Applied Machine Learning

Assignment 1: Linear classifiers

Due 14 February 2020, 11:59 PM EST

In this assignment, you will explore linear classification on the forest covertype dataset created by Jock Blackard et. al. in 1998. It is used frequently enough that scikit-learn has made it a standard dataset.

`https://scikit-learn.org/stable/datasets/index.html#covtype-dataset`

`https://archive.ics.uci.edu/ml/datasets/Covertype`

The data set includes 54 features (reals, integers, and booleans) and 7 classifications numbered 1-7: spruce/fir, lodgepole pine, Ponderosa pine, cottonwood/willow, aspen, Douglas-fir, and Krummholz. The scikit-learn implementation has converted all of these values to real numbers.

1 Importing and Partitioning the Data

1.1 Get the Data (1 pt)

Using scikit-learn, import the Forest covertype data set, shuffling it with random state 5984 in the call to fetch it. Set these to `X` and `y`

1.2 Checking the Labels (1 pt)

How many instances of each of the seven classes exist in the data set?

1.3 Choosing Subsets (1 pt)

Create data and label subsets of size 5000, 10000, and 100000, choosing the first N elements of the shuffled imported data.

1.4 Splitting the Subsets into Training and Test Sets (1 pt)

Use

`train_test_split`

to set aside 20% of each test set with random state 5984.

1.5 Remaking the Labels for Binary Classification (1 pt)

For each `y` subset, create a label vector that indicates if each label is spruce/fir (1) or not (0). Use the subsets you already created as the basis for this.

1.6 Baseline Accuracy (1 pt)

For each training subset, calculate the accuracy of a "not spruce/fir" estimator, that is, one that outputs 0 for every input. Hint: the sum of the labels in each training label subset can be used to quickly calculate this value without making a specific function.

2 Perceptron Classification

For the 5000 sample and 100000 sample subsets, you will create a Perceptron classifier that determines whether each sample point is spruce/fir or not.

2.1 Perceptron Analysis (1 pt per subset per subproblem)

For each binary data and label subset, train the `Perceptron` classifier with no added arguments, using only the training subsets.

1. What is the accuracy on the training sets and the test sets?
2. What is the confusion matrix on the test sets? Use the predicted labels as a basis for comparison to the true labels for each test set. What does the confusion matrix tell you about the classifier's ability to correctly and incorrectly predict whether each sample is a spruce/fir cover or not?
3. What are the precision and recall on the test sets?
4. Using the decision function as the threshold, plot precision and recall vs. threshold.
5. Using the decision function as the threshold, what threshold value maximizes the F score, and what is the value?

2.2 Evaluation (2 pts)

Would you say the old-fashioned Perceptron algorithm is useful for this data set? Is it better than a "not spruce/fir" classifier in any scenario? Why or why not?

3 Support Vector Machine Binary Classification

In this problem you will use the 10000 sample subset you made back in Problem 1 and will learn how to use grid search to tune hyperparameters to get the performance you want.

3.1 Scaling and Solver Speed (1 pt)

For `C=1, polydeg=3, class_weight=None`, create a both `SVC` classifer, and a `Pipeline` that uses `StandardScalar` to scale the `X` subset prior to using another `SVC` classifier. Try running both. What can you say about the relative speeds? Why do you think this is happening?

3.2 Polynomial Kernel Grid Search (2 pts per subproblem)

Without using scikit-learn's `GridSearchCV` module, create a loop that trains an SVC over each combination of the following parameters:

```
Cs = [0.001,1,100]
class_weights = [None,'balanced']
polydegs = [3,5,7]
```

If you did this right, you should have 18 different classifiers. Hint: use Python's `itertools.product` to generate the 18 option sets without using 3 nested for loops.

1. Use `cross_val_score` to generate accuracy scores for each validation set. Using the average CV score, what is the best combination?
2. Compare the average of the CV scores to the score generated by the overall training set. Which combination overfits the most? The least?
3. If you wanted a classifier that prioritized correctly classifying as many spruce/fir samples as possible without concern for misclassifying other types as spruce/fir, which combination is the best?
4. If you wanted a classifier that prioritized making sure other cover types are not classified as spruce/fir even if it means misclassifying spruce/fir types incorrectly, which combination is the best?
5. Find the accuracy score for each combination of options on the test set. Which combination actually worked the best on the test set?
6. What was the overall effect of increasing C?
7. What was the overall effect of increasing the polynomial degree?
8. Assigning balanced weights is meant to mitigate the effect of larger label counts skewing the SVM result. Does it work?

3.3 Evaluation (2 pts)

Did the CV scores you encountered give you confidence that the polynomial kernal SVM would be an improvement over the "not spruce/fir" classifier? What would you try to do differently to find a better SVM?