CITS3401 Data Exploration & Mining Project 2

Classifying Poker hands using Weka

Aleck Greenham 20362627
Ash Tyndall 20915779

27th May 2013

Introduction

This document details the analysis of Naïve Bayesian, C4.5 and Backpropagation Neural Network Classifications methods for the classification of Poker hands (modelled as permutations of 5 playing cards) as one of the nine well-defined classes. Where the analysis requirements [3] were ambiguous or incomplete, reasonable assumptions were made and documented.

Classification Method Selection

Classification methods were selected from those discussed in lectures based on how suitable each was for the classification necessary to correctly identify Poker hands. The in-program documentation of the classification methods (accessibly by right-clicking on the classifier name and selecting *properties* from the resultant context menu) was also taken into consideration.

Bayesian Classification

The Naïve Bayesian Classifier was chosen for comparison with the other classifiers used. It is based on Bayes' Theorem:

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

Where:

- $P(A|B)$ is the probability that an element in a set B is also in set A
- $P(B|A)$ is the probability that an element in a set A is also in set B
- $P(A)$ is the probability that an element from the universal set is also in set A
- $P(B)$ is the probability that an element from the universal set is also in set B

When applied to classification, $P(A|B)$ becomes the probability a record in set B, belongs to a particular class, A. Naïve Bayesian Classification calculates the probability of a record belonging to each of the available classes, and places it in the most probable class.

Record attributes are assumed to be independent to reduce the computational load in Bayesian classification. This is an appropriate assumption because each card has been split into two attributes - the value of the card and the suite - and therefore is almost entirely independent. The one exception is a hand with four of a kind; the fifth card's face value cannot be the same as the first four (because there are only 4 of each card denomination - one for each suite).

Decision Tree Classification

Decision tree classification algorithms iteratively divide records based on attributes selected to increase information gain, until a given threshold is reached. Decision tree classification algorithms differ in the way they select attributes for division.

Attributes are required to be categorical values for use with decision tree classification. This is suitable for classifying poker hands as the attributes are categorical values (card face value and suite).

Information Gain is a quantisation of the increase in information that can be achieved by subdividing a data set. It is therefore the objective of grouping the data to maximise information gain. Information gain for data set D, is defined as the difference in information or entropy before and after dividing the data:

$$Gain(A) = Info(D) - Info_A(D)$$

Where $Info(D)$ is the entropy of the entire data set:

$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

And $Info_A(D)$ is the information after a division A, into v different groups $(D_1, D_2 \ldots, D_v)$:

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

C4.5 (or the J48 implementation in Weka) was selected from the available decision tree classification methods. It maximises a quantity called *Gain Ratio*, a normalisation of Information Gain, to determine which attribute should be used to divide the data set to optimise information gain.

$$GainRatio(A) = \frac{Gain(A)}{SplitInfo(A)}$$

Where SplitInfo is defined as:

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \times \log_2(\frac{|D_j|}{|D|})$$

This iterative process of evaluation and division is stopped when all attributes have been used, each node belongs to the same class, or a pre-defined threshold has been exceeded.

Neural Network Classification

Backpropagation Neural Network Classifications (or the MultilayerPerceptron implementation in Weka) work by assigning attribute values weightings, which are continually refined by examining the error rates of recent classifications until a predefined threshold is reached. The weightings are adjusted to minimise the mean squared error between the neural network's predictions and the correct classifications.

There are 3 layers of the network: the input, hidden and output layers. Attributes are first fed into the input layer, allocated weightings and then fed into the hidden layer(s). The output values of the hidden layer(s) are are weighted and passed to the output layer. The weightings are calibrated backwards: starting with the output layer and moving through the model towards the input layer.

Implementation

Data Restructing

Weka best supports its own `ARFF` format and so it was necessary to convert the CSV training [1] and test [2] data. The tool linked to in the design document [4] timed out and produced error 500 when attempting to convert the large testing set. A substitute Python script was written, `csv_convertor.py`.

The documentation for the data attributes [5] states the “suit of card” and “class of hand” attributes are “ordinal”, yet the training data ARFF labelled all of its attributes as “numeric”. To better reflect the nature of the data and to provide improved classifcation results, the ARFF file was modified to define the card value attributes (C1 through C5) to be Weka’s “nominal specification” (Weka’s form of an enumeration).

The Python script mapped the suite attribute numbers to representative letter combinations to assist in the interpretation of the enumeration.

Creating Smaller Training Sets

The Python script `training_ratio_splitter.py` creates smaller training sets while still maintaining the ratios of each type of Poker hand present in the larger training set. The program counts the number of the lowest probable hand class, Royal Flushes, present in the larger training set and divides the full training set into smaller sets containing one Royal Flush each.

The ratio of classes is then calculated against Royal Flushes, and a segment of each class equivalent to that ratio is placed in each file. Because the training data set does not divide perfectly into integers, a small amount of card plays are discarded (36 of 25K).

Splitting Testing Set

`test_splitter.py` randomly splits the test data into a configurable amount of sets with a configurable amount of elements. Ten sets of 5000 elements were generated in this instance. To prevent any result ordering bias, the entire test set is randomised in memory and then exported into the specified number of elements in their own file.

Experiment 1: Comparison of 3 Algorithms For Classifying Poker Hands

Aim

The aim of the first experiment was to evaluate how effective each of the 3 classifiers are a categorising poker hands, as they formatted in the original training and test data sets. Each classifiers’ classification error rate was calculated in order to compare the efficacy of the different methods.

Methodology

The NaïveBayes, MultilayerPerceptron and the J48 algorithms were each trained using the complete training set `poker_hand/training` (approximately 25,000 records) and tested using 10-fold cross-validation.

Each algorithm was then re-evaluated against against 10 separate test-data files of 5000 Poker hands each and the results were averaged to attain an indication of how well the classifiers worked on non-training data.

Finally, the same algorithms were retrained against 3 small subsets of the same training data using 10-fold cross-validation to check for overfitting, or the generation of rules, conditions, or weightings due to noisy data that give poor classification performance. Where precision was significantly higher, overfitting was identified as having occurred in the larger training set.

Results

Naïve Bayes

Using the 10-fold cross-validation, the naïve bayes method achieved a correct classification rate of 49.07% and a root mean squared error of 0.24. The results shown Table 1 reveal all hands were classified as either being a hand worth nothing (0), or 1 pair (1).

The strong favouring for the lower valued hands can be explained by the large difference in the frequency in which hands worth nothing and single pairs occur (49.95% and 42.34% respectively), and the much rarer hands such as Straight Flush (SF) or Royal Flush (RF) (0.02% and 0.02% respectively). The Naïve Bayes method calculates the the most probable class for a hand as a function of the likelihood of each card being part of a hand that is in that class. The lower valued hands occur most often and are much more likely and thus all cards are more likely to belong to one of the many hands that belong to these lower classes. It does not allow for the improbable occasions where these cards belong to a more valuable hand.

The average correct classification rate for the 10 test sets was 49.24%, which is close to the value of the cross validation results. The average correct classification rate for the smaller training subsets was lower than that of the larger one at 47.95%, so no over-fitting occurred in the full training set.

Data Set	Correct Class. %	Mean Abs. error	RMS error	Weighted Avg. Precision	Weighted Avg. Recall
10-fold crossv	49.064	0.114	0.239	0.421	0.491
Test sets (avg)	*49.236*	*0.114*	*0.238*	*0.423*	*0.492*
testing/0	49.14	0.1139	0.2393	0.414	0.491
testing/1	50.16	0.1131	0.2377	0.437	0.502
testing/2	48.18	0.1135	0.2384	0.41	0.482
testing/3	49.24	0.1129	0.2371	0.418	0.492
testing/4	48.66	0.1135	0.2384	0.419	0.487
testing/5	49.72	0.1135	0.2385	0.432	0.497
testing/6	48.84	0.1135	0.2385	0.424	0.488
testing/7	49.84	0.1133	0.238	0.44	0.498
testing/8	49.52	0.1139	0.2392	0.416	0.495
testing/9	49.06	0.114	0.2394	0.416	0.491
Ratio Subsets (avg)	*47.946*	*0.114*	*0.240*	*0.429*	*0.480*
training/0	48.6795	0.1137	0.2402	0.439	0.487
training/1	47.2589	0.114	0.2405	0.42	0.473
training/2	47.8992	0.1138	0.2403	0.428	0.479

Table 1: Naïve Bayesian Classification for Poker hands for complete training set with class validation, 10 sets of test data and smaller training sets

C4.5 Decision Tree

The J48 algorithm showed a marked improvement above the Naïve Bayes on the 10-fold cross-validation with a correct classification rate of 57.54% and a root mean squared error of 0.25. Table

2 reveals was able to create rules that accommodate less likely hands such as 2 Pairs (2), 3 Pairs (3) and a Straight (S). However, it still failed to differentiate these from the more rare hands. This is not surprising for a classification method that still relies on probabilities (although they are "specialised" or weighted based on data subsets).

It is interesting to note a large dip in precision for two pairs (2). From the confusion matrix it can be seen these are often misclassified as single pairs (1). This may be because it is much easier to determine the presence of a single pair than it is to confirm the presence of 2 pairs.

The average correct classification rate of the test cases was slightly above that of the 10-fold cross validation with 57.72%. The small training subsets performed significantly worse than the full training set at 51.64%, confirming overfitting did not occur in the larger training set.

Data Set	Correct Class. %	Mean Abs. error	RMS error	Weighted Avg. Precision	Weighted Avg. Recall
10-fold crossv	57.54	0.10	0.25	0.54	0.58
Test sets (avg)	*57.716*	*0.099*	*0.246*	*0.540*	*0.577*
testing/0	57.06	0.0993	0.2472	0.524	0.571
testing/1	59.22	0.0969	0.2435	0.561	0.592
testing/2	57.14	0.0991	0.2463	0.536	0.571
testing/3	57.34	0.0988	0.2461	0.534	0.573
testing/4	58.82	0.0978	0.244	0.548	0.588
testing/5	58.24	0.0979	0.2439	0.547	0.582
testing/6	57.38	0.0992	0.2457	0.541	0.574
testing/7	57.68	0.0986	0.2457	0.544	0.577
testing/8	57.68	0.099	0.2467	0.533	0.577
testing/9	56.6	0.1	0.2496	0.53	0.566
Ratio Subsets (avg)	*51.627*	*0.108*	*0.248*	*0.477*	*0.516*
training/0	52.6811	0.1085	0.2464	0.493	0.527
training/1	51.5206	0.1081	0.2479	0.473	0.515
training/2	50.6803	0.1086	0.2493	0.465	0.507

Table 2: J45 Decision Tree Classification for Poker hands for complete training set with class validation, 10 sets of test data and smaller training sets

Neural Network

Table 3 shows Neural Networks achieved a surprising leap in correct classification rate with 93.43% and a root mean squared error of 0.07 when run using 10-fold validation, compared to that of the other two classification algorithms. This is probably due to the fact that calibrated attribute weightings can be much more sensitive to rare hands than probabilities.

Of note is the large increase in precision for 3 of a Kind (3), which may be it can be satisfied in relatively few ways, and involves only 3 cards. The precision for hands than involve 4 or more cards is significantly lower. For example, 2 pairs (2) is often misclassified as a single pair (1). This is also consistent with hands involving a smaller number of cards having a higher precision classification rate.

The execution time was considerably longer than that of the other two classification algorithms, yet in this case the huge increase in precision would justify the longer run time.

Data Set	Correct Class. %	Mean Abs. error	RMS error	Weighted Avg. Precision	Weighted Avg. Recall
10-fold crossv	93.427	0.010	0.075	0.903	0.934
Test sets (avg)	*94.360*	*0.008*	*0.063*	*0.899*	*0.944*
testing/0	93.68	0.0088	0.0662	0.888	0.937
testing/1	94.54	0.0078	0.0625	0.903	0.945
testing/2	94.58	0.0077	0.0614	0.905	0.946
testing/3	94.94	0.0073	0.0594	0.908	0.949
testing/4	94.58	0.0077	0.0619	0.903	0.946
testing/5	94.52	0.0079	0.0625	0.901	0.945
testing/6	94.64	0.0076	0.0617	0.904	0.946
testing/7	94.24	0.0082	0.0648	0.897	0.942
testing/8	94.16	0.008	0.0626	0.894	0.942
testing/9	93.72	0.0086	0.0657	0.887	0.937
Ratio Subsets (avg)	*94.805*	*0.007*	*0.056*	*0.903*	*0.948*
training/0	94.7979	0.0068	0.0559	0.903	0.948
training/1	94.7379	0.007	0.0568	0.902	0.947
training/2	94.878	0.0067	0.0546	0.904	0.949

Table 3: Neural Network Classification for Poker hands for complete training set with class validation, 10 sets of test data and smaller training sets

Conclusions

The MultiLayerPerceptron Neural Network algorithm is significantly better at classifying Poker hands than both the C4.5 Decision Tree algorithm, J48, and Naïve Bayesian algorithm, although the J48 delivers markedly better results than the Naïve Bayesian algorithm for approximately the same processing time.

Experiment 2: Investigating the Effect of Card Ordering On Classification

Aim

Cards appear in the training data files in random orders and while this does not affect the type of each Poker hand they are, it affects which attributes a particular card falls within. The aim of this experiment is to test whether this prevents the classifier methods from finding similarities between hands that have the same or similar cards, but in different orders.

Method

The training data from Experiment 1, which placed cards in each hand in random order, was ordered into two new configurations: ascending by card value, and by suite. Cards with the same card value were then sorted into suites in the first case, and those with the same suite were further sorted by card value in the second. Each of the classier algorithms in Experiment 1 was run again on the newly ordered training data using 10-fold cross-validation. The results of each algorithm were then compared to the corresponding results from Experiment 1. Where the performance of the classifiers was improved, it was confirmed that random ordering was preventing effective classifications of similar Poker hands in different orders.

Results

Naïve Bayesian

Table 4 reveals a significant increase in correct classification rate of Poker hands ordered by Card Value with 62.72%. Poker hands ordered by suite showed a much more modest improvement in classification performance with 49.38% over the original unordered hands with 49.06%.

Order	Correct Class. %	Mean Abs. error	RMS error	Weighted Avg. Precision	Weighted Avg. Recall
Unordered	49.064	0.114	0.239	0.421	0.491
By Card Value	62.715	0.092	0.217	0.617	0.627
By Suite	49.380	0.113	0.239	0.446	0.494

Table 4: Naïve Bayesian Classification for ordered and unordered Poker hands

C4.5 Decision Tree

An enormous improvement was seen in correct classification rate of C4.5 for Poker hands ordered by card value with 93.60%, compared to that of the unordered hands, 57.54%. Interesting, table 5 reveals that ordering by suite actually decreased the correct classification rate to 56.00%. This was probably because the sorting essentially standardised certain attribute values. For example, most hands would contain at least 1 heart, which is always placed at the start of the hand. This essentially gives the second attribute (the suite of the first card) the same value for all of the Poker hands, making it useless for distinguishing between records.

Order	Correct Class. %	Mean Abs. error	RMS error	Weighted Avg. Precision	Weighted Avg. Recall
Unordered	57.54	0.10	0.25	0.54	0.58
By Card Value	93.603	0.022	0.110	0.932	0.936
By Suite	55.990	0.106	0.232	0.532	0.560

Table 5: C4.5 Classification for ordered and unordered Poker hands

MultiLayerPerceptron Neural Network

Table 6 reveals a relatively small increase in correct classification rate over can be achieved by ordering the cards by card value 94.4% when compared to the classification rate for unordered hands, 93.43%. Ordering by suite again decreased the performance of the algorithm to 88.8%, down from 93.43%, which is a smaller decrease than C4.5 saw. This is again, probably due to the elimination of some attributes as distinguishable properties of different Poker hands.

Order	Correct Class. %	Mean Abs. error	RMS error	Weighted Avg. Precision	Weighted Avg. Recall
Unordered	93.427	0.010	0.075	0.903	0.934
By Card Value	94.422	0.008	0.063	0.944	0.944
By Suite	88.753	0.020	0.118	0.879	0.888

Table 6: MultiLayerPerceptron Classification for ordered and unordered Poker hands

Conclusion

Ordering Poker hands by card value improves the correct classification rates for all algorithms considered. The increase was particularly large in C4.5, but still present in Naïve Bayesian and MultiLayerPerceptron to lesser extents.

Ordering by suite shows little improvement in NaBayesian classification, but dramatically reduces the performance of C4.5 and MultiLayerPerceptron by inadvertently destroying information by making some attributes identical for most hands.

References

[1] Poker Hand Training Data, http://undergraduate.csse.uwa.edu.au/units/CITS3401/labs/poker-hand-training-true.data

[2] Poker Hand Test Data, http://undergraduate.csse.uwa.edu.au/units/CITS3401/labs/poker-hand-testing.data

[3] CITS3401 Data Exploration and Mining - Project 2, http://undergraduate.csse.uwa.edu.au/units/CITS3401/labs/proj2-2013.html

[4] Online CSV to ARFF conversion tool, http://slavnik.fe.uni-lj.si/markot/csv2arff/csv2arff.php

[5] Explanation of Poker Hand data attributes, http://undergraduate.csse.uwa.edu.au/units/CITS3401/labs/poker-hand.names

[6] Attribute-Relation File Format, http://www.cs.waikato.ac.nz/ml/weka/arff.html

Appendices

A Naïve Bayesian Confusion Matrix For Complete Training Data and Cross Validation On Randomly Order Poker Hand Training Data

```
=== Confusion Matrix ===

     a     b     c     d     e     f     g     h     i     j   <-- classified as
 11510   983     0     0     0     0     0     0     0     0 |     a = 0
  9838   761     0     0     0     0     0     0     0     0 |     b = 1
  1129    77     0     0     0     0     0     0     0     0 |     c = 2
   480    33     0     0     0     0     0     0     0     0 |     d = 3
    87     6     0     0     0     0     0     0     0     0 |     e = S
    52     2     0     0     0     0     0     0     0     0 |     f = F
    33     3     0     0     0     0     0     0     0     0 |     g = FH
     6     0     0     0     0     0     0     0     0     0 |     h = 4
     4     1     0     0     0     0     0     0     0     0 |     i = SF
     4     1     0     0     0     0     0     0     0     0 |     j = RF
```

B J48 Confusion Matrix For Complete Training Data and Cross Validation On Randomly Order Poker Hand Training Data

```
=== Confusion Matrix ===
     a     b     c     d     e     f     g     h     i     j   <-- classified as
 10100  2349    25     7     9     3     0     0     0     0 |     a = 0
  6207  4268    92    20     8     3     1     0     0     0 |     b = 1
   461   723    19     2     0     0     1     0     0     0 |     c = 2
   165   337     9     2     0     0     0     0     0     0 |     d = 3
    60    31     1     0     1     0     0     0     0     0 |     e = S
    45     9     0     0     0     0     0     0     0     0 |     f = F
     5    30     1     0     0     0     0     0     0     0 |     g = FH
     0     6     0     0     0     0     0     0     0     0 |     h = 4
     4     0     0     0     1     0     0     0     0     0 |     i = SF
     4     0     0     0     1     0     0     0     0     0 |     j = RF
```

C Neural Network Confusion Matrix For Complete Training Data and Cross Validation On Randomly Order Poker Hand Training Data

```
=== Confusion Matrix ===

     a     b     c     d     e     f     g     h     i     j   <-- classified as
 12453     2     0     0    30     8     0     0     0     0 |     a = 0
    18 10497    54    27     2     1     0     0     0     0 |     b = 1
     0  1185    21     0     0     0     0     0     0     0 |     c = 2
     1   133     0   379     0     0     0     0     0     0 |     d = 3
    84     1     0     0     8     0     0     0     0     0 |     e = S
    43     1     0     0     2     8     0     0     0     0 |     f = F
     0    33     0     3     0     0     0     0     0     0 |     g = FH
     0     0     0     6     0     0     0     0     0     0 |     h = 4
     3     0     0     0     2     0     0     0     0     0 |     i = SF
     5     0     0     0     0     0     0     0     0     0 |     j = RF
```

D Naïve Bayesian Classification For Ordered Poker Hands

Data Set	Correct Class. %	Mean Abs. error	RMS error	Weighted Avg. Prec.	Weighted Avg. Recall
10-fold crossv	62.715	0.092	0.217	0.617	0.627
Test sets (avg)	*62.312*	*0.093*	*0.218*	*0.614*	*0.623*
testing/0	63.18	0.0927	0.218	0.617	0.632
testing/1	63.3	0.0923	0.2172	0.628	0.633
testing/2	63.42	0.0916	0.2161	0.627	0.634
testing/3	61.6	0.093	0.2187	0.607	0.616
testing/4	62.22	0.0923	0.2176	0.615	0.622
testing/5	62.14	0.0928	0.219	0.607	0.621
testing/6	61.58	0.0934	0.2199	0.601	0.616
testing/7	61.58	0.093	0.2189	0.619	0.616
testing/8	61.66	0.0938	0.2207	0.601	0.617
testing/9	62.44	0.0917	0.2165	0.615	0.624

Table 7: Naïve Bayesian Classification for Poker hands ordered by card value for complete training set with class validation and 10 sets of test data

Data Set	Correct Class. %	Mean Abs. error	RMS error	Weighted Avg. Prec.	Weighted Avg. Recall
10-fold crossv	49.380	0.113	0.239	0.446	0.494
Test sets (avg)	*49.290*	*0.113*	*0.239*	*0.446*	*0.493*
testing/0	49.48	0.1135	0.2392	0.451	0.495
testing/1	49.88	0.1127	0.2378	0.451	0.499
testing/2	49.9	0.1129	0.2378	0.456	0.499
testing/3	49.38	0.1133	0.2386	0.443	0.494
testing/4	47.96	0.1132	0.2389	0.43	0.48
testing/5	49.12	0.1134	0.2392	0.443	0.491
testing/6	49.24	0.1133	0.2391	0.441	0.492
testing/7	48.8	0.1133	0.239	0.447	0.488
testing/8	49.4	0.1127	0.2377	0.448	0.494
testing/9	49.74	0.1128	0.2377	0.448	0.497

Table 8: Naïve Bayesian Classification for Poker hands ordered by suite for complete training set with class validation and 10 sets of test data