1 Linear Regression

1.1 Regression with heterogenous noise

In the standard linear regression (discussed in the lectures), we consider the model that the observed response variable y is the prediction perturbed by noise, namely

$$y = \mathbf{x}^\top \boldsymbol{\beta} + \varepsilon$$

where ε is a Gaussian random variable with mean 0, with variance of σ^2. More importantly, we have further assumed that for different observations (in the training data), the corresponding noises are identically and independently distributed. In other words, for the n-th observation $\mathbf{x}_n$, the observed response is

$$y_n = \mathbf{x}_n^\top \boldsymbol{\beta} + \varepsilon_n$$

where $\varepsilon_n \sim \mathcal{N}(0, \sigma^2)$.

This assumption is not applicable in some cases. For example, in the example of predicting the sale prices of houses, it is known that the variances for bigger houses (i.e., houses with larger $\mathbf{x}_n$ which is the square footage) tend to be bigger — one indication is that, the sale prices for bigger houses seem to have a much bigger spread or range.

In this case, we will model the data in the following way

$$y_n = \mathbf{x}_n^\top \boldsymbol{\beta} + \varepsilon_n$$

where ε_n are independently distributed but **do not have to be identically distributed**. In particular, each one could have a different variance, namely, $\varepsilon_n \sim \mathcal{N}(0, \sigma_n^2)$.

(a) Suppose our training dataset contains $\{(\mathbf{x}_n, y_n), n = 1, 2, \ldots, N\}$ such observations. Please write down the log-likelihood function of the data. This function should be a function of the data as well as $\boldsymbol{\beta}$ and all σ_n. (5 points)

(b) Derive the maximum likelihood estimate of $\boldsymbol{\beta}$, and express it in terms of the data as well as all the σ_n. You should assume σ_n is known to you — you do not need to estimate them from the data. (5 points)

1.2 Smooth Coefficients

Consider a dataset with n data points $(\mathbf{x}_i, y_i)$, $\mathbf{x}_i \in \mathbb{R}^{p\times 1}$, drawn from the following linear model:

$$y = \mathbf{x}^\top \boldsymbol{\beta} + \varepsilon,$$

where ε is a Gaussian noise. Suppose the features $x_{i1}, \ldots, x_{ip}$ for all $i = 1, \ldots, n$ have a natural ordering. Several examples have this ordering property; for example in the study of the impact of proteins on certain types of cancer, the proteins are ordered sequentially on a line. Intuitively, we can encode the natural ordering information by introducing a condition that requires the difference $(\beta_i - \beta_{i+1})^2$ cannot be large, for $i = 1, \ldots, p-1$.

(a) State the condition as a regularizer. Write the new optimization problem for finding $\boldsymbol{\beta}$ by combining both this regularization and L_2 regularization. (10 points)

(b) Find the optimal $\boldsymbol{\beta}$ by solving the problem in part (a). (5 points)

1.3 Linearly Constrained Linear Regression

Consider a dataset with n data points $(\mathbf{x}_i, y_i)$, $\mathbf{x}_i \in \mathbb{R}^{p\times 1}$, drawn from the following linear model:

$$y = \mathbf{x}^\top \boldsymbol{\beta} + \varepsilon,$$

where ε is Gaussian noise. Suppose we have another information about $\boldsymbol{\beta}$ that requires $A\boldsymbol{\beta} = \mathbf{b}$ where $A \in \mathbb{R}^{q\times p}$ and $\mathbf{b} \in \mathbb{R}^{q\times 1}$. Suppose the constraint $A\boldsymbol{\beta} = \mathbf{b}$ has a non-empty set of solutions; thus the optimization has feasible solutions. Find the maximum likelihood estimation of $\boldsymbol{\beta}$ under this constraint. (10 points)

2 Online Learning

The perceptron algorithm often makes harsh updates — it strongly biased towards the current mistakenly-labeled sample. To remedy this, suppose at ith step, the classifier is $\mathbf{w}_i$ and we want to update a little bit conservatively the classifier based on observation of $(\mathbf{x}_i, y_i)$ to the new one $\mathbf{w}_{i+1}$. Derive a new update method for the perceptron such that it makes the smallest difference from the previous model, that is, minimizes $\|\mathbf{w}_{i+1} - \mathbf{w}_i\|_2$ while the new $\mathbf{w}_{i+1}$ classifies correctly on the current sample. You need to provide the closed form analytical equation for the update rule. (10 points)

3 Kernels

The Mercer theorem states that, a bivariate function $k(\cdot,\cdot)$ is a positive definite kernel function, if and only if, for any N and any $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N$, the matrix K, where $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, is positive semidefinite. That is, all the eigenvalues of the matrix are non-negative. An alternative (but equivalent) definition states that, for every positive semi-definite matrix $A \in \mathbb{R}^{n\times n}$ and arbitrary vector $\mathbf{x} \in \mathbb{R}^{n\times 1}$, we have $\mathbf{x}^\top A\mathbf{x} \geq 0$.

Suppose $k_1(\cdot,\cdot)$ and $k_2(\cdot,\cdot)$ are kernel functions, define the following kernel matrices:

(a) $K_3 = a_1K_1 + a_2K_2$, for $a_1, a_2 \geq 0$.

(b) K_4 defined by $k_4(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})f(\mathbf{x}')$ where $f(\cdot)$ is a real valued function.

(c) K_5 defined by $k_5(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}')$

Now use the Mercer theorem to show that K_3, K_4, and K_5 are positive semidefinite matrices. (15 points)

4 Bias-Variance Trade-off **(Bonus problem)**

Consider a dataset with n data points $(\mathbf{x}_i, y_i)$, $\mathbf{x}_i \in \mathbb{R}^{p\times 1}$, drawn from the following linear model:

$$y = \mathbf{x}^\top \boldsymbol{\beta}^\star + \varepsilon,$$

where ε is a Gaussian noise and the star sign is used to differentiate the true parameter from the estimators that will be introduced later. Consider the L_2 regularized linear regression as follows:

$$\widehat{\boldsymbol{\beta}}_\lambda = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{n} \sum_{i=1}^{n} (y_i - \mathbf{x}_i^\top \boldsymbol{\beta})^2 + \lambda \|\boldsymbol{\beta}\|_2^2 \right\},$$

where $\lambda \geq 0$ is the regularization parameter. Let $X \in \mathbb{R}^{n \times p}$ denote the matrix obtained by stacking $\mathbf{x}_i^\top$ in each row.

(a) Find the closed form solution for $\widehat{\boldsymbol{\beta}}_\lambda$ and its distribution. (3 points)

(b) Calculate the bias term $\mathbb{E}[\mathbf{x}^\top \widehat{\boldsymbol{\beta}}_\lambda] - \mathbf{x}^\top \boldsymbol{\beta}^\star$ as a function of λ and some feature vector $\mathbf{x}$. (5 points)

(c) Calculate the variance term $\mathbb{E}\left[\left(\mathbf{x}^\top \widehat{\boldsymbol{\beta}}_\lambda - \mathbb{E}[\mathbf{x}^\top \widehat{\boldsymbol{\beta}}_\lambda]\right)^2\right]$ as a function of λ and some feature vector $\mathbf{x}$. (5 points)

(d) Use the results from parts (b) and (c) and the bias–variance theorem to analyze the impact of λ in the squared error. (i.e. which term dominates when λ is small or large?) (2 points)

5 Programming

In this problem, you will implement (unregularized/regularized) logistic regression for binary classification problems using two different types of optimization approaches, namely, **batch gradient descent** and **Newton's method**. Two data sets are given, one of which is text data and you will learn to construct features from this type of data. *For each of the problems, you need to report your results on both of the datasets*. Please note that there are 9 problems in total in this section. Also, *cross validation is NOT needed* for this programming assignment. **For any plot you make, you MUST at least provide title, x-label, y-label and legend (if there are more than one curves)**. Please read submission instructions carefully before submitting your report and source codes.

5.1 Data

Ionosphere This dataset contains total 351 instances and each instance has 34 attributes (features). All feature values are continuous and the last column is the class label("bad" = b = 1, "good" = g = 0). Your goal is to predict the correct label, which is either "b" or "g". We already divided the dataset into training and test sets. (`iono_train.dat` and `iono_test.dat`). Please use the training set to build your model and use the test set to evaluate your model. For more details about Ionosphere dataset, please refer to UCI website. https://archive.ics.uci.edu/ml/datasets/Ionosphere.

EmailSpam This dataset contains total 941 instances and each of them is labeled as either a spam or a ham(not spam) email. Each data instance is the text which is the content of the email (subject and body), and your goal is to classify each email as either a spam or a ham. Note: the directory contains two folders, one for training set and the other for test set. And the spam and ham emails are already divided into separate folders. i.e for building your model, you need to iterate through all the text files within `/train/spam` and `/train/ham`.

5.2 Feature Representation

An essential part of machine learning is to build the feature representation for raw input data, which is often unstructured. Once we construct the features, each data instance $\mathbf{x}_i$ can be represented as:

$$\mathbf{x}_i = (x_{i1}, ..., x_{id})$$

where x_{ij} denotes the jth feature for ith instance.

Ionosphere The dataset is well-formatted. Please directly use those values to build your model. Please do not do any normalization. Also, since there is no categorical attribute, converting to binary features (which you did for Homework#1) is not needed.

EmailSpam Since each data instance is text, we need to find a way converting it into a feature vector. In this homework, we will use the "Bag-of-Words" representation. More specifically, we will convert the text into feature vector in which each entry of the vector is the count of words occur in that text. You will be provided with the predefined dictionary (`dic.dat`), and you only

Algorithm 1 Pseudo code for generating bag-of-word features from text

```
Initialize feature vector bg_feature = [0,0,...,0]
for token in text.tokenize() do
    if token in dic //if token is in the dictionary then
        token_idx = getIndex(dic, token)
        bg_feature[token_idx]++
    else
        continue
    end if
end for
return bg_feature
```

need to care about the words that appear in that dictionary and ignore all others. (However, in practice, you need to define the dictionary based on the specific requirements of applications. This process usually comes with "removing stop words", "stemming", etc. In this homework, you don't need to consider these issues). Below is the pseudo code for generating bag-of-word features from text. For tokenization, please tokenize the string only using **whitespace and these three delimiters- '.,?'**. (http://en.wikipedia.org/wiki/Bag-of-words_model the link also provides simple example)
One example email:
hey, i have a better offer for you, offer. better than all other spam filters. Do you like accepting offer?

And given the below dictionary
[add, better, email, filters, hey, offer,like, spam,special]

we can get the feature vector as:
$[0, 2, 0, 1, 1, 3, 1, 1, 0]$

(1) (2 points). After converting all training data into feature vectors, what are the top 3 words that occur most frequently? Report the results using this format:
{(word1: # of occurrence), (word2 : # of occurrence), (word3: # of occurrence)}

5.3 Implementation

In the class, we talked about the regularized logistic regression. The regularized cross-entropy function can be written as:

$$\varepsilon(\mathbf{w}, b) = -\sum_{i=1}^{n} \{y_i \log \sigma(b + \mathbf{w}^\top \mathbf{x}_i) + (1 - y_n) \log[1 - \sigma(b + \mathbf{w}^\top \mathbf{x}_i)\} + \lambda \|\mathbf{w}\|_2^2$$

where $\mathbf{x}_i \in \mathbb{R}^d$, $\mathbf{w} \in \mathbb{R}^d$, and $\sigma(\cdot)$ is the sigmoid function. λ is regularization coefficient and b is bias parameter. Please note that we don't regularize bias term b

Stopping Criteria. For both algorithms, please run for 50 iterations.

Step size. For gradient method, we will use the fixed step size. (For Newton's method, we don't need step size.)

Initialization. For batch gradient descent, please initialize the weight $\mathbf{w}$ to 0, and b to 0.1. For Newton's method, set initial weights to the ones we got from batch gradient descent after 5 iterations (when $\lambda = 0.05, \eta = 0.01$). (Please note that Newton's method may diverge if initialization is not proper).

Extreme Condition. It is possible that when $\sigma(b + \mathbf{w}^T\mathbf{x})$ approaches to 0, $\log(\sigma(b + \mathbf{w}^T\mathbf{x}))$ goes to -infinity. In order to prevent such case, please bound the value $\sigma(b + \mathbf{w}^T\mathbf{x})$ using small constant value, $1e-16$. You can use the following logic in your code: (For other similar cases, use the same strategy)
$tmp = \sigma(b + \mathbf{w}^T\mathbf{x})$
if $tmp < 1e-16$ then
$tmp = 1e-16$

Batch Gradient Descent

(2) (3 points) Please write down the updating equation for $\mathbf{w}$ and b, for both unregularized logistic regression and regularized logistic regression. That is, how to update weights in time t using data points $X = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]$, where $\mathbf{x}_i = [x_{i1}, ..., x_{id}]$, and $y_i \in \{0, 1\}$ is the label. i.e how to update $\mathbf{w}^{t+1}$ from $\mathbf{w}^t$.

(3) (6 points) For step sizes $\eta = \{0.001, 0.01, 0.05, 0.1, 0.5\}$ and **without regularization**, implement Batch gradient descent (without cross-validation, use the whole training data for the gradient calculation).

(a) Plot cross-entropy function value with respect to the number of steps T ($T = [1, ..., 50]$) for training data, using different step sizes. (you need to make two plots, one for each dataset).
(b) Report the L_2 norm of vector $\mathbf{w}$ after 50 iterations for each step size η_i (fill in the table below)

L_2 norm (without regularization)	0.001	0.01	0.05	0.1	0.5
Ionosphere					
EmailSpam					

(4) (6 points) For step sizes $\eta = \{0.001, 0.01, 0.05, 0.1, 0.5\}$ and for regularization coefficient $\lambda = \{0, 0.05, 0.1, 0.15, ..., 0.5\}$, where λ ranging from 0-0.5 spaced by 0.05.

(a) Given $\lambda = 0.1$, plot cross-entropy function value as the number of steps T (T = [1,...,50]) for training data, using different step sizes. (you need to make two plots, one for each dataset).
(b) Please also report the L_2 norm of vector $\mathbf{w}$ after 50 iterations, for each regularization coefficient λ_i (only report when $\eta = 0.01$)
(c) Plot cross entropy function value as different regularization coefficients, for both training and test data. (x-axis will be the regularization coefficient and y-axis will be the cross

entropy function value after 50 iterations. Each plot should contain two curves, and you should make 2 (two data sets) $* \times 5$ (five different step sizes) = 10 plots).

L_2 norm (with regularization, $\eta = 0.01$)	0	0.05	0.1	0.15	0.2	0.25	0.3	0.35	0.4	0.45	0.5
Ionosphere											
EmailSpam											

Newton's method Optimization also can be done using the 2nd order technique discussed in the class, called "Newton's method". We define $\mathbb{H}$ as the Hessian matrix and $\nabla\varepsilon_t$ denotes the gradient of objective function at iteration t.

(5) (4 points) Use the notations above, write down the updating equation for $\mathbf{w}$ and b in time t, for both unregularized logistic regression and regularized logistic regression, as before.

(6) (6 points) Run the Newton's method for 50 iterations on both of datasets without regularization(you don't need to set the step size). Note: It is possible to get singular hessian. Don't try to add identity matrix to make it non-singular. Instead, try using psudo inverse. If that works, you can continue with your plots. If does not work, then provide some reasons why you get singular hessian.

(a) Plot cross-entropy function value as the number of steps T = [1,...,50] (make two plots)
(b) Report L_2 norm of vector $\mathbf{w}$ after 50 iterations
(c) Report cross-entropy function value for the test data.

(7) (6 points) Repeat the problem (6) with regularized case.(using $\lambda = \{0, 0.05, 0.1, 0.15, ..., 0.5\}$)

(8) (3 points) Briefly explain the results you got from (3),(4), using no more than 4 sentences. You should discuss about the rate of convergence, change of magnitude and any other interesting facts you have observed.

Comparison between Gradient Descent and Newton's method

(9) (3 points) Based on the results you got from (4), (7) as well as your experiments, discuss the difference between gradient descent and Newton's method in terms of convergence and computation time. Please use no more than 4 sentences.

Submission Instruction You need to provide the followings:

- Provide your answers for all of the problems **in hard copy** (if you are printing it as black-white, please make sure that you are using different line styles and colors for each curve in your plot, if there are more than one curves. The papers need to be stapled and submitted to the CS front desk. We suggest printing it as double-sided to save papers.
- Submit ALL the code and report via Blackboard. The only acceptable language is MATLAB.

– For your program, you MUST include the main function called `CSCI567_hw2.m` in the root of your folder. After running this main file, your program should be able to generate all of the results needed for this programming assignment, either as plots or console outputs. You can have multiple files (i.e your sub-functions), however, the only requirement is that once we unzip your folder and execute your main file, your program should execute correctly. Please double-check your program before submitting. You should only submit one `.zip` file. No other formats are allowed except `.zip` file. Also, please name it as `[lastname]_[firstname]_hw2.zip`

Collaboration You may collaborate. However, collaboration has to be limited to discussion only and you need to write your own solution and submit separately. You also need to list with whom you have discussed.