Middle Tennessee State University
College of Basic and Applied Sciences
Spring 2014

CSCI 7350: Data Mining
Professor: Dr. Cen Li

Project 2
(*Homework 5 & 6*)
By: Zane Colgin

Due: March 4, 2014
last modified: March 2, 2014

Project Description

This project is based on two homework assignments. The first homework was based on learning Weka, the data mining software produced by the University of Waikato. The second homework was based on classification using the ensemble approach, the feature selection approach, and principle component feature reduction method. Both homeworks used the same data set to perform various experiments upon. Note that variance is inherent to any method that relies on a random number generator. For this reason, small variations in accuracies can sometimes be disregarded due to this experimentation only sampling the random process once for each test (i.e. a single seed is used).

- First, Weka was downloaded from:
  www.cs.waikato.ac.nz/ml/weka/index_downloading.html
  Weka 3.6.10 for Java 1.7 JVM was installed for OS X. The Weka documentation was also obtained through Weka's main web site:
  www.cs.waikato.ac.nz/ml/weka/index_documentation.html
- Weka extension (wekaclassalgo) for classification algorithms was downloaded from:
  wekaclassalgos.sourceforge.net/
- *Note:* Weka should be run through the extension to access additional classifiers from plug-in
  - navigate to the wekaclassalgo directory
  - `chmod 755 run.sh`
  - `./run.sh`

Data

Data was downloaded from the UC Irvine repository.

- Download **`glass.names`** and **`glass.data`**:
  archive.ics.uci.edu/ml/machine-learning-databases/glass/
- Format these files to the C4.5 standard.
- The ID feature was disabled from participating in the classification process by using the `ignore` keyword in the newly formatted `glass.names`.

Experiments (Homework 5)

All the experiments use 10-fold cross validation.

5.1 We applied Back Propagation Neural Network (BPNN) classification method on this data using Weka. By changing the parameters one at a time, we were able to construct an educated guess on the optimal Neural Network (NN) structure for this data. The parameters we chose for this experiment included:

(a) $\mathcal{L}$, the number of layers (*default* is 0)
(b) $\mathcal{N}_\ell$, the number of nodes for each layer $\ell \in \{1, 2, ..., \mathcal{L}\}$ (*default* is $0 \; \forall \; \ell$)
(c) $\mathcal{R}$, the learning rate (*default* is 0.1)
(d) $\mathcal{M}$, momentum (*default* is 0.2)
(e) $\mathcal{T}_i$, training iterations (*default* is 500)
(f) $\mathcal{T}_m$, training mode (*default* is batch training)

Note: The Sigmoid transerFunction was left unchanged. Best accuracy given in bold type and may be applied for each row of a table in some instances. Default parameter values in tables are also in bold type.

Experiment 5.1.ab: $\mathcal{L} \in \{0, 1, 2\}$ with various $\mathcal{N}_\ell$. The first table entry of the first table ($\mathcal{N}_1 = 0, \mathcal{N}_2 = 0, \mathcal{N}_3 = 0$) gives the accuracy with no hidden layers. The first row of the first table (excluding the first element) gives the accuracy of one layer with various numbers of nodes. The first table (excluding the first row) gives the accuracy of 2 layers with various nodes per layer. The different tables (excluding the first table) give the accuracy of 3 layers with various nodes per layer. All other parameters were left at default.

$\mathcal{N}_3 = \mathbf{0}$

$\mathcal{N}_1$ / $\mathcal{N}_2$	**0**	1	2	3	5	10
0	18.69%	**35.51%**	**35.51%**	**35.51%**	**35.51%**	34.58%
1	-	**35.51%**	**35.51%**	**35.51%**	**35.51%**	**35.51%**
2	-	**35.51%**	**35.51%**	**35.51%**	**35.51%**	**35.51%**

$\mathcal{N}_3 = 1$

$\mathcal{N}_1$ / $\mathcal{N}_2$	1	2	3
1	**35.51%**	**35.51%**	**35.51%**
2	**35.51%**	**35.51%**	**35.51%**

$\mathcal{N}_3 = 2$

$\mathcal{N}_1$ / $\mathcal{N}_2$	1	2	3
1	**35.51%**	**35.51%**	**35.51%**
2	**35.51%**	**35.51%**	**35.51%**

The best result with the minimum $\mathcal{L}$ and minimum $\mathcal{N}_\ell$ were found to be $\mathcal{L} = 1$ with $\mathcal{N}_1 = 1$. We will call the set of parameters with $\mathcal{L} = 1, \mathcal{N}_1 = 1$, and all the remainder left at default, opti($\mathcal{L}, \mathcal{N}_\ell$).

Experiment 5.1.c: $\mathcal{R} \in \{0.05, 0.1, 0.2, 0.31, 0.4, 0.5, 0.6, 0.75\}$. The first row of the table gives the accuracy obtained at the specified learning rate with all other parameters left at default. The second row gives the accuracy obtained at the specified learning rate with opti($\mathcal{L}, \mathcal{N}_\ell$). We report $\mathcal{R} = 0.31$ rather than $\mathcal{R} = 0.3$, because it gave better accuracy.

$\mathcal{R}$	0.05	**0.1**	0.2	0.31
default	23.36%	18.69%	20.56%	**33.18%**
opti($\mathcal{L}, \mathcal{N}_\ell$)	**35.51%**	**35.51%**	**35.51%**	34.58%

$\mathcal{R}$	0.4	0.5	0.6	0.75
default	30.84%	27.10%	24.30%	18.22%
opti($\mathcal{L}, \mathcal{N}_\ell$)	28.97%	**35.51%**	35.05%	**35.51%**

The best result of learning rate with default values for the other parameters was given when $\mathcal{R} = 0.31$; however, when opti$(\mathcal{L}, \mathcal{N}_\ell)$ were used as parameters, the learning rate did not impact the accuracy as much. In fact, the accuracy was slightly hurt by using this as the learning rate. We will call the set of parameters with $\mathcal{R} = 0.31$ and all the remainder left at default, opti$(\mathcal{R})$.

Experiment 5.1.d: $\mathcal{M} \in \{0.0, 0.1, ..., 0.9\}$. The first row of the table gives the accuracy obtained at the specified momentum with all other parameters left at default. The second row gives the accuracy obtained at the specified momentum with opti$(\mathcal{L}, \mathcal{N}_\ell)$. The third row gives the accuracy obtained at the specified momentum with opti$(\mathcal{R})$.

$\mathcal{M}$	0.0	0.1	**0.2**	0.3	0.4
default	19.63%	21.50%	18.69%	24.77%	29.91%
opti$(\mathcal{L}, \mathcal{N}_\ell)$	**35.51%**	**35.51%**	**35.51%**	**35.51%**	**35.51%**
opti$(\mathcal{R})$	14.95%	23.83%	**33.18%**	24.30%	29.44%

$\mathcal{M}$	0.5	0.6	0.7	0.8	0.9
default	27.10%	24.77%	**33.18%**	24.30%	20.56%
opti$(\mathcal{L}, \mathcal{N}_\ell)$	**35.51%**	**35.51%**	**35.51%**	**35.51%**	**35.51%**
opti$(\mathcal{R})$	24.30%	29.44%	32.71%	29.44%	28.50%

The best result of momentum with default values for the other parameters was given with $\mathcal{M} = 0.7$. We will call the set of parameters with $\mathcal{M} = 0.7$ and all of the remainder left at default, opti$(\mathcal{M})$. When opti$(\mathcal{L}, \mathcal{N}_\ell)$ were used as parameters momentum did not impact the accuracy at all. When opti$(\mathcal{R})$ was used as a parameter the best accuracy was given with $\mathcal{M} = 0.2$, though this value was very close to the accuracy obtained with opti$(\mathcal{M})$. When we used opti$(\mathcal{L}, \mathcal{N}_\ell)$, opti$(\mathcal{R})$, and opti$(\mathcal{M})$, we were unable to increase the accuracy past 35.51%.

Experiment 5.1.e: $\mathcal{T}_i \in \{500, 1000, 5000\}$ The first row of the table gives the accuracy obtained at the specified training iterations with all other parameters left at default. The second row gives the accuracy obtained at the specified training iterations with opti$(\mathcal{L}, \mathcal{N}_\ell)$. The third row gives the accuracy obtained at the specified training iterations with opti$(\mathcal{R})$. The fourth row gives the accuracy obtained at the specified training iterations with

opti($\mathcal{M}$).

$\mathcal{T}_i$	500	1000	5000
default	**18.69%**	**18.69%**	**18.69%**
opti($\mathcal{L}, \mathcal{N}_\ell$)	**35.51%**	**35.51%**	**35.51%**
opti($\mathcal{R}$)	33.18%	33.18%	**34.11%**
opti($\mathcal{M}$)	**33.18%**%	**33.18%**%	31.31%

Once again, opti($\mathcal{L}, \mathcal{N}_\ell$) were the dominant parameter values. Training iterations had little impact on this data set. Note that with opti($\mathcal{R}$) we also tested $\mathcal{T}_i = 15000$; however, this did not improve accuracy.

Experiment 5.1.f: $\mathcal{T}_m \in \{\text{batch}, \text{online}\}$ The columns of the table indicate the set of parameters that were used. The rows indicate the training mode.

$\mathcal{T}_m$	*default*	opti($\mathcal{L}, \mathcal{N}_\ell$)	opti($\mathcal{R}$)	opti($\mathcal{M}$)
batch	18.69%	**35.51%**	34.11%	**35.51%**
online	10.28%	**35.51%**	34.11%%	34.11%

Once again, opti($\mathcal{L}, \mathcal{N}_\ell$) were the dominant parameter values. Training mode had little impact on this data set.

Our conclusion is that for this data set, layers and hidden nodes play the most important role in constructing a predictive NN. Leaving all other parameters alone we can add one layer with one hidden node and get the "optimal" accuracy using this classifier.

5.2 Naïve Bayes classification method obtained the following results:

```
Correctly Classified Instances         106          49.5327 %
Incorrectly Classified Instances       108          50.4673 %
Kappa statistic                          0.334
Mean absolute error                      0.1521
Root mean squared error                  0.3343
Relative absolute error                 71.8506 %
Root relative squared error            102.9939 %
Total Number of Instances              214
```

5.3 Decision Tree (J48) obtained the following results:

```
Correctly Classified Instances         141          65.8879 %
Incorrectly Classified Instances        73          34.1121 %
Kappa statistic                          0.5412
Mean absolute error                      0.1059
Root mean squared error                  0.2928
Relative absolute error                 50.0098 %
Root relative squared error             90.2088 %
Total Number of Instances              214
```

5.4 REPTree with numFolds = 2 as the classifier obtained the following results:

```
Correctly Classified Instances         136          63.5514 %
Incorrectly Classified Instances        78          36.4486 %
Kappa statistic                          0.492
Mean absolute error                      0.1283
Root mean squared error                  0.2761
Relative absolute error                 60.5886 %
Root relative squared error             85.0846 %
Total Number of Instances              214
```

Note that bagging with 30 iterations and REPTree as the classifier we got 74.30% accuracy. Since this method uses REPTree as the base classifier we decided to include REPTree as the fourth classifier. We will include Bagging in with experiment 6.4.

Experiments (Homework 6)

6.1 For this experiment, we applied 3 different attribute selection algorithms on the glass data and determined subsets of attributes to use for classification. The rank and order of the attributes determined by these algorithms are included in the table below.

Relief		ChiSquared		InfoGain	
(3) Mg	0.21352	(6) K	187.6856	(4) Al	0.5662
(4) Al	0.07139	(4) Al	172.0743	(3) Mg	0.5628
(8) Ba	0.06184	(8) Ba	166.3236	(6) K	0.543
(7) Ca	0.04847	(3) Mg	147.924	(7) Ca	0.472
(2) Na	0.04694	(7) Ca	146.9687	(8) Ba	0.4124
(1) RI	0.03973	(2) Na	117.2646	(2) Na	0.3346
(5) Si	0.02634	(1) RI	90.6882	(1) RI	0.3323
(6) K	0.02288	(9) Fe	20.3892	(9) Fe	0.0991
(9) Fe	0.0077	(5) Si	0	(5) Si	0

The rankings between the different methods don't completely agree with each other, though there is some correlation. Using these rankings we will choose two subsets of features for classification in the next experiment. For this project requirement we will choose a small subset (three attributes) and a larger subset (six attributes).

Subset 1: (4) Al, (8) Ba, (3) Mg

Subset 2: (4) Al, (8) Ba, (3) Mg, (7) Ca, (6) K, (2) Na

6.2 We applied the four classification methods used in experiments 5.1 - 5.4 on the new data with the subset of features we chose in experiment 6.1. The table below compares the classification performance side-by-side with results from experiments 5.1 - 5.4.

	Full Set	Subset 1	Subset 2
BPNN (with opti($\mathcal{L}, \mathcal{N}_\ell$))	35.51%	48.13%	35.51%
Naïve Bayes	49.53%	46.26%	51.87%
J48	65.89%	68.22%	67.29%
REPTree	63.55%	64.95%	63.08%

We observe that subsets of attributes can have a meaningful affect on classification. Subset 1 was able to substantially improve the accuracy of BPNN on the data set. Our conclusion is that the

selection of these attributes is very important, and more testing should be done in order to find the best subset.

6.3 We applied principle component analysis (PCA) on the original glass data. PCA transforms the data to K features where K is smaller than the dimension of the original data. We applied the four classification methods used in experiments 5.1 - 5.4 on new data with the transformed features, and compared the classification performance side-by-side with previous results.

	Full Set	$K = 5$	$K = 2$
BPNN (with opti($\mathcal{L}, \mathcal{N}_\ell$))	35.51%	57.94%	57.94%
Naïve Bayes	49.53%	51.40%	55.14%
J48	65.89%	70.56%	64.95%
REPTree	63.55%	61.21%	63.08%

We cannot conclude that using PCA will always help accuracy. However, we were able to obtain the best results for BPNN found so far in this experimentation. In general PCA did help accuracy, and it should be considered for further experimentation.

6.4 This experiment compares the base classifier and ensemble classifier performance on the original glass data. We used the Meta (classifier) - Adaboost method and the Bagging method. Within these methods we use each of the four classification methods used in experiments 5.1 - 5.4 as the base classifier. We compare the classification performance side-by-side with results from experiments 5.1 - 5.4.

	Base	Adaboost	Bagging
BPNN (with opti($\mathcal{L}, \mathcal{N}_\ell$))	35.51%	35.51%	35.51%
Naïve Bayes	49.53%	49.53%	50.93%
J48	65.89%	**79.44%**	73.83%
REPTree	63.55%	70.56%	74.30% (30 iters)%

We should note that Bagging with BPNN took the longest execution time of all tests. BPNN was not helped by either ensemble method. Naïve Bayes was only slightly helped by bagging. The tree based base classifiers performed very well with the two ensemble methods. Adaboost with J48 as the base classifier was the most predictive of all models constructed. Furthermore, we

tested PCA with $K = 5$ on the data and Adaboost with J48 as the base classifier and obtained a worse accuracy of 70.56%.

Conclusions

We conclude that there are many different methods to create classification models using training data. There are data attribute preprocessing methods such as PCA. There are attribute selection methods that play a role in determining the most desirable attributes to include in the model. There are base classifiers based on neural networks, trees, Bayes theorem, etc. There are ensemble classifiers that make use of the base classifiers. The different components of these methods all rely on their own sets of variables making it very difficult to determine the best solution. The only thing that seemed to be consistent with the obtained accuracies were the inconsistencies. For example, J48 worked well with PCA but only with $K = 5$ not $K = 2$. J48 worked well with Adaboost but not Adaboot and PCA with $K = 5$. BPNN had parameters values that worked well individually (meaning the remainder of the parameters were left at default value), but did not work well with other parameters that worked well. Every time there seemed to be a trend in improving accuracy, when we followed that trend through experimentation, the improvement quickly turned into a detriment. We must also realize that we are dealing with a limited sized training set, which introduces variance in the predictive capabilities of our model on data from outside of this set. Some methods, as well as the 10 fold cross validation, rely on a random process that is only sampled once for each experiment. Any results governed by a random process with unknown variance and has not been sampled many times should be taken lightly.