Chapter 34
Large-Scale Instance Selection Using a Heterogeneous Value Difference Matrix

Chatchai Kasemtaweechok, Nitiporn Sukkerd, and Chatchavin Hathorn

Abstract Data classification of a large-scale dataset is a common problem nowadays because the classifier model takes an overwhelming amount of time to completely learn all the data. The instance selection algorithm is a well-known technique that addresses this issue by reducing the size of the training set. Instance selection methods decrease the difficulty of data classification and improve the quality of the training data. This paper proposed a novel instance selection method using a heterogeneous value difference matrix (HVDM) distance function. The proposed method selected a set of median HVDM values in each partition as a reduced training set. We compared the proposed method with the condensed nearest neighbor (CNN) and instance-based learning (IB3) methods. Five large-scale datasets from the UCI data repository were tested with three classifier models (decision tree, neural net, and support vector machine). The accuracy and kappa of the proposed method were better than those of the other two methods, and the proposed method had a moderate reduction rate. However, the accuracy and kappa of the proposed method were nearly equal to those of the original training set.

34.1 Introduction

The extent of Internet commerce provides many extremely large datasets from which information can be extracted using data mining techniques. In addition, many operations in everyday life, such as mobile transactions by a large number of clients, can lead to the generation of large amounts of data in a database system. Even if these data are useful for data classification, the large amount of data makes the classifier model inappropriate for these problems. Because of the complexity of the learning process, it can take excessive time to create a classifier model. Moreover, the large size of the dataset results in the learning process requiring a large amount of memory space.

C. Kasemtaweechok (✉) · N. Sukkerd · C. Hathorn
Faculty of Science at Sriracha, Kasetsart University, Sriracha Campus, Sri Racha, Thailand
e-mail: Chatchai.kase@ku.th

S.-L. Peng et al. (eds.), *Sensor Networks and Signal Processing*, Smart Innovation, Systems and Technologies 176, https://doi.org/10.1007/978-981-15-4917-5_34

Data reduction is recognized as an effective method to improve the learning process for large training datasets. The instance selection algorithm is one of the most common data reduction techniques to address this difficulty. Instance selection is the main technique that decreases the complexity of data classification and improves the quality of the training data because it removes missing, redundant, and noise data from a training set. Accordingly, the classifier model focuses on collecting only the instances that have affected the classification score [1]. The selected instances enable the classifier model to predict unseen with nearly equal accuracy to the original training set.

In this paper, a new approach is presented to select a subset of the training data to keep as representative of the training set using a heterogeneous value difference matrix (HVDM) called HVDM-IS. The training set is separated into disjoint partitions based on the number of instances in the dataset. In each partition, a class representative is selected which minimizes the sum of HVDM values to other instances. After the selection process is complete, the subset of training data presents the whole quality of the data in the original training set. The proposed method was evaluated on five large-scale datasets, and the performance was compared with two other algorithms. The performance of the full training set was used as the baseline value. The reduction rate shows the reduction capacity of all concerned algorithms. Accuracy and kappa were measured from three classified models: decision tree, neural net, and support vector machine.

The rest of this paper is as follows. Section 34.2 reviews briefly previous instance selection methods. The proposed method is described in Sect. 34.3. The experimental materials and methods are explained in Sect. 34.4. The results of this experiment are shown in Sect. 34.5. Lastly, Sect. 34.6 presents the conclusion.

34.2 Instance Selection

Instance selection methods can help classifier models by reducing the size of a training set. The reduction scheme of the previous instance selection methods can be categorized into three schemes: condensation, edition, and hybrid [2].

Condensation selection focuses on collecting instances near the decision boundaries. Condensed nearest neighbor (CNN) focuses on removing the instances that are correctly classified by their nearest neighbors [3]. The group of remaining instances is collected as a consistent subset. The condensation methods collect a small amount of training data, but it is sensitive to noise.

Edition selection removes the border instances and outlier instances. Edited nearest neighbor (ENN) removes instances that are misclassified in the original training set [4]. ENN removes instances that are noisy or disagree with neighbors with high accuracy. However, the reduction rate of the edition scheme is low.

Hybrid selection includes the strengths of the above two methods with internal and noise removal processes. Hybrid selection removes noise better than condensation selection and selects a subset of training data that is smaller than for edition selection.

Instance-based learning (IB3) [5] collects the subset of instances which provide a good classification record. IB3 removes the instances which have a poor classification record later. The iterative case filtering (ICF) algorithm removes the instances which have the number of nearest neighbors in the same class greater than the number of nearest enemies (the nearest neighbors of a different class) [6]. The distances to the nearest neighbors and enemies are also used in the concept of a local set. The local set of an instance x is the set of nearest instances in the same class where the distance to x is shorter than the distance between x and its nearest enemy. The idea of a local set is used as the selecting criterion in some methods [7, 8]. Moreover, the hit miss networks (HMN) method selects instances using the graph that has a directed edge from each instance to the nearest neighbors of each different class [9]. The hit degree of a node is the number of edges directed to the same class node. The miss degree is the number of edges directed to a different class node. The HMN method removes instances if the miss degree value is greater or equal to the hit degree value. The classification accuracy of the hybrid selection is comparatively higher than those of the condensation and edition selections.

In recent years, a novel instance selection algorithm has been used that selects representative instances in each partition based on the nearest enemy information near the decision boundary [10]. Furthermore, a density-based algorithm for instance selection analyzes the density of instances in each class and keeps only the densest instances of a given neighborhood within each class [11]. Some researchers presented a new method to select a representative instance of each of the densest spatial partitions [12]. In addition, a novel instance selection method uses metric learning for transforming the input space which addresses the decision boundaries between classes. The inter-class and intra-class separation criteria are used to select the instances near to the decision boundaries [13]. The above-mentioned methods achieved high classification accuracy and reduction rates when they were tested with datasets having less than 20,000 instances. Because of the small size of the tested datasets, these instance selections did not allow any scalability analysis.

34.3 Proposed Method

This section presents the idea of the heterogeneous value difference matrix instance selection (HVDM-IS) algorithm. Figure 34.1 shows the process flow of HVDM-IS, which has two main processes: the suitable partition size calculation (SPSC) process and the median of HVDM value selection (HVDMS) process. The first process calculates a suitable partition size and forwards it onto the next process. The second process uses the number of partitions for splitting the training data. Lastly, it selects the median HVDM value as the class representative in each partition.

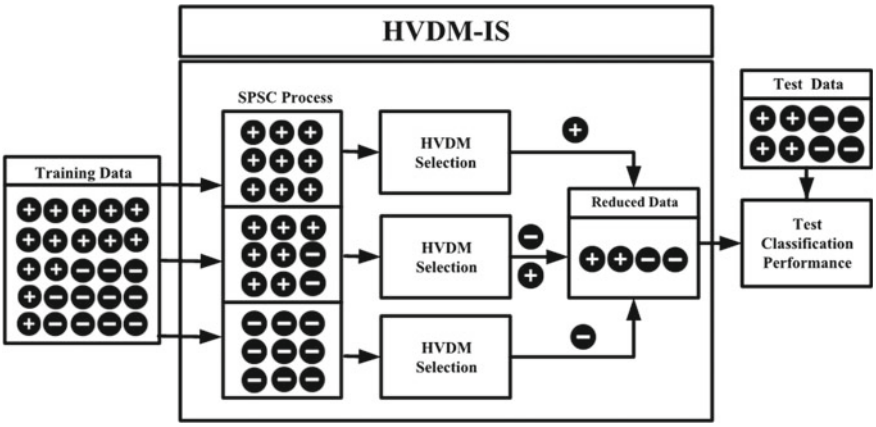

Fig. 34.1 Process flow of the HVDM-IS algorithm

34.3.1 Suitable Partition Size Calculation

The first process uses the Taro Yamane formula to calculate the number of selected instances n. The formula of Taro Yamane is shown in (34.1). Consequently, the HVDM-IS calculates the number of instances in each partition and divides the training data into n disjoint partitions. Because of its simplicity, the SPSC process is able to calculate the suitable partition size rapidly [14].

$$n = \frac{N}{1 + N(e^2)} \tag{34.1}$$

where n is the number of selected instances, N is the population (number of the overall training data) size, and e is the level of precision.

34.3.2 HVDM Selection

This section describes two related subjects: the heterogeneous value difference matrix (HVDM) and the HVDM selection process. First, the definition and formula of the HVDM distance value are explained. Lastly, the concept and pseudocode of the HVDM selection process are described.

HVDM was introduced by Wilson and Martinez [15] to compute the distance between two input vectors, and it was designed to overcome the weakness of the Euclidean distance function and the value difference matrix (VDM) distance function. The formula based on Euclidean distance works well when the attributes are linear (continuous or discrete). However, the Euclidean distance function is not suitable for nominal attributes because the values in nominal attributes are not necessarily

in any linear order. The values of some nominal attributes normally were converted from category name to a numeric value such as low (0), medium (1), and high (2) so they are not suitable for computing the numerical difference between the two values.

The VDM distance function focuses on providing an appropriate distance function for nominal attributes [16]. The conditional probability of each attribute value given a class is used to evaluate the distance between two input vectors. Each value of a nominal attribute usually appears many times among training instances. The probability calculation uses the number of occurrences of a value in each nominal attribute so the VDM is suitable for nominal attributes. However, the VDM is inappropriate for continuous attributes because of their large value range. Accordingly, the values of a continuous attribute can all potentially be unique.

The HVDM function returns the distance between two input instances x and y. It is defined as follows [15]:

$$\text{HVDM}(x, y) = \sqrt{\sum_{a=1}^{m} d_a^2(x_a, y_a)} \tag{34.2}$$

where m is the number of attributes. The function $d_a(x_a, y_a)$ returns the distance between x and y for attribute a and is defined as follows:

$$d_a^2(x_a, y_a) = \begin{cases} 1, & \text{if x or y is unknown;} \\ normalized\_vdm_a(x_a, y_a), & \text{if } a \text{ is nominal;} \\ normalized\_diff_a(x_a, y_a), & \text{if } a \text{ is linear} \end{cases} \tag{34.3}$$

The $d_a(x_a, y_a)$ function combines two distance functions for nominal and linear distance calculation. The HVDM function uses $normalized\_diff$ when the attribute is linear (discrete or continuous value). The function $normalized\_diff$ is shown in (34.4):

$$normalized\_diff_a(x_a, y_a) = \frac{|x_a - y_a|}{4\sigma_a} \tag{34.4}$$

where σ_a is the standard deviation of the numeric values of attribute a.

The function $normalized\_vdm$ is used for nominal distance calculation. The function $normalized\_vdm$ uses the square of the difference that is similar to the Euclidean distance function. The function $normalized\_vdm$ is shown in (34.5):

$$normalized\_diff_a(x_a, y_a) = \sqrt{\sum_{i=1}^{C} \left| \frac{N_{a,x,i}}{N_{a,x}} - \frac{N_{a,y,i}}{N_{a,y}} \right|^2} \tag{34.5}$$

where C is the number of classes, $N_{a,x}$ is the number of instances in the training set that have value x for attribute a, $N_{a,x,i}$ is the number of instances in the training set that have value x for attribute a, output class i, $N_{a,y}$ is the number of instances in the

training set that have value y for attribute a, and $N_{a,y,i}$ is the number of instances in the training set that have value y for attribute a and output class i.

After the training data have been divided into disjoint partitions, the HVDMS process focuses only on the selection of class representative in each partition. The method for finding the class representative is the HVDM distance value computation from a candidate to other data points in the same partition. A selected instance is a class representative that minimizes the sum of HVDM values to other instances in each partition. The pseudocode of HVDMS is shown as Algorithm 1.

There are three input parameters to the HVDMS process. First, TS is the list of instances in the current partition. Second, NS is the number of subsets in the current partition. Finally, R is the number of iterations, starting from 1 to R. In each partition, the selection process splits the training data into NS subsets. The $CreateInitialCandidates$ function randomly selects a candidate in each subset. The $CalculateSumHVDM$ function calculates the sum of the distance from the candidate to other points. After the set of selected candidates CS has been created, the $SelectRoundWinner$ function selects a candidate that has the smallest sum of HVDM distance as RW for each round-winner. In the next iteration, the $GetNearestCandidatestoRW$ function creates CS as a new set of candidates which is the nearest neighbors to the round-winner. The $CalculateSumHVDM$ and $SelectRoundWinner$ functions are called for finding RW, the new round-winner. The HVDMS process continuously repeats the above steps until the number of specified iterations is achieved. The result of the HVDMS process is S_{best} which is the instance minimizing the sum of HVDM distances to other instances.

Algorithm 1 Pseudocode of HVDM selection.

```
1: function HVDM SELECTION(TS, NS, R)
2:     CS ← CREATEINITIALCANDIDATES(TS, NS)
3:     Dist_cs ← CALCULATESUMHVDM(CS, TS)
4:     RW, Dist_rw ← SELECTROUNDWINNER(CS, Dist_cs)
5:     S_best ← RW
6:     Dist_best ← Dist_rw
7:     for N = 1 to R do
8:         CS ← GETNEARESTCANDIDATESTORW(RW, TS, NS)
9:         Dist_cs ← CALCULATESUMHVDM(CS, TS)
10:        RW, Dist_rw ← SELECTROUNDWINNER(CS, Dist_cs)
11:        if Dist_rw ≤ Dist_best then
12:            S_best ← RW
13:            Dist_best ← Dist_rw
14:        end if
15:    end for
16:    return S_best
17: end function
```

34.4 Experimental Materials and Methods

In this section, we describe the details of our experimental evaluation. Section 34.4.1 shows the list of benchmarking methods and datasets used in the experiment. The definitions of various performance measures and classification algorithms used in the evaluation of the HVDM-IS method are explained in Sect. 34.4.2.

34.4.1 Benchmarking Methods and Datasets

We evaluated the performance of HVDM-IS with three compared algorithms. The parameter settings of the compared methods are shown in Table 34.1.

The five datasets from the UCI data repository are shown in Table 34.2 [17]. In each test, 80% of the original dataset was used for training data, and the rest was used for test data. The compared algorithms were run over the training data to create a reduced training set.

Table 34.1 Parameter settings of compare methods

Compared methods	Parameters
Condensed nearest neighbor (CNN)	Mixed Euclidean distance
Instance base 3 (IB3)	Parameter $k = 3$, Upper interval = 0.9, Lower interval = 0.7, Mixed Euclidean distance
Hit miss networks (HMN-EI)	Epsilon = 0.1, Euclidean distance
HVDM-IS	Number of iterations = 5, Number of subsets = 10

Table 34.2 Description of dataset

Dataset	Number of samples	Number of attributes (Real/Integer/Nominal)	Number of classes
Fars	100,968	29 (5/0/24)	8
Census	299,284	41 (1/12/28)	3
KDD cup	494,020	41 (26/0/15)	23
Covertype	581,012	54 (0/54/0)	7
Poker	1,025,010	10 (0/10/0)	10

34.4.2 Performance Measures

The performance of HVDM-IS was compared with the other methods using three performance measures: reduction rate, accuracy, and Cohen's kappa. First, the reduction rate represents the reduction of storage capacity obtained by the method. A higher reduction rate shows that the algorithm can reduce the training data better. Second, accuracy shows the percentage of correctly classified instances. Accuracy is the most common performance indicator of classification methods. Finally, Cohen's kappa evaluates the ratio of correctly classified instances that can be attributed to a classifier itself by recompensing for random correctly classified instances. This measure is in the range from -1 to 1. A larger kappa value shows that the rating of compliance between the predicted label and the actual label is higher. The formula of Cohen's kappa is shown in (34.6) [18]:

$$\text{Kappa} = \frac{N\sum_{i=1}^{r} x_{ii} - \sum_{i=1}^{r}(x_{i+} \times x_{+i})}{N^2 - \sum_{i=1}^{r}(x_{i+} \times x_{+i})} \tag{34.6}$$

where r is the number of rows or columns in the confusion matrix, x_{ii} is entry (i, i) of the confusion matrix, x_{i+} and x_{+i} are the marginal totals of row i and column i, respectively, and N is the total number of examples in the confusion matrix.

We used three classification algorithms to evaluate the performance of the HVDM-IS and the compared methods: decision tree (J48), neural net (NET), and support vector machine (SVM). First, J48 is a Java program of the C4.5 algorithm in the Weka data mining tool [19]. C4.5 is a widely used decision tree algorithm proposed by Quinlan [20]. C4.5 builds decision trees from a training set using information entropy. Second, the NET algorithm is a multilayer perceptron (MLP) that is a feed-forward artificial neural network model trained by back propagation algorithms [21, 22]. Lastly, we used the LIBSVM algorithm that is a popular open-source library for support vector machines [23, 24]. LIBSVM supports multiclass learning and probability estimation by Platt calibration for transforming the outputs of a classification model into confidence values of predicted classes [25].

34.5 Results

In this section, we show the details of our results. Section 34.5.1 shows the reduction rate of HVDM-IS and the benchmarking methods used in the study. The classification accuracy and Cohen's kappa are reported for the evaluation of the HVDM-IS method in Sect. 34.5.2. Finally, a trade-off between the reduction rate and classification performance is discussed in Sect. 34.5.3.

34.5.1 Reduction Rate

The reduction rate shows the reduction capability of the instance selection method. It is an important performance indicator in the data reduction. From Table 34.3, the IB3 method had the highest reduction rate (89.15%) because it includes two sample removal processes: one for noise and the other for poor classification score instances for 1NN performance. Otherwise, the reduction rate of the HVDM-IS was nearly 80% of the training data size. The reduction rate of HVDM-IS was higher than that of the CNN method. The selection process of HVDM-IS focuses on the quality of the training data, so the size of the selected instances from HVDM-IS is quite large.

34.5.2 Classification Accuracy

Table 34.4 shows the classification accuracy by J48 and the standard error of HVDM-IS, the full training model, CNN, and the IB3 method. The CNN method had the best average accuracy rate. However, the size of the reduced training sets of CNN was large compared with that of the other methods. The HVDM-IS method provided the best accuracy rate on two datasets. The accuracy rate of the HMN-EI method was approximately 2.5%, higher than that of the HVDM-IS method. However, the average accuracy rate of HVDM-IS was approximately 5% lower than that of the full training model.

In Table 34.5, the average classification accuracy by NET of HVDM-IS was higher than those of the CNN and IB3 methods by about 5%. The HVDM-IS and CNN methods provided the best accuracy on two datasets. The average accuracy of the HMN-EI method was nearly equal to that of the HVDM-IS method. However, the average accuracy of HVDM-IS was nearly equal to that of the full training model.

The classification accuracy achieved by LIBSVM of HVDM-IS was much higher than by the CNN, IB3, and HMN-EI methods, as shown in Table 34.6. The classifi-

Table 34.3 Reduction rate of HVDM-IS, CNN, IB3, and HMN-EI methods

Dataset	HVDM-IS	CNN	IB3	HMN-EI
Fars (%)	77.13	54.65	74.99	47.27
Census (%)	87.07	47.73	97.66	79.03
KDD cup (%)	97.80	99.68	99.85	90.18
Covertype (%)	67.65	88.84	93.38	46.20
Poker (%)	67.49	48.52	78.89	13.25
Average (%)	79.43	67.88	89.15	55.19
Standard error	0.0584	0.1007	0.0496	0.1339

Table 34.4 Classification accuracy by J48 on five datasets

Dataset	Full training	HVDM-IS	CNN	IB3	HMN-EI
Fars (%)	79.88	78.87	79.39	78.79	79.73
Census (%)	95.65	94.64	94.56	86.64	94.20
KDD cup (%)	99.96	99.11	99.29	99.50	99.95
Covertype (%)	96.80	90.80	79.77	77.68	91.82
Poker (%)	76.31	61.40	73.24	62.69	72.02
Average (%)	89.72	84.96	85.25	81.06	87.54
Standard error	0.0483	0.0678	0.0496	0.0602	0.0509

Table 34.5 Classification accuracy by NET on five datasets

Dataset	Full training	HVDM-IS	CNN	IB3	HMN-EI
Fars (%)	46.80	47.11	42.03	34.54	46.89
Census (%)	95.12	93.20	94.78	86.83	92.56
KDD cup (%)	99.89	99.55	99.60	99.63	99.87
Covertype (%)	81.67	79.34	62.84	74.30	79.00
Poker (%)	52.92	54.03	54.54	51.47	53.24
Average (%)	75.28	74.65	70.76	69.35	74.31
Standard error	0.1084	0.1042	0.1131	0.1179	0.1050

Table 34.6 Classification accuracy by LIBSVM on five datasets

Dataset	Full training	HVDM-IS	CNN	IB3	HMN-EI
Fars (%)	45.05	45.12	41.37	40.27	44.90
Census (%)	94.84	94.24	94.53	89.14	93.04
KDD cup (%)	99.36	99.09	44.91	37.60	99.42
Covertype (%)	82.69	86.10	86.96	86.49	84.02
Poker (%)	50.12	50.12	42.10	50.30	50.12
Average (%)	74.41	74.93	61.97	60.76	73.99
Standard error	0.1131	0.1137	0.1182	0.1125	0.1477

cation accuracy for LIBSVM of HVDM-IS was higher than that of the full training model. The average accuracy of the HVDM-IS method was nearly equal to that of the HMN-EI method as shown in Fig. 34.2.

34.5.3 Cohen's Kappa

Table 34.7 shows the values for Cohen's kappa and the standard error for J48, the full training model, the HVDM-IS model, and the other methods. The results indicated

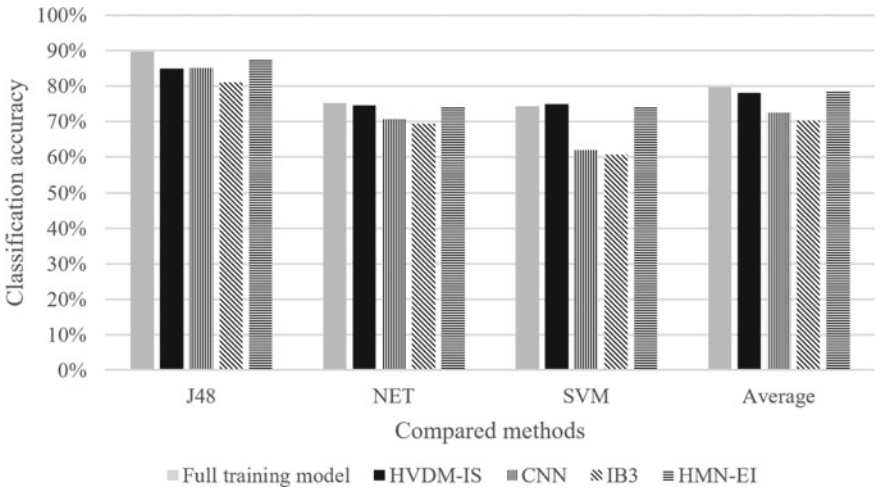

Fig. 34.2 Average classification accuracy of HVDM-IS and the compared methods

Table 34.7 Cohen's kappa by J48 on five datasets

Dataset	Full training	HVDM-IS	CNN	IB3	HMN-EI
Fars	0.7260	0.7120	0.7190	0.7110	0.7230
Census	0.4960	0.4080	0.4660	0.3350	0.5250
KDD cup	0.9990	0.9850	0.9880	0.9920	0.9990
Covertype	0.9350	0.8520	0.6770	0.6470	0.8680
Poker	0.5720	0.3050	0.5190	0.3690	0.4760
Average	0.7456	0.6524	0.6738	0.6108	0.7182
Standard error	0.0982	0.1293	0.0916	0.1207	0.0993

that CNN method had the best kappa value for three out of five datasets. The Cohen's kappa of HVDM-IS was higher than that of the IB3 method. However, the Cohen's kappa of HVDM-IS was lower than that of full training model.

In Table 34.8, NET of the HVDM-IS method yielded the best Cohen's kappa that was higher than those of the CNN and IB3 methods by about 5%. Furthermore, the Cohen's kappa of the HVDM-IS method was higher than that of full training model too. For LIBSVM, Cohen's kappa of the HVDM-IS method was much higher than those of the CNN and IB3 methods as shown in Table 34.9. The average Cohen's kappa of the HVDM-IS method was nearly equal to that of the HMN-EI method. Finally, Fig. 34.3 shows the average Cohen's kappa values of the HVDM-IS method for the three classifier models were higher than those of the CNN and IB3 methods.

Table 34.8 Cohen's kappa by NET on five datasets

Dataset	Full training	HVDM-IS	CNN	IB3	HMN-EI
Fars	0.2270	0.2500	0.2120	0.1530	0.2500
Census	0.3930	0.3880	0.4290	0.3000	0.4420
KDD cup	0.9980	0.9920	0.9930	0.9940	0.9980
Covertype	0.6290	0.6650	0.4320	0.5830	0.6540
Poker	0.1120	0.1180	0.1280	0.1030	0.0790
Average	0.4718	0.4826	0.4288	0.4266	0.4846
Standard error	0.1577	0.1563	0.1509	0.1646	0.1603

Table 34.9 Cohen's kappa by LIBSVM on five datasets

Dataset	Full training	HVDM-IS	CNN	IB3	HMN-EI
Fars	0.2170	0.2330	0.2080	0.1980	0.2140
Census	0.2360	0.0780	0.1390	0.2000	0.3520
KDD cup	0.9890	0.9850	0.2990	0.2190	0.9900
Covertype	0.7160	0.7740	0.7870	0.7810	0.7390
Poker	0.0000	0.0000	0.0000	0.0000	0.0000
Average	0.4316	0.4140	0.2866	0.2796	0.4350
Standard error	0.1820	0.1966	0.1343	0.1316	0.1571

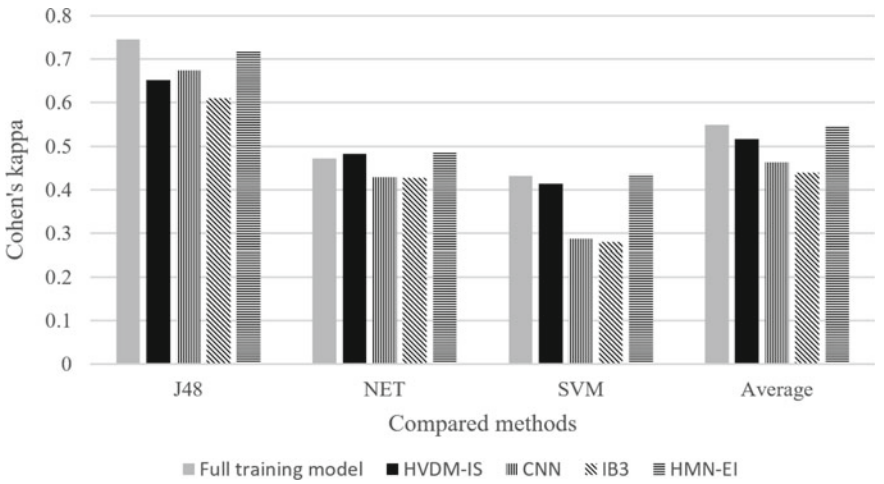

Fig. 34.3 Average Cohen's kappa of HVDM-IS and the compared methods

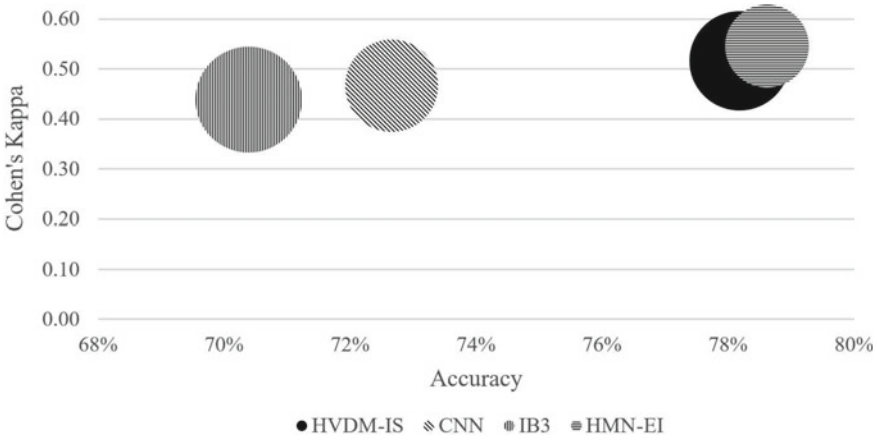

Fig. 34.4 Graphical comparison of accuracy (x-axis), Cohen's kappa (y-axis), and reduction rate (size of bubble) of HVDM-IS and the compared methods

34.5.4 Trade-Off Between Accuracy and Reduction Rate

The average classification accuracy of the HVDM-IS method was higher than the CNN (72.66%) and IB3 (70.39%) methods. Figure 34.4 presents a bubble chart providing a graphical comparison of the accuracy, Cohen's kappa, and reduction rates of the HVDM-IS and the compared methods. The HVDM-IS method had the second-highest reduction rate (79.43%). Even though the IB3 method had the highest reduction rate (89.15%), the classification accuracy of the IB3 method was the lowest. Moreover, the accuracy rate of HVDM-IS (78.18%) was nearly equal to that of the HMN-EI method (78.62%) but the HMN-EI method had a low reduction rate (55.19%).

The HMN-EI method had the highest average Cohen's kappa value (0.55) while the HVDM-IS method had an average Cohen's kappa value (0.52) which was higher than those of the CNN (0.47) and IB3 (0.44) methods. Although the HVDM-IS method had the second-highest reduction rate, it could handle the trade-off between accuracy, Cohen's kappa, and the reduction rate reasonably well, as shown in Fig. 34.4.

34.6 Conclusion

A new method was proposed to select a subset of the training data to keep as a representative training set using the heterogeneous value difference matrix (HVDM) method, called HVDM-IS. The classifier model creation takes an overwhelming amount of time when a large-scale training dataset is being processed. The data

were split into independent partitions. In each partition, a class representative was selected which minimized the sum of HVDM values to other instances. The selected instances were used for the training classification model. The accuracy and kappa were measured from three classified models: decision tree (J48), neural net (NET), and support vector machine (LIBSVM).

The results of this experiment showed that the HVDM-IS method provided classification accuracy and Cohen's kappa values that were higher than those of the CNN and IB3 methods for the NET and LIBSVM classifier models. For the J48 classifier model, the accuracy and Cohen's kappa value of the HVDM-IS method were a little lower than those of the CNN method, while the number of selected instances from the CNN method was larger. Furthermore, the classification accuracy and Cohen's kappa of the HVDM-IS method were nearly equal to those of the full training model and the HMN-EI method. However, the HMN-EI method had a quite low reduction rate.

The results of this experiment showed that the HVDM distance can help the instance selection method to calculate the appropriate distance value because the distance function combination in HVDM is applicable to the characteristics of nominal and linear distance calculation.

In future work, the HVDM-IS method could be applied in a parallel and distributed processing system, which should improve the processing speed of the method. Furthermore, we would aim to reduce the large real-world datasets with HVDM-IS and show its scalability for the big data problem.

Acknowledgements This work was financially supported by the Faculty of Science at Sriracha, Kasetsart University, Sriracha campus, Thailand.

References

1. García, S., Luengo, J., Herrera, F.: Data Preprocessing in Data Mining. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-319-10247-4
2. García, S., Derrac, J., Cano, J.R., Herrera, F.: Prototype selection for nearest neighbor classification: taxonomy and empirical study. IEEE Trans. Pattern Anal. Mach. Intell. **34**, 417–435 (2012). https://doi.org/10.1109/TPAMI.2011.142
3. Hart, P.E.: The condensed nearest neighbor rule. IEEE Trans. Inf. Theory. **14**, 515–516 (1968). https://doi.org/10.1109/TIT.1968.1054155
4. Wilson, D.L.: Asymptotic properties of nearest neighbor rules using edited data. IEEE Trans. Syst. Man. Cybern. **SMC–2**, 408–421 (1972). https://doi.org/10.1109/TSMC.1972.4309137
5. Aha, D.W., Kibler, D., Albert, M.K.: Instance-based learning algorithms. Mach. Learn. **6**, 37–66 (1991). https://doi.org/10.1007/BF00153759
6. Brighton, H., Mellish, C.: Advances in instance selection for instance-based. Data Min. Knowl. Discov. **6**, 153–172 (2002). https://doi.org/10.1023/A:1014043630878
7. González, A.A., Pastor, D.F.J., Rodríguez, J.J., Osorio, G.C.: Local sets for multi-label instance selection. Appl. Soft Comput. **68**, 651–666 (2018). https://doi.org/10.1016/j.asoc.2018.04.016
8. Leyva, E., González, A., Pérez, R.: Three new instance selection methods based on local sets: a comparative study with several approaches from a bi-objective perspective. Pattern Recognit. **48**(4), 1523–1537 (2015). https://doi.org/10.1016/j.patcog.2014.10.001

9. Marchiori, E.: Hit miss networks with applications to instance selection. J. Mach. Learn. Res. **9**, 97–1017 (2008). https://doi.org/10.5555/1390681.1390715
10. Yu, G., Tian, J., Li, M.: Nearest neighbor-based instance selection for classification. In: 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), pp. 75–80. IEEE, New Jersey (2016). https://doi.org/10.1109/FSKD.2016.7603154
11. Carbonera, J.L., Abel, M.: A novel density-based approach for instance selection. In: IEEE 28th International Conference on Tools with Artificial Intelligence, pp. 549–556. IEEE, New Jersey (2016). https://doi.org/10.1109/ICTAI.2016.0090
12. Carbonera, J.L., Abel, M.: Efficient instance selection based on spatial abstraction. In: IEEE 30th International Conference on Tools with Artificial Intelligence, pp. 286–292. IEEE, New Jersey (2018). https://doi.org/10.1109/ICTAI.2018.00053
13. Max, Z.E., Marcacini, M.R., Matsubara, T.E.: Improving instance selection via metric learning. In: 2018 International Joint Conference on Neural Networks (IJCNN), IEEE, New Jersey (2018). https://doi.org/10.1109/IJCNN.2018.8489322
14. Yamane, T.: Statistics: An Introductory Analysis, 2nd edn. Harper and Row, New York, USA (1967)
15. Wilson, D.R., Martinez, T.R.: Improved heterogeneous distance functions. J. Artif. Intell. Res. **6**, 1–34 (1997)
16. Stanfill, C., Waltz, D.: Toward memory-based reasoning. Commun. ACM. **29**, 1213–1228 (1986). https://doi.org/10.1145/7902.7906. ACM, New York
17. UCI machine learning repository, http://archive.ics.uci.edu/ml
18. Triguero, I., Derrac, J., Garcia, S., Herrera, F.: A taxonomy and experimental study on prototype generation for nearest neighbor classification. IEEE Trans. Syst. Man. Cybern. **42**, 86–100 (2012). https://doi.org/10.1109/TSMCC.2010.2103939
19. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. SIGKDD. Explor. **11**(1), 10–18 (2009). https://doi.org/10.1145/1656274.1656278
20. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, MA (1993)
21. Rosenblatt, F.: Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington, DC (1961)
22. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. In: Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1, pp. 318–362. MIT Press, Cambridge (1986)
23. Corinna, C., Vladimir, V.: Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995). https://doi.org/10.1023/A:1022627411411
24. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. **2**(3), 1–27 (2011). https://doi.org/10.1145/1961189.1961199
25. Platt, J.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. Adv. Large Mar. Classif. **10**(3), 61–74 (1999). https://doi.org/10.1007/s10994-007-5018-6