# ISYE 6740, Fall 2024, Homework 4

100 points

Prof. Yao Xie

## 1. Optimization (35 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. (10 points) Show step-by-step mathematical derivation for the gradient of the cost function $\ell(\theta)$ in (1).

   **Answer:**

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

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

$$\frac{\partial}{\partial \theta} \left[ (y^i - 1)\theta x^i \right] = \left(y^i - 1\right) x^i$$

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

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

**Answer:**

**Data:** Training data $(x^i, y^i)$ for $i = 1, 2, \ldots, m$
**Result:** Optimized parameter $\theta^*$
Initialize: parameter $\theta^0$;
**do**
  $\theta^{t+1} \leftarrow \theta^t + \gamma_t \sum_{i=1}^m \left[ \frac{x^i \exp\{-\theta x^i\}}{1+\exp\{-\theta x^i\}} + (y^i - 1)\, x^i \right]$;
**while** $\|\theta^{t+1} - \theta^t\| > \epsilon$;

**Algorithm 1:** Gradient Descent

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.

**Answer:**

**Data:** Randomly sample a small subset $S_t$, $t = 1, 2, ...$
**Result:** Optimized parameter $\theta^*$
Initialize: parameter $\theta^0$;
**while** $t \leq$ *max(t) and* $\|\nabla \ell(\theta)\| \geq$ *pre-specified tolerance value* **do**
  **for** *each subset* $S_t$ **do**
    $\theta^{t+1} \leftarrow \theta^t + \gamma_t \sum_{i=1}^m \left[ \frac{x^i \exp\{-\theta x^i\}}{1+\exp\{-\theta x^i\}} + (y^i - 1)\, x^i) \right]$;
  **end**
**end**
$\theta^{t+1} = \theta^t$;

**Algorithm 2:** Stochastic Gradient Descent

Gradient Descent is computationally expensive especially working with huge dataset. In the other word, if the dataset is huge, we choose Stochastic Gradient Descent. However, the Stochastic Gradient Descent is based on small subset of the sample, it might be more nosier than Gradient Descent.

4. (15 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.

**Answer:**

From training problem (1), we have the log-likelihood function for logistic regression:

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

Gradient of the Log-likelihood: taking the first derivative):

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

Hessian Matrix: taking the second derivative:

$$\begin{aligned}
\ell''(\theta) &= \sum_{i=1}^{m} \frac{d}{d\theta} \left( \frac{x^i \exp\{-\theta x^i\}}{1 + \exp\{-\theta x^i\}} + (y^i - 1) x^i \right) \\
&= \sum_{i=1}^{m} \frac{d}{d\theta} \left( \frac{x^i \exp\{-\theta x^i\}}{1 + \exp\{-\theta x^i\}} \right) \\
&= \sum_{i=1}^{m} \frac{-(x^i)^2 \exp\{-\theta x^i\}(1 + \exp\{-\theta x^i\}) - x^i \exp\{-\theta x^i\} \left(-x^i \exp\{-\theta x^i\}\right)}{\left(1 + \exp\{-\theta x^i\}\right)^2} \\
&= \sum_{i=1}^{m} \frac{-(x^i)^2 \exp\{-\theta x^i\}}{(1 + \exp\{-\theta x^i\})^2} \\
&< 0.
\end{aligned}$$

Because the Hessian matrix is negative, the $l(\theta)$ is concave. The gradient descent will achieve a unique global optimizer,

## 2. Bayes Classifier for spam filtering (35 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

$$V = \{\textsf{free, money, no, cap, crypto, real, cash, prize, for, you, big, chance, pizza, is, vibe}\}.$$

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

- free money no cap
- crypto real money
- real cash prize money for you

and 4 example non-spam messages

- big free chance
- no cash no pizza
- money money money
- pizza is big vibe for real

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 $i$th 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.

   **Answer:**

   $$P(y = 0) = \frac{3}{7}, \quad P(y = 1) = \frac{4}{7}$$

   $V = \{$free, money, no, cap, crypto, real, cash, prize, for, you, big, chance, pizza, is, vibe$\}$.

   **spam:**

   free money no cap [1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

   crypto real money [0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]

   real cash prize money for you [0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0]

   **non-spam:**

   big free chance [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0]

   no cash no pizza [0, 0, 2, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0]

   money money money [0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

   pizza is big vibe for real [0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 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 $\theta_{0,1}$, $\theta_{0,6}$, $\theta_{1,2}$, $\theta_{1,15}$ 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.)

**Answer:**

Joint Probability:

$$P(X, y) = \prod_{i=1}^{m} P(x^{(i)}, y^{(i)})$$

Where $m$ is the number of messages.
Based on Bayes rule:

$$P(X, y) = \prod_{i=1}^{m} P(y^{(i)}) P(x^{(i)}|y^{(i)})$$

The probability of the dataset is the product of probabilities for each individual message.

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

Then:

$$P(X, y) = \prod_{i=1}^{m} P(y^{(i)}) \prod_{k=1}^{d} \theta_{c,k}^{x_k}$$

d is the number of words in V.
Take the log-likelihood:

$$\log P(X, y) = \sum_{i=1}^{m} \log P(y^{(i)}) + \sum_{i=1}^{m} \sum_{k=1}^{d} x_k^{(i)} \log \theta_{y^{(i)},k}$$

We want to maximum the $P$, however, it is constrained problem, we need to use the Lagrangian. The constrain is:

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

$$\mathcal{L}(\log P(X, y)) = \sum_{i=1}^{m} \log P(y^{(i)}) + \sum_{i=1}^{m} \sum_{k=1}^{d} x_k^{(i)} \log \theta_{y^{(i)},k} + \lambda_c \left(1 - \sum_{k=1}^{d} \theta_{0,k}\right)$$

To find the maximum, we take the derivative of $\mathcal{L}(\log P(X, y))$

$$\frac{\partial \mathcal{L}(\log P(X, y))}{\partial \theta_{c,k}} = \frac{1}{\theta_{c,k}} \sum x_k^{(i)} - \lambda_c = 0$$

$$\theta_{c,k} = \frac{\sum x_k^{(i)}}{\lambda_c}$$

Since

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

$$\sum_{k=1}^{d} \theta_{c,k} = \sum_{k=1}^{d} \frac{\sum x_k^{(i)}}{\lambda_c} = 1$$

we get

$$\lambda_c = \sum_{k=1}^{d} \sum x_k^{(i)}$$

plug in

$$\theta_{c,k} = \frac{\sum x_{c,k}^{(i)}}{\sum_{k=1}^{d} \sum x_{c,k}^{(i)}}$$

The numerator is the number of all word k in c, the denominator is the number of all words in c.
$\theta_{0,1} = \frac{1}{13}$
$\theta_{0,6} = \frac{2}{13}$
$\theta_{1,2} = \frac{3}{16}$
$\theta_{1,15} = \frac{1}{16}$

3. (15 points) Given a test message "money for real", using the Naive Bayes classier 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.

**Answer:**

Based on Bayes rule:

$$P(y \mid x) = \frac{P(x \mid y)P(y)}{P(x)} = \frac{P(x \mid y)P(y)}{\sum_y P(x \mid y)P(y)}$$

For the test message "money for real", given:

$$P(x \mid y = c) = \prod_{k=1}^{n} \theta_{c,k}^{x_k}, \quad c = 0, 1$$

$\theta_{0,2} = \frac{3}{13}$
$\theta_{0,6} = \frac{2}{13}$
$\theta_{0,9} = \frac{1}{13}$
$\theta_{1,2} = \frac{3}{16}$
$\theta_{1,6} = \frac{1}{16}$
$\theta_{1,9} = \frac{1}{16}$
spam posterior calculation:

$$P(y = 0 \mid x) = \frac{P(x \mid y = 0)P(y = 0)}{\sum_y P(x \mid y)P(y)} = \frac{P(x \mid y = 0)P(y = 0)}{\sum_y P(x \mid y)P(y)}$$

$$= \frac{\theta_{0,2}^{x_2} \cdot \theta_{0,6}^{x_6} \cdot \theta_{0,9}^{x_9} \cdot p(y = 0)}{\sum_y P(x \mid y)P(y)} = \frac{0.0011704272}{\sum_y P(x \mid y)P(y)}$$

non-spam posterior calculation:

$$P(y = 1 \mid x) = \frac{P(x \mid y = 1)P(y = 1)}{\sum_y P(x \mid y)P(y)} = \frac{P(x \mid y = 1)P(y = 1)}{\sum_y P(x \mid y)P(y)}$$

$$= \frac{\theta_{1,2}^{x_2} \cdot \theta_{1,6}^{x_6} \cdot \theta_{1,9}^{x_9} \cdot p(y = 1)}{\sum_y P(x \mid y)P(y)} = \frac{0.00041852678}{\sum_y P(x \mid y)P(y)}$$

Based on the probability, it is spam.

## 3. 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.

   **Answer:**

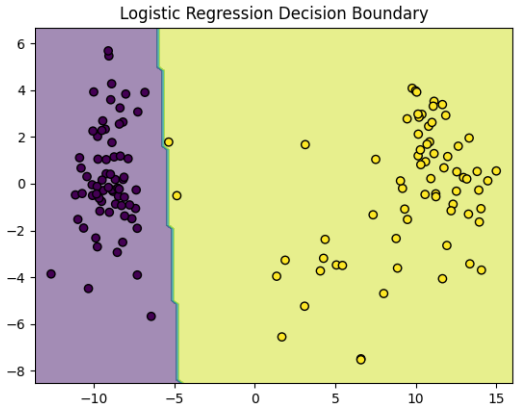
   The accuracy for three classifier:

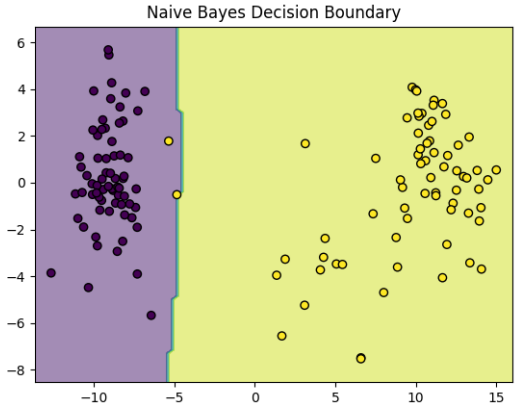
   Naive Bayes: 94.118%
   Logistic Regression: 94.118%
   KNN: 94.118%

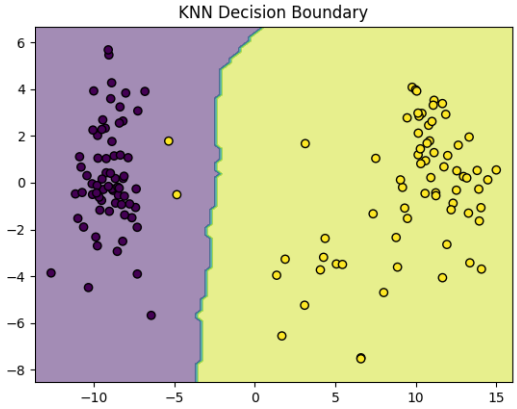
   I got the same accuracy results from the three classifiers, it maybe due to the simple dataset or a particular seed used.

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.

   **Answer:**




The boundary line of Naive Bayes is slightly curved compared to logistic regression. KNN produces a more complex boundary line than the other two classifiers. This is because KNN is a non-parametric model that based on the nearest neighbors.