Dirasat, Pure Sciences, Volume **32**, No. **1**, 2005

Instance Reduction Techniques for Arabic and Hindu Digit Recognition

Khalil M. el Hindi, Eman F. Issa and Areej M. Abu Kar **

ABSTRACT

Instance-based learning is competitive, in terms of classification accuracy, in many applications to more sophisticated learning techniques such as neural networks. However, instance based learning requires storing a large training set which requires large storage space and hence long classification time. Several general instance reduction techniques have been developed to deal with this problem. This work presents several instance reduction techniques that are designed for the digit recognition problem. The techniques construct a set of prototypes, which are retained instead of the complete training set. We also present a distance function that is used to measure the distance between an instance and a prototype. The new methods are compared with some of the best-known reduction techniques on the handwritten digit recognition problem. The results show that the techniques presented provide the best combination of amount of reduction and classification accuracy for both Arabic and Hindu digits.

KEYWORDS: Instance-Based Learning, Instance Reduction, K Nearest Neighbors Algorithm, Hand-written Digit Recognition.

1. INTRODUCTION

In Hindi et al. (2004A), a machine's ability to recognize Hindu and Arabic digits was compared using several machine learning techniques: the first nearest neighbor (Aha et al., 1991), Naive Bayesian (Michie et al., 1994), and feed forward neural networks using the Rprop algorithm (Riedmiller et. al., 1993). The obtained classification accuracies, using the three algorithms, were much higher for Hindu digits. It was also found that, for Hindu and Arabic digit recognition, the classification accuracy of the first nearest neighbor algorithm is competitive to the neural network approach and much better than that of the Naïve Bayesian method.

The K Nearest Neighbor (KNN) is a simple form of Instance-Based Learning. IBL systems can be efficiently and easily trained; the training phase consists of simply retaining a number of classified examples, which is called the training set or instance memory. However, to produce good classification accuracy Instance-Based learning systems, usually, retain a large number of classified instances. This does not only require a large storage space but also slows down the classification process, which may hinder their applicability in real life applications, such as the recognition of vast amounts of digits.

This problem was addressed in the literature using different approaches: Instance reduction techniques (Wilson et al., 2000), Indexing techniques (Wess et al., 1993), and early stopping techniques (Hindi, 2003).

In Hindi (2004B) we addressed the problem of long classification time in a different way. We combined three classification methods to classify Hindu and Arabic digits. The methods are Naïve Bayesian, and neural network and Instance-Based learning. The idea was to use Instance-Based learning (the slow method) only when the other faster methods fail to classify a new instance with sufficient certainty. However, that method did not deal with the problem of the high memory requirement of Instance-Based learning.

Instance reduction techniques attempt to retain only the important instances and discard the rest. This usually comes at a cost, a reduction in classification accuracy (Wilson et al., 2000 for an excellent comparison of such techniques). Indexing techniques, on the other hand, attempt to index instances to speed up the retrieval

* KASIT, University of Jordan, Amman, Jordan. Received on 10/3/2004 and Accepted for Publication on 16/9/2004.

process. Unfortunately, these techniques are suitable when the number of attributes is relatively small and become less effective when the number of attributes increases. Hindi (2003) presented some early stopping criteria that can be used to avoid searching the whole instance memory. Just like indexing, early stopping techniques do not attempt to reduce the number of retained instances; they just attempt to speed up the classification process. In Hindi (2004C) a comparison was made between early stopping techniques and some instance reduction techniques with respect to the classification accuracy and the percentage of the training set that needs to be searched or retained.

This work presents new reduction techniques dedicated to the digit recognition problem, where each instance is represented as a bit vector. The techniques produce a set of prototypes that are retained instead of the training set. A new instance is classified using the most similar prototype. This work also presents a new distance metric that is used to measure the similarity (distance) between an instance and a prototype. The results show that the new reduction techniques provide the best compromise of amount of reduction and classification accuracy.

Section 2 describes instance-based learning while section 3 reviews some of the best known instance reduction techniques. Section 4 presents some new instance reduction techniques and the distance function we use to measure the distance between an instance and a prototype. Section 5 compares the new techniques and some of the best known instance reduction methods. Section 6 summarizes our conclusions and highlights some directions for future work.

2. The K-Nearest Neighbor Algorithm (KNN)

The KNN algorithm is a simple form of instance-based learning. Instance-based learning is a machine-learning method that retains a number of classified examples in an instance memory. To classify a new instance, the KNN algorithm retrieves the k nearest neighbor instances and assumes that the class with majority votes is the class of the new instance. k is usually chosen as an odd integer such as 1 or 3.

Table 1: A sample of handwritten Arabic and Hindu digits.

Digit	Arabic	Hindu
Zero		
One		
Two		
Three		
Four		
Five		
Six		
Seven		
Eight		
Nine		

Despite being simple, instance-based learning can deliver good classification accuracy in many applications, comparable to those achieved by more sophisticated approaches such as neural networks and decision trees (Stanfill et al., 1986; Hindi et al., 2004A). Instance-Based learners are good when the target function is complex but can be approximated using several local functions.

The instance memory (the training set) is usually represented as a table, where each row represents a classified instance and each column represents an attribute (feature). One attribute is designated as the class attribute.

To measure the similarity between two instances, a similarity (distance) function is used (Stanfill et al., 1986; see also Wilson et al., 1997 for an excellent survey of such functions). The function that we use in this work, to measure the distance between two instances, is the HVDM function (Wilson et al., 1997).

$$HVDM(X,Y) = \sqrt{\sum_{a=1}^{m} d_a^2(x_a, y_a)}$$

where *X* and *Y* are two instances, *m* is the number of attributes, x_a and y_a are the values for attribute *a* in instances *X* and *Y*, respectively. The distance function, d_a, between two values depend on the type of the attribute (symbolic or numeric):

$$d_a(x,y) = \begin{cases} vdm_a(x,y), & \text{if } a \text{ is symbolic else} \\ \dfrac{|x-y|}{a_{\max} - a_{\min}} & \text{if } a \text{ is numeric} \end{cases}$$

$$vdm_a(x,y) = \sum_{c=1}^{C} \left(\frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right)^2$$

where:

- $a_{\max}$ and $a_{\min}$ are the maximum and minimum values of attribute *a*.
- $N_{a,x}$ is the number of instances in the training set that has the value *x* for the attribute *a*.
- $N_{a,x,c}$ is the number of instances in the training set that have value *x* for attribute *a* and belong to output class *c*.
- *C* is the total number of output classes in the problem domain.

The main drawback of instance-based learning is the need to store many instances, which requires large storage space and increases the classification time. Another drawback is that they do not produce high level description of concepts that can be easily understood by people such as rules or decision trees and also they usually fail to provide good explanation. The justification for their classifications is usually 'that is the class of the most similar example(s)'.

3. Instance Reduction Techniques

Many instance reduction techniques were developed over the years to reduce the storage requirements and speed up the classification time of instance-based learning. Wilson et al. (2000) presents an excellent comparison between many of these techniques. Instance reduction techniques can be classified into two categories incremental and decremental. Incremental techniques start with an empty reduced set and augment it with more instances to improve classification accuracy. On the other hand, decremental techniques start with the complete training set and remove from it instances that seem to have minimal effect on the classification accuracy.

Wilson et al. (2000) compare these techniques with respect to the amount of reduction they achieve and the generalization accuracy obtained. They conclude that ELGrow (Caemron-Jones, 1995) and Explore (Caemron-Jones, 1995) are among the best reduction techniques with respect to the amount of reduction achieved. While ENN and RENN are among the best techniques with respect to the classification accuracy obtained. DROP3 (Wilson et al., 2000) have the best mix of storage reduction and classification accuracy while DROP5 (Wilson et al., 2000) seems to have the best classification accuracy. In the following subsections, we review each of these methods.

The Edited Nearest Neighbor Rule

The Edited Nearest Neighbor (ENN) (Wilson, 1972) algorithm is a decremental algorithm. It starts with the whole training set and removes from it any instance that does not agree with the majority of its k nearest neighbors (k is usually 3). It, therefore, eliminates noisy instances as well as instances that are close to enemy instances (instances of a different class). It does not greatly reduce the storage requirement. It is, therefore, considered as a noise elimination method rather than a storage reduction method. The Repeated ENN (RENN) is essentially the same as ENN but it repeatedly applies the above process until all remaining instances have the same class as the majority of their neighbors. Therefore, RENN is a more effective storage reduction technique.

Encoding-Length-Based Methods

Cameron-Jones (1995) developed what he calls "Pre/All" method, which was later called the ELGrow method[1]. This method consists of two phases a growing and a pruning phase. In the growing phase, each instance in the training set is added to the reduced set if doing so results in a lower cost than not adding it. This phase is followed by a pruning phase, during which instances in the reduced set are removed if doing so lowers the cost. The cost is estimated as follows

$$COST(m, n, x) = F(m, n) + m\log_2(C) + F(x, n - m) + x\log_2(C - 1)$$

where n is the number of instances in the training set, m is the number of instances in the reduced set, and x is the number of instances seen so far that are misclassified

[1] The ELGrow name was given to this method by Wilson (2000) to better distinguish from other methods.

by instances in the reduced set. F(m,n) is the cost of encoding which m instances from the n available are retained and is defined as

$$F(m,n) = \log^*(\sum_{j=0}^{m} C_j^m) = \log^*(\sum_{j=0}^{m} \frac{n!}{j!(n-j)!})$$

where $\log^*(x)$ is the sum of the positive terms of $\log_2(x)$, $\log_2(\log_2(x))$, etc.

The Explore method (Cameron-Jones, 1995) adds a third phase, the mutation phase. During this phase 1000 mutations are performed. Each mutation attempts to add an instance to the reduced set, remove an instance from it, or swap an instance in the reduced set with an instance that was left out. The resulting change is retained only if it lowers the cost.

Both the ELGrow method and the Explore method are quite good with respect to the amount of reductions performed. Moreover, the generalization accuracy of the Explore method is good.

The DROP Family of Techniques

Wilson et al. (2000) presents several algorithms DROP1-DROP5 that take careful note of the order in which instances are removed. Before we review these techniques, we need to introduce some notation. The nearest enemy of an instance P is the nearest instance with a different class. The associates of an instance P are those instances that have P as one of their k nearest neighbors.

DROP1

DROP1 (Decremental Reduction Optimization Procedure 1) removes an instance, P, if at least as many of its associates in the reduced set would be classified correctly without it.

DROP2

DROP2 removes an instance, P, only if at least as many of its associates in the original training set (including those that may have been pruned) would be classified correctly without it. Also, DROP2 attempts to remove center instances (that are far from enemy instances) before boarder instances (that are close to enemy instances). This is done by sorting the instances in the reduced set by the distance to their nearest enemy.

DROP3

Because DROP2 removes center instances, noisy instances, which are typical border instances, may be retained. This makes DROP2 sensitive to noisy instances. Therefore, DROP3 uses a noise filtering technique, such as ENN before applying DROP2.

DROP4

DROP4 is the same as DROP3 but it uses a more careful noise filtering phase. The noise-filtering pass removes an instance if it satisfies two conditions: 1) if it is misclassified by its k nearest neighbors, and 2) if it does not hurt the classification of other instances.

DROP5

DROP5 modifies DROP2 by adding a pre-pass to it. During this new pass instances that are near their enemy are removed first and proceeds outwards. After this pass, DROP2 is applied until no further improvement can be made. The first pass serves to remove noisy instances in addition to many internal instances.

4. Bit-Vector Dedicated Reduction Techniques

In this section, we present four instance reduction techniques that are suitable for digit instances, where each digit is presented as a bit vector. The basic idea is to build a prototype for each cluster of similar instances. The prototype is simply the sum of all these instances (which are bit vectors). The methods, therefore, differ from the general instance reduction techniques in that the resulting reduced set is a set of prototypes not actual instances. The methods are all incremental; they start with an empty reduced set and augment it with more prototypes. The methods differ mainly in the way they produce clusters.

The first method, which we call I2I (Instance to Instance), begins with a randomly chosen seed instance as a prototype. It repeatedly finds the nearest instance to this instance, and adds it (vector addition) to the prototype if it is of the same class (digit). This neighbor is then marked as covered. This process is repeated until the next nearest instance belongs to a different class. Then another uncovered seed is selected and the process continues until all instances are covered by some prototype. The complete algorithm is presented figure 1.

The second method, which we call I2I2, adds another phase to the above method. In the second phase, each incorrectly classified instance in the training set is added to the set of prototypes as a new prototype. This phase increases the size of the reduced set and improves the classification accuracy.

Algorithm I2I

Input: A training set T.

Output : a set of prototypes (the reduced set).

```
Prototypes={};      // begin with an empty reduced set
mark all instances as uncovered
For each instance I in T
        If I is uncovered then
            let P = I
            mark I as covered
            P.participants=0; // this is used to count the
                              //size of the cluster covered by P
        let N be nearest neighbor of I;
        While N is uncovered and N.class = P.class do
            P = P + N;  / /vector addition
            mark N as covered
            P.participants ++;
            let N be next nearest neighbor of I;
Prototypes = Prototypes + P; // add P to the reduced set
Return Prototypes
```

Figure 1: The algorithm finds a set of prototypes, where each prototype is the sum vector of a cluster of similar instances.

The third technique which we call I2P (Instance to Prototype), differ from the first technique in the way it finds the next neighbor. It finds the next neighbor to the current prototype, that is being constructed, not to the seed instance. The fourth technique, which we call I2P2, adds a second phase to the third technique, which is the same as the second phase in I2I2 i.e. all instances in the training set that are incorrectly classified by I2P are added to the reduced set.

4.1. Measuring the Distance between an Instance and a Prototype

It is essential for the presented techniques to measure accurately the distance between an instance, *X*, and a prototype, *P*. This function is used during the reduction phase by algorithms *I2P* and *I2P2*, which measure the distance between the prototype that is being constructed and training instances. It is also used during the classification (generalization) phase, to find the distance between a new (previously unseen) instance and the retained prototypes. We developed the following distance function which gave good results.

$$dist(X, P) = \sum_{a=1}^{m} W_a * (VDM_a(x_a, p_a) + 1)$$

where

x_a is the value of attribute a of instance X.

p_a is 1 if the actual value of attribute a of prototype P is greater than 0 and 0 otherwise.

W_a is the weight of the attribute a in prototype P and is calculated as follows

$$W_a = \begin{cases} 1 - \dfrac{P_a}{L_p} \; if & X_a = 1 \\ \dfrac{P_a}{L_p} & if \; X_a = 0 \end{cases}$$

where, L_p is the number of instances that were used to construct prototype P. We also call them P's participating instances.

The function takes two factors into consideration when measuring the distance between an instance and a prototype with respect to attribute a. The first is the *VDM* distance between $X_{a,,}$ and p_a, and the second is the weight of P_a itself. P_a is a nonnegative integer value that represents the sum of the corresponding attribute value of P's participating instances. P_a is a value between 0 and the total number of participating instances. It is 0 if the corresponding attribute value of all participating instances is 0 and it is the same as the number of the participating instances (if the corresponding attribute value is 1 in all participating instances).

Of course, the VDM value is computed for the values that appear in the training instances, which are 0 and 1. Therefore, if the P_a value is more than 1 it is considered 1 and the VDM distance between X_a and 1 is used.

The value P_a also serves as a weight for attribute a. If P_a is high that means many of the participating instances have 1 as a value for attribute a. Therefore, if the corresponding value in instance X is 1 then instance X should be considered similar to prototype P. Accordingly, the VDM distance is reduced by multiplying it by a value less than 1. The weight value $1-(Pa/L_p)$ proved to be good. If P_a is the same as L_a then the weight is 0.

However, if the P_a value is high while the corresponding X_a value is 0 then the *VDM* distance is slightly decreased. This is done by multiplying the *VDM* distance by the weight value P_a/L_a.

5. Comparing the Different Techniques

In this section, we present the results we obtained using all different reduction techniques on the Hindu and Arabic hand written digits. In these experiments, we used two data sets of Arabic and Hindu digits. The first consists of 2000 hand-written Arabic digit images (instances) and the other consists of 2000 hand-written Hindu digit images. Twenty volunteers took part in building the datasets. Each volunteer was asked to write down each Hindu and Arabic digits 10 times. These digits were later scanned as 30 by 30 bitmap images. These images were right aligned, and no other preprocessing step was performed on them.

Ten-fold cross validation technique was used in all experiments. In each experiment, 90% of each data set was used as a training set and the remaining 10% was used as a test set. This was repeated 10 times for each algorithm, replacing each time the 10% test set with another 10% of the training set.

Table (2) shows the classification accuracies and the size of the reduced sets we obtained using each of the above techniques. The first row shows the classification accuracies obtained using the first nearest neighbor algorithm using the complete data sets (100%). The table shows that the ENN and RENN approaches are the worst reduction techniques with respect to the amount of reduction achieved. On the other hand, ELGrow and Explore are the best reduction techniques in terms of the amount of reduction achieved, however the classification accuracies achieved are not that good. DROP2 achieves the best combination in accuracy and amount of reduction among the DROP family of techniques. These results are all consistent with those reported by Wilson et al. (2000), using other datasets. The table also shows that three of the presented methods, namely I2I2, I2P, and I2P2 achieve better classification accuracies than all other techniques. The best of these techniques is the I2P2 technique, which achieves classification accuracies close to that achieved by the first nearest neighbor algorithm (1NN), but using only 39.94% and 52.01% of the original training sets in the Hindu and Arabic cases respectively. In fact, the classification accuracy of the I2P2 technique is lower than that of the first nearest neighbor algorithm only by 0.85% in the Hindu case and by 1.9% in the Arabic case.

Our results are also consistent with those reported in Hindi et al. (2004A), in that Hindu digits are actually easier to recognize than Arabic digits. Not only we have better classification accuracy in the Hindu case, but also most reduction techniques achieved larger amount of reduction in the Hindu case. The only exceptions are in the case of the ENN and RENN algorithms, where we achieved larger amount of reduction in the Arabic case. In fact, this also indicates that the Hindu digits are easier to recognize than Arabic digits. Recall that ENN and RENN are considered noise filtering techniques (because they remove instances that are inconsistent with their nearest neighbors), therefore, the fact that they eliminate more Arabic instances indicate that the two algorithms find them more noisy than Hindu digits, and hence the lower classification accuracy.

Table 2: The classification accuracy (Accr %) and the percentage of the retained dataset (size%) using different reduction techniques.

Technique	Hindu		Arabic	
	Accr%	Size %	Accr%	Size %
1NN	95.15	100	85	100
ENN	92.35	94.16	79.8	84.28
RENN	92.3	93.89	79.3	80.06
ELGrow	81.5	4.14	64.45	4.19
Explore	84.35	5.07	68.3	5.51
Drop1	80.65	10.8	70.9	15.37
Drop2	90.5	18.37	77.15	21.44
Drop3	84.45	17.52	77.15	21.44
Drop4	89.9	18.13	77.65	22.67
Drop5	88.4	14.79	76.55	20.46
I2I	83.5	25.9	76.25	41.31
I2I2	90.55	30.63	80.7	47.02
I2P	93.85	39.61	83.1	51.51
I2P2	94.15	39.94	83.1	52.01

6. Conclusion

In this work, we presented new instance reduction techniques that are designed for the digit- recognition problem. We compared the new techniques with some of the best-known reduction techniques. Our results show that the new techniques achieve the best combination of classification accuracy and amount of reduction. The I2P2 algorithm yields classification accuracy close to that obtained by the first nearest neighbor algorithm, while retaining and thus searching only 39.94% and 52.01% of the training set in the Hindu and Arabic cases, respectively. Our results are also consistent with those reported in Hindi et al. (2004A) in that the Hindu digits are easier to recognize. They have higher classification

accuracy and the size of the reduced Hindu set (by most reduction techniques) is smaller than the reduced Arabic set. We expect that the techniques presented can be useful for similar pattern recognition problems such as the letter recognition problem. How effective these techniques are for such problems is a subject for future research.

REFERENCES

Aha, D. W., Kibler, D. and Albert, M. K. 1991. Instance Based Learning Algorithm, *Machine Learning*, 6: 37-66.

Caemron-Jones, 1995. Instance Selection by Encoding Length Heuristic with Random Mutation Hill Climbing, *Proc. of the Eighth Australian Joint Confion AI*, 99-106.

Hindi, K., Abu Kar, A. and Al Qaddomi, G. 2004A. Comparing the Machine Ability to Recognize Hand-Written Hindu and Arabic Digits, Submitted to Pattern Recognition Letters.

Hindi, K., Abu Kar, A. 2004B. Combining Heterogeneous Classifiers for Classifying Hand-Written Hindu and Arabic Digits, *Dirasat*, 31 (2).

Hindi, K. 2004C. Instance-Based Digit Recognition, Submitted to *AMSE*, France.

Hindi, K. Early-Halting Criteria for Instance-Based Learning, *Proc. of ACS/IEEE International Conference on Computer Systems and Applications*, AICCSA03, Tunisia.

Michie, D., Spiegelhalter, D. J. and Taylor, C. C. 1994. Machine Learning, Neural and Statistical Classification, (edited collection). New York: Ellis Horwood.

Riedmiller, M., Braun, H. 1993. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm, *Proc. of the IEEE International Conference on Neural Networks, San Francisco*, 586-591.

Stanfill, C. and Waltz, D. 1986. Toward Memory-based Reasoning. *Communications of the ACM*, 29:12: 1213-1228.

Wess, Stefan, Klaus-Dieter Althoff, and Michael M. Ritcher, 1993. Using k-d Trees to Improve the Retrieval Step in Case-Based Reasoning. Topics in Case-Based Reasoning, *First European Workshop*, Springer-Verlag, 67-181.

Wilson, D. R. and Martinez, T. R. 2000. Reduction Techniques for Instance-Based Learning Algorithms. *Machine Learning*, 38: 257-286.

Wilson, D. R. and Martinez, T. R. 1997. Improved Heterogeneous Distance Functions. *Journal of Artificial Intelligence Research (JAIR)*, 6 (1):1-34.

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

طرق جديدة لتقليل عدد الأمثلة مخصصة للأرقام العربية والهندية

خليل هندي، وايمان عيسى، واريج أبو كار*

ملخص

يعتبر أسلوب التعلم المعتمد على الامثلة من الطرق البسيطة التي تنافس في كثير من التطبيقات أساليب أكثر تعقيدا مثل الشبكات العصبية الاصطناعية وذلك من حيث دقة التصنيف، ولكنها تتطلب تخزين العديد من الامثلة مما يتطلب سعة تخزينية عالية ويقلل سرعة التصنيف. وقد تم تصميم العديد من الطرق لتقليل عدد الامثلة وذلك عن طريق تحديد الامثلة المهمة والاحتفاظ بها فقط. يقدم هذا البحث طرق تقليل جديدة مخصصة للارقام العربية والهندية تقوم ببناء نماذج مناسبة يحتفظ بها بدلا من الاحتفاظ بجميع الامثلة، كما يقدم البحث مقياسا لقياس درجة التشابه بين هذه النماذج والامثلة. وقد تمت مقارنة هذه الطرق الجديدة بالعديد من الطرق المشهورة لتقليل الامثلة على الارقام العربية والهندية وقد ثبت أنها توفر أفضل توازن بين دقة التصنيف ونسبة الامثلة المحتفظ بها في هذا المجال.

* كلية الملك عبد الله الثاني لتكنولوجيا المعلومات، الجامعة الأردنية. تاريخ استلام البحث 2004/3/10، وتاريخ قبوله 2004/9/16.