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 sub-dividing 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 [**?**] 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.

Data Import

Experimentation

Initial Experiment

For our intial experiment, we decided to use the NaiveBayes, MultilayerPerceptron and the J48 algorithm to test against our full sized training set `poker_hand/training`. For the experimental parameters, we elected to cross-validate 10-fold, and repeat the number of interations of the algorithms 10 times.

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
hand.names

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