# Heriot Watt University

Overfitting- Decision Trees & Neural Networks

Data Mining and machine learning

Job van dieten – jv71 – H00153550

01/12/2016

### 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 were used, which allowed the processing of large quantities of data to be done without human error.

### 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.

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

### Variation in performance with varying learning parameters in Decision Trees

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. One variables range was tested independent from the others, meaning no permutations of the variables were used. When one variable was tested, the others were set to their default.

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 1 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 1, 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 2

### Variation in performance with varying learning parameters in Neural Networks

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. The *momentum* parameter, which adds a fraction of the previous *epoch*’s weight update to the current one. The classification accuracy plummets from an average of 97% to 15% with a momentum shift between 0.8 and 1. The parameters *hidden layer and nodes* make the most significant difference in the classification accuracy. Shown in figure

### Variation in performance per different metrics (TP Rate, FP Rate, Precision, Recall, F Measure, ROC Area)

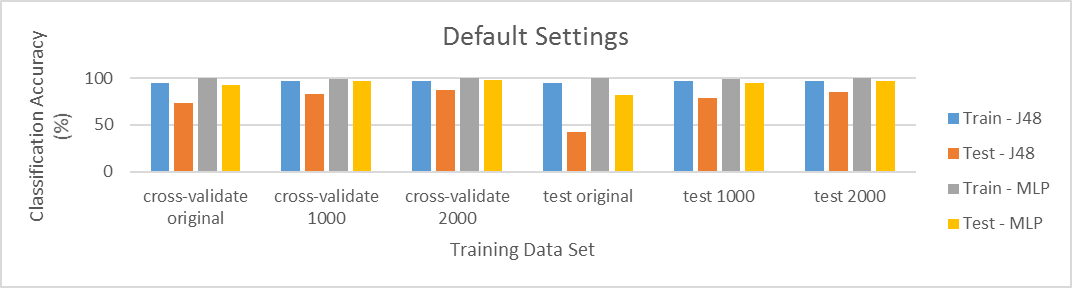


Figure 1:Different Training Set Size Results

Table 2: Variables for J48/J48Consolidated classifiers

Table 1: Variables for Multi Layer Perceptron classifier

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | Range | Step size | Default |
| Pruning | Yes/No | - | Yes |
| Pruning Confidence | 0.05-0.35 | 0.05 | 0.25 |
| Binary Split | Yes/No | - | No |
| Instances per Leaf | - | steps pf 1 | 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| 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,  65 nodes |
| Learning rate | 0-0.8 | 0.1 | 0.3 |
| Momentum | 0-1 | 0.1 | 0.2 |
| Validation  Threshold | 10-80 | 10 | 0 |

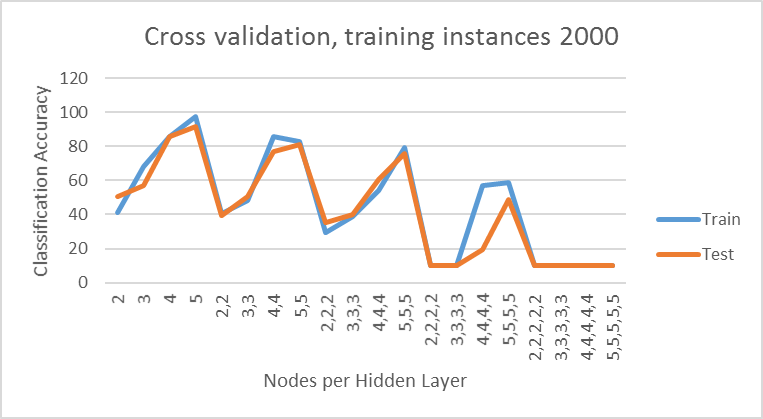


Figure 2: MLP Nodes per Hidden Layer Results

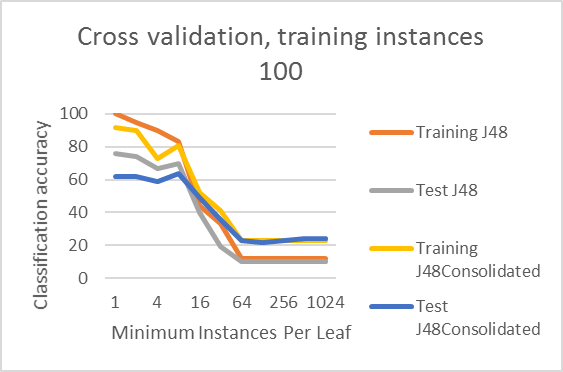


Figure 3: J48 vs J48Consolidated Results

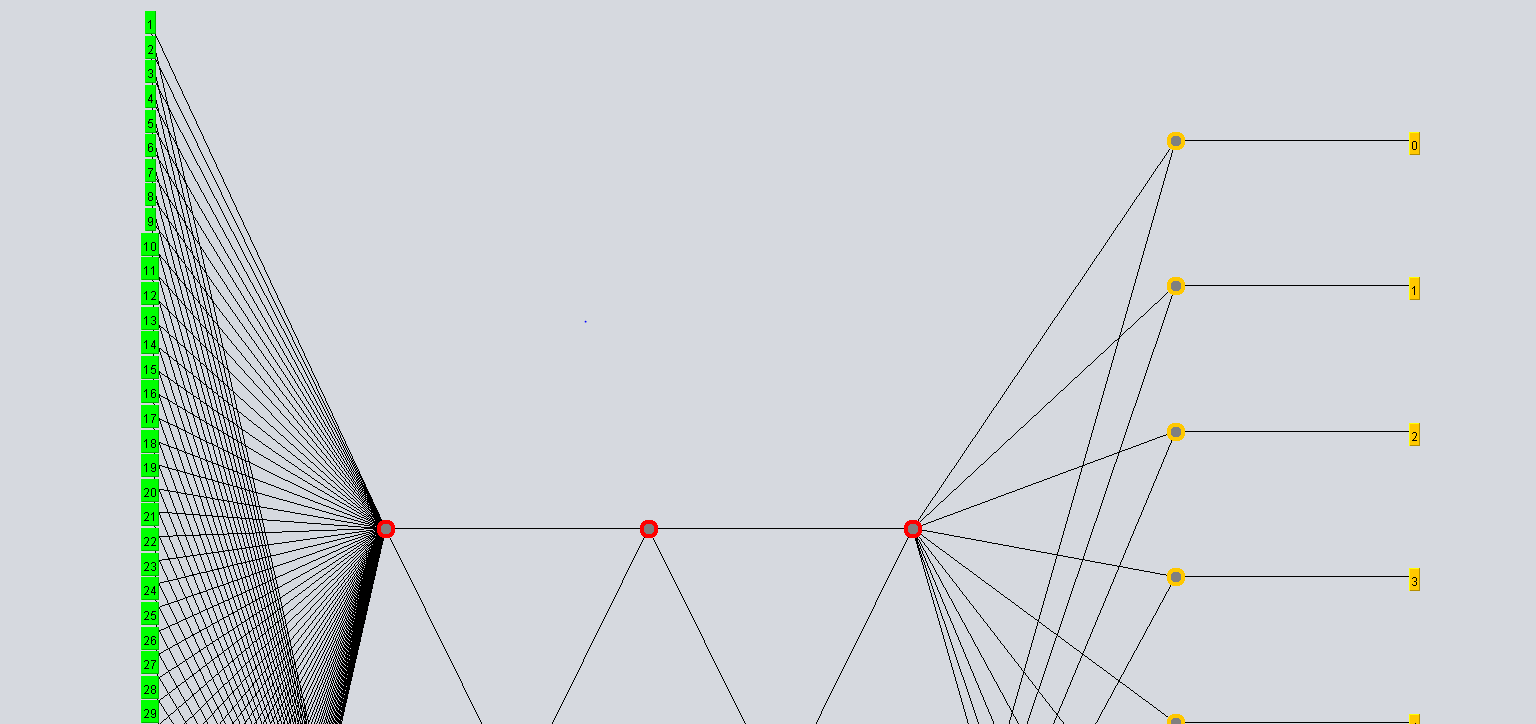


Figure 4: Top Half of MLP