Predicting the Roosevelt National Forest Cover Type

Minji Pack

Andrew ID: mpack
Masters in Biotechnology
Innovation and Computation
2016-2018 Batch

Introduction

By implementing machine learning pipeline, I get to know why each step is required, what steps are required, and how machine learning pipeline works. In this assignment, I have implemented machine learning pipelines of data transformation, and model selection. In data transformation, a normalizer is applied to numerical features. Also, one hot encoding is used for the features containing very few values since those features seemed to be categorical features.

In model selection, I have tried multiple classification models since this given data set, a finite number of labels, clearly shows that this is a classification problem. I made use of a three classifiers: logistic regression, Stochastic Gradient Descent, KNN. To evaluate each model performance, I used multi-class classification accuracy. I also measured the residuals between predicted labels and true labels and removed outliers that had detected.

Dataset

In the given dataset[1], there are 15120 observations data with labels as a given training set. I used the given training data set into two parts: train data set and validation data set. This is because I need labels to evaluate my performance of my models. Therefore, I need data set with labels. For both data set, I also split labels from data set.

There exists **7 labels** and **53 features** in my datasets. As features, the first ten features are [4]

1. Elevation
2. Aspect
3. Slope
4. Horizontal Distance to Hydorology
5. Vertical Distance to Hydrology
6. Horizontal Distance to Roadways
7. Hillshade at 9 am
8. Hillshade at noon
9. Hillshade at 3pm
10. Horizontal Distance to nearest wildfire ignition points
11. Four binary features indicates the types of wilderness area.
12. Forty binary features indicate soil types

Hence, my training set as well as validation set contains such feature information. For labels, we have seven forest cover types:

1. Spruce/Fir
2. Lodgepole Pine
3. Ponderosa Pine

4 Cottonwood/Willow
5 Aspen
6 Douglas-fir
7 Krummholz

But my both train label set and validation label set will contain 0-6, indicating 1-7 label, respectively.

1 **Train Set:** 10131 instances
2 **Validation Set:** 4989 instances

1 Module Design

First, I have separated into machine learning package, data folder, and main.go. Main.go is the part that runs overall experiments. A data folder cotains the whole dataset I mentioned above. The main part is machine learning package. Under machine learning package, it contains every module that is related to machine learning pipeline:

1 **Models:** Contain the machine learning algorithms I have implemented: Logistic Regression and Stochastic Gradient Descent. These are designed in the following way
   - **Base Struct**: Contains the model hyperparameters and the weights
   - Model Initialization: I have created a 'Create`AlgoName`' method associated with struct, which will accept user inputs for hyperparameter values. It will also do any other necessary initialization.
   - **Model Learning**: I have made the 'Fit' method associated with the struct, which will take the training set as an input and apply the model algorithm to train parameter weights.
   - **Model Prediction**: I have made the 'Predict' method associated with the struct, which will take the test set(only data, no labels) as input and provide predicted labels for users.
   - **Helper Methods**: These help me perform training and prediction and keep code clean.
2 **Transformers**: They contain the data transformations described below. They are designed similar way to the models - with a base struct, the initialization and the transformation to be applied to them.
3 The other modules: Preprocessors, Evaluators and other data utility functions have been created and defined for use.

Under machine learning package, model and transformers consists of objects in a way following object oriented programming. What I mean by object oriented programming is that a object structure containing properties exists and have a constructor when used to create a object. With regard to the object, we can implement multiple methods as much as we want that updates the properties in the object.

2 The Machine Learning Pipeline

The section describes each components of ML Pipeline.

2.1 Data Cleaning

I have explored the whole dataset with Excel. There were no missing values, so I have not done Data Cleaning implementation.

2.2 Preprocessing

The function of preprocessor is that reading the file in a correct format and divide whole data into data fields and labels. For data utility, it deals with how to split data as well as labels into train dataset and train labels, and validation dataset and validation labels. I have divided the whole data into training and validation data sets with a 0.33 ratio split(i.e. 67% training and 33% validation). Not using test data since it is provided but labels not available to us

2.3 Data Transformations

2.3.1 Normalizer

Different numerical features might have different feature variance(high or low) which affects final model performance. It is important to have minimum variance within a feature for all features which improves performance.

$$X_{norm} = \frac{X - \overline{X}}{X_{std}}$$

Here $\overline{X}$ is the row mean of feature matrix X, and X_{std} is the row standard deviation of feature matrix X.

I have not applied Normalization to the binary valued columns since they are already One hot encoded and we shouldnt use normalization on such features.

2.3.2 Outlier Removal

This subsection describes how I detect outliers and remove them from the training dataset.

After proceeding all the previous steps, we can recognize which instances are causing problems by comparing true labels with our predictions from the model we have utilized. This is the method of Residual Analysis for detecting and removing outliers. In Regression problems this can be done easily comparing true values with predictions on training data.

$$R = |y_{true} - y_{pred}|$$

where R is the residual. The samples with high overall residuals can outliers. They should be removed before proceeding. To apply this to classification, I applied this formula:

$$R = 1.0 - confidenceScore(y_{pred})$$

This means follows - I tried to measure residuals by measuring the confidence of the classifier for the true label. A high confidence is indicative of high residual error. This can be more easily done with Residual plots.

Table 1 Comparision between results without outliers removal and results applying outliers removal

1	Training without outliers removal	Testing without outliers removal	Training	Testing
LR	0.6000	0.5859	0.6018	0.5863

2.4 Model Algorithms

2.4.1 Multiclass Logistic Regression(with L2 Regularization)

Logistic Regression is the way to capture the linear features for a classification problem. The way of converting traditional Logistic Regression meant for two labels to many labels is by applying **One vs. All training**. This means that we have to one-hot encode the labels, and maintain different sets of weights for each label. However, there is a high chance of being overfitting. The way to solve is to apply Regularization. How it works is to assign small weights instead of an extreme value and very close to zero for the less likely features. With regularization, it helps to avoid overfitting. Unlike L1 Regularization, L2 Regularization prevents scarcity.

2.4.2 Multiclass Stochastic Gradient Descent(with L2 Regularization)

Similar to Logistic Regression, Multiclass SGD is a linear classifier. It is unique because it adds randomness and also allows for online learning - which means that we can keep adding and using new training data even after 1 step of learning is complete. This is a very special thing which very few classifiers have.

2.4.3 K-Nearest Neighbours

This is more of an in-memory classifier which is known to work well on some datasets(mainly images). It stores whole train set, and uses combination of distances of test set from train data and corresponding labels, to find its own label.

Results

Table 2 shows the results of the technique applied on the Development data set. I have for all of my own implementations, also setup a GoML implementation to compare how I do against standard library implementations. I have additionally run experiments to tune my hyperparameters for the models both for GoML and for my own implementation. The sets of preprocessing and transformations discussed above has been applied.

Table 2 Results of the technique applied on the Validation data set

1	GoML Training	GoML Testing	My Implementation Training	My Implementation Testing
SGD	0.6010	0.5847	0.1642	0.1555
LR	0.6010	0.5849	0.6018	0.5863
KNN	0.1471	0.1329	-	-

As can be seen for Logistic Regression I perform as well as the standard library implementation. For SGD, it seems that I have missed something in my implementation or have some bug. Since time for project submission was up I have submitted for now, but will try more after this. In the standard library implementation, SGD performs a little worse than Logistic Regression can be seen. Finally KNN can be seen to have very bad performance in general.

Conclusion:

Throughout this project, I have explored external libraries, i.e., gonum and goml and get to know about them; how to install external libraries in go, how to import libraries with annotation, and also call functions in external libraries. As a result, I become more confident with go with others' codes and refer to not only go official document but also other documents associated with external libraries. Another

invaluable thing I have learned is to get to familiar with object oriented programming. Before this project, I have made use of some object structures, but I did not understand it fully. However, during the project, I had to use struct, construct objects by getting input and initialize them. It was a great experience.

References

1. https://www.kaggle.com/c/forest-cover-type-prediction, Accessed 25th November, 2017
2. https://www.slideshare.net/KayleighBeard/forestcovertype, Accessed 25th November, 2017
3. https://www.slideshare.net/danielgribel/forest-cover-type-prediction-56288946, Accessed 25th November, 2017
4. David Morales-Hidalgo *"Tree Cover Assessment"*, Page 6(2006)