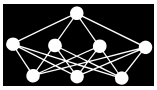


SMOOTHING DECISION BOUNDARIES TO AVOID OVERFITTING IN NEURAL NETWORK TRAINING

Khalil el Hindi[*], *Mousa AL-Akhras*[†]

Abstract: This work addresses the problem of overfitting the training data. We suggest smoothing the decision boundaries by eliminating border instances from the training set before training Artificial Neural Networks (ANNs). This is achieved by using a variety of instance reduction techniques. A large number of experiments were performed using 21 benchmark data sets from UCI machine learning repository, the experiments were performed with and without the introduction of noise in the data set. Our empirical results show that using a noise filtering algorithm to filter out border instances before training an ANN does not only improve the classification accuracy but also speeds up the training process by reducing the number of training epochs. The effectiveness of the approach is more obvious when the training data contains noisy instances.

Key words: *Artificial neural network, instance-based learning, instance reduction, instance selection, machine learning, noise filtering, overfitting, over learning, prototype selection*

Received: December 20, 2010
Revised and accepted: July 16, 2011

1. Introduction

This work addresses an important problem in the field of machine learning, which is the problem of overfitting. This is a common problem for machine learning algorithms that use a variable sized-representation of hypotheses, such as Artificial Neural Networks (ANNs) and decision trees. Overfitting has a negative effect on classification accuracy; it simply means that although the classifier's error measured on the training data is small, the classification error measured on new instances,

[*]Khalil el Hindi
Computer Information Systems Department, King Abdullah II School for Information Technology
The University of Jordan, Amman 11942 Jordan, E-mail: hindi@ju.edu.jo
[†]Mousa AL-Akhras
Computer Information Systems Department, King Abdullah II School for Information Technology
P.O.Box 13835, The University of Jordan, Amman 11942 Jordan, E-mail:
mousa.akhras@ju.edu.jo

unseen during training, is high. It is the classification error on unseen instances what matters, since the main objective of machine learning is to be able to deal with new situations. This is what makes overfitting an important issue in machine learning.

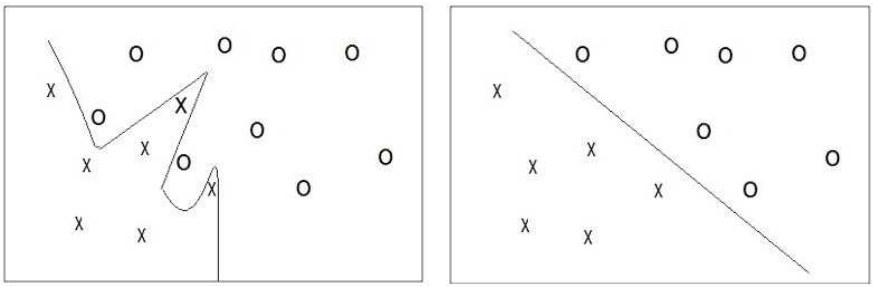
Overfitting occurs in two situations: when the training set contains noisy instances and when the training instances are not good representatives of the instance space [14]. Both of these situations are common in real life applications. Real life data sets are rarely free of noise. Noise occurs when an instance is wrongly assigned to a different class than its correct one. Also, real life data sets usually contain a small subset of the instance space which makes them less likely to be good representatives of the instance space.

In this work, we suggest eliminating border instances which tend to be noisy instances or hard-to-learn untypical instances. This should help in smoothing the edges to avoid overfitting and speed-up the training process. Although, we believe that this is a general approach that may improve the classification accuracy of not only ANN, but also of decision trees, our experiments and results, reported in this work, are limited to ANN.

This work is a more elaborate work than the one presented in [8], focusing in particular on training data sets that may contain noisy instances.

The suggested method smoothes the decision boundary by eliminating the idiosyncrasies of the training set by filtering out near-border instances from the training set, as these instances are a major source of overfitting. We do this by applying an instance reduction algorithm [20] on the training set to eliminate near-border instances before we train an ANN. The aim of using an instance reduction algorithm as a pre-training phase is to smooth the decision boundary and thus makes it less likely for an ANN to overfit the training data. By removing border instances, the effect of noise can be reduced or even eliminated, as most noisy instances would lie near the borders.

Fig. 1 shows the decision boundaries of a hypothetical classification problem where the Os represent instances of one class and the Xs represent instances of a different class.



(a) Before eliminating border instances (b) After eliminating border instances

Fig. 1 *Decision boundaries before and after eliminating border instances.*

Fig. 1a shows the decision boundaries in the presence of border instances where some of them could be noisy instances. Training an ANN to classify the Os and the Xs is equivalent to finding a line that separates the Os from the Xs (as this is a linear separable problem). This is a much harder job for the points in Fig. 1a than it is for the points in Fig. 1b which has the border instances eliminated. Finding a line that separates the Os from the Xs in Fig. 1a requires more training epochs, which increases the chances of overfitting and reduces classification accuracy for new instances.

To test the effectiveness of the proposed method on improving the classification accuracy and the training speed of ANN, we used 21 data sets from UCI machine learning repository [6]. We performed two sets of experiments: in the first set, we used the data sets without any modifications; in the second set, however, we introduced some noise, and experimented with different noise ratios.

The results show substantial improvement in both classification accuracy and learning speed, reflected as a reduction in the number of training epochs, when noise filtering algorithms were used to preprocess the training data.

The rest of this paper is organized as follows: Section 2 reviews the problem of overfitting and the typical methods for dealing with it. Section 3 reviews the instance reduction algorithms we use in this work. In section 4, we discuss the results of smoothing the decision boundary in terms of classification accuracy and speed of training. Section 5 presents our conclusions.

2. Typical Methods for Dealing with Overfitting

Overfitting is an important issue that was addressed in different ways in the literature. In decision tree learning, for example, there are two main methods: either to stop growing the tree when further growing is not based on statistically significant data, or to grow the full tree and then prune it. There are two main pruning techniques: reduced error pruning and rule post pruning [15]. Reduced error pruning avoids overfitting by pruning the resulting tree bottom up as long as that pruning does not increase the classification error. In rule post pruning, the tree is first converted into rules and then the rules are independently pruned.

In ANNs, overfitting is avoided using two main methods [14]. The first exploits the fact that overfitting usually occurs as a result of overtraining, therefore, it is natural to try to avoid overfitting by avoiding overtraining. Overtraining occurs when the ANN is trained for a large number of epochs. One approach to determine a good number of epochs is by using a validation set which is taken from the training set before training begins. The validation set is used during training to decide when we should stop training. When the error, measured using the validation set, stops declining and starts growing, we stop training and record the number of epochs at which the error started to grow. To determine the validation set, the data set is usually randomly split into 3 equal parts. The described process is repeated three times (3-fold cross validation) using one partition as a validation set each time. The average number of training epochs is then computed. Then, an ANN is trained on the whole training set, including the validation set, for the computed average number of epochs. In this way, overtraining and consequently overfitting are less likely to occur.

The other approach is called weight decaying [14]; it simply pulls the weights towards zero. The intuition behind this is that overfitting does not occur early in the training process but rather occurs late in the training process. This is because at the beginning the weights are initialized to small random numbers close to zero. This means that the neural network at this stage has not yet been affected by the idiosyncrasies of the training data. It is when the weights grow in size, probably due to overtraining, that the neural network becomes affected by the idiosyncrasies of the training data. Therefore, decreasing the weights or pulling them towards zero helps avoid overfitting.

As can be seen, the discussed approaches for overfitting avoidance depend on the nature of the learning algorithm. However, the approach we discuss in this work is independent of the learning algorithm, making it suitable for any machine learning algorithm that may overfit the training set, such as decision trees, classification rules, or neural networks.

3. Instance Reduction Techniques

Instance reduction algorithms have been extensively studied in the context of Instance-Based Learning (IBL) [2]. Instance-based learners are lazy in the sense that they do little work during training, deferring most of the work to the classification stage. During training, they simply store the training data and when a new instance is to be classified, they search their memory for the most similar instance(s) and use the class with the majority of votes as the predicted class for the new instance. Their accuracy tends to increase as the training set size increases but this comes at a cost in terms of their memory requirement and classification time. For this reason, a wide variety of instance reduction techniques [1, 2, 7, 9, 10, 11, 16, 17, 18, 19, 20] were developed in an attempt to alleviate these drawbacks by identifying and retaining the most relevant instances only. Wilson and Martinez [20] compare between many instance reduction techniques that may be used to reduce the number of instances needed for IBL. Some instance reduction algorithms may help in removing hard-to-learn, near-border instances that may cause overfitting.

Instance reduction techniques can be categorized into pure instance reducers and noise filters. Pure instance reducers are much more effective in reducing the size of the training set than noise filters [20]. They tend to keep a small proportion of the training set consisting of a few center instances. An instance is considered to be a center instance if it has the same class as that of the majority of its k nearest neighbors; otherwise it is considered a border instance. Noise filtering algorithms, on the other hand, are less effective at reducing the size of a training set; they tend to retain the majority of center instances and remove near-border instances that are close to instances that belong to other classes (enemy instances).

Border instances tend to be noisy instances or untypical, hard-to-learn instances. Due to the negative effect of noisy instances on the classification accuracy, even some pure instance reduction algorithms include a noise filtering stage, e.g., DROP3 and DROP5 [20]. We call these techniques instance reducers with noise elimination.

In the following subsections, we review the instance reduction techniques we use in this work categorized into noise filters, pure instance reducers, and instance reducers with noise elimination.

3.1 Noise filters

The following three algorithms for removing border instances and retaining center instances are used in our experiments.

1. **The Edited Nearest Neighbor algorithm (ENN)**: This algorithm removes an instance if it does not agree with the majority of its k nearest neighbors (with $k = 3$, typically). This edits out noisy instances as well as near-border cases, leaving smoother decision boundaries. It also retains all internal points, which keeps it from discarding most training examples [18]. Therefore, it is ineffective as a reduction algorithm. This algorithm was shown by [20] to retain on average 83.34% of the original data set. In our experiments, using 21 data sets (before inserting noisy instances), it retained on average 85.49% of the original data set.

2. **The Repeated ENN (RENN)**: This algorithm applies the ENN algorithm repeatedly until all remaining instances are consistent with the majority of their k-neighbors (have the same class), which widens the gap between classes and smoothes the decision boundary even further (compared to the ENN). This algorithm was shown by [20] to retain on average 80.92%. It retained on average 83.92% of the 21 data sets we used in our experiments. It is not a huge reduction but as would be expected it retains fewer instances compared to the ENN.

3. **ALL-kNN**: ALL-kNN [17] extends ENN in another way. This algorithm makes k iterations. At the ith iteration, it flags as bad any instance that is not classified correctly by its i nearest neighbors. After completing all iterations, the algorithm removes all instances flagged as bad. This algorithm was shown by [20] to retain on average 77.44%. In our experiments, it retained on average 80.76% of the 21 data sets we used. Again, it is not a huge reduction since it eliminates only the border instances however it eliminates more instances than ENN and RENN.

3.2 Instance reducers

We included these two algorithms in our experiments because they are among the most effective instance reduction algorithms; they retain a very small proportion of the training data and maintain a relatively good classification accuracy when used with the kNN algorithm [20].

1. **Encoding Length (ELGrow)**: This algorithm [7] begins with a growing phase that takes each instance in the training set (T) and adds it to the reduced set if that results in a lower cost than not adding it. After all instances are seen, pruning is done, where each instance in the reduced set is removed if doing so lowers the cost of the classifier. This algorithm was shown by [20]

to retain on average 1.83% of the data sets and in our experiments it retained on average 2.02% of the considered data sets.

2. **Encoding Length (Explore)**: This algorithm [7] begins by growing and pruning the reduced set (S), using the ELGrow method, and then performs 1000 mutations to try to improve the classifier. Each mutation attempts adding an instance to S, removing one from S, or swapping one in S with one in T-S, and keeps the change if it does not increase the cost of the classifier. The generalization accuracy of the Explore method (as an IBL) is quite good empirically, and its storage reduction is much better than most other reduction algorithms. According to [20], this algorithm retains on average 2.01% of the original data set. In our experiments, it retained on average 2.26% of the considered data sets. It is, therefore, one of the most effective instance reduction algorithms.

3.3 Instance reducers with noise filtering

Due to their obvious negative effect on the accuracy of IBL, even some instance reducer algorithms which tend to retain border instance, do that after a noise filtering stage. Two such techniques were considered in this work.

1. **DROP3**: This algorithm [20] has a high accuracy and a reasonable storage reduction. It uses a noise-filtering pass, such as ENN, before starting to remove noisy instances. The algorithm starts by sorting S on an attempt to remove the centre points before the border points. The instances are sorted by distance to their nearest enemy (nearest neighbor with a different class). So instances far away from their nearest enemy are removed first. This removes center points and retains border points; this makes sense only because noisy instances are removed in the first stage. According to [20] this algorithm retains on average 14.31% of the original data set. In our experiments, it retained on average 16.85% of the considered data sets.

2. **DROP5**: This algorithm [20] considers removing first the instances that are nearest to their nearest enemy, and proceeds outward. This serves as a noise-reduction pass, but will also cause most internal points to be removed as well. After this pass, the furthest-to-nearest pass is performed repeatedly until no further improvement can be made. This algorithm was shown by [20] to retain on average 16.09% of the original data set. In our experiments, it retained on average 17.86% of the considered data sets.

4. Training ANN using the reduced sets

We believe that border instances are the primary cause of overfitting; after all, even human beings find untypical examples confusing and difficult to learn. Therefore, it is natural to assume that these instances would require a large number of training epochs and possibly leading to overfitting and thus a reduction in classification accuracy. If this hypothesis is true then, eliminating such instances before training an ANN should help reduce the number of training epochs and the likelihood of

overfitting. This is especially true if the border instances are noisy instances which they tend to be.

To verify this hypothesis, we conducted two sets of experiments, in the first, we used the original data sets without any modifications; in the second, we modified the training sets so that they contain some noisy instances. The experiments were performed on 21 benchmark data sets obtained from the UCI machine learning repository [6]. We trained ANNs on the whole training sets as well as on the reduced sets produced using the discussed instance reduction algorithms. Each algorithm was applied on each training set to obtain a reduced set. Each reduced set, in addition to the full set, was used to train 4 ANNs that differ only in the number of neurons in the hidden layer.

Each ANN used is a 2-layer network with biases. The hidden layer consists of sigmoid units, while the output layer consists of linear units. Levenberg Marquardt algorithm [12, 13], as implemented in MATLAB®, was used to train each ANN. The Mean Square Error (MSE) [3] was used as the performance function.

The number of neurons in the input layer is equal to the number of attributes in the data set and the number of neurons in the output layer is equal to the number of classes (number of class attribute values) in the data set. Each data set was used to train ANNs with 2, 3, 5, and 10 neurons in the hidden layer. We used 10-fold cross validation in all experiments [4, 14]. Therefore, in total, the full set and each reduced set was used in 40 experiments. The figures in the tables below show the average results of the 40 experiments for the full set and for each reduced set.

In each experiment, we determined the number of training epochs using a validation set that consists of one third of the training data. The ANN was trained on two thirds of the training set until the error on the validation set starts rising. This procedure was repeated three times (3-fold cross validation) swapping each time the validation set by another third of the training instances. The number of training epochs performed before the error on the validation set starts rising was recorded each time. Finally, the ANN was trained on the whole training set for the average number of training epochs with an upper limit of 10000 epochs as a maximum.

4.1 The results of using noise-free training sets

In this section, we present the results we obtained using the full sets and the reduced sets without deliberately inserting any noisy examples. Fig. 2 summarizes the results in terms of the classification accuracy of the neural networks. The figure shows the average classification accuracy obtained on the 21 data sets while Tab. I shows the average classification accuracy for each benchmark data set obtained using the 4 ANNs (with 2, 3, 5, and 10 hidden neurons).

It is clear from Fig. 2 that training neural networks on the reduced sets obtained using noise filtering techniques gave the best results. The average classification accuracy obtained by the noise filtering algorithms ALL-kNN, ENN, and RENN is 71.13%, 71.35% and 71.68% respectively, while the average classification accuracy obtained using the full training set is only 67.87%.

ALL-kNN, ENN, and RENN gave better results (compared to the full set) on 18, 18, and 17 data sets out of the 21 data sets used respectively. These figures are

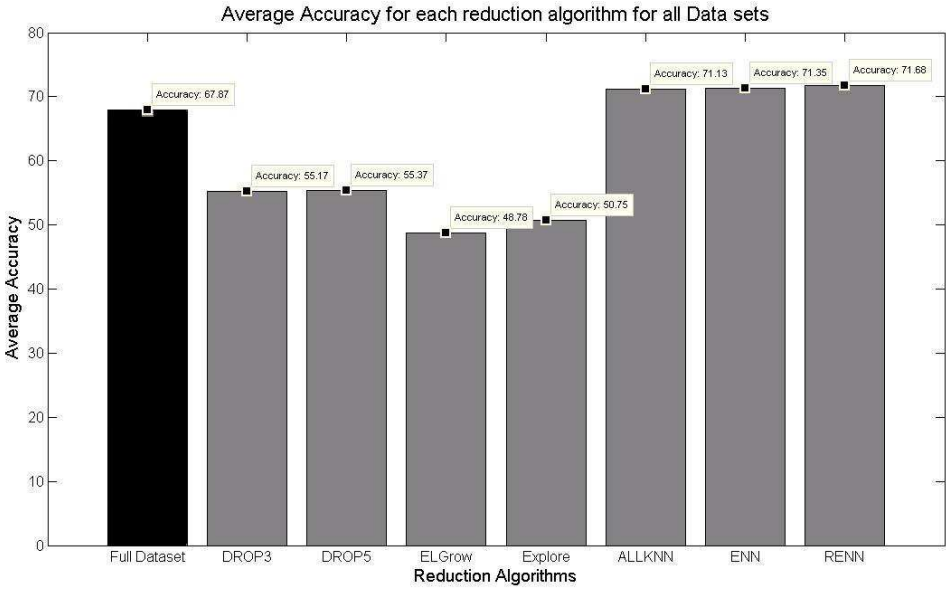


Fig. 2 *Accuracy per reduction algorithm over all data sets.*

shown in boldface in Tab. I. ALL-kNN, ENN, and RENN gave worse results only on 3, 3 and 4 data sets respectively.

In fact, the three algorithms gave significantly better results than the full set on 13 data sets, shown in boldface and with an asterisk in the table, at 99% significance level and a degree of freedom of 39 (since every data set was used in 40 experiments; i.e. 10 fold cross validation with four structures of 2, 3, 5, and 10 neurons in the hidden layer). ALL-kNN, ENN, and RENN gave significantly worse results on only 2, 1, and 2 data sets, respectively. These figures are marked with an asterisk in the table. On the remaining data sets the results were neither significantly better nor significantly worse.

This may sound surprising given the fact that no noisy examples were deliberately inserted, but in fact it is not. These algorithms remove near-border instances and these instances tend to be either noisy instances or untypical hard-to-learn instances. In either case, they are not good representatives of the instance space, and this is evident by the improvement in accuracy we obtained.

On the other hand, the average classification accuracy, obtained using the pure instance reduction techniques, i.e. ELGrow and Explore, are 48.78% and 50.75% respectively, which are much worse than the results obtained using the full sets. This is expected because they retain a very small proportion of the data sets. It seems that some of the instances they remove are typical instances that are useful for training the ANNs. The accuracy achieved EL-Grow, which is 48.78%, may seem worse than a random selection, but this would be the case only if all data sets contained only two class values (Boolean functions), but this is not always the case in the used data sets.

	Reduction Algorithms Classification Accuracy							
Data Set	**Full data set**	**DROP3**	**DROP5**	**ELGrow**	**Explore**	**ALL kNN**	**ENN**	**RENN**
Anneal	87.26	50.38	53.28	39.56	42.32	**88.88**	**89.55***	**88.19**
Australian	82.43	59.09	60.36	56.78	67.21	**84.82***	**83.84***	**84.57***
Breast Cancer (WI)	94.06	90.57	88.17	87.55	90.59	**96.67***	**96.81***	**96.92***
Flag	27.80	24.40	17.49	17.69	17.84	**31.05***	**28.65**	**32.49***
Glass	42.23	31.76	34.43	**42.40**	39.66	**49.07***	**48.55***	**49.10***
Heart	69.26	67.96	68.24	64.44	65.83	**80.19***	**80.83***	**82.04***
Heart (Cleveland)	70.72	64.91	67.80	60.55	65.46	**78.12***	**79.84***	**80.62***
Image Segmentation	71.67	36.37	31.96	29.17	25.89	**76.55***	**77.08***	**77.68***
Ionosphere	79.12	43.60	64.89	61.96	56.14	**84.53***	**83.60***	**84.73***
Iris	93.67	85.00	86.50	71.33	72.00	**96.00***	**95.67***	**95.67***
LED Creator	32.33	**46.35**	**33.33**	**38.00**	**46.48**	**46.98***	**49.58***	**48.55***
Liver (Bupa)	65.29	56.34	56.72	51.15	51.15	58.49*	60.28*	59.99*
Pima Diabetes	74.77	59.84	60.87	58.42	66.42	73.70	74.28	73.67*
Promoters	70.30	59.45	57.84	50.86	51.11	**73.50***	**70.80**	**70.50**
Sonar	71.42	68.79	68.10	60.92	60.51	**72.75**	**73.98***	**74.33***
Soybean (Large)	18.97	15.34	12.31	13.30	12.23	**27.28***	**27.13***	**31.09***
Vehicle	67.34	49.12	50.51	36.05	36.44	62.94*	**67.56**	66.96
Voting	92.51	91.55	76.93	65.38	64.54	**95.28***	**95.23***	**95.23***
Vowel	33.34	27.98	28.18	16.14	18.94	**34.10**	**34.15**	**33.44**
Wine	94.96	74.56	83.57	62.78	71.89	**95.65**	**95.67**	**95.67**
Zoo	85.83	55.28	61.39	40.00	43.06	87.22	85.28	83.89
Average	**67.87**	**55.17**	**55.37**	**48.78**	**50.75**	**71.13**	**71.35**	**71.68**

Tab. I *Accuracy achieved by the ANNs on the full and reduced sets for the used data sets.*

The classification accuracy of DROP3 and DROP5 are 55.17% and 55.37%. These algorithms are less effective at reducing the size of the data sets than Explore and ELGrow. They retain more examples than Explore and ELGrow, which explains why they gave better classification results. We believe that their results are not as good as those obtained by noise filters because they do remove some center instances which hurts the accuracy.

In short, our results indicate that the more training examples we present to the neural network the better accuracy we obtain. The only exception is when the noise filtering algorithms are used. They obviously reduce the size of the training sets, but at the same time, increase the classification accuracy.

The next issue to consider is the number of training epochs. If removing border instances makes learning easier, then it should reduce the number of training epochs. Fig. 3 summarizes the results in terms of required training epochs for the neural networks averaged over the used 21 data set, while Tab. II shows the average number of epochs needed to train the neural network for each benchmark data set to reach an acceptable low error. The results show that pure instance reduction techniques have the best results in reducing the number of epochs. The average number of epochs needed on the reduced sets produced by DROP3, DROP5, EL-Grow and Explore, are 1787.45, 2302.22, 134.97, and 163.47, respectively. These are huge reductions compared to the average number of epochs required on the full training sets which is 6406.58. They represent 28%, 36%, 2.1% and 2.55% of the number of epochs required to train the full data sets. This is expected since these algorithms produce really small data sets, making it a lot easier for the ANNs to learn or even memorise the training instances without extracting the important features necessary to classify new instances, and hence the reduction in accuracy.

However, the noise filtering algorithms (ALL-kNN, ENN and RENN) also achieved impressive results of 3497.15, 3615.41, and 3253.89 epochs, respectively. These figures represent 54.58%, 56.43%, and 50.79% of the number of epochs required on the full data sets. This is a substantial reduction by all means, confirming that learning became much easier after removing the border instances using these noise filtering algorithms. This reduces the training time not only because the number of epochs is smaller, but also, because at each epoch the ANN is presented with a smaller number of training instances.

4.2 The results of using noisy training sets

In this section, we discuss the results of applying reduction algorithms on noisy data sets. We deliberately inserted noise in the same 21 data sets that we used in the last section. Noise was inserted by changing the class of randomly chosen instances. We inserted 5%, 10% and 20% noise in each data set and re-performed the experiments described above. Namely, we applied the 7 reduction techniques on each noisy set and used the resulting reduced set for training the ANNs. We also used the full noisy data sets for training the ANNs for the sake of comparison. Again, we used ANNs with 2, 3, 5 and 10 hidden neuron in the hidden layer.

Tab. III shows the average classification accuracy we obtained using all ANNs trained using every data set with and without noise (0% noise, repeated from Tab. I for the ease of reference).

The results show that the accuracies of the ANNs trained using the full and the reduced sets degrades as the noise ratio increases, however, the noise filtering algorithms were consistent in giving better results than the full noisy sets regardless of the noise ratio. This is a clear indication that applying the noise filtering algorithms actually does remove some noisy instances and thus improves the classification accuracy.

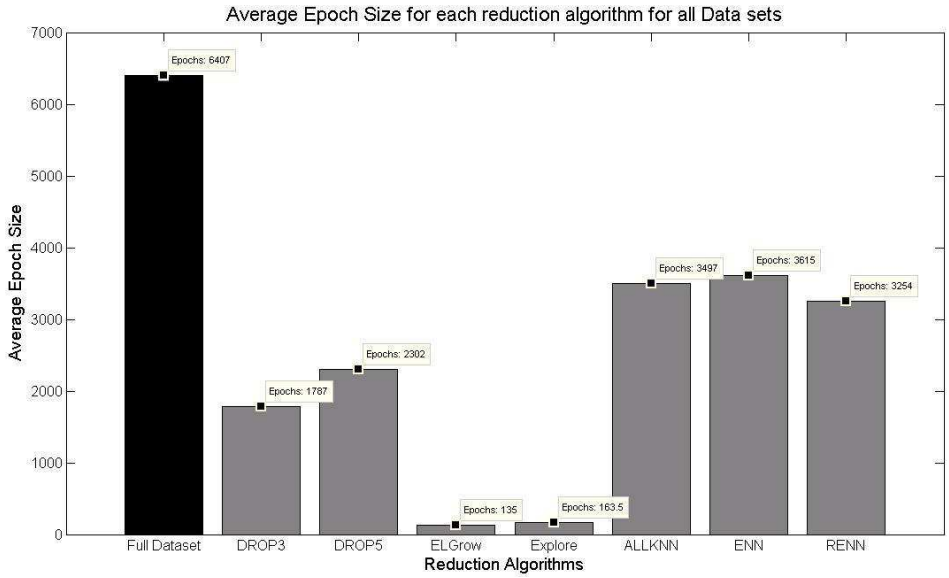


Fig. 3 *Required number of epochs per reduction algorithm over all data sets.*

The results show that the noise filtering algorithms were especially effective for small to moderate noise ratios. In fact, with 5% noise ratio, they gave better accuracy than the noise-free data sets and for 10% noise ratio they gave almost the same classification accuracy as the noise-free data sets. They only gave worse results than the noise-free data sets results with 20% noise ratio. This is understandable because these algorithms become less effective at eliminating noisy instances when there is a large number of them. This suggests that there is some room for further improvement.

It also shows that the RENN algorithm is the best noise filtering algorithm among the considered algorithms for our purpose of ANN training. Recall that the RENN repeatedly applies the ENN, until we are left with the instances that agree with the majority of their neighbors. This explains why it gave better results than the ENN; it was simply able to eliminate more noisy instances but it too found it more difficult to remove all noisy instances when there were many of them. Note that the ALL-kNN was even less effective than ENN. This is surprising because according to [20], ALL-kNN gave better results in terms of storage reduction and classification accuracy than ENN and RENN when used with the kNN algorithm.

Fig. 4 shows the difference in the achieved accuracy between the ANNs trained using the full data sets with 0%, 5%, 10% and 20% noise and ANNs trained using the reduced sets. It is evident from the figure that as the noise ratio increases, the effectiveness of the noise filtering algorithms becomes more apparent (i.e. the gap in accuracy between the ANNs trained on the full noisy sets and ANNs trained on the reduced sets increases). Again, the best noise filtering algorithm, for our purposes, is RENN. On the other hand, the ANNs trained on the full data set

	Reduction Algorithms Training Epochs							
Data Set	**Full data set**	**DROP3**	**DROP5**	**ELGrow**	**Explore**	**ALL kNN**	**ENN**	**RENN**
Anneal	5013.50	651.18	1169.45	9.98	9.20	3220.70	2829.85	3209.30
Australian	10000.00	58.50	983.95	6.25	6.25	1150.83	2327.45	818.20
Breast Cancer (WI)	8800.38	48.73	64.05	7.60	6.85	82.78	76.45	71.70
Flag	6928.70	49.25	191.75	7.80	11.20	2517.78	2184.85	941.85
Glass	9694.93	3936.75	4017.90	8.65	12.65	8273.88	9548.20	8630.43
Heart	9484.33	60.28	309.60	6.65	6.30	891.95	742.90	227.55
Heart (Cleveland)	9826.20	84.90	337.60	6.38	6.53	1318.58	2391.15	1082.50
Image Segmentation	9502.03	532.13	2401.80	22.78	22.43	6932.78	7627.05	7124.38
Ionosphere	4835.03	45.78	50.58	5.90	6.18	129.98	514.80	88.43
Iris	4409.85	330.18	454.03	8.65	9.38	830.78	1039.28	1084.33
LED Creator	6091.43	2580.13	3614.58	303.55	277.60	3615.45	3169.48	2860.10
Liver (Bupa)	9771.73	7248.35	7106.95	6.38	6.95	9899.05	9197.23	8856.43
Pima Diabetes	9754.25	6223.70	9541.68	8.28	7.40	9372.13	9357.18	9167.15
Promoters	53.08	19.05	16.85	5.78	5.88	47.40	46.25	46.90
Sonar	1050.58	42.25	127.75	5.65	5.75	140.55	82.50	81.83
Soybean (Large)	7350.55	1899.20	1938.88	11.30	13.63	7135.60	6939.18	5895.68
Vehicle	9551.79	6678.03	7276.00	24.10	23.60	9239.23	9481.68	9346.33
Voting	3816.83	30.68	51.03	6.00	6.70	111.05	108.23	108.23
Vowel	8071.90	6938.33	8628.20	2357.98	2973.63	8142.70	7870.98	8301.78
Wine	362.03	44.58	35.28	6.38	6.53	328.85	330.40	330.40
Zoo	169.15	34.43	28.68	8.30	8.25	58.23	58.58	58.33
Average	**6406.58**	**1787.45**	**2302.22**	**134.97**	**163.47**	**3497.15**	**3615.41**	**3253.89**

Tab. II *The average number of epochs required by the ANNs on the full and reduced sets.*

has better accuracy than the ANNs trained on the reduced sets produced using DROP3, DROP5, ELGrow, and Explore.

The increase in the number of training epochs is another indication that noisy examples do actually make learning more difficult. Tab. IV shows that they increase the number of required training epochs. Tab. IV shows the average number of

Noise Ratio	Full data set	DROP3	DROP5	ELGrow	Explore	ALL kNN	ENN	RENN
0%	67.87	55.17	55.37	48.78	50.75	71.13	71.35	71.68
5%	63.07	56.78	55.29	51.26	52.83	70.41	70.75	70.95
10%	57.42	54.85	52.33	49.95	51.84	67.08	67.70	67.82
20%	50.46	50.78	45.65	47.56	48.16	59.07	60.51	62.64
Average	**56.98**	**54.14**	**51.09**	**49.59**	**50.94**	**65.52**	**66.32**	**67.14**

Tab. III *The results of training the ANNs using the reduced noisy data sets.*

training epochs with and without noise (0% noise, repeated from Tab. II for the ease of reference). Note that the number of required training epochs tends to increase as the noise ratio increases.

The noise filtering algorithms proved to be effective at reducing the number of training epochs. It is true that pure instance reduction algorithms DROP3, DROP5, ELGrow and Explore require even fewer training epochs but this came at the expense of the classification accuracy as is evident from Tab. III. The noise filtering algorithms, on the other hand, maintain a good balance between accuracy and training speed. With respect to the number of training epochs, RENN emerged, once again, as the winner requiring only 43.23% of training epochs required on the full noisy data sets. The other two noise filtering algorithms ALL-kNN and ENN required 47.74% and 52% of the training epochs required on the full sets.

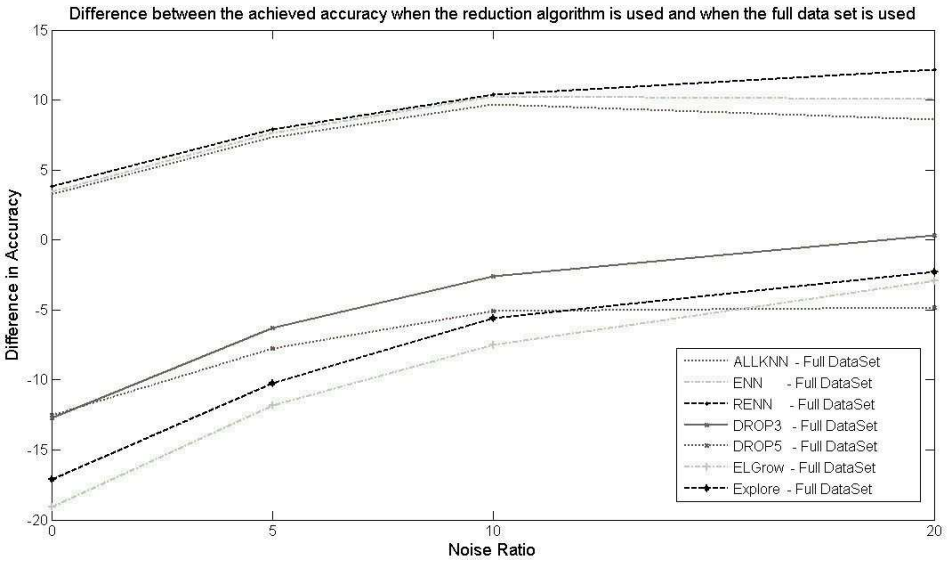


Fig. 4 *The difference in accuracy between the full noisy sets and the filtered sets.*

Noise Ratio	Full data set	DROP3	DROP5	ELGrow	Explore	ALL kNN	ENN	RENN
0%	6060.17	1697.72	2210.41	134.67	163.06	3160.51	3289.21	2975.00
5%	7736.07	1888.98	2378.54	170.87	176.50	3446.86	3853.15	3274.46
10%	7805.09	1774.95	2551.21	110.21	166.81	3701.51	3957.99	3438.10
20%	8130.45	1976.56	3240.55	78.20	134.91	4153.35	4499.24	3519.53
Average	**7890.54**	**1880.17**	**2723.43**	**119.76**	**159.41**	**3767.24**	**4103.46**	**3410.70**
Percentage	**100.00%**	**23.83%**	**34.52%**	**1.52%**	**2.02%**	**47.74%**	**52.00%**	**43.23%**

Tab. IV *The number of training epochs required for noisy data sets.*

5. Conclusions

Our empirical results confirmed our hypothesis that eliminating border instances makes learning easier since these instances represent either noisy or untypical hard-to-learn instances. This was evident because we obtained better classification accuracy when we trained the ANNs after eliminating these border instances. Also, training required substantially smaller number of epochs which means less training time. Training require less time, also, because at each epoch an ANN is presented with fewer examples. ALL-kNN, ENN, and RENN proved to be very effective algorithms at eliminating border instances. They proved to be especially effective when the training sets contained a small to moderate noise ratios. RENN proved to be the best algorithm in achieving better classification accuracy and, at the same time, requiring the smallest number of training epochs and thus increasing the training speed. On the other hand, pure instance reduction techniques such as DROP3, DROP5, ELGrow, and Explore, gave bad classification results.

References

[1] Aha D. W., Kibler D., Albert M. K.: Instance-Based Learning Algorithms. Machine Learning,**6**, 1991, pp. 37–66.

[2] Aha, D. W. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms, International Journal of Man-Machine Studies, **36**, 1992, pp. 267–287.

[3] AL-Akhras M., Zedan H., John R., ALMomani I.: Non-intrusive speech quality prediction in VoIP networks using a neural network approach, Neurocomputing, **72**, 10-12, 2009, pp. 2595–2608.

[4] AL-Akhras M., ALMomani I., Sleit A.: An Improved E-MODEL Using Artificial Neural Network VoIP Quality Predictor, NEURAL NETWORK WORLD, **21**, 1, 2011, pp. 3–26.

[5] Askarunisa A., Ramaraj N.: An Algorithm for Test Data Set Reduction for Web Application Testing, NEURAL NETWORK WORLD, **21**, 1, 2011, pp. 27–43.

[6] Blake C., Merz C.: UCI Repository of Machine Learning Databases [Online]. Irvine, CA: University of California, Department of Information and Computer Science, 1998. Available from: http://www.ics.uci.edu/ mlearn/MLRepository.html.

[7] Cameron-Jones R. M.: Instance Selection by Encoding Length Heuristic with Random Mutation Hill Climbing. In: Proceedings of the Eighth Australian Joint Conference on Artificial Intelligence, 1995, pp. 99–106.

[8] El Hindi K., AL-Akhras M.: Eliminating Border Instance to Avoid Overfitting. IADIS International Conference Intelligent Systems and Agents 2009 (ISA 2009), 21–23 June, 2009, Algarve, Portugal.

[9] Garcia S., Cano J. R., Herrera F.: A Mimetic Algorithm for Evolutionary Prototype Selection: a scaling up approach. The Journal of The Pattern Recognition Society, **41**, 8, 2008, Elsevier Ltd. 2008.

[10] Gates G. W.: The Reduced Nearest Neighbor Rule. IEEE Transactions on Information Theory, **18**, 3, 1972, pp. 431–433.

[11] Hart P. E.: The Condensed Nearest Neighbor Rule. IEEE Transactions on Information Theory, **14**, 1968, pp. 515–516.

[12] Levenberg K.: A Method for the Solution of Certain Non-Linear Problems in Least Squares, The Quarterly of Applied Mathematics, **2**, 2, 1944, pp. 164–168.

[13] Marquardt D. W.: An Algorithm for the Least-Squares Estimation of Nonlinear Parameters, SIAM Journal of Applied Mathematics, **11**, 2, June 1963, pp. 431–441.

[14] Mitchell T. M.: Machine Learning, 1997, New York, NY, USA McGraw-Hill.

[15] Quinlan J. R.: C4.5: Programs for machine learning, 1993, Morgan Kaufmann, San Mateo, CA.

[16] Ritter G. L., Woodruff H. B., Lowry S. R., Isenhour T. L.: An Algorithm for a Selective Nearest Neighbor Decision Rule. IEEE Transactions on Information Theory, **21**, 6, November 1975, pp. 665–669.

[17] Tomek I.: An Experiment with the Edited Nearest-Neighbor Rule. IEEE Transactions on Systems, Man, and Cybernetics, **6**, 6, June 1976, pp. 448–452.

[18] Wilson D. L.: Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. IEEE Transactions on Systems, Man, and Cybernetics, **2**, 3, 1972, pp. 408–421.

[19] Wilson D. R., Martinez T. R.: Improved Center Point Selection for Radial Basis Function Networks. In: Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA'97), 1997.

[20] Wilson D. R., Martinez T. R.: Reduction Techniques for Exemplar-Based Learning Algorithms. Machine Learning, **38**, 3, 2000, pp. 257–286.