ML Exercise 2 - Group 19

Classification Tasks

Lukas Stanek
Technical University of Vienna
Karlsplatz 13
1040, Vienna

Marten Sigwart
Technical University of Vienna
Karlsplatz 13
1040, Vienna
e1638152@student.tuwien.ac.at

—- Thomas Appler
Technical University of Vienna
Karlsplatz 13
1040, Vienna
—-

1. INTRODUCTION

Our task was to apply 4 different classification algorithms to 4 different data sets. Hereby, we had to experiment with different parameter settings of the classification algorithm, evaluate the performance by choosing appropriate performance measures and compare results among classifiers and data sets. We also had to perform some pre-processing steps on the data prior to applying the classifiers and study the impact of those pre-processing methods.

1.1 4 data sets

We chose four different data sets for the task. We chose data sets which varied in size, varied in the number of attributes and varied in the number of instances. Further we tried to pick data sets with different numbers of classes. Some data sets had missing values to take into account.

The four data sets we chose are:

- Letter Recognition
- Internet Advertisements
- Leukemia
- Congressional Voting

1.1.1 Data Set 1: Letter Recognition

TODO

1.1.2 Data Set 2: Internet Advertisements

TODO

1.1.3 Data Set 3: Leukemia

TODO

1.1.4 Data Set 4: Congressional Voting

TODO

1.2 4 classification algorithms

TODO

1.2.1 Algorithm 1: K-Nearest Neighbors

TODO

1.2.2 Random Forest: Random Forrest

TODO

1.2.3 Decision Tree - 48J: Decision Tree

TODO

1.2.4 Bayes Network: Bayes??

TODO

2. DATA SET 1: LETTER RECOGNITION

The objective for this data set is to predict a letter shown on a rectangular, black and white, display. There are 16 numerical features provided for determining the correct letter. These attributes provide information about statistical properties of the letter like total number of pixels, width, mean of x-axis pixels, etc. All of these values were then scaled into a range from 0 through 15. The 26 capital letters to predict in this set were taken from 20 different fonts. The letters are more or less evenly spread over the 20.000 instances, within a range of 734 to 813 occurrences. This data set contains no missing values. For comparable results 10-fold cross validation was used to measure the algorithms precision. As for performance criteria with this data set only the percentage of correctly classified instances will be considered. The reason for this being that if a letter is wrongly classified, it doesn't matter which other letter was recognized instead of the correct one.

2.1 Preprocessing

As the data set was only provided as CSV file without header row, we added a header row for displaying the attribute names in Weka and therefore facilitating the analysis of the results.

2.2 K-Nearest Neighbors

In order to be able to compare the results we always used the same number of neighbors: 1,2,3,4,5,6,8,10,16,20,40,100. The algorithm is generally very fast, even with 100 neighbors, it is usually finished within 30 seconds.

2.2.1 Default implementation

At first we started out using the default KNN implementation of Weka with default settings. The defaults are a *LinearNN* search algorithm with the *Eucledian distance* as distance function. We started out with increasing the amount of neighbors starting at one. Though with this settings we got the best results using just the nearest neighbor.

K	% Correct	% Wrong
1	95.96	4.04
2	94.925	5.075
3	95.635	4.365

.

In order to improve the results we started with weighting the neighbors after their distance which helped us further improving the results. The best results where achieved by weighting the neighbors *1/distance*. While the results yielded by the weighting function *1-distance* where also better than without weighting, the were not as good.

K	% Correct	% Wrong
1	95.96	4.04
2	95.99	4.01
3	96.04	3.96
4	**96.115**	**3.885**
5	96.03	3.97
6	96.015	3.985
8	95.75	4.25

.

With high values for K the prediction rates get worse.

K	% Correct	% Wrong
40	90.915	9.085
100	84.3	15.7

2.2.2 Manhattan Distance

did not improve our prediction, although again when the neighbors were weighted by their distance the success rate improved.

2.2.3 Chebyshev Distance

also did not improve the result, as it only takes the distance between the attributes which are farthest away from each other as result. As for weighting again, the results improved slightly when using weighted neighbors.

2.2.4 KD Tree Search Method

improves the run time of the classification significantly. The time for evaluating the models with 10-folds cross validation decreased by 66%.

2.3 Random Forest

The default settings for Random Forests are 100 Trees with unlimited depth, the number of randomly chosen features per tree is calculated by *log_2(#predictors) + 1*.

Number of Trees.

We started out with the default settings and varying the number of trees, starting at 100 trees. The result yielded was already at 96.41% accuracy. Further increasing the number of trees improved the result continuously, but also increased the computation time.

# Trees	% Accuracy	Computation time (s)
50	96.10%	3.1s
100	96.41%	6.55s
200	96.53%	12.33s
300	96.51%	19.13s
400	96.54%	24.46s
500	96.58%	30.66s

.

Though the performance is improving with the number of trees, the improvements are getting smaller. While an increase of trees from 50 to 100 trees results in a gain of 0.3 percent, the increase from 400 to 500 trees only gains 0.04 percent accuracy.

2.3.1 Increasing Execution Slots

For computing a 10-fold cross-validation at 500 trees on one single thread, the taken time was 5:55min. In order to improve this, the number of execution slots (=threads) can be increased. When repeating the same operation with 10 slots our computation time improved to 1:56min.
Using 50 threads we calculated a model with 1000 trees, which yielded the best result so far with an accuracy of *96.63%*.

2.3.2 Modifying number of features per tree

Through varying the number of features which will be selected per tree, we tried to increase the accuracy.
Results for *#Features: 3*:

# Trees	% Accuracy
100	96.51%
200	96.66%
300	96.78%
400	96.75%

2.3.3 Modifying tree depth

2.4 Decision Tree - 48J

TODO

2.5 Bayes Network

TODO

2.6 Conclusion

TODO

3. DATA SET 2: INTERNET ADVERTISEMENTS

TODO

3.1 Preprocessing

TODO what preprocessing was done

3.2 K-Nearest Neighbours

TODO

3.3 Random Forest

TODO

3.4 Decision Tree - 48J

TODO

3.5 Bayes Network

TODO

3.6 Conclusion

TODO

4. DATA SET 3: LEUKEMIA

TODO

4.0.1 Preprocessing

TODO what preprocessing was done

4.0.2 K-Nearest Neighbours

TODO

4.0.3 Random Forest

TODO

4.0.4 Decision Tree - 48J

TODO

4.0.5 Bayes Network

TODO

4.0.6 Conclusion

TODO

5. DATA SET 4: CONGRESSIONAL VOTING

TODO

5.0.1 Preprocessing

TODO what preprocessing was done

5.0.2 K-Nearest Neighbours

TODO

5.0.3 Random Forest

TODO

5.0.4 Decision Tree - 48J

TODO

5.0.5 Bayes Network

TODO

5.0.6 Conclusion

TODO

6. CONCLUSION

TODO