Heriot Watt University

03/12/
2016

Overfitting- Decision Trees & Neural Networks

DATA MINING AND MACHINE LEARNING

JOB VAN DIETEN – JV71 – H00153550

1 Data Generation

All data was generated using the command line commands, an example for the *J48* classifier:

```
java -cp weka.jar weka.Run J48 -t training_dir -x 10 -C pruning_confidence
```

and for the *Multi-Layer Perceptron* (*MLP*):

```
java -cp weka.jar weka.Run MultilayerPerceptron -t training_dir -x 10 -H 2
```

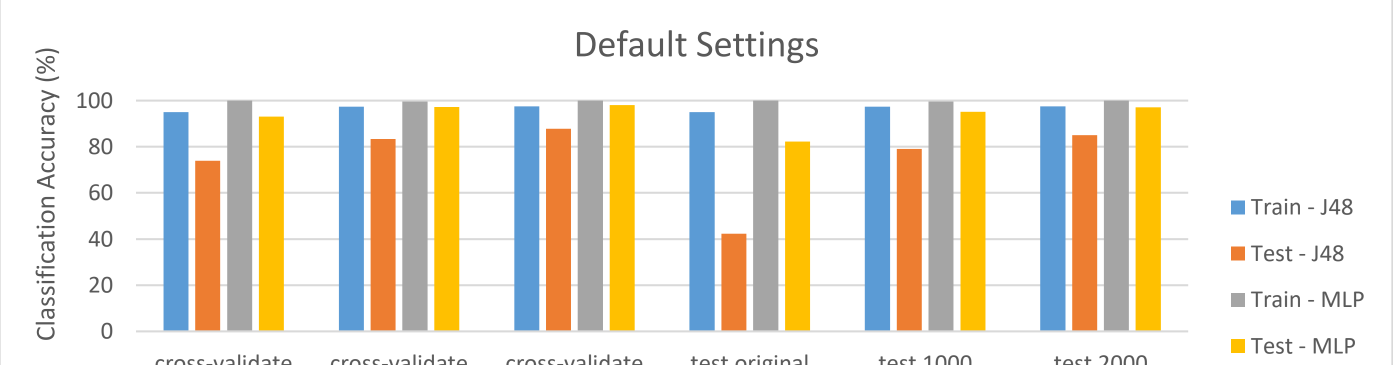
To generate and process the large amounts of data, Powershell Scripts[1] were used, which allowed the processing of large quantities of data to be done without human error. These output the results in a format allowing for quick insertion into excel for creating graphs. Three scripts were created, one for the data calculations of the *Decision Trees*, with a parameter input to determine the classifier used. The data is output to text files, the name of which includes the variable name and value that was tested. All data is ordered in folders for the different test and training sets. The same procedure is used for the *MLP*, for which a separate script was used due to the change in variables. The final script made reads the text files, stores the important information in arrays and outputs all information per training data size in a text file that can be copied into excel easily, and allows the generation

2 Variation in performance with size of the training set

The effect of the difference in size of the training set is the same for both Neural Networks and Decision trees. Figure 1 shows the variation in classification accuracy obtained, under default settings, for the *J48* classifier and the *Multi-Layer Perceptron*. As shown, both for cross-validation and testing on a test set, the accuracy of both classifiers increases when the training data size increases. From the perspective of overfitting, the increase in training data size correlates to an increase in accuracy on the test set, showing that the larger the training set, the less over-fitting occurs. One reason for this occurrence could be that the added instances in the training data set come from the test set, meaning that not only can the classifiers adapt to more noise examples, the actual test set contains less variables to test on. For this test set this is an unlikely reason, since the test set still contains more instances than the biggest training set. The suspected reason is that the increase in the training data set size creates more noise, which if it contains a pattern, will be easier to identify with more instances. If the noise is random, the additional instances will stop the classifiers from producing patterns that do not exist, but are found due to the small set of data in the original training set. The data sets represent handwritten digits, which will contain a large amount of noise, because handwriting is unique to most people.

3 Variation in performance with change in the learning paradigm (Decision trees versus Neural Nets)

When it comes to recognizing the handwritten digit, the Neural Network has parameters that allow it to achieve near perfect classifications. Figure 1 shows, under default settings, that the best classifications for both testing and training data sets are when the training data set contains 2000 instances. The cross-validated results at this data set was even higher, however since it is not tested on all possible data, the chosen best performance is a more realistic choice. The *J48* classifier performed with 97.6% and 85% classification accuracy on the training and data set respectively, however, the *MLP* performed with 100%



and 97.9%. The difference on the testing set is 12.5%, a significant difference. The *J48 Consolidated* classifier had a minimally higher accuracy, with 85.4% accuracy on the test set. This means that the *MLP* has a lower tendency to overfit data than a *Decision Tree* classifier. The reason the *MLP* gains a higher accuracy, is because it is a biologically inspired computation method, modelled after the synaptic connections in the human brain. The human brain demonstrates the ability in real life to learn to identify patterns with a lot of noise. The biological approach, with learning occurring over several epochs, makes it possible for the *MLP* to teach itself the training data set classifications with 100% accuracy. This result is not entirely indicative of a good result on the test set, however the approach explains the high accuracy gained. The *J48* and *J48 Consolidated* classifiers approach the data set from a machine perspective. Decision trees do not handle small perturbations in the data well, and can overfit the data sets. Compared to *Neural Networks*, it is, relatively, an ad hoc heuristic. *Decision Trees* advantage over *MLP*'s are in speed of calculations, which is a major weakness for the *MLP*, for although the accuracy is higher, the time taken to generate and teach the neurons is much longer than the construction of a decision tree.

4 Variation in performance with varying learning parameters in Decision Trees

The command used to run the *Decision tree* is:

java -cp *weka.jar weka.Run J48 -t training_dir -T test_dir*

The two chosen classifiers were *J48* and *J48Consolidated*. Both use the *C4.5* decision tree construction algorithm, however were *J48* utilizes the whole training data as one sample, whereas *J48Consolidated* samples several sets to create the decision tree. Both classifiers were run through the same set of variables, as shown in table 1 with, the set of samples for *J48Consolidated* set to 5. These permutations of the variables were used. When one variable was tested, the others were set to their default.

Variable	Range	Step size	Default
Pruning	Yes/No	-	Yes
Pruning Confidence	0.05-0.35	0.05	0.25
Binary Split	Yes/No	-	No

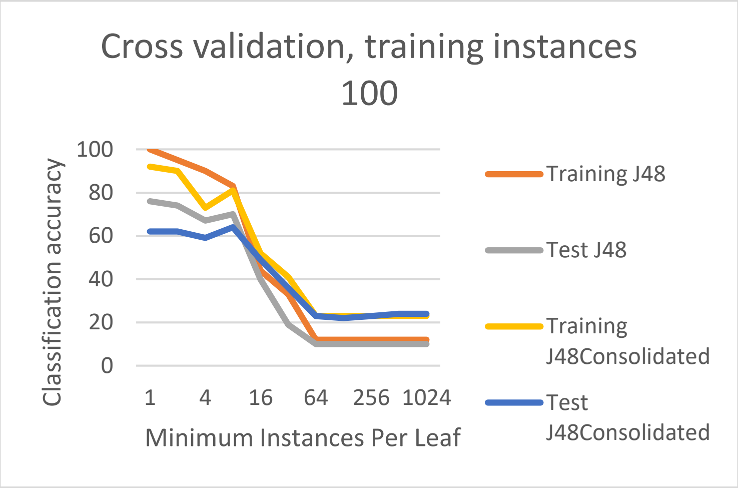
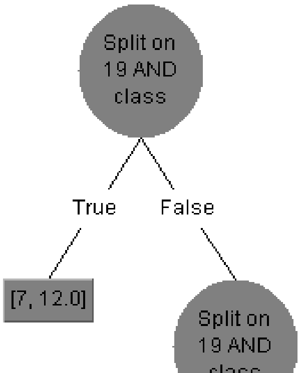


Figure 2: J48 vs J48Consolidated Results

The first thing noticed in the data produced is that 3 out of the 4 variables in table 1 have no significant effect on the classification accuracy. The variable that has a profound effect is the number of instances per leaf of the decision tree. Figure 2 shows that with a lower leaf amount, *J48* had a higher classification accuracy in both training and cross validation testing phases. However, as the number of instances per leaf increases, the accuracy hits a bottom of 10%, whereas *J48Consolidated* hovers around the 23%. For every training set size, the classification accuracy follows a pattern of decline with the decrease of minimum instances per leaf. From the perspective of overfitting, in the case of figure 2, both classifiers reduce the amount of overfitting with a higher minimum of instances per leaf. In the case of *J48Consolidated* the test and training errors start to equalize after 16 minimum instances per leaf, however for *J48*, the error



on training set and test set only merge after 64 minimum instances per leaf.

The nature of the data set, recognizing handwritten digits, may warrant a more customized approach to the classification. In this case Weka provides the *User Classifier* option, in which the user can construct a tree semi-manually. To do this the user needs to identify certain attributes as key to the identification of the digit. To do this, the correlation of the attributes was determined using the *CorrelationAttributeEvaluator*. This evaluation showed that for all datasets, some groups of attributes were less correlated than others. From this an attempt was made to create a custom classifier, as shown in Figure 3. Two class values were split from the attribute with the highest correlation to the class, and then a *J48* classifier was used for the rest class values. This resulted in only 40% of the instances being correctly identified. Using the *User Classifier* requires intricate knowledge of the dataset.

5 Variation in performance with varying learning parameters in Neural Networks

The command used to run the *MLP* is:

```
java -cp weka.jar weka.Run MultilayerPerceptron -t training_dir -T test_dir
```

The parameter ranges, shown in table 2, chosen for *learning rate, validation threshold,* and *epochs*, resulted in no significant variation in test set or training set classification accuracy. With an increase in epochs, a difference of maximum 0.2% was found, which was deemed insignificant. The trivial effect of epochs on the classification accuracy may be due to the source of data, or the efficiency of the back-propagation method.

Variable	Range	Step size	Default
Epochs	100-1900	200	500
Layers and nodes	2-5 hidden layers, 2-5 nodes per layer	1	1 hidden layer, 33 nodes
Learning rate	0-0.8	0.1	0.3
Momentum	0-1	0.1	0.2
Validation Threshold	10-80	10	0

change deemed necessary is then overridden by the previous weight update, and thereby wrong classifications are made, with the *MLP* only getting some correct classifications through some luck.

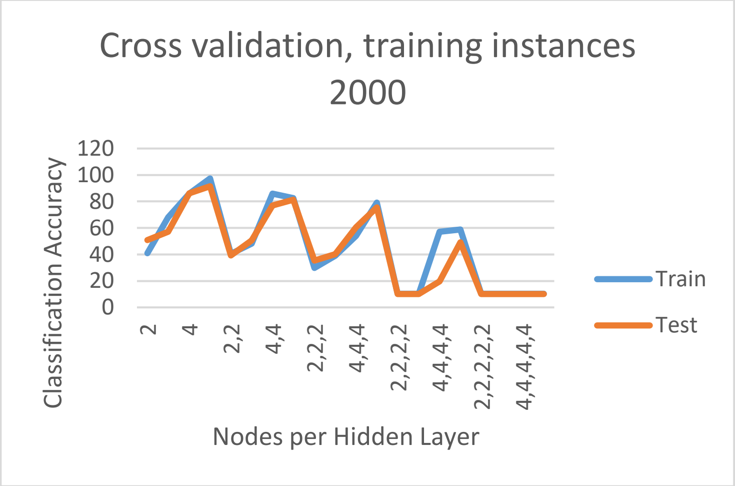


Figure 4: MLP Nodes per Hidden Layer Results

The parameters *hidden layer and nodes* make the most significant difference in the classification accuracy. Shown in figure 4, the classification accuracy reaches its peaks at 1 layer, with the highest result coming from the maximum number of nodes tested. The default value is determined by $\frac{classes+instances}{2} \approx 32.$ For this value the accuracy is 98.5%, 7% more than the maximum value from figure 4. This leads to the conclusion that a higher number of nodes in 1 hidden layer produces better classification results, although it is worth mentioning that the correlation of number of nodes to classification accuracy may be logarithmic in nature, approaching the maximum accuracy quickly. Therefore, if the application of the classifier is time-sensitive, to stop the *MLP* from taking too long to train, the ideal number of nodes may be lowered.

The *Neural Network* approach minimizes overfitting very well. Figure 4 shows that the testing classification accuracy is largely similar to the accuracy on the training data, the main exceptions being at 4 layers of 4 and 5 nodes. The optimum result, at the default values for all parameters, resulted in a difference in accuracy of 2.9% between the training and test set.

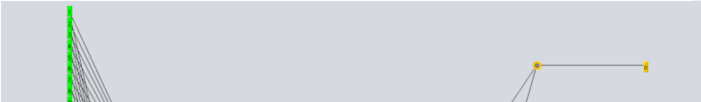


Figure 5 shows the *MLP* with 3 layers of 2 nodes each, displayed by the Weka Neural Network Toolbox. The GUI only shows the top half of the neural net, but the connections are clearly visible.

6 Variation in performance per different metrics

The metrics *True Positive Rate* (*TP Rate*), *False Positive Rate* (*FP Rate*), *Precision*, *Recall*, *F-Measure*, and *ROC Area* are mathematical representations of different perspectives on the classification accuracies. The weighted averages are a general description of how the entire data set faired in the classification, however may lead to incorrect assumptions about individual class value performance.

For example, table 3 shows the confusion matrix for the *User Classifiers* algorithm designed in part 4. It shows that the splitting the class values 7 and 8 resulted in these not being considered at all in the classification.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	398	29	54	0	4	15	10	0	0	0
1	0	322	49	14	73	19	27	0	14	0
2	4	46	311	47	1	16	73	0	8	0
3	0	4	9	361	1	53	81	0	17	0
4	49	31	22	1	225	108	47	0	25	0
5	2	0	52	33	4	35	234	0	150	0
6	0	180	25	5	2	11	288	0	5	0
7	15	137	109	68	73	58	41	0	12	0
8	1	24	52	150	5	79	87	0	108	0
9	5	5	16	81	38	213	33	0	116	0

Table 3:Confusion Matrix for User Classifier Decision Tree

In these cases, the classification diverted to other digits. The metrics for this data set show how deceptive the weighted average of the metrics can be when evaluating the performance of the classifiers, and by extension, the classification accuracy. The metrics table for the *User Classifier* show this, as displayed in table 4. This shows that the weighted average for the classification for the true positive rate, which is the classification accuracy, is 40%. This does not reflect the accuracy on individual classes very well, since the digit 0 was classified with an accuracy of 78%, and the digits 7 and 9 were not classified at all. A different metric could be chosen to better represent the performance of the classifier. *F-Measure* uses a

fact it is worth noting that in most cases the metrics were equal.

A purpose for which these metrics could be used, is if the purpose of the classifier is to only identify some digits with a high accuracy, where the rest of the class values are not important. In this case a classifier with a low average classification accuracy, but with a high *F-Measure* for the desired digit would be more suitable. This may cut down on computational resources needed.

Class	**TP Rate**	**FP Rate**	**Precision**	**Recall**	**F-Measure**	**ROC Area**
0	0.78	0.016	0.84	0.78	0.809	0.882
1	0.622	0.099	0.414	0.622	0.497	0.761
2	0.615	0.084	0.445	0.615	0.516	0.765
3	0.686	0.087	0.475	0.686	0.561	0.8
4	0.443	0.044	0.528	0.443	0.482	0.7
5	0.069	0.124	0.058	0.069	0.063	0.472
6	0.558	0.137	0.313	0.558	0.401	0.71
7	0	0	0	0	0	0.5
8	0.213	0.075	0.237	0.213	0.225	0.569
9	0	0	0	0	0	0.5
Weighted average	0.4	0.067	0.331	0.4	0.356	0.667

Table 4: Metrics Evaluation Data for User Classifier Decision Tree