{0,14} + \log \theta_{0,16} + \log \theta_{0,17} + \log \theta_{0,21}$$

Since many of these words have $\theta_{0,k} = 0$, we get $-\infty$, meaning $P(x|y=0) = 0$

For $y = 1$ (Negative Class):

$$\log P(y=1) + \sum x_k \log \theta_{1,k}$$

$$= \log \frac{4}{7} + \log \theta_{1,3} + \log \theta_{1,4} + \log \theta_{1,14} + \log \theta_{1,16} + \log \theta_{1,17} + \log \theta_{1,21}$$

$$= -0.5596 + (-2.0149) + (-2.7081)$$

$$+ (-2.7081) + (-2.7081) + (-2.7081) + (-2.7081) = -16.1149$$

Since $\log P(y = 1|x)$ is finite while $\log P(y = 0|x) = -\infty$, we classify it as **Negative**.

Test Message 2: "Best movie ever, amazing experience"

$$[0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0]$$

For $y = 0$ (Positive Class):

$$\begin{aligned}\log P(y = 0) + \sum x_k \log \theta_{0,k} \\ &= \log \frac{3}{7} + \log \theta_{0,4} + \log \theta_{0,7} + \log \theta_{0,8} + \log \theta_{0,9} + \log \theta_{0,20} \\ &= -0.8473 + (-2.5649) + (-2.5649) + (-2.5649) \\ &+ (-2.5649) + (-2.5649) = -13.6718\end{aligned}$$

For $y = 1$ (Negative Class): Since none of these words appear in negative class $P(x|y = 1) = 0$, so $\log P(x|y = 1) = -\infty$

Since $\log P(y = 0|x)$ is finite while $\log P(y = 1|x) = -\infty$, we classify it as **Positive**.

Test Message 3: "This service is amazing"

$$[0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0]$$

For $y = 0$ (Positive Class): Since "is" is missing from positive class, its probability is 0, leading to $P(x|y = 0) = -\infty$.

For $y = 1$ (Negative Class): Since amazing" does not appear in the negative class, its probability is 0, making $P(x|y = 1) = -\infty$ as well.

Final decision is **Inconclusive** (as there is not enough training data).

4. Comparing classifiers: Divorce classification/prediction (30 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/ml/datasets/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. (15 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.

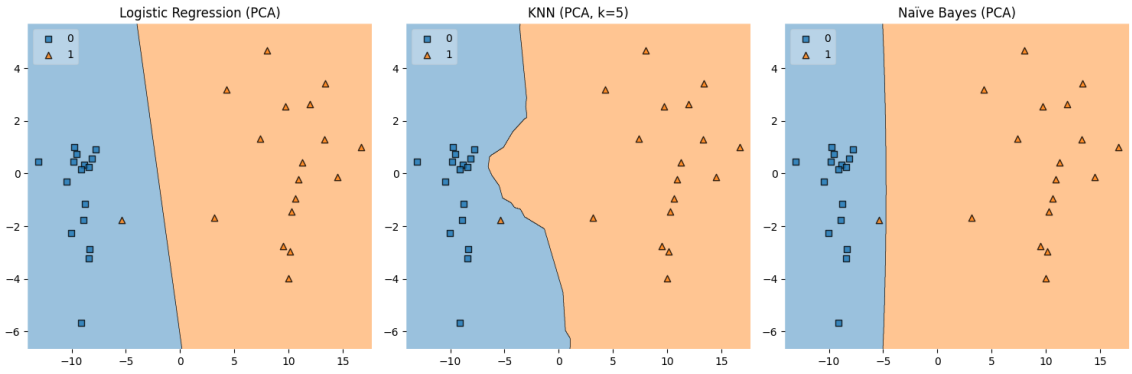
- Logistic Regression - Training Accuracy: 1.000000, Test Accuracy: 0.941176
- KNN (k=5) - Training Accuracy: 0.977941, Test Accuracy: 0.970588
- Naïve Bayes - Training Accuracy: 0.977941, Test Accuracy: 0.970588

Among the three classifiers, KNN (k=5) and Naïve Bayes achieved the highest test accuracy (97.06%), while Logistic Regression performed slightly worse (94.12%). The perfect training accuracy of Logistic Regression suggests possible overfitting, which could explain its slightly lower test accuracy. KNN and Naïve Bayes perform well, likely due to their ability to capture non-linear relationships in the dataset. KNN benefits from its flexibility in decision boundaries, while Naïve Bayes assumes feature independence, which may align well with the dataset's structure. Overall,

KNN performs best, likely because it can adapt well to variations in the data without making strong assumptions about feature distributions.

2. (15 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.

- Logistic Regression (PCA) - Training Accuracy: 0.9779, Test Accuracy: 0.9706
- KNN (PCA, k=5) - Training Accuracy: 0.9853, Test Accuracy: 0.9706
- Naïve Bayes (PCA) - Training Accuracy: 0.9779, Test Accuracy: 0.9706



Logistic Regression produces a nearly linear boundary, reflecting its assumption of a linear relationship between features and the decision function. KNN (k=5) results in a more flexible, non-linear boundary that adapts to local variations in the data, making it effective for capturing complex patterns. Naïve Bayes assumes feature independence and models a Gaussian distribution, resulting in a smooth, slightly curved boundary. Despite these differences, all classifiers achieve high test accuracy (97.06%), indicating that the two classes are well-separated in the reduced two-dimensional PCA space. The strong performance of KNN suggests that local decision-making is beneficial in this dataset, while Logistic Regression and Naïve Bayes perform well due to the natural separability of the data after PCA transformation.