Term Project Report for STAT745

STAT745 Term Project
Supervised Dataset: SPLICE

Dec 10, 2010

Abstract In this part of the project we are analyzing a supervised dataset called SPLICE. For our analysis we used 3 different types of learners: Linear regression, k-NN and a tree learner called M5. We first found the optimum parameters for k-NN and M5 by using LOOCV and the associated correct classification percentages. Then we tried to observe if variable selection could improve the performance. For variable selection, we used Wrapper algorithm and saw that for all learners the performance increased. However, the increase amount for k-NN was considerably better than the increase amount in other learners. Lastly, we wanted to observe if the testing strategy has an effect on the performance and we compared the performance values of LOOCV (leave-one-out cross validation) with variable selection to those of 10FCV (10 fold cross validation) with variable selection. We saw that linear regression and M5 are not affected by the testing strategy very much, whereas for k-NN 10FCV provided an increase of 10% in correct classification rate.

1 Introduction

Firstly I manually investigated the dataset to see the number of variables as well as instances and to see if there are any missing values. The SPLICE dataset consists of 3175 instances that are defined by 61 variables among which 1 is the class variable. The class variable can have 2 values, i.e. the problem we have is a two class problem. For each class (ei and ie) there are 1648 and 1527 instances, which means that we have a relatively balanced dataset in terms of class distributions.

My further analysis of the supervised data will consist of the following steps:

– ***Finding the optimum parameters:*** In that step, I will apply different learning algorithms on the SPLICE data and try out different parameter values to find the optimum parameters of each algorithm. The cross validation strategy adopted

Ekrem Kocaguneli
Lane Department of Computer Science and Electrical Engineering
West Virginia University
Morgantown, WV 26505, USA
E-mail: ekocagun@mix.wvu.edu

in this step will be leave-one-out cross validation (from now on LOOCV). In the following steps of the analysis, I will use the parameter values that I found in that step. Furthermore, in that step I will be able to observe the behavior of the learning algorithms.
- ***Affect of Variable Selection on Performance:*** The main aim of this step is to observe, if it is possible to increase the performance of the techniques applied on the dataset in the previous step. For that purpose, I will use a Wrapper algorithm, whose details will be provided in the related subsection.
- ***Affect of Cross Validation on Performance:*** In that step I will use 10-fold cross validation (from now on 10FCV) and I will compare the results of 10FCV to the results of LOOCV. I will try to interpret these results in terms of bias and variance.

For afore mentioned analysis steps, the learning techniques that were used are linear regression, k-nearest-neighbor (k-NN) and classification-and-regression-trees (M5P). All these algorithms are implemented as part of a data mining toolkit called WEKA, which I used for that part of the project. The only algorithm that needs clarification among the afore listed algorithms is probably M5P. It implements the algorithm that was invented and developed by R. Quinlan and Yong Wang, which is used as a classification tree in that project.

2 Results

2.1 Finding the optimum parameters

In that part I will be applying LOOCV to each algorithm and I will report the accuracy values to see what the optimum parameters are for the algorithms.

Linear regression does not have a parameter to optimize in the sense that k-NN does. With LOOCV, linear-regression correctly classified 2682 instances out of 3175, which means 84.47% of all the instances were correctly classified.

k-Nearest-Neighbor algorithm lets us use different number of neighbors to optimize. Since this is a two class problem, I will use odd number of neighbors, so that one of the two classes will have the majority in selected k nearest neighbors. The k values that I tried and the percentage of the correctly classified instances are given in table According to this table, k-NN classifier is relatively successful: Note that all the k values that I have tried yielded more than 75% accuracy. Furthermore, all the k values yielded quite similar results: Lowest accuracy was yielded by $k = 1$ with correct classification rate of 75.78%, whereas the highest accuracy was yielded by $k = 5$ wiht correct classification rate of 77.71%. Since $k = 5$ yielded the highest correct classification rate, I will be using $k = 5$ when comparing different learners. One caution that is needed to be done here is that for the

Another learner that I used is a tree learner, which is M5 [**?**]. The available parameter that I could use for that learner was the maximum amount of instances that are allowed to be in the leaf nodes. The amount of instances allowed to be in the leaf nodes has to do with the depth of the tree, i.e. the less number of instances one chooses to be on the leaf nodes, the deeper the tree is going to become. So I tried out different number of instances on the leaf nodes, which are 50,100,200,400,800. The tree that is built with at most 50 instances on the leaf node will be a deeper tree, when compared to the tree of 800 instances in the leaf node. Below in Figure 2, the deeper tree with at

k Value	Correctly Classified	Percentage
$k = 1$	2406	75.78
$k = 3$	2451	77.20
$k = 5$	2461	77.51
$k = 7$	2451	77.20
$k = 9$	2428	76.47
$k = 11$	2453	77.48

Fig. 1: The percentages of correct classifications for different k values. Although the values are very close to one another, the highest correct classification percentage is attained at *k=5*.

most 50 instances allowed in the leaves attained the higher prediction accuracy with 92.56% correct classification rate.

One important caution to be made regarding tree learner is that the run times are very high. That is understandable to some extent, because we are using LOOCV and we have a dataset size of 3175 instances, which means that we are building 3175 trees (one for each test instance).

Min. Instances In Leaf Node	Correctly Classified	Percentage
n = 50	2939	92.56
n = 100	2913	91.74
n = 200	2862	90.14
n = 400	2839	89.41
n = 800	2424	76.34

Fig. 2: Correct classification rates for different trees. Unlike k-NN, different trees perform much differently depending on how deep they are allowed to grow. For example, in this example the deepest tree is the one with at most 50 instances in the leaf nodes. We will use this tree as the optimum tree in our comparisons.

Now we have the optimum parameters for k-NN ($k = 5$) and for M5 (maximum instance in the leaves = 50) and we know the performance of linear regression, we can compare their performances. In Figure 3 we see the performance of learners with their optimum parameters. As can be seen from Figure 3, the best performance is attained by M5, which is a tree learner. In the learners that I choose, there is a regression-based (linear regression), a similarity-based (k-NN) and a rule-based learner (M5). When the performance values are taken into consideration, we can say that rule-based learner works better on SPLICE dataset. From best to worst the order of the 3 learners used here are: 1) M5, 2) linear regression and 3) k-NN.

Learner	Optimum Parameter	Correctly Classified	Percentage
Linear Regression	N/A	2682	84.47%
k-NN	$k = 5$	2461	77.51%
M5	Max. Instance Size = 50	2939	92.56%

Fig. 3: Comparison of algorithms with their optimum parameters according to LOOCV. With these settings adopted, the best performing algorithm for SPLICE dataset turns out to be M5. In other words, tree learner performs better than a linear regression based learner or a similarity-based learner.

2.2 Affect of Variable Selection on Performance

Now that we have the ordering for the methods and performance values according to LOOCV, we can observe if it is possible to improve these performance values (i.e. if we can better classify the data) by introducing variable selection algorithms. There are a bunch of different ways to select variables in a dataset such as principal component analysis (PCA), linear discriminant analysis (LDA) and Wrapper. Among these variable selection methods, PCA is actually a dimension alteration mechanism rather than a variable selection mechanism, in the sense that it alters an N-dimensional space to a D-dimensional space, where $N \geq D$. However, we can pick up the most important variables yielded by PCA and still use it as a variable selector. Wrapper on the other hand compares all the combinations of the variables according to a performance criteria and according to a given learner, which means that if we have n variables in a dataset there are $2^n - 1$ different combinations of variables to be compared.

Since, Wrapper considers all different combinations and give the combination that yielded the lowest error rate, I am going to use Wrapper for feature selection. Wrapper needs a base learner to compare different variable couplings. We have a relatively big dataset of 3175 instances and 60 independent variables to compare, so Wrapper will require a lot of CPU cycles. Therefore, we have better choose a fast running base learner. We have noted earlier that M5 required a long run-time, and linear regression attained a better performance rate than k-NN. Because of those reasons, I chose the linear regression as the base learner for Wrapper.

The attributes selected by Wrapper are given in Figure 4. Wrapper has selected 35 variables out of 60. The naming convention in Figure 4 is that v stands for variable and + indicates if it is a plus or a negative variable (there are 30 plus and 30 negative variables in SPLICE) and finally the number after + sign indicates the number of the variable. The next thing we will do will be to run our algorithms with the SPLICE dataset, where only the listed 35 variables will be used.

Selected Variables

v-26	v-13	v-6	v+2	v+14
v-20	v-12	v-5	v+3	v+16
v-18	v-11	v-4	v+4	v+18
v-17	v-10	v-3	v+5	v+21
v-16	v-9	v-2	v+6	v+27
v-15	v-8	v-1	v+7	v+28
v-14	v-7	v+1	v+10	v+29

Fig. 4: The variables selected by Wrapper.

The performance of the learners after variable selection is given in Figure 5. As can be seen, the variable selection improves all the algorithms. However, the improvement amount is questionable. For example for linear regression and M5, the improvement is almost negligable (less than 1%). On the other hand, the variable selection really helps k-NN and improves its performance by 3%. My explanation to that scenario is that k-NN uses the Euclidean distance, which is calculated via variable values, therefore if we use better explaining attributes then the Euclidean distance helps us more and hence results in better classification. So our summary of that experiment is that, variable

selection helps to improve the performance of a learner, but the amount of improvement depends on the learner.

Learner	Correctly Classified	Percentage Now	Percentage Before
Linear Regression	2688	84.66%	84.47%
k-NN	2547	80.22%	77.51%
M5	2933	92.66%	92.56%

Fig. 5: The number/percentage of correct classifications after variable selection and performance comparison before and after variable selection through Wrapper. Note that variable selection improved the performance in all cases. However, for linear regression and M5, the improvements are negligable, whereas for k-NN, the improvement is close to 3%.

2.3 Affect of Cross Validation on Performance

In this subsection, we will use the 3 algorithms with the optimum parameters. However, this time we will use 10FCV instead of LOOCV and we will try to observe how the performance or the order of the algorithms change. Depending on those observations, we will try to see if there is a bias towards a learner. In this setting, we will again use the variables that were selected by Wrapper in the previous setting.

In Figure 6 we see the performance of learners when subject to different testing strategies. It is surprising to see that linear regression and M5 perform almost exactly the same under LOOCV and 10FCV, whereas there is a considerable performance increase for k-NN. From that point, it is difficult to interpret as to why the performance improves about 10% for k-NN when we change the testing strategy, however, we can say that k-NN works better under 10FCV.

Learner	Percentage LOOCV	Percentage 10FCV
Linear Regression	84.66%	84.56%
k-NN	80.22%	92.34%
M5	92.66%	92.60%

Fig. 6: Comparison of performance between LOOCV and 10FCV.