Homework II

STAT/CS 5810/6655 - Spring semester 2024

Please upload your solutions in a single pdf file in Canvas. Any requested plots should be sufficiently labeled for full points. Include any code requested.

Unless otherwise stated, programming assignments should use built-in functions in your chosen programming language (Python, R, or Matlab). However, exercises are designed to emphasize the nuances of machine learning and deep learning algorithms - if a function exists that trivially solves an entire problem, please consult with the TA before using it.

Please be clear in your explanations and provide the necessary steps to reach the solution. You are not required to type the solutions, but unorganized work and difficult to read handwriting may be subject to point deductions to the TA discretion.

1. (**(10 pts)**) Read through the paper "Reconciling modern machine learning practice and the classical bias-variance trade-off" by Belkin et al. (2019). You can find it on Canvas. Write a short (1-2 paragraph) summary of the paper.

2. **Maximum Likelihood Estimation (10 pts)**
   Consider a random variable $\mathbf{X}$ (possibly a vector) whose distribution (pdf or pmf) belongs to a parametric family. The density or mass function may be written as $f(x;\boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is called the parameter, and can be either a scalar or vector. For example, in the univariate Gaussian distribution, $\boldsymbol{\theta}$ can be a two dimensional vector consisting of the mean and the variance. Suppose the parametric family is known, but the value of the parameter is unknown. It is often of interest to estimate this parameter from observations of x.

   Maximum likelihood estimation is one of the most important parameter estimation techniques. Let $x_1, \ldots, x_n$ be i.i.d. (independent and identically distributed) realizations drawn from $f(x;\boldsymbol{\theta})$. By independence, the joint distribution of the observations is the product

$$L(\boldsymbol{\theta}) = \prod_{i=1}^{n} f(x_i;\boldsymbol{\theta}).$$

Viewed as a function of $\boldsymbol{\theta}$, this quantitiy is called the likelihood of $\boldsymbol{\theta}$. It is often more convenient to work with the log-likelihood,

$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log f(x_i; \boldsymbol{\theta}).$$

A maximum likelihood estimator (MLE) of $\boldsymbol{\theta}$ is any parameter

$$\hat{\boldsymbol{\theta}} \in \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log f(x_i; \boldsymbol{\theta}).$$

If the maximizer is unique, $\hat{\boldsymbol{\theta}}$ is called *the* maximum likelihood estimator of $\boldsymbol{\theta}$.

(a) **(5810 - 10 pts, 6655 - 5 pts)** Let $X_1, \ldots, X_n$ be an i.i.d. sample from an exponential distribution with the density function

$$p(x; \beta) = \frac{1}{\beta} e^{-\frac{x}{\beta}},\ 0 \leq x < \infty.$$

Find the MLE of the parameter β. Given what you know about the role that β plays in the exponential distribution, does the MLE make sense? Why or why not?

(b) **(6655 - 5 pts)** Let $X_1, \ldots, X_n$ be i.i.d. sample from the following pdf

$$p(x; \theta) = \begin{cases} \frac{1}{\theta}, & \text{if } 0 \leq x \leq \theta \\ 0, & o.w. \end{cases}$$

Write down the likelihood and the log-likelihood functions. Find the maximum likelihood estimator (MLE) of the parameter θ. **Hint:** You won't be able to use calculus on this problem to get the MLE. Instead argue that the likelihood function is a decreasing function of θ and use that fact to find the MLE.

3. **Logistic Regression as ERM (10 pts).** Consider training data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$ for binary classification and assume $y_i \in \{-1, 1\}$. Show that if $L(y, t) = \log(1 + \exp(-yt))$,then

$$\frac{1}{n} \sum_{i=1}^{n} L(y_i, \boldsymbol{w}^T \boldsymbol{x}_i + b)$$

is proportional to the negative log-likelihood for logistic regression. Therefore ERM with the logistic loss is equivalent to the maximum likelihood approach to logistic regression.
Clarification: In the above expression, y is assumed to be -1 or 1. In the notes, we had $y \in \{0, 1\}$. So all you need to do is rewrite the negative log-likelihood for logistic regression using the ± 1 label convention and simplify that formula until it looks like the formula above. In particular, you can show (and should argue) that the posterior distribution in this case can be written as

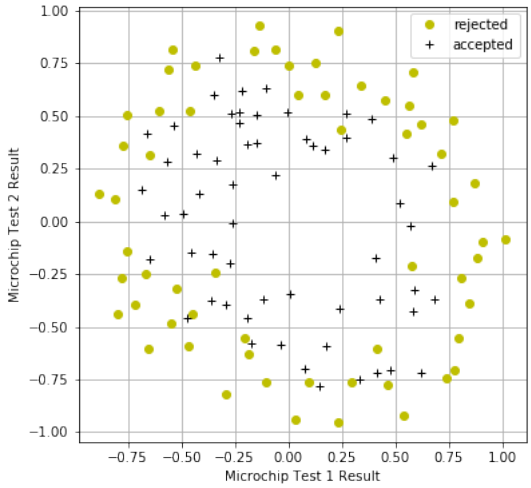
$$P(y|\boldsymbol{x}; \boldsymbol{w}, b) = \frac{1}{1 + e^{-y(\boldsymbol{w}^T \boldsymbol{x} + b)}}.$$

You can then use this representation to show the equivalence.

4. **Surrogate Losses (5 pts)**. In class, we discussed the fact that minimizing the 0-1 loss for classification is intractable, motivating the use of surrogate losses. Typically the least squares loss function is used in regression. Can we also use this loss as a surrogate loss for classification? Discuss why or why not. Can you think of any other surrogate losses for the 0-1 loss? If you can't think of any on your own, you may need to do some searching on the Internet.

5. **Gradient descent with regularized logistic regression (35 pts).** In this exercise, you will implement regularized logistic regression to predict whether microchips from a fabrication plant pass quality assurance (QA). During QA, each microchip goes through various tests to ensure it is functioning correctly. Suppose you are the product manager of the factory and you have the test results for some microchips on two different tests. From these two tests, you would like to determine whether the microchips should be accepted or rejected. To help you make the decision, you have a dataset of test results on past microchips in `ex2data2.txt`, from which you can build a logistic regression model. Data is available at Canvas – Files/Data. Stater Code is provided at Canvas – File/Stater Code/HW2.

   The figure below shows that our dataset cannot be separated into positive and negative examples by a straight line. Therefore, a straightforward application of logistic regression will not perform well on this dataset since logistic regression will only be able to find a linear decision boundary.



   One way to fit the data better is to create more features from each data point. The following map $\Phi(\mathbf{x})$ maps the features into all polynomial terms of x_1 and x_2 up to the sixth power as

$$\tilde{\mathbf{x}} = \Phi(\mathbf{x}) = \begin{bmatrix} 1 & x_1 & x_2 & x_1^2 & x_1x_2 & x_2^2 & x_1^3 & \dots & x_1x_2^5 & x_2^6 \end{bmatrix}^T.$$

   A logistic regression classifier trained on this higher-dimension feature vector will have a more complex decision boundary and will appear nonlinear when drawn in our 2-dimensional plot. While the feature mapping allows us to build a more expressive classifier, it is also more susceptible to overfitting. In this exercise, you will implement regularized logistic regression to fit the data and also see for yourself how regularization can help combat the overfitting problem.

Define $\tilde{\mathbf{x}}_i$ as the non-linear transformation for i-the sample. Recall that logistic regression is a linear classifier. That is, it has the form $f(\mathbf{x}) = \tilde{\mathbf{w}}^T\tilde{\mathbf{x}}$ where $\tilde{\mathbf{w}} = [w_0, \mathbf{w}^T]^T$ where w_0 corresponds to the offset and $\mathbf{w}$ is the weight vector. The classifier assigns a data point $\mathbf{x}$ to class 1 if $f(\mathbf{x}) \geq 0$ and to class 0 otherwise. Here y_i refers to the labels for the i-th sample, $y_i = 1$ means rejected, $y_i = 0$ means accepted. The loss function for regularized logistic regression can be written as

$$\ell(\mathbf{w}, w_0) = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log\left(\frac{1}{1+\exp(-\tilde{\mathbf{w}}^T\tilde{\mathbf{x}}_i)}\right) + (1-y_i)\log\left(\frac{\exp(-\tilde{\mathbf{w}}^T\tilde{\mathbf{x}}_i)}{1+\exp(-\tilde{\mathbf{w}}^T\tilde{\mathbf{x}}_i)}\right)\right] + \lambda||\mathbf{w}||^2,$$

where λ is the regularization parameter that is set by the user.

(a) (**10 pts**) Derive the gradient of this loss function with respect to the weights and offset.

(b) (**2 pts**) Load the data into the software you prefer (Python, R, Matlab, ...), transform the features into $\tilde{\mathbf{x}}$. Randomly shuffle the data and then split them into training and testing sets. Report how you split the data.

(c) (**5810 - 20 pts**) Apply regularized logistic regression to the data using built-in methods in the language of your choice. Try setting $\lambda = 0.1$. Report, if possible, the value of the objective function (i.e. the loss function) at the optimum and the test error.

(d) (**6655 - 20 pts**) Implement logistic regression from scratch, i.e., update $\tilde{\mathbf{w}}$ with gradient descent using the expression you derived in part (a). Try setting $\lambda = 0.1$. Vary your learning rate until you get good convergence results. Report the test error, learning rate, how you initialized the weights and offset, and the value of the objective function (i.e. the loss function) at the optimum. Describe your termination criterion. The framework of the program has been provided in the Starter code, you can find it on Canvas – File/Stater Code/HW2. Since the Starter Code is written in Python, you do not need to use it, especially if you choose a different language. However, it may be helpful to look at the starter code for a general framework.

(e) (**3 pts**) Try setting $\lambda = 0, 1, 5, 10$. Use the same training and testing set from part (b). Report the test errors. Briefly comment on your results. I.e., for which values of λ does it appear to overfit or underfit the data?

6. **LDA and QDA (10 pts)**. Assume that you believe your data follows a parametric model given by the class-conditional densities given in (1) and you want to classify between two classes $k \in \{0, 1\}$, with class prior probabilities π_k. Your data has p features implying that $x, u_k \in \mathrm{R}^p$ with class-conditional densities:

$$P(x|C_k) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}}\exp\left[-\frac{1}{2}(x-u_k)^T\Sigma_k^{-1}(x-u_k)\right]. \tag{1}$$

(a) Assuming $\Sigma_0 = \Sigma_1$ find the Bayes classifier that classifies the data. **Hint:** To do this you need to estimate the parameters u_k and Σ_k. To do this, use maximum likelihood estimation (MLE). Then plug in the estimated parameters into the formula for the Bayes classifier.

(b) (**6655**) If you relax the previous assumption and assume that $\Sigma_0 \neq \Sigma_1$, find the new Bayes classifier. This gives the Quadratic Discriminant Analysis (QDA) classifier. Why does it have that name?

7. **Spam detection (20 pts)** In this problem, you will apply several classifiers to the problem of spam detection using a benchmark database assembled by researchers at Hewlett-Packard. Download the file `spambase.data` from Canvas in the "Files/Data" section and issue the following commands to load the data. In Matlab:

```
z = dlmread('spambase.data',',');
rng(0); % initialize the random number generator
rp = randperm(size(z,1)); % random permutation of the indices
z = z(rp,:); % shuffle the rows of the data matrix
x = z(:,1:end-1);
y = z(:,end);
```

In Python:

```
import numpy as np
z = np.genfromtxt('spambase.data',dtype=float, delimiter=',')
np.random.seed(0) #Seed the random number generator
rp = np.random.permutation(z.shape[0]) #random permutation of the indices
z = z[rp,:]  #shuffle the rows of the data matrix
x = z[:,:-1]
y=z[:,-1]
```

In R:

```
#Code created for UNIX/Mac Environment, R Version = 3.5
set.seed(2^16-1) #Set Seed

#Assuming you have downloaded the spambase data to your Downloads folder.  Change as necessary
setwd("~/Downloads")

#Read the spambase data.  The following 2 lines will create a R dataframe with 58 columns.
#The Outcome Variable is named Y
#All other feature variables are named V1 through V58
spam_base <- read.csv("spambase.data", header = F)

#Randomly Shuffle the Dataframe on Rows
spam_base <- spam_base[sample(nrow(spam_base)),]

#Prepare feature set x as a dataframe and outcome vector y separately
x = spam_base[,-58] #Excluding the last column which is y
y = spam_base[,58] #Selecting only the last column y
```

Note: If you copy and paste the above, you may need to delete and retype the single quote to avoid

an error. This has to do with the optical character recognition of pdfs on some systems.

Here x is $n \times d$, where $n = 4601$ and $d = 57$. The different features correspond to different properties of an email, such as the frequency with which certain characters appear. y is a vector of labels indicating spam or not spam. For a detailed description of the dataset, visit the UCI Machine Learning Repository, or Google 'spambase'.

Apply LDA, QDA, logistic regression, and Naive Bayes to the dataset. Use cross-validation to calculate your test error. Report the test error for each case using a confusion matrix. You may use any built-in methods for this. As a sanity check, what would be the test error if you always predicted the same class, namely, the majority class from the data?