Problem Set 2 (Theory 25 points)

June 2024

1 Problem 1 (6pts)

Both random forest and tree boosting are techniques we can use to improve the accuracy of simple models. The difference between the two methods is mainly in how individual trees are built and combined to form the final prediction. Describe the two mechanisms when used for classification.

2 Problem 2 (4pts)

One way to construct an ensemble of base models is through bagging. Each base model is constructed by training a decision tree (or some other predictive model) on a set of data points drawn from a data set using random sampling with replacement.

Theoretically speaking, even if a large number of samples are used to construct each tree, we would still have a set of out-of-bag data points. Derive the limiting probability that any data point is used for at least once in training if each tree is trained with n samples (n is large) and there are k independent trees.

3 Problem 3 (3pts)

Let ρ be the correlation coefficient of X and Y. Show that after any affine transformation of X and Y, $X' = aX + b$ and $Y' = cY + d$, the correlation coefficient of X' and Y' is still ρ.

4 Problem 4 (3pts)

Assume the true model of a problem is $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + U$, but X_2 is unobserved, so we use the model $Y = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{U}$ where the residual is mean zero and uncorrelated with any of the Xs. Give one condition under which the estimate $\hat{\beta}_1$ will be unbiased in this misspecified model. If $\text{cov}(X_1, X_2) \neq 0$, derive an expression of the bias of $\hat{\beta}_1$

5 Problem 5

In this problem, we study the effect of outliers in parameter estimation. In all parts of this problem, assume we have a finitely large number of data points, so we do not need to worry about the systems being underdetermined. Assume X is fixed.

5.1 Part I (2pts)

When estimating the model $Y = aX$ where X is a one dimensional variable and a is a scalar, can we change one Y in the data set to make a equal to any desired value a_0? Provide a short proof is we can and a counterexample if we cannot.

5.2 Part II (2pts)

If we were to estimate the model $Y = aX + b$ instead, can we change one Y to fit any desired a_0 and b_0? Provide a short proof is we can and a counterexample if we cannot.

5.3 Part III (2pts)

Now we want to estimate a k-dimensional model $Y = \beta X$ where β and X are both k-dimensional. How many Ys do we need to change (at least) to fit any desired β_0? Provide a short proof.

6 Problem 9 (3pts)

Given a simple neural network that takes in a one dimensional input and passes it directly to the output layer with one linear node with a swish activation function. The final objective function is MSE. Let the parameters in the linear node be β and α where output = swish($\beta x + \alpha$).

$$L(\beta, \alpha) = (\text{observation} - \text{output})^2$$

Given a single data point x, y, derive the gradient of the loss function with respect to the parameters α and β

Case study (45 pts in total)

1 Predicting ad clicking behavior with LASSO, elastic net, deci-

decision tree, random forest, and XGBoost.

In this exercise, we will predict users' ad clicking behavior using a LASSO model, a elastic net model, a decision tree, a random forest, and XGBoost.

## Loading the python packages

[ ]:
```python
# Load libraries



import warnings
warnings.filterwarnings('ignore')

#import os

import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
%config InlineBackend.figure_format = 'retina'

# for higher resolution
from IPython.display import set_matplotlib_formats
set_matplotlib_formats('svg','pdf')

# nice format for matplotlib https://tonysyu.github.io/raw_content/
 ↪matplotlib-style-gallery/gallery.html
plt.style.use('bmh')
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from pandas import read_csv, set_option
from pandas.plotting import scatter_matrix
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split, KFold, cross_val_score,␣
 ↪GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.pipeline import Pipeline
from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier,␣
 ↪RandomForestClassifier, ExtraTreesClassifier
from sklearn.metrics import classification_report, confusion_matrix,␣
 ↪accuracy_score

#Libraries for Deep Learning Models
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
#from keras.optimizers import SGD
from tensorflow.keras.optimizers import SGD
```

## Loading the Data

[ ]:
```
# load sample dataset (only load the first 300000 rows) (1pts)
```

[ ]:
```
#Explore the data (print the first and last 5 rows) (2pts)
```

[ ]:
```
#remove nas from dataframe by simply dropping these rows (2pts)
```

[ ]:
```
    #create Y and X. Y is the "click" column, and X is the other 19 columns (Do not␣
 ↪use ['id', 'hour', 'device_id', 'device_ip']) (1pts)
```

[ ]:
```
#use sklearn one-hot encoder to transform string variables in x to categorical␣
 ↪columns. make sure to save the data as a sparse matrix. (3pts)
```

[ ]:
```
#do a train-test split. Use the first 90% of the data as training. (1pts)
```

1.1 LASSO and ElasticNet

First use a sklearn LASSO model with alpha=0.005 (regularization penalty) to predict clicking behavior. Report the prediction accuracy. Then use sklearn elastic net with alpha = 0.005 and l1_ratio=0.5 (l1_ratio is a number from 0 to 1 which represents the portion of L1 penalization in the total penalization term)

[ ]:
```
#initialize and training the model (1pts)
```

```
[ ]: #testing the model (2pts)
```

```
[ ]: #initialize and training the elastic net (2pts)
```

```
[ ]: #testing the elastic net (2pts)
```

```
[ ]: #Compare the two results. What are the differences? What do you think is␣
      ↪causing these differences? 2pts
```

1.2 Decision tree and random forest

```
[ ]: #build a single decision tree model using gini impurity and roc_auc scoring␣

      ↪(2pts)
[ ]:
     #do a grid search on the max-depth variable [3,10,None]. (3pts)
[ ]:
     #print the auc of the optimal model applied on the test set (3pts)
[ ]:
     #build a random forest using gini impurity and roc_auc scoring (2pts)
[ ]:
     #do grid search to tune n_estimators and max_depth -- 'max_depth': [3, 10,␣

[ ]:  ↪None],'n_estimators': [10,50,100,200]. (3pts)

                                  #print the performance of the best model (3pts)
```

1.3 XGBoost

Use the XGBoost classifier to predict clicking behavior. Fine-tune n_estimators over [10, 50, 100] and η (eta) over $[0.01, 0.05, 0.1]$. Use roc_auc as the scoring criterion for CV.

```
[ ]:
```

```
[ ]: #initialize the parameter grid and the model (2pts)
```

```
[ ]: #do grid search (3pts)
```

```
[ ]: #print the performance of the best model (2pts)
      ↪model, which ones are likely underfitted/overfitted. (3pts)
     #Looking at the CV results of the decision tree, random forest, and XGBoost␣
```

Additional Problems

1 Optimal classification algorithm (15 pts)

We have the following dataset $\{x_i, y_i\}_{i=1}^n$ that a model specified as

$$y_i = \mathbb{1}(f(x_i)g(h_i) > 0)$$

where y_is are binary outcomes, x_is are vector samples of categorical variables and h_is are unobserved vector samples of other variables.

1.1 Derivation

Assuming you have a large training dataset, design an algorithm that produces a label y_i for each input data x_i (without observing h_i). Describe your algorithm (pseudo-code).

1.2 Implementation

Implement this algorithm. Train the model with the training dataset (train_data1.csv) and output the labels for the test set (test_data1.csv). You save your trained model and submit a function that loads the model properly and outputs a prediction of each row in the test data. Your score will depend on the F1 score of your predictions. The known categorical variables are *category*1 and *category*2, and the true labels are in the *label* column.

2 Random forest (25pts)

A random forest is an ensemble of decision trees. In this exercise, write a python code that implements a random forest from scratch. Different trees are fitted with different subset of data (with all of the columns).

1. Write a tree class that initializes a single decision tree.
2. Write a random forest class that initializes n_tree trees with different subsets of data.

3. Write a get_prediction method in the random forest class that computes the average prediction (if continuous) or majority vote prediction (if categorical) based on the forest.

All of your trees should be based on the same hyperparameters: max_leaf (maximum number of leaves), min_gain (minimum information gain based on entropy to make another split), and max_depth (maximum depth of the tree). Train the model with train_data2.csv and save the model weights. Submit a function that loads the trained model properly and produces a prediction for each row in test_data2.csv. Your final performance will be evaluated by the MSE of the test dataset.