JOURNAL OF KOREA MULTIMEDIA SOCIETY, VOL. 9, NO. 6, JUNE 2006(pp. 762-772)

Instance Based Learning Revisited: Feature Weighting and its Applications

Doo Heon Song[†], Chang Hun, Lee[††]

ABSTRACT

Instance based learning algorithm is the best known lazy learner and has been successfully used in many areas such as pattern analysis, medical analysis, bioinformatics and internet applications. However, its feature weighting scheme is too naïve that many other extensions are proposed. Our version of IB3 named as eXtended IBL (XIBL) improves feature weighting scheme by backward stepwise regression and its distance function by VDM family that avoids overestimating discrete valued attributes. Also, XIBL adopts leave-one-out as its noise filtering scheme. Experiments with common artificial domains show that XIBL is better than the original IBL in terms of accuracy and noise tolerance. XIBL is applied to two important applications - intrusion detection and spam mail filtering and the results are promising.

Keywords: Instance base learning, Feature weighting, Value Distance Metric, machine learning, noise filtering

1. INTRODUCTION

The term "instance based learning" is often used confusingly to refer to either a specific algorithm that David Aha had developed [1] or a learning model that stores a part (or all) of example instances in training phase and uses k-nearest neighbor principle in test phase. A large population of k-nearest neighbor research community often regarded the latter as an extension of edited k-nearest neighbor algorithm with symbolic attributes and therefore overlooked its creative paradigm and/or effectiveness in machine learning problems.

※ Corresponding Author : Doo Heon Song, Address : (449-710) 571-1 Mapyong Dong Yong-In Yong-In SongDam College, TEL : +82-31-330-9231, FAX : +82-31-330-9239, E-mail : dsong@ysc.ac.kr
Receipt date : April 18, 2006, Approval date : June 13, 2006
[†] Dept. of Computer Games & Information, Yong-in SongDam College, Yong-In, Kyungki Do, 449-710, Korea
[††] Dept. of Computer Engineering, Konkuk University, Korea
(E-mail : chlee@konkuk.ac.kr)

However, a set of instance based algorithms as a machine learning paradigm has developed with common characteristics that it delays the processing of its input information until it has a specific request (*defer*) and replies to queries by combining information from its stored examples (*demand-driven*) and delete the constructed query and any intermediate results (*discard*)[2]. Thus, machine learning algorithms with above characteristics are now referred to "***lazy learners***" in that they do nothing but store examples until a query arrives to the system. Other learning algorithms which find a set of rules or structures (e.g. decision tree, neural network) in training are called as "***eager learners***" in contrast [3]. Thus, the term "instance based learning (IBL)" in this paper only refers to the pioneer work of a lazy learning paradigm done by David Aha.

Just like C4.5 [4] among many decision tree based algorithms, IBL has been used as its original form in many recent interesting problems such as anti-spam filter [5,6], medical problems [7], and pattern recognition problems [8]. People also have put a great deal of efforts to increase its function-

alities [9,10] and to extend its representation to rule set [11], relational learning [12] or probabilistic formalization [13].

Very often, the original IBL was criticized by its feature-weighting scheme, which assumes a uniform distribution of weights and updates them by simple reward/penalty function with misclassified instances. People have tried to improve the original IBL by searching for the right weights of attributes with various bias and evaluation functions [2,9] or by using conditional probability [10].

In this paper, we also try to improve the feature-weighting scheme by a search with statistical measure - backward stepwise regression. We will also improve the original distance function and measurement by cognitive scientific results. [14, 15] We applied our version of the extended IBL (XIBL) to two practical problems - Intrusion Detection [16,17] and Korean Spam-mail Filter [18] and obtained promising results.

In section 2, we explain the original IBL and its family. Then, we describe the extensions we give to the IB3, which is the most popular implementation of IBL family algorithms in section 3, followed by the experimental results with theoretical common data sets [19] in section 4. Two practical applications that used XIBL as an engine or a part of the system will be explained in section 5. Some further discussion and the limitations of this research will conclude the paper in section 6.

2. INSTANCE BASED LEARNING: THE MODEL

2.1 IBL Framework

The concept of IBL can be summarized as Fig. 1.

The pre-processor performs a normalization process of the input instances in that all numeric attribute values of an instance should be represented within the range of 0 and 1 by the formula below.

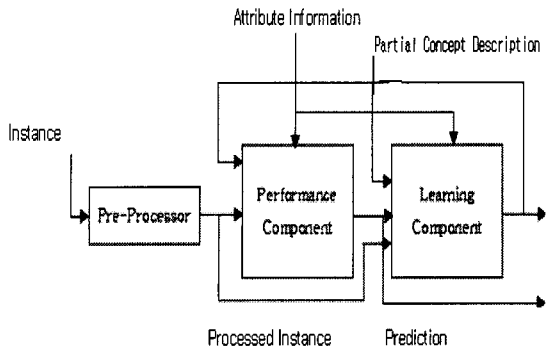

Fig. 1. The IBL framework.

$$\text{Normalize\_attribute}(x_a, a) = \frac{x_a - a_{\min}}{a_{MAX} - a_{\min}}$$

x_a=attribute in instance, a=predictor attribute, a_{Max}=Maximum bound of a, a_{min}=Minimum bound of a

All input instances are stored in the memory called partial concept description (PCD) after pre-processing. This pre-processing does the training phase.

When a test instance is coming into the system with normalized values, the performance component compared the input instance with each member of the PCD with a predefined similarity function below and takes k most similar instances from PCD. Then the component makes a prediction of the class of the input by voting procedure among them.

$$\text{Similarity}(x,y) = \frac{1}{\sqrt{\sum_{i \in F} \textit{Attribute\_difference}(x_i, y_i)}}$$

where

Attribute_difference (x_i, y_i) =

a. $(x_i - y_i)^2$ *if attribute i is numeric-valued*

b. 1 if x_i *and* y_i *have different symbolic values*

c. 0 if x_i *and* y_i *have the same symbolic value*

Among the k-nearest instances in the PCD, the majority-voting scheme is used to predict the class of the input instance.

The learning component may update the PCD with various filters - class boundary filter, noise

filter, etc. The original IBL framework assumes the incremental learning. However, it can be easily changed to the batch learning if the learning component precedes the performance component with some known examples. Thus, the above IBL framework completes the basic IB1 algorithm. Overall, IB1 is very similar to the symbolic attribute value extension of the k-nearest neighbor rules.

2.2 IBL with Partial Instances

2.2.1 IB2: Reducing Storage Requirements

Since IB1 stores all training instances, one may argue that IB1 is too much memory consuming and therefore too slow to some problems with many examples. Mathematical analysis can easily show that if we keep only the boundary instances, the performance would be similar to that of IB1 [1]. Thus, IB2 has a memory filter such that it keeps only the predictive value is sufficiently different from the current PCD members by a threshold value.

2.2.2 IB3: Noise Filter

IB2 is very sensitive to the noise. Thus, IB3 keeps the prediction history of each member of PCD and discards some members with significantly poor predictions. The voting procedure also uses only k most similar "acceptable" instances in that regard. Thus, the procedure of IB3 is summarized in Fig. 2.

2.2.3 IB4: Attribute Weighting

IB3 performs as well as C4.5 in most domains. However, when there are irrelevant attributes, since the algorithm averages the distances over all attributes from one instance to another, IB3 may fail to find salient feature(s). Thus, IB4 takes a weighted average of similarity function and attribute-weight updating procedure. As a principle, IB4 is supposed to have no worse performance than IB3

```
Key: T: Test set  P: Set of predictor attributes
  t: The target attribute  k: # of most similar instances used
  a, b : Confidence thresholds for accepting/dropping
  PCD: Partial concept description
Train (T,P,t,k,threshold,a,b)
    1.  for each xi in T do
      1.1 xi = Preprocess(xi, P)
      1.2 prediction = Performance(xi, P, t, k, a)
      1.3 Learn(xi,prediction,t,threshold,b)
Performance(x, P, t, k)
  1. S ← ø
  2. ∀yi ∈ PCD:S ← SU < yi , Similarity(x, yi, P)>
  3. KSET ← k_most_similar_acceptable_instances(S,k)
  4. return Target_value_prediction(KSET,t,k)
Learn(x, ,prediction,t,threshold,b)
      1. If (|xi,-prediction| > threshold) then PCD ←
      PCD U xi
      2. S = set of stored instances no dissimlar than
      the k^th most similar acceptable instance
      3. Update_prediction_records(S)
      4. Discard_significantly_poor_instances(S,b)
```

Fig. 2. IB3 Algorithm.

but in real world domains, IB4 is often worse than IB3 so that people tends to use IB3.

3. XIBL: FLEXIBLE VALUE DISTANCE AND STEPWISE WEIGHTING FUNCTION

In an IBL algorithm, the method for extracting knowledge from training data is represented by the data itself; that is, the method is not based on the conceptual method that focuses on attributes as shown in C4.5. The knowledge acquired through this method is called Partial Concept Description (PCD) as described in section 2. This algorithm is accurate in many aspects and could be compared to that of C4.5. Although the learning speed is slower than that of C4.5, the validation speed of the new data according to the learned results and the stability according to data change are better than those of C4.5.

The two biggest features of the IBL algorithm are the method for determining the instances to be stored in PCD and the method for measuring the distances between instances. Among four algo-

rithms of IBL family, the IB3 algorithm that has noise filtering scheme and IB4, which is the attribute-weighting IB3 algorithm, are most noticeable. However, in practice, IB3 is used more frequently because IB3 is superior to IB4 in many aspects. These empirical reports suggest that the attribute-weighting scheme of IBL should be significantly improved.

Our version of IB3, re-implemented in C++, called XIBL in this paper has three important changes from the original IB3 as following.

First, XIBL avoids overestimating symbolic attributes by applying Value Difference Metric (VDM) [14] whereas original IB3 took the 'winner-takes-all' strategy. IB3 has the normalization process in it in order to calculate the distance between two instances. Real-valued attributes usually have a distribution between 0 and 1. However, IB3's strategy gives 1 if the attribute values are different and 0 if they are the same. This inevitably overestimates symbolic attributes over real-valued attributes.

Value Difference Metric (VDM) takes into account the overall similarity of classification of all instances for each possible value of each feature. Using this method, a matrix defining the distance between all values of a feature is derived statistically, based on the examples in the training set. The distance δ between two values for a specific feature is defined in below equation [14].

$$\delta(V_1, V_2) = \sum_{i=1}^{n} \left| \frac{C_{1i}}{C_1} - \frac{C_{2i}}{C_2} \right|^k$$

where V1, V2 denote two points in the example space and C_k and C_{ki} denote the number of examples labeled as class k and i'th value of class k in respectively.

There also have been many other Stanfill's VDM-like extensions of discrete value metric [15]. We test three of them that can be directly implemented in our system. N1 is the original VDM [14] and N2 uses Euclidian distance and N3 consid

$$\text{N1}: \quad normalized\_vdm1_a(x,y) = \sum_{c=1}^{C} \left| \frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right|$$

$$\text{N2}: \quad normalized\_vdm2_a(x,y) = \sqrt{\sum_{c=1}^{C} \left| \frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right|^2}$$

$$\text{N3}: \quad normalized\_vdm3_a(x,y) = \sqrt{C^* \sum_{c=1}^{C} \left| \frac{N_{a,x,c}}{N_{a,x}} - \frac{N_{a,y,c}}{N_{a,y}} \right|^2}$$

Fig. 3 Three VDM improvements.

ers the sample size from N2.

Second, IBL is sensitive to noise, and in order to solve this problem, we have applied a Leave-one-out noise filter [10], which is based upon mathematical statistics.

Third, in setting the weighted value for attributes, we have applied the method of estimating attribute value according to a backward stepwise regression, which was statistically validated, instead of a reward/penalty method from among the learning-based methods adopted by IB4. Among N attributes we have, we compare the accuracy of full N attributes and those of when only N-k attributes are considered. The difference of the two means the importance of k attributes in classification. Hence, we set the weights of attributes proportional to the difference in a training set. The pseudo code of attribute weighting with the backward stepwise regression principle is shown as Fig. 6 in section 5.2.

The performance of XIBL, which has been developed in this method, is better than those of C4.5 and the original IBL. The instances in PCD are 8-10% of the learned instances and they have stability especially in reacting with mass data [17]. One may argue that XIBL's weight assigning procedure and leave-one-out noise filtering will have speed deficiency. However, all overhead of the system is put in training phase thus it would not be a significant problem in real world applications. Fig. 4 shows the pseudo-code of our XIBL used in these experiments.

```
Key: T: Training set P: Set of attributes
VDM: Symbolic value distance metrics TE: Test set
k : Number of most similar instances used
PCD: Partial concept description
W : Weight of each attributes
Train( T , P , VDM , k ,W)
1. PCD={}
2. VDM = Make_VDM(T , P)
3. for each xi ∈ T do
classValue=NearstNeighbor(P, PCD , VDM,W,k, xi)
if classValue not equal targetValue
then PCD = PCD + xi
LeaveOneOut(PCD, k, VDM, P)
NearstNeighbor(P, PCD, VDM, W, k , x)
1. Update each attribute's boundary values for
numeric type
2. for each y i ∈ PCD do Similarity(yi,x,W)
3. KSET=k_most_similar_instance( PCD , k)
4. return Vote(KSET)
LeaveOneOut( PCD , k , VDM , P)
1. NOISE={}
2. for each xi ∈ PCD do
2.1 classValue=NearstNeighbor(P, PCD , VDM, xi)
2.2 if classValue not equal targetValue then
                NOISE = NOISE + xi
3. PCD = PCD − NOISE
Test(TE, P, VDM, k)
1. for each xi ∈ TE do
1.1 return NearstNeighbor(P, PCD , VDM,k, xi)
```

Fig. 4. XIBL Pseudo code.

4. ARTIFICIAL DOMAIN EXPERIMENTS

4.1 Test Set

We choose 6 artificial domains from standard UCI machine learning data sets [19]. We conduct a 10-fold cross validation (CV) for each set except hayes-roth set where we apply 5-fold CV due to the number of available instances. All sets except liver-disorder have more than 70% of nominal-valued attributes. The characteristics of the data sets are summarized in Table 1.

4.2 Normal Data Test Results

First, we test 8 different implementations of IB3. IB3 stands for the original implementation. N1, N2, and N3 in their names denote that they use N1, N2, and N3 type of VDM in nominal valued attributes in respectively. W in the name of an implementation means it has feature-weighting procedure of our invention. All results are averaged and compared by t-test of 95% confidence interval and summarized in Table 2. Since liver-disorder data set has no nominal value attribute, we can only test feature-weighting effect in that data set.

First, we can verify the feature-weighting effect by comparing IB3W with IB3. IB3W does not have VDM function. Among 6 data sets, IB3W is superior to IB3 in 3 domains and inferior to one domain (hayes-roth). It may be weak evidence that the feature-weighting scheme is really helpful.

Table 1. DATA SET CHARACTERISTICS

	INSTANCES	CLASSES	ATTRIBUTES	NOMINAL ATT
ANN	7200	3	21	14
HAYES-ROTH	132	3	5	5
HOUSE-84	435	2	16	5
HYPOTHYROID	3163	2	25	19
LIVER-DISORD	345	2	6	0
SICK	3772	2	29	22

Table 2. RESULTS WITH NORMAL DATA

	IB3	IB3W	IB3-N1	IB3W-N1	IB3-N2	IB3W-N2	IB3-N3	IB3W-N3
ANN	98.17	98.09	98.17	98.52	98.17	98.52	98.09	98.52
HAYES-ROTH	63.64	59.09*	63.64	75.00+	65.91+	80.00+	65.36+	86.36+
HOUSE-84	40.69	46.90+	40.45	43.24+	40.50	43.90+	42.26	45.38+
HYPOTHYROID	95.15	96.87	97.72+	96.39	96.96+	96.68+	97.82+	96.39
LIVER-DISORD	64.35	77.39+						
SICK	94.68	96.25+	91.01*	95.07	96.15	96.50+	93.47	94.97

KEY: +: STATISTICALLY BETTER THAN IB3 *: STATISTICALLY WORSE THAN IB3

Second, comparing IBL-N1, IBL-N2, and IBL-N3 with IBL in 5 domains, again, VDM helps a little (superior in 1 or 2 domains) but not overwhelming. Among 3 VDM functions, it looks like N2 or N3 is better than N1 though. However, if we compare IB3 with algorithms that have feature weighting and VDM, especially IB3W-N2, the accuracy increases significantly for most domains we tested.

4.3 Noise Tolerance

The noise can be classified into two groups. The attribute noise means that a certain amount of information other than the class attribute is inaccurate. On the other hand, the class noise means that for a certain amount of data, the class attribute is incorrect. We test both cases by assigning a random value other than the original one to some attributes by designed rate. The noise rate starts from 5% to 20% by 5%. Then, we compute the degradation rate of the accuracy of IB3 and IB3W-N2 by the formula below.

Degradation = (accuracy (normal) - Min (accuracy (noise))) / accuracy (normal) * 100(%)

In attribute noise experiment that Table 3 summarizes, IB3W-N2 tends to be more stable than IB3 although the difference is not that far. However, in terms of accuracy when the algorithm performs the poorest, still IB3W-N2 is better than IB3 in most data sets.

In class noise, IB3 turns out to be very unstable in most domains. IB3W-N2 is more stable than IB3 except liver-disorder where IB3W is still far better than IB3 in accuracy. The noise filter of IB3WN2 is leave-one-out [10] and above experiments proves that it is quite useful in noise-prone domains.

5. REAL WORLD APPLICATIONS OF XIBL

5.1 Intrusion Detection

5.1.1 System Model

Snort [20] is a well-known signature-based network intrusion detection system. It sequentially processes the packets that occur from the network according to each protocol specification, compares the attack signatures, and then signals when they conform to the corresponding stages. The machine learning-based intrusion detection system, on the

Table 3. Attribute Noise Tolerances

	IB3(%)			IB3W-N2(%)			WN2-IB3
	Norm	Min	Rate	Norm	Min	Rate	Difference
Ann	98.17	96.66	1.54	98.52	97.08	1.46	-0.08
hayes-roth	63.64	44.67	29.80	80.00	57.58	28.03	-1.78
House-84	40.69	32.65	19.76	43.90	38.34	12.67	-7.09
Hypothyroid	95.15	93.64	1.59	96.68	95.83	0.88	-0.71
liver-disord	64.35	46.97	27.01	77.39	54.40	29.71	2.70
Sick	94.68	91.64	3.21	96.50	94.71	1.85	-1.36

Table 4. Class Noise Tolerances

	IB3(%)			IB3W-N2(%)			WN2-IB3
	Normal	Min	Rate	Normal	Min	Rate	Difference
Ann	98.17	98.14	0.03	98.52	97.77	0.76	0.73
hayes-roth	63.64	33.24	47.77	80.00	60.49	24.39	-23.38
house-84	40.69	33.58	17.47	43.90	39.09	10.96	-6.52
Hypothyroid	95.15	69.99	26.45	96.68	72.62	24.89	-1.56
liver-disord	64.35	57.82	10.14	77.39	61.47	20.57	10.43
Sick	94.68	72.36	23.57	96.50	76.11	21.13	-2.45

other hand, creates an event for each packet from the network and learns the normal and abnormal patterns using the events.

The basic ideas of our proposed model are as follows:

In order to examine the truth of the Snort attack signals, the rate of packets that are classified into abnormal patterns is obtained through examining a certain number of packets that are classified in machine learning-based intrusion detection by back tracking at the time of giving the alarm. The signal is given to the administrator when the obtained rate for the abnormal pattern exceeds the standard value. Further, in order to detect the attacks not perceived by Snort, when the consecutive abnormal frequency exceeds a certain limit after examining the classified events of the machine learning-based intrusion detection system, the administrator will be informed of the attacks that are not perceived by Snort. We shall define the former abnormal rate as "Alpha-Cut" and the latter consecutive abnormal numbers as "Beta-Pick"[17].

The signals of the intrusion detection system are generally classified as the following: true positive (when a normal state is to be noticed as normal), false positive (when a normal state is to be noticed as abnormal), false negative (when an abnormal state is to be noticed as normal), and a true negative (when an abnormal state is to be noticed as abnormal). The empirical problem of using IDS is not just detecting an attack but there are too many false alarms to manage thus our effort is centered on decreasing false alarms while keeping the accuracy (detecting attacks) as much as possible.

5.1.2 Experiment

For the experiment, we used Snort 1.8.1 and followed the basic environment setting distributed by the website [20]. The learning algorithms used in the machine learning-based intrusion detection system were XIBL and C4.5 [4].

Our experiment data set is organized as follows. First, we randomly collect 70% of real attacks from DARPA 1998 data set and part of normal data set in contiguous time frame with about the same size in packets. In results, we have 44726 packets and 45453 normal packets. Then, we conduct a 10-fold cross validation on this data set.

We choose the best PCD (for XIBL) or the best tree (for C4.5), which contains noise-filtered packet vector from above procedure. The best PCD contains 365 normal packets and 294 abnormal packets or about 8% of training data with 99.4% fitting rate. With this PCD, we test 30% of attack packets and about the same size of normal packets. The result shows that 88% of normal packets are classified into "normal" but only 68% of attack packets are correctly classified.

Table 5 shows the results of the performance tests of the system when single and using Snort, XIBL, and C4.5, as learning engines, formed combined mode systems. Among the 137 attack data, 42 true alarms were observed by Snort. There were 42 true alarms in the XIBL combined model that applied Alpha-Cut and there were 10 true alarms in the combined model that applied C4.5. Generally, combined models are expected to show fewer frequent true alarms than does the Snort single model. When XIBL was applied, the rate of true alarms was maintained at 100%, which was the same as that of the Snort single model, while it produced a rather disappointing result of 23.8% when C4.5 was applied.

As for the new true alarms that are not detected when Snort single model is used, the XIBL-combined model that applies Beta-Cut gave 81 true alarms, whereas the C4.5-combined model gave 35 true alarms. Of the 137 attacks, the Snort single model detected only 16 while it missed detecting attacks in the remaining 121. The XIBL-combined model detected 16 attacks, detected by Snort, and 13 additional attacks, which were not detected by Snort. That is, it missed 108 attacks. The C4.5-combined model detected 8 attacks among the 16

Table 5. An Integrated analysis example of performance test

Original Data	Single System	Combined System		
	Snort	Algorithm / Filter	XIBL	C4.5
Abnormal (137times try)	Abnormal (42 times detect)	Alpha-cut	Abnormal (42times detect)	Abnormal (10 times detect)
			Normal	Normal
	Normal (95 times miss)	Beta-pick	Abnormal (81 times detect)	Abnormal (35 times detect)
			Normal	Normal
Normal	*Abnormal (22,787 alarm)*	Alpha-cut	*Abnormal (12,791 alarm)*	*Abnormal (21,403 alarm)*
			Normal	Normal
	Normal	Beta-pick	*Abnormal (43 alarm)*	*Abnormal (53 alarm)*
			Normal	Normal

attacks detected by Snort and 7 additional attacks. It missed 122 attacks.

In terms of the number of alarms, the results of the experiment above show that the number of alarms in the combined models decreases to the level of 56.75% of those of the Snort model. In terms of the quality of alarms, the alarms detected by the combined models include most of the alarms detected by the Snort model and they also include the alarms not detected by the Snort model. This, however, inevitably requires sacrifice of the new false alarms, which are not found in the Snort model. However, considering the fact that the new false alarms are not as frequent and that it is possible to detect new attacks that were previously undetected, the application of this model depends upon the circumstances of the system of the administrators.

5.2 Korean Spam mail Filtering

5.2.1 The Setup

Keyword pattern based filtering software programs such as Microsoft Outlook are weak to multimedia-based spam mails and also vulnerable to the variations of known patterns. We design and implement a prototype of spam-mail filtering software for Korean Language that overcomes the drawbacks of Outlook type filtering algorithms.

First, we interpret spam mail filtering as a two-class learning problem trained on message vectors where email contents are analyzed and reformed as follows.

$$Msg = < \vec{n}_1, \vec{n}_2, \vec{n}_3, ..., \vec{n}_n >$$

where each attribute n_x corresponds to a unique noun and if the noun n_x appears in the message, n_x is 1 and 0 otherwise.

5.2.2 Learning process

Our goal is to develop a stable learning engine based on the form of message vector shown above. Thus, the first step of learning is to determine the set of attributes from training examples. We use Mutual Information [21] to measure relationship between a noun and class of E-mail.

$$\mathrm{MI}(N;C) = \sum_{n\in\{0,1\},C\in\{0,1\}} P(N=n, C=c)\cdot \log \frac{P(N=n, C=c)}{P(N=n)\cdot P(C=c)}$$

As in [4], we compute the mutual information (MI) of each candidate attribute N with the category-denoting variable C, and select the attributes with the m highest MI - scores.

Then, we apply XIBL as the main learning engine. Fig. 5 shows the weight decision procedure. **IBL** in Fig. 8 denotes the original IB3.

```
for (i=1 ; I<size(A);i++)
  for(j=1; j<size(T);j++)
   Power(Ai)=|Acc(IBL(TA-Ai ,ej,1) - Acc(IBL(TA,ej,1)|
Weight(Ai)=Power(Ai) / sum(Power(Ai)
```

*IBL(TA,e,k) : With a set A of attributes, find from instances T the closest k to new instance e
*Acc(e) : classification accuracy of new instance e

Fig. 5. Attribute Weight decision.

Acc calculates the classification accuracy of **IBL** with attribute set A of Training set T. TA-Ai means dropping attribute Ai from each training example. Then, we calculate normalized weight. Fig. 6 shows the overall processes of our system.

5.2.3 Experiment Summary

We performed an experiment on Korean E-mail corpus with spam mail filter using place of dispatch tracking technique. We collected 100 messages and 50 web documents in computer science field, which are classified into legitimate mail. 150 commercial messages are classified into spam mail.

Leave-one-out [10] algorithm is the train-test error rate estimator. When there are n sample, it trains n-1 sample and tests the rest one example. This procedure repeats n times. The number of attributes selected by MI was 10. The number of extracted nouns T was used as trigger that precipitates place of dispatch tracking. We measured classification accuracy, corresponding to T. When T was 10, the accuracy was increased to 96.7%. Fig. 7 summarizes the results.

6. CONCLUDING REMARKS

In this paper, we reviewed the original IBL prin-

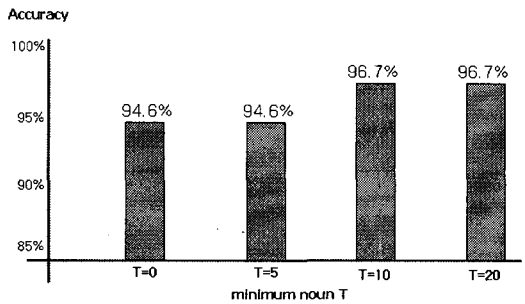

Fig. 7. the result of spam-mail experiment.

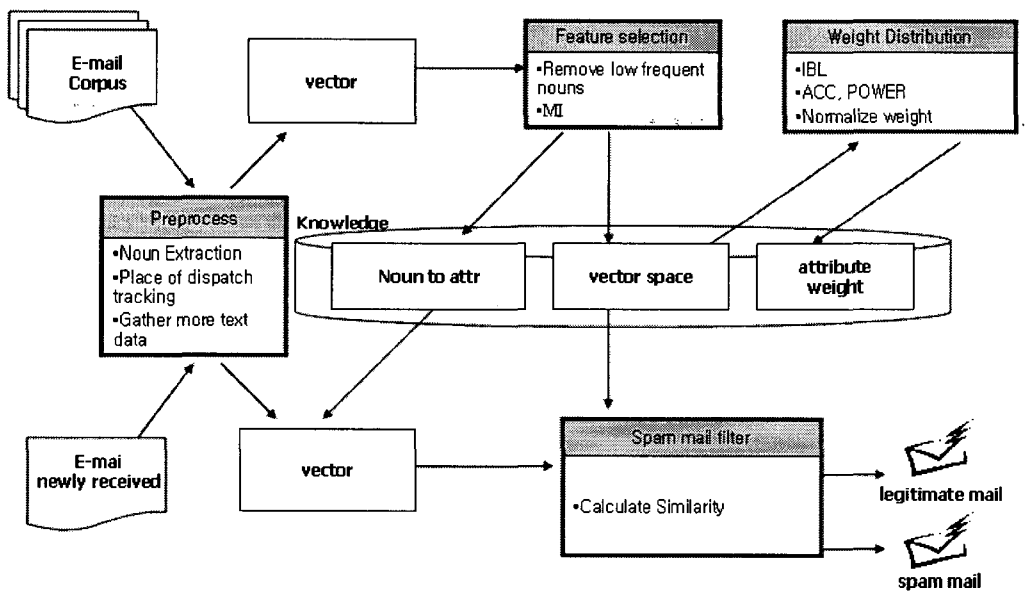

Fig. 6. Spam mail filtering system.

ciples and gave a series of meaningful improvements on IB3. The new feature-weighting scheme works nicely in most artificial domains we tested. The VDM effect is also confirmed. Overall, with leave-one-out as the noise-filtering scheme, XIBL is more noise tolerant.

XIBL has been used as a part of the system or the learning engine for two real applications. For the network intrusion detection system, XIBL combined with snort system significantly reduces false alarms without losing its accuracy and it is far better than snort alone and snort combined with C4.5. For spam-mail filtering problem, XIBL also works nicely that for all conditions we tested, over 90% of the spam mails are detected.

However, enhancing with new mass-data environment such as bioinformatics and web analysis, XIBL has clear limitations that it is slow and it still stores about 8 % of the input in training. In order to achieve a significant improvement in above areas, it may adopt a new concept from other lazy learners [3] but until then, XIBL is highly acceptable in practice.

7. REFERENCE

[1] D. W. Aha, *A Study of Instance-Base Algorithms for Supervised Learning Task: Mathematical, Empirical, and Psychological Evaluations*, Ph.D. Dissertation, University of California, Irvine, 1990.

[2] D. W. Aha, "Feature weighting for Lazy Learning Algorithms," in *Feature Extraction, Construction, and Selection:A Data Mining Perspective* (H. Liu and H Motada (eds.)), Kluwer Academic Publishers, pp 1-20, 1998.

[3] D. W. Aha (eds.), *Lazy Learning*, Kluwer Academic Publishers, 1997.

[4] J. R. Quinlan, *C4.5:Programs for Machine Learning*, Morgan Kaufmann, 1993.

[5] I. Androutsopoulos, G. Paliouras, V. Karkaletsis, G. Sakkis, C. Spyropoulos, P. Stamatopoulos, "Learning to Filter Spam E-Mail: A Comparison of a Naïve Bayesian and a Memory-Based Approach," *Proc. of the Machine Learning and Textual Information Access Workshop of the 4th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pp. 1-3, 2000.

[6] G. Sakkis, I. Androutsopoulos, G. Paliouras, V. Karkaletsis, C. Spyropoulos, P. Stamatopoulos, "A Memory-based Approach to Anti-Spam Filtering in Mailing Lists," *Information Retrieval* 6, pp. 49-73, 2003.

[7] S. Dzeroski, S. Schulze-Kremer, K. R. Heidtke, K. Siems, D. Wettschereck, and H. Blockeel. "Diterpene structure elucidation from 13 CNMR spectra with inductive logic programming," *Applied Artificial Intelligence*, 12(5): pp. 363-384, 1998.

[8] M. S. Lew, T. S. Huang, K. Wong, "Learning and Feature Selection in Stereo Matching," *IEEE Transactions On Pattern Analysis And Machine Intelligence*, Vol. 16, No 9, 1994.

[9] P. Domingos, "Context sensitive feature selection for lazy learners," *AI Review* 11, pp. 227-253, 1997.

[10] S. Cost, S. Salzberg, "A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features," *Machine Learning* 10, pp. 57-78, 1993

[11] P. Domingos, "Unifying Instance-based and Rule based Induction," *Machine Learning* 24, pp. 141-164, 1996 .

[12] W. Emde, D. Wettschereck, "Relational Instance-based Learning," in *Proc. Of 13th International Conference on Machine Learning*, L. Saitta(eds.), Morgan Kaufmann, pp. 122-130, 1996,

[13] P. Kontkanen, P. Myllymaki, T. Silander, H. Tirri, "Bayes Optimal Instance-based Learning," in *Lecture Notes in AI*, Vol. 1398, pp. 77-88, 1998.

[14] C. Stanfill, D. Waltz, "Toward memory-based reasoning". *Communications of the ACM* 29, pp. 1213-1228, 1986.

[15] D. Randall Wilson, T. R. Martinez, "Improved Heterogeneous distance Functions," *Journal of Artificial Intelligence Research* 6, pp. 1-34, 1997.

[16] D. H. Song, I.W. Weon, C. H. Lee, "The Utility of Packet level decision in Misuse Intrusion Detection System: An analysis of DARPA dataset toward a hybrid behavior based IDS," *Proc. of the 3rd APIS*, Istanbul, Turkey, pp. 214-218, 2004.

[17] I. W. Weon, D. H. Song, C. H. Lee, "Effective Intrusion Detection Model through the Combination of a Signature-based Intrusion Detection System and a Machine Learning Intrusion Detection System," in *Journal of Information Processing and Engineering*, to be published, 2006.

[18] H. J. Ha, D. H. Song, C. H. Lee, "Design and Implementation of Korean Spam mail Filter by Tracking the Dispatch Information and Instance-based Learning," in *Proc. of Third East Asian Language Processing and Information Technology Conference*, Mongolia, pp. 369-373, 2003.

[19] http://www.ics.uci.edu/~mlearn/MLRepository.html, the UCI Machine Learning Repository, 2006.

[20] SNORT, http://www.snort.org, 2005.

[21] G. Salton, M. J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983.

Doo-Heon Song

He received a BS degree in Computer Science from Seoul National University and an MS degree in Computer Science from the Korea Institute of Science Technology in 1983. He received his PhD Certificate in Computer Science from the University of California in 1994. From 1983~1986, he was a researcher at the Korea Institute of Science and Technology. He has been a professor at the Department of Computer Games & Information, SongDam College, Korea, since 1997. His research interests include data mining, machine learning, security, vision, ITS, and game intelligence.

Chang-Hun Lee

He received a BS degree in Mathematics from Yonsei University, Korea, in 1980, and an MS and a Ph.D. in Computer Science from the Korea Institute of Science and Technology in 1987 and 1993 respectively. From 1996~2000, he was the head of the Information Technology Center at Konkuk University. He has been a professor at the department of Computer Engineering, Konkuk University, Korea, since 1980. His research interests include artificial intelligence, operating system, embedded systems, and security.