CS4341 Introduction to Artificial Intelligence

# HW 4 A term 2013

Team: Bohao Li (bli) and Tushar Narayan (tnarayan)

Initial Experiments:

We imported the provided .arff file into Weka. On the Explorer window, we observed that the 10 difference attributes for the game of Connect-4 were listed.

To start off, we did not apply any filters on these attributes. After observing each attribute’s statistical information on the Preprocess tab, we opened up the Classify tab. For the Classifier, we chose the MultilayerPerceptron function. Under the Test Options section, we specified 5 folds for Cross-validation. In the drop-down for specifying the attribute to classify results by, we chose “(Nom) Winner”. We then ran the experiment. The results of this first experiment are:

Scheme:weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

Relation: connectFour

Instances: 80

Attributes: 10

FourInARow

UnblockedThreeInARow

HalfblockedThreeInARow

UnblockedTwoInARow

HalfblockedTwoInARow

SplitThreeInARow

TotalPieceValue

TurnCount

CurrentTurn

Winner

Test mode:5-fold cross-validation

… (truncated)…

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances 55 68.75 %

Incorrectly Classified Instances 25 31.25 %

Kappa statistic 0.3671

Mean absolute error 0.3225

Root mean squared error 0.5225

Relative absolute error 65.4954 %

Root relative squared error 105.3352 %

Total Number of Instances 80

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure ROC Area Class

0.711 0.343 0.727 0.711 0.719 0.769 1

0.657 0.289 0.639 0.657 0.648 0.769 2

Weighted Avg. 0.688 0.319 0.689 0.688 0.688 0.769

=== Confusion Matrix ===

a b <-- classified as

32 13 | a = 1

12 23 | b = 2

This was a very basic first experiment, to observe what the results would look like and to gain some experience with using Weka. However, with an accuracy of 68.75%, this was a very promising first try.

The next experiment we ran was to observe how the removal of attributes would contribute to or take away from the accuracy of the MultilayerPerceptron function. We removed three attributes, which left the following seven attributes:

1. FourInARow
2. UnblockedThreeInARow
3. HalfblockedThreeInARow
4. UnblockedTwoInARow
5. SplitThreeInARow
6. CurrentTurnWinner

We tested this model with 5-fold cross-validation. We observed 71.25% of instances were correctly classified.

We then removed the UnblockedTwoInARow attribute, which resulted in an accuracy of 72.5% on the training set.

After conducting a number of experiments with the removal and addition of different attributes from among the ten provided attributes, we eventually reached an accuracy of 76.25% by using the following eight attributes:

1. FourInARow
2. UnblockedThreeInARow
3. HalfblockedThreeInARow
4. SplitThreeInARow
5. TotalPieceValue
6. TurnCount
7. CurrentTurn
8. Winner

This implied that the attributes relating to two connections in a row - either unblocked or halfblocked - were not very important in determining the winner of a specific game. We postulated that this would be highest accuracy that we would find, considering that the eight features used seemed logically important to determining the accuracy of the prediction.

However, it turned out that we were able to reach 85% accuracy by removing the TurnCount and CurrentTurn attributes. Further, the best accuracy we were able to reach was 86.25%, was was observed when the FourInARow attribute was also removed. This was the attribute set with the highest accuracy.

# Attribute set with the highest accuracy:

We first selected the attributes manually based on our experience with the previous experiements. A few attributes are considered redundant:

1. Four in a row

As this attribute has the same value in all of the provided training sets, this attribute is deselected.

1. Turn count

It seems illogical to decide the result of the game based on the number of turns that have occurred.

1. Half blocked two in a row

This attribute can show little about the result of the game as it does not constitute any clear threats.

For a fivefold cross validation, deselecting these attributes increased the accuracy from 68.75% to 70%. After further experimentation, we deselected more attributes and below is our best solution for the highest accuracy of 86.25% using fivefold cross validation.

=== Run information ===

Scheme:weka.classifiers.functions.MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

Relation: connectFour-weka.filters.unsupervised.attribute.Remove-R4-weka.filters.unsupervised.attribute.Remove-R4-weka.filters.unsupervised.attribute.Remove-R7-weka.filters.unsupervised.attribute.Remove-R6-weka.filters.unsupervised.attribute.Remove-R1

Instances: 80

Attributes: 5

UnblockedThreeInARow

HalfblockedThreeInARow

SplitThreeInARow

TotalPieceValue

Winner

Test mode:5-fold cross-validation

=== Classifier model (full training set) ===

Sigmoid Node 0

Inputs Weights

Threshold -1.6239155109574668

Node 2 6.321944918882007

Node 3 8.071884101344423

Node 4 -3.959066541224748

Sigmoid Node 1

Inputs Weights

Threshold 1.6250668579678238

Node 2 -6.322158479395499

Node 3 -8.072075261754119

Node 4 3.95799108703256

Sigmoid Node 2

Inputs Weights

Threshold 1.1609256334559772

Attrib UnblockedThreeInARow -7.126200794482693

Attrib HalfblockedThreeInARow -11.263483099311143

Attrib SplitThreeInARow -4.869333082002804

Attrib TotalPieceValue 4.723306204631735

Sigmoid Node 3

Inputs Weights

Threshold -5.339553502707845

Attrib UnblockedThreeInARow -3.069795311011722

Attrib HalfblockedThreeInARow -1.2543555944565683

Attrib SplitThreeInARow -7.853221979075916

Attrib TotalPieceValue -7.1817851665989005

Sigmoid Node 4

Inputs Weights

Threshold 3.7436379664023245

Attrib UnblockedThreeInARow 4.415984812388216

Attrib HalfblockedThreeInARow 4.951659189793525

Attrib SplitThreeInARow -3.8553029973802557

Attrib TotalPieceValue -2.1122668936093714

Class 1

Input

Node 0

Class 2

Input

Node 1

Time taken to build model: 0.06 seconds

=== Stratified cross-validation ===

=== Summary ===

**Correctly Classified Instances 69 86.25 %**

**Incorrectly Classified Instances 11 13.75 %**

Kappa statistic 0.7197

Mean absolute error 0.2282

Root mean squared error 0.3579

Relative absolute error 46.3416 %

Root relative squared error 72.1489 %

Total Number of Instances 80

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure ROC Area Class

0.889 0.171 0.87 0.889 0.879 0.893 1

0.829 0.111 0.853 0.829 0.841 0.893 2

Weighted Avg. 0.863 0.145 0.862 0.863 0.862 0.893

=== Confusion Matrix ===

**a b <-- classified as**

**40 5 | a = 1**

**6 29 | b = 2**

In this neural network, there are five attributes:

1. UnblockedTreeInARow
2. HalfblockedThreeInARow
3. SplitThreeInARow
4. TotalPieceValue
5. Winner

Using tenfold cross validation, the accuracy dropped to 77.5%. However, after removing the TotalPieceValue attribute, the accuracy of tenfold cross validation rose to 80%, which was the highest accuracy that we obtained for the tenfold cross validation on the MultiplayerPerceptron.

We saved this model as “Neural Net Input for Weka-modified.arff”.

We also used various attribute filters to select the most valuable attributes using a more scientific approach. Below are the results we have thus far:

1. BestFirst+CfsSbusetEval:

UnblockedThreeInARow, HalfblockedThreeInARow, UnblockedTwoInARow, TurnCount

1. Ranker+ChiSquaredAttributeEval:

TurnCount(ranked 14.998),

HalfblockedThreeInARow(ranked 12.343),

UnblockedTwoInARow(ranked 9.858),

UnblockedThreeInARow(ranked 8.574)

1. Ranker + InfoGainAttributeEval:

TurnCount(ranked 0.16281),

HalfblockedThreeInARow(ranked 0.12687),

UnblockedTwoInARow(ranked 0.10743),

UnblockedThreeInARow(ranked 0.09165)

CurrentTurn (ranked 0.00207)

A very interesting result from these rankings is that the one attribute that we considered useless: TurnCount got the highest rank in all three types of filtering methods. Inspecting the result of the Information Gain Attribute Eval Ranking, we noted that TurnCount has an outstanding information gain of 0.16281, compared to CurrentTurn’s 0.00207, which we consider to be a promising attribute in the first place.

Further inspections reviewed that TurnCount performs very well if selected as the only used attribute in building the neural network.

Below is the result of neural networks built using single attributes, with the hidden layer field set to the default value of ‘a’.

|  |  |
| --- | --- |
| **Attribute** | **10-fold cross-validation accuracy** |
| TurnCount | 62.5 |
| UnblockedThreeInARow | 58.75 |
| HalfblockedThreeInARow | 61.25 |
| UnblockedTwoInARow | 67.5 |
| HalfblockedTwoInARow | 56.25 |
| SplitThreeInARow | 58.75 |
| TotalPieceValue | 53.75 |
| CurrentTurn | 46.25 |

We can see that TurnCount actually ranks the second in the list, right after UnblockedTwoInARow. We didn’t include either when selecting attributes by hand to build our neural network. Viewing the result we can conclude that when selecting the attributes, we have to take their coverage into account. FourInARow is a very powerful attribute, but it’s useless because no board has this attribute. UnblockedThreeInARow is also powerful, but it doesn’t provide us much information as only 9 out of boards 80 boards have this attribute. UnblockedTwoInARow, although not lethal, is owned by 25 out of 80 boards and all boards have a TurnCount value that’s greater than 0.

We proceeded to building a neural network using the most promising attributes as provided using the filter:

1. TurnCount(ranked 0.16281),
2. HalfblockedThreeInARow(ranked 0.12687),
3. UnblockedTwoInARow(ranked 0.10743),
4. UnblockedThreeInARow(ranked 0.09165)

We expected even better accuracy; however, the resulting accuracy in five-fold cross-validation is only 77.5 percent. In ten-fold cross-validation, the accuracy dropped to an embarrassing 72.5 percent, which demonstrates no improvement at all compared to the 72.5 percent with no attribute filtering.

We speculate that the value of individual attributes are important in building a neural network but the combination of different attributes still counts towards coming up with the best predictions. A neural network that has only the best attributes ranked by individual value might not perform as well as the ones that contains combinations of different attributes that perform much better together than the individual contributing attributes.

Going back to the table containing the accuracy of neural networks generated using only one attribute, we conducted another experiment: we deselected the least accurate attribute to generate a neural network using the remaining attributes and recorded the network’s accuracy.

Starting with all 10 attributes:

|  |  |
| --- | --- |
| **Most Recently Deselected Attribute** | **10-fold cross-validation accuracy** |
| None | 72.5 |
| FourInARow | 71.25 |
| CurrentTurn | 73.75 |
| TotalPieceValue | 78.75 |
| HalfBlockedTwoInARow | 78.75 |
| SplitThreeInARow | 77.5 |
| UnblockedThreeInARow | 71.25 |
| HalfBlockedThreeInARow | 63.75 |
| TurnCount | 67.5 |

We can see a general pattern that removing the low accuracy attributes can actually increase neural network accuracy, the top accuracy being 78.75 percent, only slightly lower than the known maximum accuracy of 80 percent.

This method of deselecting low value attributes can potentially be useful in building neural networks in the future.

Experimenting With Hidden Layers

1. Experimenting the number of hidden layers

Using the deselect least accurate attribute experiment result presented above, we decided to conduct the experiment on hidden layers using these attributes:

1. UnblockedThreeInARow
2. HalfblockedThreeInARow
3. UnblockedTwoInARow
4. HalfblockedTwoInARow
5. SplitThreeInARow
6. TurnCount
7. Winner

These attributes create a neural network with total accuracy of 78.75 percent using default settings. The default version contains one hidden layer and 4 hidden nodes.

|  |  |
| --- | --- |
| **Hidden Layer Setting** | **5-fold cross-validation accuracy** |
| 0 | 70 |
| a | 78.75 |
| a, a | 72.5 |
| a, a, a | 56.25 |
| a, a, a, a | 56.25 |
| a, a, a, a, a | 56.25 |

With one hidden layer the accuracy of the network increased, showing that one hidden layer is useful in making accurate prediction.

As we can see as we increase the number of hidden layers the accuracy of 5-fold cross-validation decreases. We speculate that the accuracy reduction results from overfitting the data.

An interesting fact is that the accuracy doesn’t decrease after the hidden layer number is above 3. We speculate that as the layers increase the effects generated by the initial data can no longer penetrate into the later layers, making the later layers redundant in terms of filtering information. It might also be related to Weka’s algorithms ignoring the extra hidden layers as they increase the complexity of the computation.

2. Experimenting the number of hidden nodes with one hidden layer

We later conducted an experiment with the effect on accuracy of the number of hidden nodes in one hidden layer.

|  |  |
| --- | --- |
| **Number of Hidden Nodes** | **5-fold cross-validation accuracy** |
| 0 | 70 |
| 1 | 70 |
| 2 | 73.75 |
| 3 | 77.5 |
| 4 | 78.75 |
| 5 | 78.75 |
| 6 | 80 |
| 7 | 80 |
| 8 | 78.75 |
| 9 | 78.75 |
| 10 | 80 |
| 20 | 78.75 |
| 30 | 78.75 |
| 50 | 78.75 |

From the result we observe that the accuracy increases with the number of hidden nodes in the first layer until the number of hidden nodes reaches 6. The accuracy then oscillates between 78.75 percent and 80 percent with no obvious differences.

The initial low accuracy is due to the underfitting of data. As the hidden nodes increase the data is better fitted. Further increase of nodes didn’t contribute much to reduce the error, nor did it increase the error.

We also observed that as the number of nodes increases, the time taken to train the neural networks increases as well. Although not reflected in this experiment, too many hidden nodes might overfit the data create large generalization error.

3. Experimenting the number of epochs

We conducted this experiment on the effect of the number of epochs to the accuracy of the neural network. We had the preconception that the more a neural network is trained, the more accurate it will be. Using the attributes used in the last two experiments and one hidden layer with 6 hidden nodes, we recorded the results:

|  |  |
| --- | --- |
| **Number of Epochs** | **Accuracy** |
| 1 | 56.25 |
| 10 | 63.75 |
| 50 | 67.25 |
| 100 | 71.25 |
| 200 | 78.25 |
| 300 | 78.75 |
| 400 | 78.75 |
| 500 | 80.00 |
| 800 | 78.75 |
| 1500 | 77.5 |
| 5000 | 71.25 |
| 100000 | 72.5 |

The accuracy of the neural network increased following the initial increase of the number of epochs. The increase rate of accuracy gradually reduced until the number of epochs reached 500, where we have the maximum accuracy.

Against our initial assumption, the accuracy value actually decreased as we continue to increase the number of epochs. The accuracy dropped to the value of 71.25 at 5000 epochs. To test this effect further, we provided the network 100, 000 epochs to train. It took a few minutes to get the result, but it’s not drastically different from the 71.25 percent of the 5000 epochs.

We conclude the the accuracy of the neural network will drop with the increase of epochs but will not drop below a certain level and will most probably be higher than trained with only a few dozen epochs.

The reason of the decline of accuracy is unknown. We speculate that as the neural network is trained for too many times, some unwanted error compounds which results in a deviation from the neural network’s optimal state.

Ensembling

We then proceeded to set up an experiment in Weka using the following two datasets:

1. Neural Net Input for Weka.arff

This was the provided .arff file containing all the ten attributes for the Connect-4 datasets.

1. Neural Net Input for Weka-modified.arff

This was the modified .arff file containing only the following four attributes that generated an accuracy of 80% with tenfold cross validation:

1. UnblockedThreeInARow
2. HalfblockedThreeInARow
3. SplitThreeInARow
4. Winner

We set up the experiment type to be cross-validation with 10 folds with the Classification option selected. For Iteration Control, the Number of repetitions was set to 1.

We wanted to ensemble the results; in order to accomplish that objective we used the following four algorithms:

1. MultilayerPerceptron
2. NaiveBayes
3. RandomForest
4. AdaBoostM1

The AdaBoost algorithm is one of the ensembling methods mentioned in the description of this project. Running the MultilayerPerceptron algorithm against the RandomForest allowed us to compare the generated neural network against a decision tree algorithm.

After running the experiment, we switched to the Analyse tab to examine the Test output. We tested with the corrected Paired T-Tester, with the significance set to the default of 0.05. On performing the test, we observed the following results:

Tester: weka.experiment.PairedCorrectedTTester

Analysing: Percent\_correct

Datasets: 2

Resultsets: 4

Confidence: 0.05 (two tailed)

Sorted by: -

Date: 10/3/13 11:17 PM

Dataset (1) function | (2) bayes (3) trees (4) meta.

----------------------------------------------------------------------

connectFour (10) 72.50 | 71.25 75.00 80.00

'connectFour-weka.filters (10) 80.00 | 73.75 82.50 78.75

----------------------------------------------------------------------

(v/ /\*) | (0/2/0) (0/2/0) (0/2/0)

Key:

(1) functions.MultilayerPerceptron '-L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a' -5990607817048210779

(2) bayes.NaiveBayes '' 5995231201785697655

(3) trees.RandomForest '-I 10 -K 0 -S 1' -2260823972777004705

(4) meta.AdaBoostM1 '-P 100 -S 1 -I 10 -W trees.DecisionStump' -7378107808933117974

We observed that ensembled AdaBoost algorithm definitely performed better than the neural network, while the decision tree (RandomForest) performed second best. We observed that NaiveBayes perfomed the worst when using all the ten attributes.

When using the modified version of the attributes, we see that the neural network performed better than the AdaBoost. It was interesting to observe that the RandomForest algorithm worked the best. However, we believe that that is because the RandomForest algorithm functions by constructing a number of different decision trees and then picks the best, which allows it to select the best attributes for the particular training data sets.

On averaging the decision tree and the neural network for the modified attribute set, we see that we observe an accuracy of 81.25%, which still performs better than the 78.75% accuracy of the AdaBoost algorithm. We cannot provide the weights of the attributes for this experiment since Weka does not display that information.