CISC 5800: Machine Learning

FINAL PROJECT

Xinyi (Jonathan) Liu | December 12, 2015

Contents

Introduction 1

Methods 1

Results 3

Conclusion 7

Introduction

Using the "Adult" dataset hosted on UCI's Machine Learning Repository, the target of this project is to build classification functions utilizing different machine learning algorithms, and optimize those functions' prediction accuracies. For this purpose, several steps of data cleaning were first implemented to transform data into a better shape for modeling process, and two algorithms used to build final machine learning functions for this project were logistic regression and support vector machine algorithms. As the result, with multiple parameter selection and function restructure, both of two models have successfully reached good levels of accuracy, and great amount of valuable experience was obtained through this project.

- Description of "Adult" Dataset

This dataset was extracted from the 1994 census database, which contains series of demographic features including age, employment status, education, etc. The format of this dataset is mixture of characters and numbers, and not all features in the dataset are providing unique and useful information. Therefore data preparation is highly necessary before any modeling attempt.

Methods

- Data preparation

As stated above, the original dataset contains a mixture of data types, and there are other issues existed in this dataset that requires certain data cleaning process to be performed. Therefore, in order to efficiently clean the dataset, R programming language was utilized to finish this task.

First, by utilizing "as.factor" and "as.numeric" functions in R, all character features were transformed into numeric values. Then, because the range of number features in this dataset are highly diversified, a scaling step was taken using the formula shown below (See Formula 1), in order to coerce all number figures to have similar scales. In detail, each value was first subtracting the column mean from this value, then dividing this value by the standard deviation of its column. After scaling, the dataset was able to better represent the information hidden behind its figures.

Formula 1. Scaling formula

$$\frac{x - \overline{x}}{\sigma_x}$$

- Split training and testing data sets

After data cleaning and scaling, the last step is to separate training set and testing set. This was accomplished by utilizing R's "sample.split" function provided by "CaTools" package, which randomly split the dataset into separate subsets, while maintaining same distributions of the target variable in those subsets. The seed was set to value 42 in order to reproduce same split result every time running this function. The training/testing split ratio was set to 0.75, meaning training set contains 75% of observations from original dataset, and testing set contains 25%.

- Feature Selection

In "Adult" dataset, the fourth feature "education" and the fifth feature "education_num" are representing the same information, therefore the "education" column was removed because number column was easier for calculation. The rest 13 features were all selected after series of attempts that remove different features from dataset all resulted in decrease in prediction accuracies.

- Modeling

In this project, logistic regression and support vector machine algorithms were utilized. While logistic regression implemented through customized programming, support vector machine, given its complexity of computing support vectors, was implemented by using "fitsvm" function provided in Matlab's Statistics and Machine Learning Toolbox.

- Benchmark

By using MLE to predict the target, which is to classify all observations into the same class that appeared most in dataset, all observations would be labelled as <= 50K. The accuracy of this method is 0.76, which can be considered as the benchmark of prediction accuracy for this project. Any classification accuracy produced by functions built in this project that beats this benchmark would be considered as having good performance.

Logistic Regression

There are three customized functions been built to implement logistic regression algorithm. "learnLogisticWeights" function takes in a starting classification vector $\mathbf{w}_0$, then utilized gradient decent to update (learn) its value for given number of iterations. "logisticClassify" function utilizes "sigmoidLikehood" function to compute likelihood that an input observation should be labelled as each class, and returns the class label that has higher probability.

To optimize learning process, different parameter settings were tested for updating the values of $\mathbf{w}$, including different figures of lambda, different gradient decent step sizes, and different amount of iterations. For all tests, "L1" and "L2" regularizations were also attempted along with normal logistic regression.

Support Vector Machine (SVM)

Support Vector Machine was also used for classification in this project. To implement SVM algorithm, Matlab's "fitcsvm" function provided by Statistics and Machine Learning Toolbox was utilized.

Similar to the tests on logistic regression, different parameters were attempted in order to find optimal set or configurations, including number of iterations, fraction of outliers to be excluded, and the box constraint.

Results

Logistic Regression

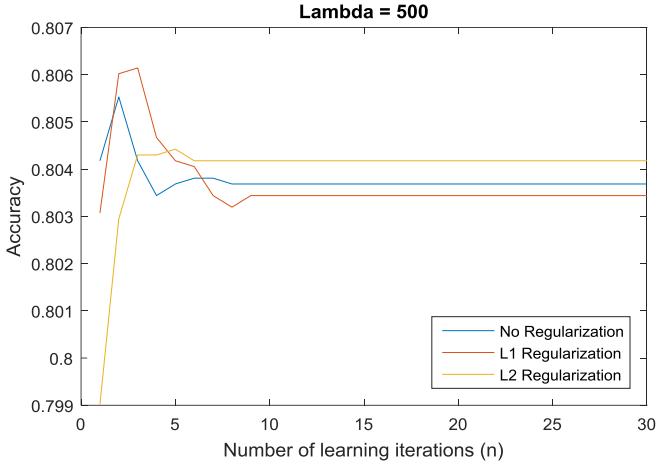
First parameter tested was number of iterations the function run to update weight vector **w**. To obtain a straightforward understanding of how different number of iterations affect testing accuracy, while testing other parameters, test set accuracy were always calculated and recorded from one iteration through thirty iterations of learning steps.

The second parameter tested was lambda, which was used in regularization process to limit **w** from getting too large. The smaller lambda is, the stronger effects it would cause on updating process. The results are shown in table 1, and the best result is illustrated in Figure 1.

Table 1. Change of Accuracy using different Lambda

Step Size	Lambda	Best N	L0 Accuracy	L1 Accuracy	L2 Accuracy
0.01	1	2	0.8055	0.4924	0.6576
0.01	500	3	0.8042	0.8061	0.8043
0.01	5000	2	0.8055	0.8055	0.8060
0.01	10000	2	0.8055	0.8057	0.8060
0.01	30000	2	0.8055	0.8057	0.8057

Figure 1. Change of Prediction Accuracy
Step Size = 0.01
N = 1:30
Lambda = 500



Based on the result, Lambda = 500 illustrated a slightly better accuracy than other values. Therefore the optimal lambda selected was 500.

With an optimal lambda found from previous step, the next parameter tested was the starting weight vector $\mathbf{w_0}$, the default setting was all zeroes, and then different range of random values were tested. To better compare the result, seed was set to 42 using rng(42) function in Matlab when generating random values. Results are shown in Table 2, and the best result is illustrated in Figure 2.

Table 2. Change of Accuracy using different w0

Step Size	Lambda	Best N	L0 Accuracy	L1 Accuracy	L2 Accuracy	w0
0.01	500	3	0.8042	0.8061	0.8043	All 0
0.01	500	3	0.8041	0.8059	0.8042	0 to 1
0.01	500	1	0.8061	0.8054	0.8002	-3 to 3
0.01	500	1	0.8063	0.8066	0.8026	-5 to 5
0.01	500	1	0.8060	0.8061	0.8039	-8 to 8

Optimal w0 using random values in range -5 to 5												
-1.255	4.507	2.320	0.987	-3.440	-3.440	-4.419	3.662	1.011	2.081	-4.794	4.699	3.324

As the result, random values in range -5 to 5 were selected as w_0.

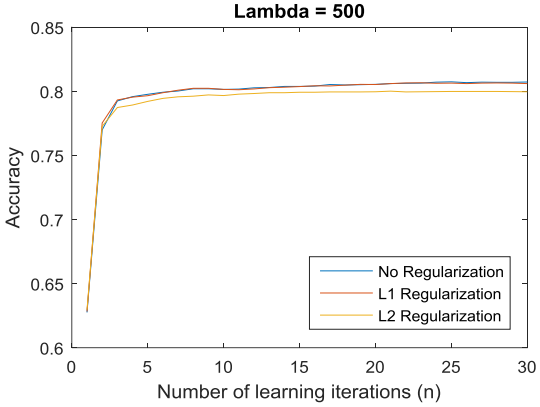
Last parameter tested was the step size used for gradient decent. Small step size may generate good result but takes longer to reach the optimal point. Results are recorded in Table 3, and the best result is illustrated in Figure 3.

Table 3. Change of Accuracy using different step size

Step Size	Lambda	Best N	L0 Accuracy	L1 Accuracy	L2 Accuracy	w0

0.01	500	1	0.80627	0.80663	0.80258	-5 to 5
0.05	500	3	0.78980	0.78686	0.77899	-5 to 5
0.005	500	5	0.80614	0.80651	0.80135	-5 to 5
0.001	500	25	0.80737	0.80639	0.79988	-5 to 5
0.0005	500	61	0.80639	0.80700	0.80012	-5 to 5

Figure 3. Change of Prediction Accuracy
Step Size = 0.001
N = 1:30
Lambda = 500



According to the result, step size of 0.001 was considered as the optimal option.

Support Vector Machine (SVM)

In order to get better result, while testing parameter sets, the following parameters were also customized:

KernelFunction: linear, nonlinear (rbf), formulas shown below:

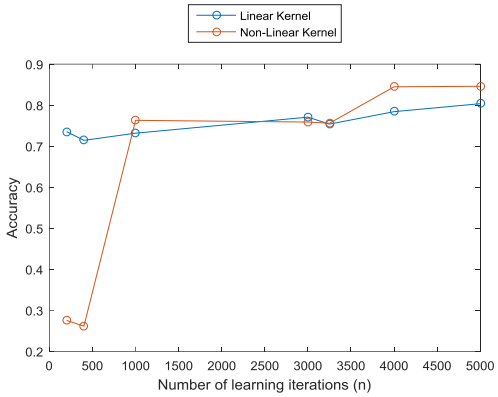
Value	Description	Formula
`'gaussian'` or `'rbf'`	Gaussian or Radial Basis Function (RBF) kernel, default for one-class learning	$G(x_1, x_2) = \exp\left(-\|x_1 - x_2\|^2\right)$
`'linear'`	Linear kernel, default for two-class learning	$G(x_1, x_2) = x_1' x_2$

Kernel Scale: 'auto', this requires a random number generating process, and to produce comparable results, the random seed was set to 42 every time "fitsvm" function was called.

Standardize: False, since the input data set has already been standardized through data preparation.

The first parameter tested was the number of iteration that Matlab would run to train the model. Although larger amount of iteration can raise the model's prediction accuracy, it cost longer time to compute. In order to find a good balance between number of iterations and time consumption, several number of iteration were tested, and the results were shown below.

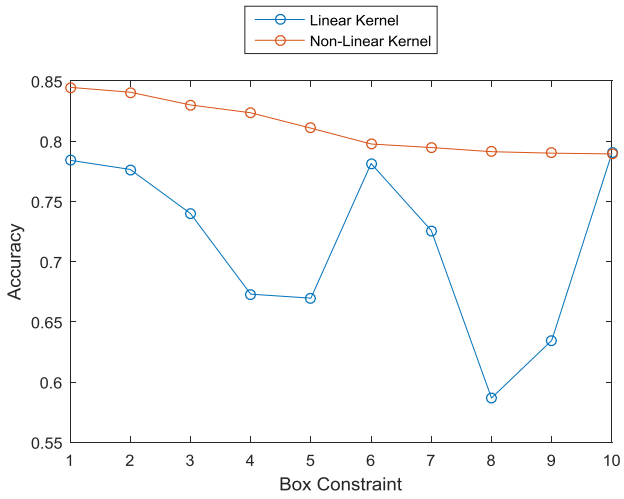
# of iteration	Linear Kernel Accuracy	Non-Linear Kernel Accuracy
200	0.735380835	0.275429975
400	0.714250614	0.261547912
1000	0.731695332	0.763022113
3000	0.770761671	0.758845209
3250	0.753808354	0.755896806
4000	0.784152334	0.844717445
5000	0.803562654	0.846068796



Interestingly, the result shows that nonlinear kernel performed extremely badly with small amount of iteration, but then rocketed to the same level as linear kernel, and eventually beat the performance of linear kernel method. Given the high time consumption of non-linear kernel method, 4000 iteration was selected even though higher amount of iterations may give higher accuracy.

The second parameter tuned was box constraint, which controls the maximum penalty imposed on margin-violating observations, therefore it is able to prevent overfitting. The larger box constraint is, the less support vector the function returns. Therefore it might reduce the complexity of classification model. Results are shown below:

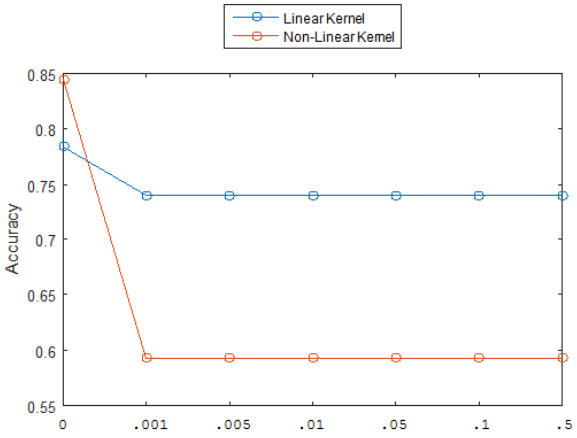
Box Constraint	Linear Kernel Accuracy	Non-Linear Kernel Accuracy
1	0.784152334	0.844717445
2	0.776535627	0.840663391
3	0.73992629	0.83009828
4	0.672972973	0.823587224
5	0.66965602	0.810810811
6	0.781449631	0.797665848
7	0.725798526	0.794717445
8	0.586732187	0.791277641
9	0.634520885	0.79004914
10	0.79004914	0.789434889



Seems increasing box constraint did not help improving the accuracy.

The third parameter tested was OutlierFraction, which means the fraction of observations that been considered as outliers and excluded from learning process. By tuning this parameter, some extreme outlier can be excluded from training process and hence may able to improve classification accuracy. Results are listed below:

Fraction of Outliers	Linear Kernel Accuracy	Non-Linear Kernel Accuracy
0	0.784152334	0.844717445
0.001	0.74017199	0.592751843
0.005	0.74017199	0.592751843
0.01	0.74017199	0.592751843
0.05	0.74017199	0.592751843
0.1	0.74017199	0.592751843
0.5	0.74017199	0.592751843



Similar with box constraint, the outlier fraction change shown no improvement in overall accuracy.

As the result, the best accuracies of each method are:

Logistic Regression: 0.80737

SVM: 0.844717445

Conclusion

In this project, both logistic regression and support vector machine algorithms were implemented to classify observations in "Adult" dataset. Both models were able to beat the benchmark after tuning, with similar amount of computing time. Scaling/Standardization and transforming dataset into numeric values had shown a great value in reducing computation time costs and raising classification accuracies.

While parameter tuning, logistic regression has shown some space for tuning, which started with low accuracy and was able to boost accuracy after some parameter changes. On the contrary, Support Vector Machine had shown good performance with default parameters, and shown little improvement with parameter tuning. This might be caused by optimization gap between professionally developed SVM function and the simple implement of logistic regression function used in this project.