Homework 2, Machine Learning, Fall 2022

***IMPORTANT* Homework Submission Instructions**

1. All homeworks must be submitted in one PDF file to Gradescope.
2. Please make sure to select the corresponding HW pages on Gradescope for each question.
3. For all theory problems, please type the answer and proof using Latex or Markdown, and export the tex or markdown into a PDF file.
4. For all coding components, complete the solutions with a Jupyter notebook/Google Colab, and export the notebook (including both code and outputs) into a PDF file. Concatenate the theory solutions PDF file with the coding solutions PDF file into one PDF file which you will submit.
5. Failure to adhere to the above submission format may result in penalties.

All homework assignments must be your independent work product, no collaboration is allowed. You may check your assignment with others or the internet after you have done it, but you must actually do it by yourself. **Please copy the following statements at the top of your assignment file**:

Agreement 1) This assignment represents my own work. I did not work on this assignment with others. All coding was done by myself.

Agreement 2) I understand that if I struggle with this assignment that I will reevaluate whether this is the correct class for me to take. I understand that the homework only gets harder.

1 Convexity (Chudi)

Convexity is very important for optimization. In this question, we will explore the convexity of some functions.

1.1

Let $h(x) = f(x) + g(x)$ where $f(x)$ and $g(x)$ are both convex functions. Show that $h(x)$ is also convex.

1.2

In mathematics, a concave function is the negative of a convex function. Let $g_1(x), g_2(x), ..., g_k(x)$ be concave functions and $g_i(x) \geq 0$. Show that $\log(\prod_{i=1}^{k} g_i(x))$ is concave. (Note the base of log function is e.)
Hint 1: the sum of two concave function is concave.
Hint 2: $\log(a \times b) = \log(a) + \log(b)$.
Hint 3: log function is concave and monotonically increasing in $\mathbb{R}^+$.

1.3

A line, represented by $y = ax + b$, is both convex and concave. Draw four lines with different values of a and b and highlight their maximum and minimum in two different colors. What does this reveal about the convexity of the maximum and minimum of these lines respectively?

1.4

Now we consider the general case. Let $f(x)$ and $g(x)$ be two convex functions. Show that $h(x) = \max(f(x), g(x))$ is also convex or give a counterexample.

1.5

Let $f(x)$ and $g(x)$ be two convex functions. Show that $h(x) = \min(f(x), g(x))$ is also convex or give a counterexample.

1.6

Since convex optimization is much more desirable then non-convex optimization, many commonly used risk functions in machine learning are convex. In the following questions, we will explore some of these empirical risk functions.
Suppose we have a dataset $\{\boldsymbol{X}, \boldsymbol{y}\} = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$ that we will perform classification on, where $\boldsymbol{x}_i \in \mathbb{R}^p$ is a p-dimensional feature vector, and $y_i \in \{-1, 1\}$ is the binary label. Please identify and show by proof whether the following empirical risk functions are convex with respect to weight parameter vector $\boldsymbol{\omega}$. In the model below, $\boldsymbol{\omega} = [\omega_1, \omega_2, ..., \omega_p]^T \in \mathbb{R}^p$, $b \in \mathbb{R}$ are the weight and bias. We would like to show convexity with respect to $\boldsymbol{\omega}$ and b.

(a)

$$L(\boldsymbol{\omega}, b) = \sum_{i=1}^{n} \log(1 + e^{-y_i(\boldsymbol{\omega}^T \boldsymbol{x}_i + b)})$$

(b)

$$L(\boldsymbol{\omega}, b, C) = \frac{1}{2}||\boldsymbol{\omega}||_2^2 + C \sum_{i=1}^{n} \max(0, 1 - y_i(\boldsymbol{\omega}^T \boldsymbol{x}_i + b)), \quad C \geq 0$$

1.7

Consider the lasso regression problem. Given a dataset $\{\boldsymbol{X}, \boldsymbol{y}\} = \{\boldsymbol{x}_i, y_i\}_{i=1}^n$ where $y_i \in \mathbb{R}$, please identify and show by proof whether the following empirical risk function is convex with respect to the weight parameter vector $\boldsymbol{\omega}$.

$$L(\boldsymbol{\omega}, b, C) = \|\boldsymbol{y} - (X\boldsymbol{\omega} + b\mathbf{1}_n)\|_2^2 + C\|\boldsymbol{\omega}\|_1, \quad C \geq 0$$

2 Gradient Descent and Newton's Method (Chudi)

Optimization algorithms are important in machine learning. In this problem, we will explore ideas behind gradient descent and Newton's method.
Suppose we want to *minimize* a convex function given by

$$f(\boldsymbol{x}) = 2x_1^2 + 3x_2^2 - 4x_1 - 6x_2 + 2x_1x_2 + 13.$$

2.1

(a) Find the expression for $\nabla f(\boldsymbol{x})$, the first derivative of $f(\boldsymbol{x})$.

(b) Find the expression for $Hf(\boldsymbol{x})$, the Hessian of $f(\boldsymbol{x})$.

(c) Compute *analytically* the value of $\boldsymbol{x}^*$, at which the optimum is achieved for $f(\boldsymbol{x})$. Hint: use the first two parts.

2.2

Gradient descent is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. Since $f(\boldsymbol{x})$ is strictly convex and differentiable, gradient descent can be used to find the unique optimal solution.

Recall that gradient descent takes repeated steps in the opposite direction of the gradient of the function at the current point until convergence. The algorithm is shown below. Given an initial point $\boldsymbol{x}_0$ and a step size α,

- $t = 0$
- while $\boldsymbol{x}_t$ is not a minimum
  - $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \alpha\nabla f(\boldsymbol{x}_t)$
  - $t = t + 1$
- end

For each iteration, $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \alpha\nabla f(\boldsymbol{x}_t)$. Suppose $\alpha = 0.1$. Show that $\boldsymbol{x}_{t+1}$ converges to $\boldsymbol{x}^*$ obtained in 2.1 as $t \to \infty$.

Hint 1: Let A be a square matrix. $I + A + A^2 + \ldots A^k = (I - A)^{-1}(I - A^{k+1})$.

Hint 2: Let A be a square matrix. A^k converges to zero matrix if all eigenvalues of matrix A have absolute value smaller than 1.

2.3

Will $\boldsymbol{x}_{t+1}$ converge to $\boldsymbol{x}^*$ when α grows larger? Hint: the previous question's calculations should provide you the answer.

2.4

Gradient descent only uses the first derivative to make updates. If a function is twice differentiable, is the second derivative able to help find optimal solutions? Newton's method takes the second derivative into consideration and the key update step is modified as $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - (Hf(\boldsymbol{x_t}))^{-1}\nabla f(\boldsymbol{x}_t)$. The algorithm is shown below.

- $t = 0$
- while $\boldsymbol{x}_t$ is not a minimum
  - $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - (Hf(\boldsymbol{x_t}))^{-1}\nabla f(\boldsymbol{x}_t)$
  - $t = t + 1$
- end

(a) Using the initial point $\boldsymbol{x}_0 = [1, 2]^T$, give the value of $\boldsymbol{x}_1$ that Newton's method would find in one iteration. Does this method converge in one iteration? If not, give the value of $\boldsymbol{x}_2$ that Newton's method would find and report whether $\boldsymbol{x}_2$ is equal to $\boldsymbol{x}^*$.

(b) Using the initial point $\boldsymbol{x}_0 = [1, 2]^T$ and a fixed step size $\alpha = 0.1$, give the value of $\boldsymbol{x}_1$ that gradient descent would find in one iteration. Does this method converge in one iteration? If not, give the value of $\boldsymbol{x}_2$ that gradient descent would find and report whether $\boldsymbol{x}_2$ is equal to $\boldsymbol{x}^*$.

(c) According to the results above, which method converges faster? Please briefly explain why it happens.

3 Bias and Variance of Random Forest

In this question, we will analyze the bias and variance of the random forest algorithm. **Bias** is the amount that a model's prediction differs from the target value in the training data. An underfitted model can lead to high bias. **Variance** indicates how much the estimate of the target function will alter if different training data were used. An overfitted model can lead to high variance.

3.1

Let $\{X, y\} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ be the training dataset, where $\boldsymbol{x}_i$ is a feature vector and $y_i \in \{-1, 1\}$ is the binary label. Suppose all overlapping samples are consistently labeled, i.e. $\forall i \neq j$, $y_i = y_j$ if $\boldsymbol{x}_i = \boldsymbol{x}_j$. Show that 100% training accuracy tree can be achieved if there is no depth limit on the trees in the forest. Hint: the proof is short. One way to proof it is by induction.

3.2

Let $D_1, D_2, ..., D_T$ be random variables that are identically distributed with positive pairwise correlation ρ and $Var(D_i) = \sigma^2, \forall i \in \{1, ..., T\}$. Let $\bar{D}$ be the average of $D_1, ..., D_T$, i.e. $\bar{D} = \frac{1}{T}(\sum_{i=1}^T D_i)$. Show that

$$Var(\bar{D}) = \frac{1-\rho}{T}\sigma^2 + \rho\sigma^2. \tag{1}$$

3.3

In the random forest algorithm, we can view the decision tree grown in each iteration as approximately D_i in Problem 3.2. Given this approximation, briefly explain how we expect the variance of the average of the decision trees to change as T increases.

3.4

As T becomes large, the first term in Eq(1) goes away. But the second term does not. What parameter(s) in random forest control ρ in that second term?

4 Coding for SVM (Henry)

4.1

Generate a training set of size 100 (50 samples for each label) with 2D features (X) drawn at random as follows:

- $X_{neg} \sim \mathcal{N}$([-5, -5], 5·I_2) and correspond to negative labels (-1)
- $X_{pos} \sim \mathcal{N}$([5, 5], 5·I_2) and correspond to positive labels (+1)

Accordingly, $X = \begin{bmatrix} X_{pos} \\ X_{neg} \end{bmatrix}$ is a 100×2 array, Y is a 100×1 array of values $\in \{-1, 1\}$. Draw a scatter plot of the full training dataset with the points colored according to their labels.

4.2

Train a linear support vector machine on the data with $C = 1$ and draw the decision boundary line that separates positives and negatives. Mark the support vectors separately (e.g., circle around the point).

4.3

Draw decision boundaries for 9 different C values across a wide range of values between 0 and infinity. Plot the number of support vectors vs. C (plot on a log scale if that's useful). How does the number of support vectors change as C increases and why does it change like that?

4.4

Now, try to rescale the data to the [0,1] range and repeat the steps of the previous question (4.3) and over the same range of C values. Are the decision boundaries different from those in the previous question? What does this imply about (a) the geometric margin and (b) the relative effect of each feature on the predictions of the trained model?

4.5

Try using boosted decision trees (using any boosted decision tree algorithm you can find) with and without changing the scaling of the dataset and plot both decision boundaries on one plot. Do the plots differ? Is that what you expected? (Make sure you use the same random seed for both runs of your algorithm.)

5 Coding for Logistic Regression (Henry)

Download the Movie Review data file from Sakai named moviereview.tsv. The data file is in tab-separated-value (.tsv) format. The data is from the Movie Review Polarity data set (for more details, see http://www.cs.cornell.edu/people/pabo/movie-review-data/). Currently, the original data was distributed as a collection of separate files (one movie review per file). The data we are using converted this to have one line per example, with the label 0 or 1 in the first column (the review is negative or positive) followed by all of the words in the movie review (with none of the line breaks) in the second column. We provide a dictionary file to limit the vocabulary to be considered in this assignment. The dictionary.txt uses the format: [word, index], which represents the k^{th} word (so it has the index k) in the dictionary.

5.1

In this section, your goal is to derive the update formula for the logistic regression when optimizing with gradient descent. Gradient descent is as follows:

To optimize J(θ, x, y)
choose learning rate η
t as current time, T as max time
$\theta_0 \rightarrow$ initial value
while t < T: **do**
  Update: $\theta_j \rightarrow \theta_j - \eta \cdot \frac{\partial J(\theta, \mathbf{x}, y)}{\partial \theta_j}$
end while

Assume you are given a data set with n training examples and p features. We first want to find the negative conditional log-likelihood of the training data in terms of the design matrix $\mathbf{X}$, the labels $\mathbf{y}$, and the parameter vector θ. This will be your objective function $J(\theta)$ for gradient descent. Here we assume that each feature vector $x_{i,\cdot}$ contains a bias feature, that is, $x_{i,1} = 1 \ \ \forall i \in 1, ..., n$. Thus, $x_{i,\cdot}$ is a $(p+1)$-dimensional vector where $x_{i,1}$ =1. As such, the bias parameter is folded into our parameter vector θ.

(a) Derive the negative log-likelihood of $p(y|\mathbf{X}, \theta)$. (Hint: look at the course notes.)

(b) Then, derive the partial derivative of the negative log-likelihood $J(\theta)$ with respect to θ_j, $j \in 1, ..., p+1$.

(c) The update rule is $\theta_j \rightarrow \theta_j - \eta \cdot \frac{\partial J(\theta)}{\partial \theta_j}$. Based on the update rule, write the gradient descent update for parameter element θ_j.

We will implement this momentarily.

5.2

In this section, you are going to do some preprocessing. You are going to create something similar to a bag-of-words feature representation, where the feature vector $x_{i,v}$ for movie review i and vocabulary word v in the dictionary is set to 1 if movie review i contains at least one occurrence of vocabulary v. An example of the output from your pre-processing for one movie review might be as follows: 20:1 22:1 30:1 35:1 45:1 ....., which expresses that vocabulary words 20, 22, 30, 35, 45, etc. are in the movie review. The feature vector ignores words not in the vocabulary of dict.txt. You can use csv and math packages in python.

5.3

In this section, implement a sentiment polarity analyzer using binary logistic regression. Specifically, you will learn the parameters of a binary logistic regression model that predicts a sentiment polarity (i.e., the label) for the corresponding feature vector of each movie review using gradient descent, as follows:

- Initialize all model parameters to 0.
- Use gradient descent to optimize the parameters for a binary logistic regression model. The number of times gradient descent loops through all of the training data (number of epochs). Set your learning rate as a constant η=0.1.
- Perform gradient descent updates on the training data for 30 epochs.

Do not hard-code any aspects of the data sets into your code. You should only use the packages that are provided. You may use sklearn for creating the train/test split. You can use csv and math packages in python.

Let us provide a note about efficient computation of dot-products. In linear models like logistic regression, the computation is often dominated by the dot-product $\theta^T \mathbf{x}$ of the parameters $\theta \in \mathbb{R}^p$ with the feature vector $\mathbf{x} \in \mathbb{R}^p$. When a standard representation of $\mathbf{x}$ is used, the dot-product requires $O(p)$ computation, because it requires a sum over all entries in the vector: $\theta^T \mathbf{x} = \sum_{j=1}^{p} \theta_j x_{\cdot,j}$ However, if our feature vector is represented sparsely, we can observe that the only elements of the feature vector that will contribute a non-zero value to the sum are those where $x_{\cdot,j} \neq 0$, since this would allow $\theta_j x_{\cdot,j}$ to be nonzero. This requires only computation proportional to the number of non-zero entries in $\mathbf{x}$, which is generally very small for model compared to the size of the vocabulary.

6 Boosted Decision Trees and Random Forest (Henry)

In this assignment, you are going to fit and tune random forest and boosted decision trees to predict whether a given passenger survived the sinking of the Titanic based on the variables provided in the data set. You may use sklearn and gridsearch.

6.1

First, download the Titanic.csv file from Sakai, and drop the columns that have "na" values from the data set. Convert the gender variable into a binary variable with 0 representing male, and 1 representing female. Then split 80% of the data into train and 20% of the data into test set.

6.2

Fit a random forest and boosted decision tree model on the training set with **default** parameters, and estimate the time it required to fit each of the models. Which model required less time to train? Why do you think that is? (Hint: look at the default values of the parameters.)

6.3

Choose a range of parameter values for each of the algorithms (tell us what parameters you played with), and tune on the training set over 5 folds. Then, draw the ROC and provide the AUC for each of the tuned models on the whole training set (in one figure) and test set (in another figure). Comment on the similarities/differences between the two models (if any).

7 Generalized Additive Models

In this question, you are going to fit a generalized additive model (GAM) to predict the species of penguins on a given dataset. You can download the dataset penguins_trunc.csv from Sakai. The dataset is preprocessed and you can use it directly.

Split 80% of the data into a training set and 20% of the data into a test set. Please use one of the following methods to fit a GAM model: (1) logistic regression with ℓ_1 penalty, (2) Explainable Boosting Machine, (3) FastSparse, (4) pyGAM. Report training and test accuracy and plot the shape function for each feature on a separate plot.

Hint 1: For logistic regression with ℓ_1 penalty, you can use sklearn packages. You need to first transform each continuous feature into a set of binary features by splitting on the mid-point of every two consecutive values realized in the dataset, i.e., CulmenLength ≤ 32.6, CulmenLength ≤ 33.3. You can then fit regularized logistic regression to these indicator variables and construct the component function for variable j by weighting the indicator variables that pertain to feature j by their learned coefficients and adding them up. You can use sklearn LogisticRegression. Please remember to set penalty to "ℓ_1" and algorithm to "liblinear".

Hint 2: Explainable Boosting Machine is a tree-based, cyclic gradient boosting generalized additive modeling package. It can be pip installed for Python 3.6+ Linux, Mac, Windows. More details about the algorithm and

usage are available in https://github.com/interpretml/interpret and https://interpret.ml/docs/ebm.html.
Hint 3: FastSparse implements fast classification techniques for sparse generalized linear and additive models in R. To install this package, please follow the instruction here https://github.com/jiachangliu/fastSparse/tree/main/installation. Example code about how to run FastSparse is available https://github.com/jiachangliu/fastSparse/tree/main/application_and_usage and example code to visualize shape functions is https://github.com/jiachangliu/fastSparse/tree/main/visualization.
Hint 4: You may also use pygam package to fit a GAM model which is available here https://github.com/dswah/pyGAM.