Homework 3 and 4, Machine Learning, Fall 2023

***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 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.
4. You must show work or give an explanation for every question. "Yes" and "No" are not sufficient answers. Always show the steps of your calculations. Only partial credit can be given if you do not explain your answer.
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 statement at the top of your assignment file**:

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

1 Boosting (Theory) 10 pts (Yiyang)

In lecture, we learned how AdaBoost works from both statistical and probabilistic perspectives. In this problem, we will look closer into AdaBoost for multi-class classification.

(a) [Exponential Loss and Negative Log-likelihood Loss] In a previous lecture, we proved that the expected value of the exponential loss for Adaboost is minimized at

$$f^*(x) = \frac{1}{2} \ln \frac{P(Y=1|x)}{P(Y=-1|x)}$$

which is one-half of the log odds. Based on this result, we have obtained that

$$p = \Pr(Y=1|x) = \frac{e^{2f^*(x)}}{1+e^{2f^*(x)}}$$

Now, we would like you to maximize the expected value of the negative binomial log-likelihood loss (which is the same as minimizing the expected value of the binomial log-likelihood loss) instead, which can be written as

$$\max_{f(x)} E_{Y\sim D(x)} - \log(1+e^{-2Yf(x)}).$$

Prove that

$$f^*(x) = \frac{1}{2} \ln \frac{P(Y=1|x)}{P(Y=-1|x)}$$

still holds.

(b) [Multiclass Exponential Loss] For a K-class classification problem, for each class $c \in \{1, \cdots, K\}$, consider the coding $Y^{(c)} = (Y_1^{(c)}, \cdots, Y_K^{(c)})^T$, with the labels as

$$Y_k^{(c)} = \begin{cases} 1 & c = k \\ -\frac{1}{K-1} & c \neq k \end{cases}$$

Let $f = (f_1, \cdots, f_K)^T$ with $\sum_{k=1}^K f_k = 0$. We can define the loss function as

$$L(Y, f) = \exp\left(-\frac{1}{K} Y^T f\right).$$

Prove that the expected value of $L(Y, f)$ is

$$\mathbb{E}\left[\exp\left(-\frac{1}{K}\sum_{k=1}^K Y_k f_k\right)\right] = \mathbb{E}\left[\exp\left(-\frac{1}{K-1} f_k\right)\right]$$

Hint: Recall that, for a real valued $a \neq 0$ and $a \neq 1$, $\frac{1}{a(a-1)} = \frac{1}{a-1} - \frac{1}{a}$.

(c) [Multiclass Optimization] Based on the expected value for the loss function we derived from (b), and the Lagrange multiplier for constraints,[1] we can get the objective function as

$$\min_f \mathbb{E}\left[\exp\left(-\frac{1}{K-1} f_k\right)\right] - \lambda \sum_{k'=1}^K f_{k'}. \tag{1}$$

Derive the minimizer f^* of the expectation of $L(Y, f)$, subject to the zero-sum constraints, and find the optimized $f_k^*(x)$ and $P(Y = Y^{(k)}|x)$. Please express $P(Y = Y^{(k)}|x)$ as a function of K and f^* (and not λ). Hint: do not eliminate λ before taking a derivative.

[1] This will be discussed in detail in the section on convex optimization, but you do not need to know much convex optimization for this problem.

2 Random Forest (Theory) 10 pts (Ishaan)

(a) Two friends, Geoff and Carina, are each trying to create a Random Forest classifier to fit their data. Both are familiar with how to construct Random Forests, but the two friends used different sampling methods. When they test their models, Geoff gets a very small average correlation ρ_1 between his model's trees, while Carina's average correlation ρ_2 is much higher. Let's give some more details.

1. Each random forest classifier is composed of a large number n of decision trees – each, when given an input, outputs a binary classification of 0 or 1.
2. Assume the expected accuracy of any individual tree on this input is p, where $0 < p < 1$, and the variance of the binary output is σ^2.
3. Given a new input, the random forest classifier will take the aggregate of the tree's results and use a majority vote to create the final classification.

First, show how the variance $Var(\bar{X})$ for the average result of the Random Forest, where $\bar{X}$ can be rewritten in terms of the individual outputs X_i of the trees, will differ between Geoff and Carina. Hint: What terms must we include to calculate the variance of the sum of multiple variables?

Second, for each of Geoff and Carina, derive the probability that the aggregated result of the forest deviates from the expected value by more than a small positive value ϵ. Compare these two equations for the situation when n approaches ∞ to discuss which person will be more likely to have a "better" random forest model (or at least a model with a better guarantee): mathematically, what's the benefit of having a model with near-zero correlation? Hint: What does the average output of the trees converge to when n becomes very big (and what theorem shows this)? Hint 2: use Chebyshev's Inequality, which states that for a random variable X with finite mean μ and finite variance σ^2 and strictly positive real number k, we can state that: $P(|X - \mu| \geq k) \leq \frac{\sigma^2}{k^2}$.

(b) One classifier we've already learned about is k-nearest neighbors – now, let's try to understand the difference between these two classifiers in high-dimensional spaces. Consider a dataset of N samples uniformly distributed in a d-dimensional unit hypercube. We aim to investigate how KNN and Random Forests interact with this dataset in the context of local sampling.

1. **For KNN**: Given a point x in this space, derive the expected radius r_k required to encompass its k nearest neighbors. Use the fact that the volume of a hypersphere within this radius r_k can be derived as: $V_k = C_d * r_k^d$, where C_d is a constant that depends on d and represents the volume of a *unit* hypersphere in d dimensions (more on this soon), and V_k is the volume of the hypersphere in which the k nearest neighbors of a point x are expected to lie. Hint: Use the idea that these points are uniformly distributed. The expected number of data points in the hyperspheres is the total number of data points N times the fraction of the hypercube's volumes that the hypersphere occupied.
2. **For Random Forests**: When training a decision tree on a bootstrapped sample (sampled with replacement), derive the expected number of unique samples E[unique] drawn from the original dataset. Hint: Calculate the probability that a given point is NOT chosen in a single draw.
3. **Direct Comparison**: Let's try and use the results from 1 and 2 to explore one of the biggest issues of KNN's in high-dimensional space.
    - An interesting fact about C_d, which represents a constant used to compute the volume of a hypersphere for d dimensions, is that as d increases, the value for C_d actually peaks and then decreases for large (> 5) d. Therefore, for very high dimensions, the volume of a *unit* hypersphere becomes vanishingly small compared to the hypercube enclosing it. In KNN, intuitively reason in one to two sentences what happens to the "required" radius r_k as the dimension d grows to big numbers, holding all else constant, and what this indicates about the discriminative power of KNNs for high dimensional data? In other words, how effective is KNN when it is under high-dimensional cases? You may assume that $N * C_d > k$ in this case where k is a small fraction of a large N.

- In contrast, what percent (give a numeric answer) of data will be represented in a bootstrapped sample (there are N total bootstrap samples) when N is very large (assume sampling with replacement)? Compute a ratio $\frac{V_{RF}}{V_k}$ between V_{RF}, which is the proportion of the entire data space N that is covered by E[unique] points (i.e., the number available to an RF), and the volume V_k covered by KNN to justify which classifier is more robust in high-dimensional situations. Hint: $(1-\frac{1}{n})^n \approx \frac{1}{e}$ when n is really large.

3 Variable Importance for Trees and Random Forests (Coding) 20 pts (Stephen)

In this question, you will investigate several measures of variable importance for trees and random forests. Software packages often lack detail and/or use slightly different definitions of variable importance for trees and random forests. There is not necessarily a "best" measure. The purpose of this question is for you to calculate a few measures yourself in order to learn about some the issues involved in variable importance for trees and random forests.

Recall that for each node t in a decision tree $\mathcal{T}$, we grow the decision tree by finding the split — defined by a variable X_{j_t} and a vector of cutoffs s_t (*e.g.*, $X_{j_t} \leq s_t$ in the binary tree case) — that maximizes the reduction in the impurity measure, $\Delta I(s_t, j_t, t)$. For decision trees, one measure of variable importance for a variable X_j is the total reduction in the impurity measure attributable to X_j:

$$\text{Imp}^{\mathcal{T}}(X_j) := \sum_{t \in \text{nodes}(\mathcal{T})} \mathbb{1}_{[j=j_t]} \Delta I(s_t, j_t, t). \tag{2}$$

Now, suppose there is a different variable $X_{\tilde{j}_t}$, where $\tilde{j}_t \neq j_t$, and cutoff $\tilde{s}_t$ that results in nearly as large of a reduction in the impurity measure. Should this count towards the variable importance of $X_{\tilde{j}_t}$ even though it is not used for the split at node t of tree $\mathcal{T}$? Well, $X_{\tilde{j}_t}$ is not important to $\mathcal{T}$ at node t in the sense that $\mathcal{T}$ does not require $X_{\tilde{j}_t}$ at node t in order to achieve its predictive performance on the training dataset. However, suppose a tree $\tilde{\mathcal{T}}$ were trained on a second training dataset drawn from the same distribution. The large reduction in the impurity measure due to $X_{\tilde{j}_t}$ on the first dataset suggests a reasonably high likelihood that $X_{\tilde{j}_t}$ would have a higher reduction in the impurity measure on the second dataset than X_{j_t}, in which case $X_{\tilde{j}_t}$ would have a higher variable importance (as calculated by Equation (2)) than X_{j_t} with respect to tree $\tilde{\mathcal{T}}$ at node t. In this sense, we say that X_{j_t} is *masking* the *potential variable importance* of $X_{\tilde{j}_t}$. A measure of variable importance that more effectively captures the variable importance *and* potential variable importance is one that includes the impurity measurement that would occur if $X_{\tilde{j}_t}$ were used as the split, even though it is not. We define this measure as follows:

$$\text{Imp}_s^{\mathcal{T}}(X_j) := \sum_{t \in \text{nodes}(\mathcal{T})} \mathbb{1}_{[j=j_t]} \Delta I(s_t, j_t, t) + \mathbb{1}_{[j=\tilde{j}_t]} \Delta I(\tilde{s}_t, \tilde{j}_t, t), \tag{3}$$

where $X_{\tilde{j}_t}$ and cutoff $\tilde{s}_t$ constitute the best *surrogate split*. In other words, for each node, find the best split and the best surrogate split (which is chosen among all of the variables that are not used in the best split). If the variable used in either split is X_j (note that by construction both splits cannot use X_j), then add the reduction in the impurity measure to the importance of X_j. Note that Equations (2) and (3) are sums over *all* nodes in the tree.

We have not yet discussed the way to define the "best" surrogate split. For a binary tree, we define the best surrogate split by the variable $X_{\tilde{j}_t}$, where $X_{\tilde{j}_t}$ is not the actual best split variable X_j, and cutoff $\tilde{s}_t$ that maximize the following predictive similarity measure:

$$\lambda(j_t, s_t, \tilde{j}_t, \tilde{s}_t) = \frac{\min(p_L, p_R) - (1 - P_{L_{j_t} L_{\tilde{j}_t}} - P_{R_{j_t} R_{\tilde{j}_t}})}{\min(p_L, p_R)} \tag{4}$$

where p_L is the proportion of observations such that $X_{j_t} < s_t$, p_R is the proportion of observations such that $X_{j_t} \geq s_t$, $P_{L_{j_t} L_{\tilde{j}_t}}$ is the proportion of observations such that $X_{j_t} < s_t$ and $X_{\tilde{j}_t} < \tilde{s}_t$, and $P_{R_{j_t} R_{\tilde{j}_t}}$ is the

proportion of observations such that $X_{j_t} \geq s_t$ and $X_{\tilde{j}_t} \geq \tilde{s}_t$. Choosing the best surrogate split by maximizing $\lambda(j_t, s_t, \tilde{j}_t, \tilde{s}_t)$ is the method used by `MATLAB`, for example.

For random forests, we can extend the decision tree measures of variable importance by simply averaging over all of the trees. For a random forest $\mathcal{F}$ composed of trees $\{\mathcal{T}_i\}_{i=1}^M$, the analog to $\text{Imp}^{\mathcal{T}}$ is given by:

$$\text{Imp}^{\mathcal{F}}(X_j) := \frac{1}{M} \sum_{i=1}^{M} \text{Imp}^{\mathcal{T}_i}(X_j) \tag{5}$$

There is also another method, as discussed in class, that uses permuted "out-of-bag" samples:

$$\text{Imp}^{\mathcal{F}}_{\text{OOB}}(x_j) := \frac{1}{M} \sum_{i=1}^{M} \text{error}_{\text{OOB}}(\mathcal{T}_i, x_j^{\text{perm}}) - \text{error}_{\text{OOB}}(\mathcal{T}_i, x_j) \tag{6}$$

where $\text{error}_{\text{OOB}}(\mathcal{T}_i, x_j)$ is the out-of-bag error of tree $\mathcal{T}_i$ predicting on bootstrap sample i with X_j unadjusted and $\text{error}_{\text{OOB}}(\mathcal{T}_i, x_j^{\text{perm}})$ is the out-of-bag error of tree $\mathcal{T}_i$ predicting on bootstrap sample i with X_j randomly permuted. See the class notes for more discussion of this variable importance measure. We will use the least-squares error. When taking the average in Equations (5) and (6), if a variable is not used in the decision tree, ignore this term in taking the average. Only average over trees in which variable X_j is used in a split.

For this question, use the accompanying training and test data, `train.csv` and `test.csv`. You will find $n = 500$ and $n_{\text{test}} = 100$ observations of a binary outcome Y and five binary covariates, $X_1, \ldots, X_5$. You may use any combination of your own code or an existing package to answer any part of this problem. If you choose to use a package, make sure you read the documentation carefully to understand exactly what it is doing. For simplicity you will use decision stumps (*i.e.*, decision trees with one split)[2]. Use the Gini index as the impurity measure. If a variable is not used in a decision tree or decision forest, it is not important to that model, so report it as 0.

(a) Grow one decision stump on the training dataset and answer the following questions.

  (i) Describe clearly (or draw) the decision stump based on the best split and the decision stump based on the best surrogate split.

  (ii) Report the variable importance measurements from Equations (2) and (3) for this stump based on the best split and best surrogate split. Does this suggest any variable(s) are more important than the others?

  (iii) Report the mean misclassification error of predictions on the test dataset of both decision stumps from part (a)(i).[3]

(b) Grow a random forest of decision stumps on the training dataset with $K = 1, \ldots, 5$ randomly selected variables available for each stump. Use $M = 1000$ stumps and $B = 0.8 \times n$ bootstrap samples. The following question parts should be done for each K (ideally with your numerical results summarized in a single table for each part). For each part, discuss any dependence of your answers on K and why this may have occurred.[4].

  (i) How many times is each variable used for the best split? How many times is each variable used for the best surrogate split? We may consider one variable more important than another if it appears more often in either splits or surrogate splits. Which variable is most important?

[2]Random forests are usually composed of fully grown decision trees. In this problem we use decision stumps only. Stumps are just special cases of decision trees (they have only one split) so all of the variable importance measures we defined for decision trees apply to decision stumps.

[3]In other words, if $\{\hat{y}_i\}_{i=1}^{n_{\text{test}}}$ are the predictions of the model and $\{y_i\}_{i=1}^{n_{\text{test}}}$ are the actual labels, then the error you should evaluate is $\frac{1}{n} \sum_{i=1}^{n_{\text{test}}} \mathbf{1}_{\hat{y}_i \neq y_i}$.

[4]Parts (b) and (c) ask you to interpret your answers based on K and B. There is not one correct answer, you will be graded on the overall quality of your response.

(ii) Compute the variable importance measures in Equations (5) and (6). Does this suggest any variable(s) are more important than the others according to these criteria? Recall that when using Equation (2) to compute variable importance for decision stumps, the phenomenon of "masking" can hide the potential variable importance of some variables. When using Equations (5) and (6) to compute variable importance for random forests, does masking hide the importance of some variables as much as it does with decision stumps? (On the topic of masking, you do not need to provide a numerical answer, just a brief discussion of the role of masking).

(iii) Compute the mean misclassification error on the test data using two methods. In the first method, use the majority vote of the stumps as the prediction and compute the loss. In the second method, find the predictions of each stump, compute the loss on each, and average the results. Which method is correct for computing the prediction error of the random forest?

(c) Grow a random forest of decision stumps with $B = q \times n$ bootstrap samples, for each of $q \in \{0.4, 0.5, 0.6, 0.7, 0.8\}$. Let each stump choose from $K = 2$ randomly selected variables (this is the closest to the default choice of $\sqrt{p} \approx 2.23$) and $M = 1000$ stumps. The following question should be answered for each B (ideally with your numerical results summarized in a single table for each part). For each question, discuss any dependence on B and why this may have occurred.

(i) Compute the variable importance measurements in Equations (5) and (6). Does this suggest any variable(s) are more important than the others?

4 GAM (Coding) 5 pts (Allan)

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) FastSparseGAM/L0Learn, (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 the penalty to "l_1" and the algorithm to "liblinear". Remember to avoid data leakage during feature binarization.

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. Since the shape function plots for each feature are interactive in EBM, please download the figure and insert it in your notebook, the detailed step is shown here.

Hint 3: FastSparseGAM/L0Learn implements fast classification techniques for sparse generalized linear and additive models in python. To install this package, please follow the instructions here https://github.com/ubc-systopia/L0Learn/tree/master. Example code about how to run FastSparse using python interface is available https://github.com/ubc-systopia/L0Learn/blob/master/python/tutorial_example/example.py. You can always install L0Learn in Colab.

Hint 4: You may also use pygam package to fit a GAM model which is available here https://github.com/dswah/pyGAM.

5 Boosting Algorithm Practice (Coding) 5pts (Stark)

In this assignment, you are going to fit and tune 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 the XGBoost library.

(a) First, download the Titanic.csv, 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. Convert the rest of the data into appropriate data types. Then split 80% of the data into train and 20% of the data into the test set using `train_test_split`.

(b) Use sklearn's Adaboost and XGBoost to implement two classifiers for survival prediction and use five-fold cross-validation to tune some parameters of the Adaboost and XGBoost classifiers. Comment on their performance difference using metrics of your choice on the test set. Use decision tree as the base estimator for both boosting algorithms.