CSCI 447 Assignment 1 Report

Clint Cooper and Emily Rohrbough

***Abstract*—WEKA is a tool used for evaluating learning algorithms. We were assigned to experiment and learn about the tool, ten machine learning algorithms, and three loss functions. We construct and evaluate these algorithms on ten distinct data sets. Our results indicate that the No Free Lunch Thoerum is in fact true. Through careful parameter tuning and experimentation methods, we were able to draw conclusions about the algorithms' ability to classify data for different types of data sets. With our conclusions about which algorithms are effective on which types of data sets, we make suggestions about when to use certain classifiers over others.**

All code used for this experimental analysis is located at http://github.com/Keydrain/CSCI477-2015/tree/master/Project1 and utilizes the Univeristy of Waikato ARFF standard [1].

Keywords*—WEKA, Loss Functions, Machine Learning Algorithms

I. Introduction

There are countless machine learning algorithms that classify data based on feature vectors. In 1995 Wolpert argued that there is no best machine learning algorithm for classification problems [2]. Most data sets have unique classes and features associated with instances of the data, making it unlikely that there exists a generic algorithm to address the differences between data sets and classify them each optimally.

To assess the effectiveness of various machine learning algorithms at classifying various data sets, ten algorithms were each run on ten different data sets obtained from the UCI Repository [3]. The machine learning classification algorithms included Simple Nearest Neighbor, K-Nearest Neighbor, Naive Bayes, Logistic Regression, decision tree, RIPPER, Sequential Minimal Optimization, Feedforward Neural Networks, Kernel Neural Networks, and Adaboost. Three different loss functions were used to assess the algorithms' classification efficiency for each data set: Hinge-Loss, Logistic-Loss, and Square-Loss.

II. Problem Statement and Hypotheses

Wolpert suggested that there is no best machine learning algorithm [2]. However, there are machine learning algorithms that perform best on certain types of data sets. We will test ten different algorithms on ten carefully chosen data sets to try to determine when to use a particular algorithm over another. We now introduce a hypothesis for the performance of the algorithms on each of our data sets. Table 1 displays the data set characteristics used in determining our hypotheses.

A. Artificial Characters [4]

Due to the noise and outliers it is unlikely that Adaboost will perform well. Since the data is not linearly separable, the SMO and Logistic Regression algorithms will misclassify a lot of instances.

The data set is likely large enough for sufficient training of the Neural Networks, they should do a satisfactory job of classifying the instances. If the decision tree and RIPPER manage to avoid overfitting the data, they might also offer accurate classifications. The nearest neighbor algorithm's performance will depend heavily on the amount of noise in the data set; if it is reasonable they should do a good job of classifying the instances, with the K-NN likely offering a performance improvement over the Simple-NN. Since the data is independent we expect that Naive Bayes will perform well.

B. Car Evaluation [5]

The data set has a relatively low number of instances and only six attributes. This will make it difficult for algorithms that rely on distinct features to classify, such as the decision tree. There may also be problems with dividing up such a small data set into a training set, evaluation set, and testing set, these factors make it likely that the decision tree will perform poorly. There may also be issues training the Neural Networks on such a small data set, resulting in poor classifications.

The nearest neighbor algorithms should perform well since nearby instances in this particular data set are likely in the same class. Adaboost should also perform well since there shouldn't be a lot of noise or outliers.

C. Connect-4 [6]

Algorithms that suffer from overfitting, such as the decision tree and RIPPER, will likely struggle with classification. The Neural Network algorithms also suffer from overfitting data, but there may be enough instances for them to train on and avoid those issues. We suspect that the similarities between the attributes of instances will also introduce noise that might cause algorithms like Adaboost and SMO to perform poorly.

We suspect that the nearest neighbor algorithms will perform alright, along with the Naive Bayes, because they are not as affected by noisy data and tend not to overfit data.

D. Contraceptive Method Choice [7]

The data set should be large enough to train the Neural Networks and decision tree on and they will likely perform the best. The excessive noise and number of outliers will make it difficult for Simple-NN, K-NN, or RIPPER to offer accurate classifications. Naive Bayes will lose accuracy due to the lack of independence in the data and the SMO and Logistic Regression algorithms will have problems separating the data (as well as the noise), leading to mis-classifications.

Data Set	Num. Instances	Num. Attributes	Num. Classes	Linearly Separable	Noise Expected
Artificial Characters	6,000	7	10	N	Y
Car Evaluation	1,728	6	4	N	Y
Connect-4	67,557	42	3	N	N
Contraceptive Method Choice	1,473	9	3	N	Y
Ecoli	336	8	8	N	Y
Letter Recognition	20,000	16	26	N	Y
Tic-Tac-Toe Endgame	958	9	2	Y	N
Wine	178	13	3	Y	N
Yeast	1,484	8	10	N	N
Zoo	101	17	7	N	Y

TABLE I. THE DETAILS OF THE TEN DATA SETS CHOSEN FROM THE UCI REPOSITORY TO TEST ON EACH ALGORITHM.

E. Ecoli [8]

The size of the data set will make it difficult to train Neural Networks, Decision Trees, or the RIPPER algorithm on. This makes it unlikely that any of them will perform well. However, they use a decision tree to intuitively model how the instances are classified so the decision tree algorithm might actually end up classifying the data accurately if it can imitate their methodology [9]. Logistic Regression and SMO will both have problems separating the data, resulting in unsatisfactory classifications.

The Simple-NN and K-NN will likely perform well given the attributes are relatively unique between classes. The Naive Bayes algorithm may perform well if the collected data is independent.

F. Letter Recognition [10]

Assuming most of the letters suffered from minimal distortion the two nearest neighbors algorithms should do an alright job of classifying the instances because the basic structure of each letter should still be similar. There should not be dependence between sets so Naive-Bayes should do a good job of classifying the letters. Both of the Neural Network algorithms should perform well because they have a large set of data to train on.

The algorithms that are affected heavily by noise will likely not perform very well. The decision tree, RIPPER, and Adaboost will likely have problems with features that have been distorted too much and will make bad classification decisions based on those distortions.

G. Tic-Tac-Toe Endgame [11]

The decision tree algorithm should work very will since the tree should be able to be kept small while still offering very good accuracy. The RIPPER algorithm and the Neural Networks should perform well because the data set seems to be large enough to train them on and there should not be too much noise or too many outliers. The data is also linearly separable, making it possible for the SVM and Logistic Regression Algorithms to perform well.

The Simple-NN and K-NN algorithms might have performance issues since there might be neighbors that are in a different class because of a small difference in the placement of Xs and Os on the Tic-Tac-Toe board.

H. Wine [12]

This is a relatively small data set that is easy to classify. Due to the size and nature, noise is sparse and should not impact the effectiveness of the various classification algorithms. However, due to the number of instances, algorithms such as neural networks and trees may not perform well.

In particular, we expect Naive Bayes and Adaboost to perform well on this set. Logistic may not perform as well given the clustering of the data relations.

I. Yeast [13]

The data set is probably large enough to train the Neural Networks sufficiently, so they should be able to classify the instances accurately. The decision tree algorithm should work very well since the data is very similar to the their intuitive classification scheme. If the data is unique between classes, the K-NN and Simple-NN algorithms should also perform well.

The algorithms that depend on linear separability (SMO and Logistic Regression) will have problems with misclassification due to the data being difficult to separate.

J. Zoo [14]

There are various attributes that are likely common across multiple classes such as ”toothed” and ”breathes” because these attributes are common with many animals in various classifications. This will make it difficult for algorithms like Naive-Bayes (which relies on the data being independent) to classify the instances. The nearest neighbor algorithms may also suffer from bad performance because of their assumption that nearby neighbors are in the same class, for example: a seasnake and a bass share many common attributes such as eggs, aquatic, predator, toothed, legs, and backbone but are classified differently. This type of noise may also make it difficult for algorithms like Adaboost to accurately classify instances. Furthermore, the data set only has 101 instances, making algorithms that rely on large diverse training sets inaccurate such as Feedforward-Neural Networks and Kernel Neural Networks.

We suspect that algorithms like the decision tree and RIPPER that depend on feature based decisions to lessen the classification choices will perform well. The only drawback to using these algorithms on this data is the possibility of over fitting the data.

III. Algorithms Implemented

The following ten machine learning algorithms were used: Simple Nearest Neighbor, k-Nearest Neighbor, Naive Bayes, logistic regression, decision tree, RIPPER, Sequential Minimal Optimization, feedforward neural networks, kernel neural networks, and Adaboost. Each was run on the 10 data sets chosen for experimentation.

A. k-Nearest Neighbor

K-Nearest Neighbor (k-NN) is an lazy learning algorithm used to classify an object based on the majority vote of its k nearest neighbors [15]. The number of nearest neighbors k is a user defined integer, and the euclidean distance is used to determine the them [16]. Typically k is an odd number to eliminate potential ties between neighboring classes [16].

When $k = 1$, k-NN is referred to as simple nearest neighbor. This is a special case of k-NN, where the class of x_i is predicted to be the class of the closest training sample [16].

When $k >$**2**, k-NN takes the closest neighbors to determine the likelihood of x_i falling into a class [15]. This can be described as the following:

$$P(C_i)|x) = \frac{P(x|C_i)P(C_i)}{P(x)} = \frac{k_i}{K}.$$

The bias of k-NN is assuming the votes or density estimation of the neighbors of x_i are a true representation of what the classification of x_i is [17].

B. Naive Bayes

Naive Bayes is a linear classifier that uses Bayes' Theorem as a conditional probability model to learn the conditional probability $P(C|d)$ of class set C given data point d [16] [17]. Bayes' Theorem is:

$$p(C|d) = \frac{P(C)P(d|C)}{P(d)} \equiv P(C)\prod_{i=1}^{n} P(x_i|C)$$

where C is a set of classes of size k, D is a vector $\langle d_1, d_2, \cdots, d_n\rangle$ of n independent features, and P(d) is effectively constant [15].

Naive Bayes combines this conditional probability model with a decision rule to separate the data set. The algorithm used in the experiments used the MAP (maximum a posterior) decision rule, represented in the following function [16]:

$$\underset{k\in K}{\operatorname{argmax}}\{P(C_k)\prod_{i=1}^{n} P(x_i|C_k)\}.$$

The bias of Naive Bayes is that it is linear [15]. Also, if the the $P(x_i|C_k) = 0$, then that value is dropped from the training which results in the full product to be zero [15].

C. Logistic Regression

Logistic regression uses a logistic function, which estimates the probability of a the relationship between a categorical dependent feature and independent feature(s) [17]. The goal is to maximize the probability $P(C|D)$, where C is a set of classes and D is a vector of independent features [18]. For models in d-dimensional feature space, a weight w_i is assigned and used to determine the probability as follows [15]:

$$p(C|X) = \frac{1}{1 - \exp(w_o + \sum_{i=1}^{d} w_i x_i).}$$

Overall, the maximum conditional log likelihood, W, is used to determine the actual classification of D.

$$W = \underset{w}{\operatorname{argmax}}(\prod_{j} P(C^j|x^j, w).$$

The bias of logistic regression is that it depends on a linear decision boundary [18].

D. decision tree: J48

The decision tree classifier implemented models the ID3 algorithm [15]. This creates a decision tree composed of decision nodes that are used as a function to classify the entropy of an attribute [16]. When given an input x, the tree can quickly determine its classification due to the tree's easy IF-THEN rule conversion [16].

Entropy is an issue that occurs in the creation of the decision tree [15]. To handle this, the maximum entropy reduction is desired. Let $p\pi$ be the number of positive examples in a partition of D and $n\pi$ be the number of negative examples in a partition of D, then:

$$\underset{i}{\operatorname{argmax}}(gain(x_i)) \equiv \underset{i}{\operatorname{argmax}}[I(p\pi, n\pi) - E(x_i)]$$

where $I(p\pi, n\pi)$ is the information content of a partition and $E(x_i)$ is the expected entropy if split at x_i [15] [16]. To determine the probability of a point falling within a particular partitioning, $IV(x_i)$ is used to use the gain ratio [15]:

$$\underset{i}{\operatorname{argmax}}[gainratio(x_i)] \equiv \operatorname{argmax}[\frac{gain(x_i)}{IV(x_i)}]$$

where

$$IV(x_i) = -\sum_{j=1}^{m_i} \frac{p_i + n_i}{p + n} lg(\frac{p_i + n_i}{p + n}).$$

Decision trees prefer very shallow trees of distinct identifiers to split the data such that partitions fall within hypercubes [15]. Additionally, gain ratio does not account for the overfitting of data, although it does penalize for the use of distinct identifiers [15].

E. RIPPER

The Repeated Incremental Pruning to Produce Error Reduction algorithm, also known as RIPPER, is an IREP-based propositional rule learner [19]. RIPPER attempts to find the propositional and FOL rule that will best classify a data set [15]. The algorithm takes a set of examples D and splits it into two sets, a growing set D^g and a pruning set D^p [19]. RIPPER uses D^g to grow a rule r' for each class, and then uses D^p to evaluate its worth [15]. Immediate pruning occurs when r' is compared to the previously grown rule r. This process is continued until an entire rule set is developed [19] [15].

RIPPER runs the risk of overfitting the data, as it has a binary bias of evaluating equality and favors longer rules [15].

F. Support Vector Machine: SMO

A support vector machine is a typical nonparametric classifier that acquires a decision function through the use of a training set of input-output pairs [20]. The sequential Minimal Optimization (SMO) algorithm is included in the LibSVM package [21]. The SMO algorithm is a well-known tool for separating an n-class classification problem into n two-class problems to construct a set of hyperplanes y which are "classifier boundaries" with a margin width of p [17] [20]. The minimum margin is desired for y so the upper bound, 1, is the likelihood of x_i being C and the lower bound, -1 is the opposite [15]. This results in the following:

$$min(p) = min(\frac{1}{2}W^T W)$$

such that

$$C(x) = \begin{cases} +1 \; if \; y \geq 1 \\ -1 \; if \; y \leq -1 \end{cases}$$

where

$$y = W^T + b.$$

Support vector machines run under the assumption that there is no noise in the data [15]. Additionally, Support vector machines are biased to be linearly separable and treats the data as such [15].

G. Feedforward Neural Network

Feedforward neural networks, also known as multi-layer perceptions, is a biologically inspired classification algorithm that consists of organized layers [16] [22]. Data enters through the input layer and continues through c-layers, where c is the number of classification per data set [22]. In each layer a node evaluates the input to give a weight, or similarity measure, that is added to the previously outputted weight [22]. Once the final layer is reached, the largest outputted weight determines the classification associated with the input value [16].

Feedforward neural networks are biased towards the information received from the previous hidden node, although no information is every passed pack to the previous node [16]. This assumes the learning will be improving as more inputs are presented, i.e. neural networks are universal approximators [15].

H. Kernel Neural Network: KBF Network

KBF Network is a radial basis function network that uses non-linear radical basis functions to classify an input x_i [23]. RBF Networks are a two-layered feedforward network [23]. Like Feedforward neural networks, data is passed through an input layer, through hidden layers which performs a set of Gaussian RBFs to determine a weight, or similarly measure, and then is outputted as a linear summation [24]. The NBF Network trains in an unsupervised manner, resulting in k-means clustering which are used as a basis for fitting data [25]. This allows RBF newtworks to determine the classification C of an input x_i.

KBF Networks tend to perform poorly when noise is present and they are costly to execute when data sets are large [23]. However, they are good at interpolation and learn quickly [23].

I. Adaboost

Adaboost, also known as Adaptive Boosting, is an ensemble learning algorithm that generates a complementary base-learners to train the next learner on the mistakes of the previous learner [15] [26]. This ultimately generates a 'strong' learner from a combination of 'weaker' learners that have a error probability less than 50 percent, i.e. boosting [15]. During the training process, the error probability is assigned to each sample of the training set D^t which is used [17]. At each iteration, a new weak learning is added to the ensemble, and the weight vector is adjusted to focus on the instances that have been misclassified [17]. This is why decision stumps are often used [16].

Adaboost is biased towards choosing the incorrectly classified data to increase the selection of the wrong data for the next training set [15]. This indicate that small training sets will put Adaboost at a disadvantage of accurately choosing classifiers .

IV. Experimental Approach

Three loss functions were used to analyze each algorithm's ability to correctly classify the 10 data sets chosen. These loss functions were hinge loss, squared loss, and log loss. Additionally, the parameters of each classification algorithm and loss function had to be tuned to each data set. All experiments were conducted on campus in the computer science lab. Our training runs involved minimal tuning of the feature and instead relied heavily on already established standards in the WEKA environment. The parameters that we did modify were to specify five for the number of nearest neighbors and to optimize with regard to the mean squared value. No further tuning of tactic was implemented for this series of experiments.

A. Loss Functions

The objective of the algorithms used in the experiments were to see which classified best for each data set. Knowing this, several loss functions were examined before choosing hinge loss, squared loss, and log loss. It was determined these three loss functions analyzed classification functions well. Let p be the be the prediction and y a label of x_i [15] [27].

1) hinge loss: hinge loss can be interpreted as how comfortable or confident an algorithm's is with its classification.

$$loss_{Hinge}(p, y) = \text{argmax}(0, 1 - yp)$$

2) Squared Loss: Squared loss can be interpreted as the accuracy of an algorithm's classification.

$$loss_{Squared}(p, y) = \frac{1}{|D|} \sum_{x_i \in D} (p - y)^2$$

3) log loss: log loss can be interpreted as the probability of an algorithm accurately making a classification.

$$loss_{Log}(p, y) = \log(1 + exp(-yp))$$

B. Turning Process

We opted to create training sets for most of our original data sets. These training and testing data sets are partitioned from the original data 80/20 respectively. This relationship is known as the Pareto Principle [28]. We believe there is past evidence to warrant the ratio 80/20 viable for this exercise. Additionally, our preliminary testing of the various algorithms indicated that WEKA's suggested defaults would satisfy most of the parameter tuning. Regardless, K-NN, decision tree, and RBF Networks still required specific variable tuning.

For k-NN, a value for k needed to be picked. Initially our approach was to iterate over 3-10 for k and choose the minimum k that resulted in the best performance according to correctness. This approach was deemed unrealistic given the amount of time it took to run the tests, so we let $k = 5$. Five was chosen because it allowed k-NN to choose from a fairly large number of nearest neighbors, while hopefully avoiding inaccurate outputs that could have resulted from using more neighbors that could have been 'too' far away.

For decision tree, we had the option for allowing pruning or letting the tree grow in full. With the intentions of comparing each algorithm in their most basic state, we turned tree pruning off.

For the KBF Network algorithm, we had the choice of using logistic regression or linear regression for the ridge value. We chose to use logistic regression because the logistic function was used in both the logistic regression algorithm and the log loss performance algorithm

V. Results

300 total tests were performed. Of the 300 tests, only 285 were actually completed. The following tests failed after running for several hours:

- Feedforward Neural Network for the Artificial Characters, Connect-4, Letter Recognition and Yeast data sets.
- Logistic regression for the Artificial Character and Letter Recognition data sets.
- Support vector machine for the Connect-4 data set.

The results of the algorithms ran, in relation to the data sets can be viewed in Figures 1, 2, and 3.

A. hinge loss

Hinge loss can be interpreted as how comfortable or confident an algorithm is with its classification. Knowing this, we were able to interpret the confidence of each algorithm according to the calculated $loss_{Hinge}(p, y)$. Figure 1 shows these results.

The many of the graphs in Figure 1 tend towards a zero for various data sets in multiple algorithms. Occasionally a 'hinged' outlier appears within the data indicating a particular data set did poorly in relation to that particular algorithm. An interesting example is comparing simple nearest neighbors to k-nearest neighbors for the Connect-4 and Artificial Character data sets. For simple nearest neighbor, both had a high error rate, compared to in k-NN were both significantly improved their error.

Overall it can be seen the hinge function suggests the Artificial Character, Connect-4, and Letter Recognition data sets could not be confidently predicted given the classification algorithms used in the experiments. The logistic regression algorithm performed poorly overall considering three data sets were unable to conclude under this evaluation.

B. Squared Loss

Squared loss can be interpreted as the accuracy of an algorithm's classification. Knowing this, we were able to interpret the accuracy of each algorithm according to the calculated $loss_{Squared}(p, y)$. Figure 2 shows these results.

An interesting aspect of the graphs seen in Figure 2 is that for each data set used as input for each algorithms, the accuracy of classification is relatively similar. All error is within a range of about .1 percent for each. The exception to this is the Tic-Tac-Toe Endgame data set, it had a relatively high error percentage range across the board; it also had the worst error the most often. This could be that is the only data set that has two classifications.

Overall, the squared loss performance function suggests the predicted classifications are typically close to the actual classifications, usually with each data set experiencing 20 percent misclassification. It seems that the support machine vector algorithm performed the best, with the smallest squared loss, while the feedforward neural networks earning the largest squared loss.

C. log loss

Log loss can be interpreted as the probability of an algorithm accurately making a classification. Knowing this, we were able to interpret the probability of each algorithm according to the calculated $loss_{Log}(p, y)$. Figure 3 shows these results.

We used the WEKA log loss-flag which ended up resulting in a density mass number divided by the total number of instances. Initially we expected a probability between 0-1, however, given the different loss values presented, we made an analysis anyways. Ultimately it seemed that most functions performed relatively well. There were a few extreme outlier that skewed the data. The yeast, as well as the letter recognition data set, performed relatively poorly. Both had lower probability of accurately making a classification.

Overall the Adaboost algorithm, feedback neural network, and logistic regression were determined to perform the best according to accuracy. It was difficult to determine the worst algorithms given the extreme outliers in our results.

VI. Conclusion

It can be concluded that the tests performed could have done better. For each of the Data sets we refer heavily to Figures 1, 2 and 3 located below.

A. Artificial Characters

The results for the Artificial Characters data set indicate that most of the algorithms had difficulty classifying this set. We had hypothesised that Adaboost, SMO, and Logistic would have a difficult time properly classifying this set, while neural networks and Naive Bayes would perform better at classifying. Of the tests that concluded, Adaboost and SMO performed better than we had predicted while Logistic was unable to finish and generate results. Surprisingly Neural Networks and Naive Bayes performed worse than we had expected. It is difficult to pull accurate conclusions from this data set due to the missing data from runs which had not produced a result. Also note that the log loss function generated very large values for this data set.

B. Car Evaluation

The performance evaluations of each algorithm suggested each algorithm performed well on the car evaluation data, thus car evaluation was a good data set for evaluating classifier algorithms. In our hypothesis we said the algorithms that relied on distinct features (decision trees, neural networks, etc) would perform worse than the algorithms that relied on nearby instances (nearest neighbor). We can see that all three of the loss functions indicate that decision trees and neural networks performed better than we had original predicted. Nearest Neighbor and Logistic regression however, conformed with our hypothesis and achieved the lowest levels of loss.

C. Connect-4

Connect-4 is a relatively large data set with similar instances that we believed would cause the decision tree, Adaboost, SVM and RIPPER algorithms to perform poorly. This was almost true; the evaluation given by the square loss function indicated they performed no better or worse, while log and hinge loss gave slightly higher error for kernel neural networks and decision trees. Additionally, we stated the other algorithms would perform relatively well in comparison. Our results agreed with this prediction. We can see from the loss functions that logistic regression and Naive Bayes classified the Connect-4 data set the best, along with the nearest neighbors algorithms.

D. Contraceptive Method Choice

In our hypothesis we said neural networks and the decision tree algorithms would perform the best, while all other algorithms would struggle to correctly classify the data set. According to the log loss, this hypothesis wasn't fully true; decision tree performed poorly according to log loss, but we did accurately predict simple nearest neighbor and SVM algorithms would do poor. It was a surprise to see logistic regression was considered to have the best square evaluation. Overall, it seems the Contraceptive data set was a difficult data set to accurately classify by all of the algorithms we tested.

E. Ecoli

The number of instances in this data set is relatively small and as such, we predicted that the size would negatively impact the trainability of Neural Networks, decision tree and RIPPER. We suspected Nearest Neighbors would perform better due to the uniqueness between classes. Our results indicate that Nearest Neighbor and Decision Trees did indeed performed well, while we obtained mixed results from the other algorithms in our hypothesis. Neural Networks in particular obtained dissenting results between square loss and hinge loss. Feedforward Neural Network appears had varying error when comparing its square loss value to its hinge loss, as did the Kernel Neural Network, but with opposite errors. Given this discontinuity, continued testing would be the wisest decision to achieve a more accurate result. RIPPER did not conform to our hypothesis as expected and instead performed better in all three loss measurements.

F. Letter Recognition

In our hypothesis, we had predicted that nearest neighbors, Naive Bayes, and the Neural Networks would perform well, while algorithms that are effected by noise would not. Our results concluded that RIPPER, Naive Bayes, and Feedforward Neural Network did not perform well while Kernel Neural Network and Adaboost achieved better-than-expected results. Given this, we concluded that Adaboost does not appear to be as hampered by variable features, as it displayed robustness on this particular data set. Looking at the other algorithms, it appears that this data set was relatively easy to classify and evaluate.

G. Tic-Tac-Toe Endgame

Tic-Tac-Toe was an interesting data set to work with given it only has two classes. We had thought the decision tree, RIPPER, neural network, SVM and logistic regression algorithms would preform well. It can be seen that this hypothesis was not necessarily true. The squared loss evaluation, showed S-NN and logistic regress were the only two without high margins of error, while the hinge function suggested all the algorithms were confident in their classification, except SVMs, Naive Bayes, and logistic regression. Conclusions can be drawn from the log loss as well and it shows RIPPER and decision tree did not perform very well. This means our hypothesis was incorrect.

H. Wine

Given the nature of the data and commentary from the author, we expected many of our algorithms to perform well on this data set. We specified that Neural networks and Decision Trees would not perform as well as Naive Bayes and Adaboost. Our results indicate that both Naive Bayes and Adaboost scored low with regard to square loss, but better with log loss. Adaboost also performed well with hinge, but Naive Bayes performed very poorly. Neural networks performed well with regard to hinge and log loss but very poorly for square loss. Notice that all algorithms have a poor rating in square loss for this data set. Realizing that the square loss is great for all algorithms, the nearest neighbors generated some of the best results.

I. Yeast

The yeast data set was unable to converge and give a true evaluation of the logistic regression algorithm, while the confidence of the SMO indicated yeast was not linearly separable for classification. We had said these two algorithms would perform the worst. We also thought the algorithms that relied on uniquely distinct identifiers would do well. The can be seen in the results that this is true. Given this generalization, Naive Bayes and RIPPER gave the best classification for this data set.

J. Zoo

This data set was concluded to be very interesting, as it was a unique set containing only one reoccurring object. This alone indicated to use the data set would be difficult to classify by any algorithm and would likely occur in significant error. We did however believe the RIPPER and decision tree algorithms would perform best. We were surprised to see small error for both squared and log loss functions, however the classification confidence of most algorithms were small, resulting in high error. Overall, all algorithms were able to classify the data set well due to the fact each instance was only by itself, but as interesting as this was, it did not results in significant evaluation for any of the algorithms.

VII. Future Works

We could draw from our conclusions that the No Free Lunch Theorem is indeed true, however we were not overly thrilled with the run time or performances of the algorithms we experimented on. There are certain things we would do differently in the future if we were to conduct this experiment again. There are three things we would like to change.

First, we would make sure to choose data sets that were reasonable classification data sets. The zoo data set did not provide a meaningful performance analysis, although it was interesting to see the algorithms would technically work.

Second, for the experimental approach we would better tune our parameters. We typically the default parameters because we were interested in keeping each algorithm as 'basic' or 'unspecialized' as possible for what we felt would be a fair algorithm comparison. For example, decision trees do not necessarily have to prune, although prune should help with overfitting the data. Additionally, different parameter tuning could help us with cutting down our run time; we had several tests ran for 1+ hours, and for 300 tests this took a considerable amount of time. This doesn't account for the tests that never finished running after being left for several hours.

Third, we would consider running each algorithm, for each data set, multiple times to have a better performance evaluation/comparison. With each iteration performed, a more accurate performance evaluation would be obtained. Ultimately, the run-time of the algorithms prevented us from comparing this. Initially we intended to iterate each algorithm with each data set ten times, which would have resulted in 3,000 tests to graph and draw conclusions from.

Overall this assignment has been a true learning experience between exploring the WEKA environment to researching the learning algorithms.

Acknowledgment

The authors would like to thank Sam Micka for his contribution to this work. He helped locate the data sets used in experimentation and develop our experimental hypotheses.

References

[1] G. Paynter, "Attribute-relation file format (arff)," 2002.

[2] D. Wolpert and W. Macready, "No free lunch theorems for search," in *Technical Report SFI-TR-95-02-010*, Feb 1996.

[3] M. Lichman, "UCI machine learning repository," 2013.

[4] M. Botta, "UCI machine learning repository: Artificial characters data set," 1992.

[5] M. Bohanec, "UCI machine learning repository: Car evaluation data set," 1997.

[6] J. Tromp, "UCI machine learning repository: Connect-4 data set," 1995.

[7] T. Lim, "UCI machine learning repository: Contraceptive method choice data set," 1997.

[8] K. Nakai, "UCI machine learning repository: Ecoli data set," 1996.

[9] P. Horton and K. Nakai, "A probabilistic classification system for predicting the cellular localization sites of proteins.," in *Ismb*, vol. 4, pp. 109–115, 1996.

[10] D. Slate, "UCI machine learning repository: Letter recognition data set," 1991.

[11] D. Aha, "UCI machine learning repository: Tic-tac-toe endgame data set," 1996.

[12] M. Forina and S. Aeberhard, "UCI machine learning repository: Wine data set," 1991.

[13] K. Nakai, "UCI machine learning repository: Yeast data set," 1996.

[14] R. Forsyth, "UCI machine learning repository: Zoo data set," 1990.

[15] J. Sheppard, "Lecture notes in machine learning: Soft computing," September 2015.

[16] T. Mitchell, *Machine Learning*. McGraw-Hill Science, Engineering, and Math, 1997.

[17] E. Alpaydin, *Introduction to Machine Learning, 2nd Ed.* Massachusetts Institute of Technology, 2010.

[18] T. Jaakkola, "Machine learning: Lecture 5," 2004.

[19] W. W. Cohen and F. E. Rule, "Ripper," 2002.

[20] "Function approximation," in *Support Vector Machines for Pattern Classification*, Advances in Pattern Recognition, pp. 265–296, Springer London, 2005.

[21] C. Chang and C. Lin, “Libsvm- a library for support vector machines,” 2014.

[22] D. Weenink, “Feedforward neural networks 1,” April 2004.

[23] J. Bullinaria, “Radial basis function networks: Introduction,” 2014.

[24] B. Frey, “Neural networks and kernal methods.”

[25] E. Frank, “Rbfnetwork: Classes that implement radial basis function networks.”

[26] “Adaboost,” 2015.

[27] “Loss functions,” 2015.

[28] M. Newman, “Power laws, pareto distributions and zipf’s law,” *Contemporary physics*, vol. 46, no. 5, pp. 323–351, 2005.

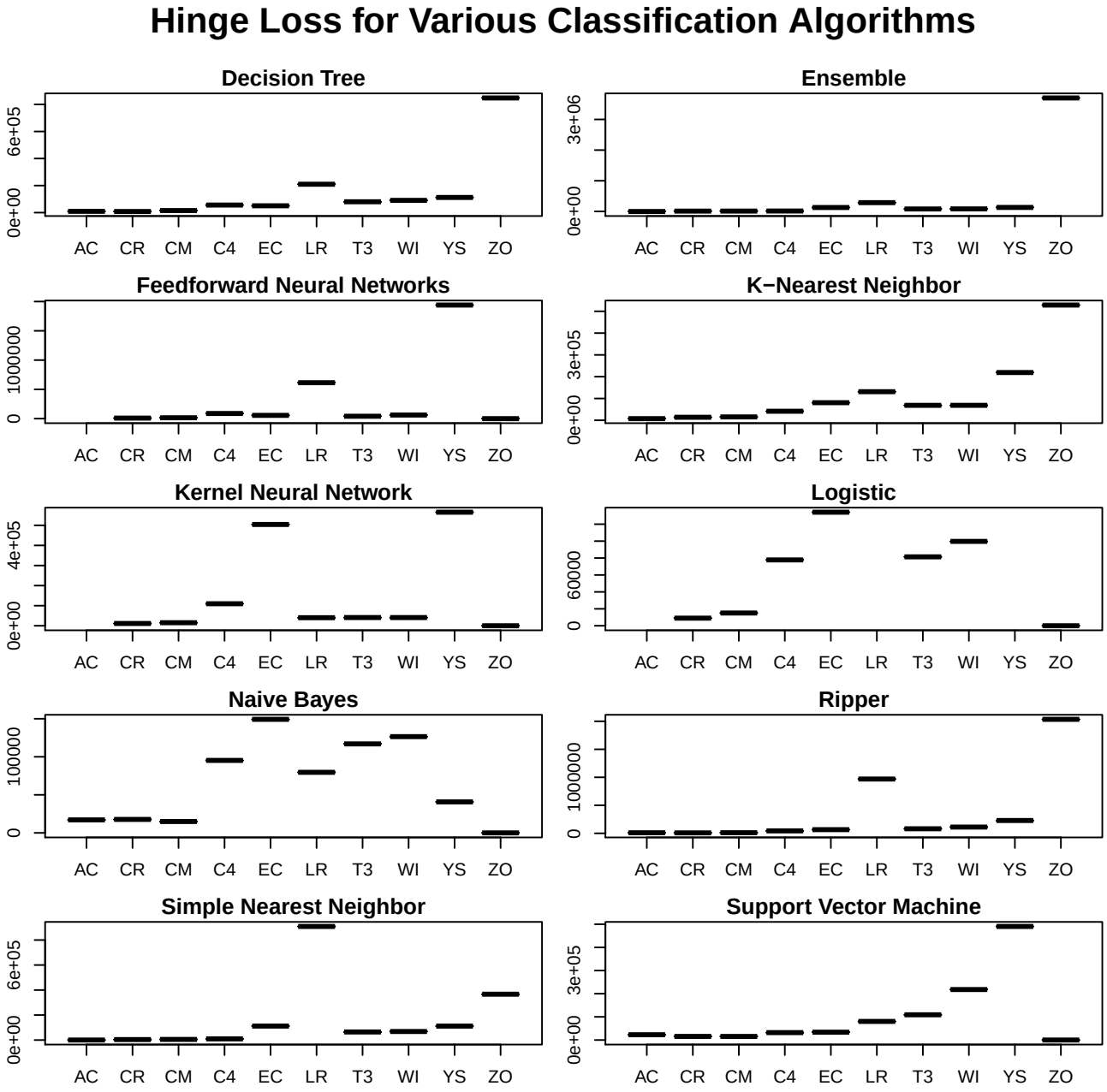

Fig. 1. hinge loss values for each of the data sets given the algorithm used. Note that data sets are labeled as follows: AC::Artificial Characters, CR::Car Evaluation, CM::Contraceptive Method Choice, C4::Connect-4, EC::Ecoli, LR::Letter Recognition, T3::Tic-Tac-Toe Endgame, WI::Wine, YS::Yeast, ZO::Zoo

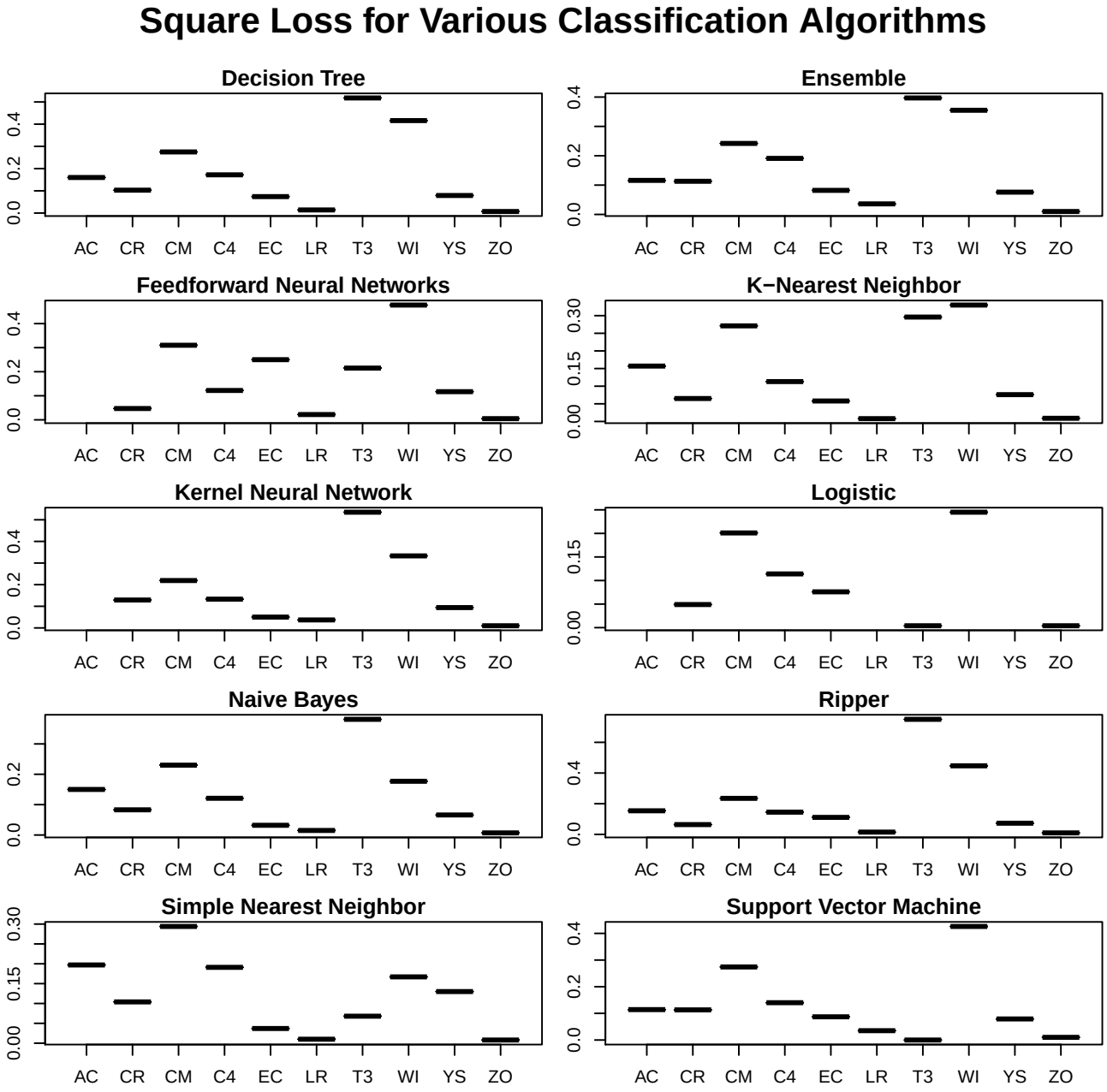

Fig. 2. square loss values for each of the data sets given the algorithm used. Note that data sets are labeled as follows: AC::Artificial Characters, CR::Car Evaluation, CM::Contraceptive Method Choice, C4::Connect-4, EC::Ecoli, LR::Letter Recognition, T3::Tic-Tac-Toe Endgame, WI::Wine, YS::Yeast, ZO::Zoo

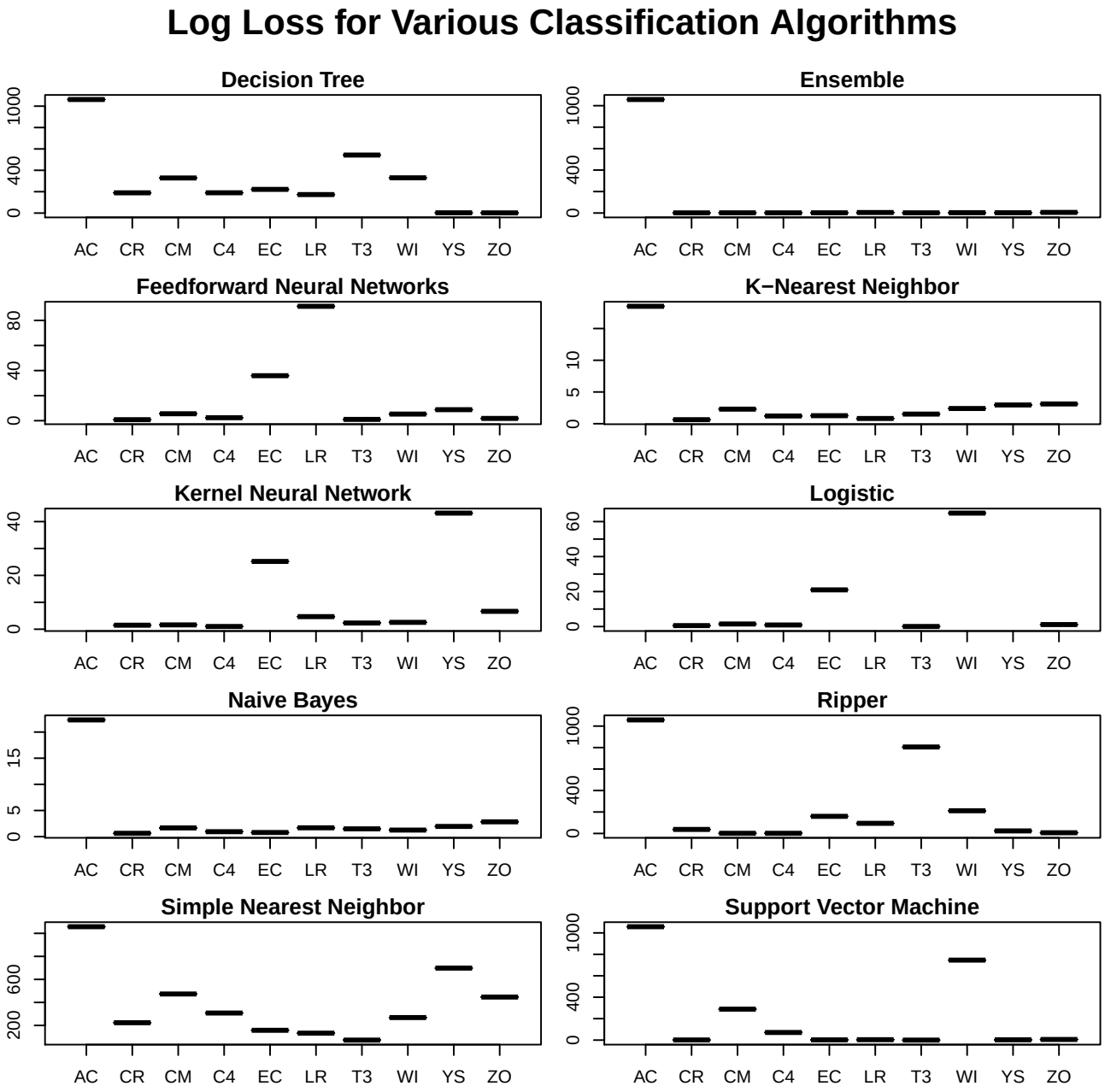

Fig. 3. log loss values for each of the data sets given the algorithm used. Note that data sets are labeled as follows: AC::Artificial Characters, CR::Car Evaluation, CM::Contraceptive Method Choice, C4::Connect-4, EC::Ecoli, LR::Letter Recognition, T3::Tic-Tac-Toe Endgame, WI::Wine, YS::Yeast, ZO::Zoo