POLL: multiclass classification from binary classifiers through random sampling

Nithi Bunrupunthunad Jittat Fakcharoenphol Thanawin Rakthanmanon
Department of Computer Engineering,
Kasetsart University.
Bangkok 10900, Thailand.
E-mail: nithi@thai.com, jtf@ku.ac.th, g4465014@ku.ac.th.

August 30, 2003

Abstract

We apply a sampling technique in combinatorial optimizations to the problem of multiclass classification. Various methods have been used to construct a multiclass classifier from a set of binary classifiers. Suppose that there are n classes. A simple majority vote (known as max-win), while giving very reliable answers, need a quadratic number of comparisons. Other methods, by cascading binary classifiers, reduce the number of comparison to $O(n)$ but suffer from the build-up of the failure probability. This implies that they do not scale well with n. We show that by using random sampling, we need only $O(n \log n)$ comparisons while retaining the good performance of the majority method.

1 Introduction

In many machine learning methods, a binary classifier is easy to construct, while, in most applications, a multiclass classifier is needed. There are quite a few techniques invented to remedy this problem [2, 4, 8, 14]. Notably, by making all-pair comparisons, a simple majority vote [3] (denoted as MAX-WIN later on in the paper) gives a very reliable answer. It, however, needs $O(n^2)$ applications of binary classifiers for the problem with n classes. The popular Decision Directed Acyclic Graphs [11] reduces the number of comparison down to $O(n)$ while suffering from an exponentially large failure probability because it builds on a cascading path of classifiers. By reducing the depth of the path to $O(\log n)$,[1] a recent approach called Adaptive Directed Acyclic Graphs [7] greatly improves the success probability while retaining a linear run time. Specifically, it fails only with a polynomially large probability. These approaches do not scale very well with the number of classes.

We apply a well known sampling technique in combinatorial optimizations to the problem of multiclass classification. The technique has a successful application on many cut problems mostly due to Karger [5, 6]. We show that by using random sampling, we need only $O(n \log n)$ comparisons while retaining the good performance of the majority method.

1.1 The problem

Given an unknown data point d in the universe Ω, the *multiclass classification problem* is to determine which of the n classes d belongs to. There are various method for solving this problem.

[1] If not specified otherwise, all logarithms in this paper are base 2.

In this paper we focus on methods which use one-against-one binary classifiers described as follows. For each pair $\{i, j\} \in [n] \times [n]$, where $i \neq j$, we have a binary classifier $A_{i,j} : \Omega \to \{i, j\}$ that answers whether d belongs to i or j. Normally, it is expected at if d belongs to the i-th class, $A_{i,j}(d)$ should return i. It is not clear what $A_{j,k}(d)$ would return when d does not belong in either j or k. For a pair $\{i, j\}$ that a comparison is performed, we say that j is an *opponent* of i and vice versa.

The algorithm MAX-WIN performs all $\binom{n}{2}$ one-against-one classifications. For each i, it picks all $n-1$ possible opponents, runs binary classifiers, and keeps track of the score s_i that i wins, i.e.,

$$s_i = |\{(i, j) \in [n] \times [n], i \neq j : A_{i,j}(d) = i\}|.$$

It then outputs that d belongs to the highest score class.

1.2 Our results

Our basic result shows that if MAX-WIN gives a "clear-cut" answer (to be defined formally later). By having only $O(n \log n)$ comparisons we get the same answer with high probability (i.e., with probability at least $1 - 1/n^c$ for some $c > 0$). We note that this success probability goes to 1 as n grows. This is in contradiction with the other approaches.

We further give an analysis of MAX-WIN. We show that, with a mild assumption, if each binary classifier gives correct answers with constant probability greater than half, with overwhelming probability MAX-WIN gives a "clear-cut" answer, satisfying our assumption for the first result. We also mention a more sophisticated analysis of MAX-WIN which is based on the similarity measure of the query points and the classes.

1.3 A note on technique

Our algorithm is analogous to polling. Suppose that MAX-WIN gives a correct answer. By sampling only l opponents for each i, on average we should get the same answer. One can see this paper as an analysis of a sufficient sampling size.

It can also be seen as a sparsification technique as in Karger's result, since this problem is essentially to find the node with the highest in-degree.

1.4 Previous results

The problem stems from the classification using Support Vector Machines (SVMs) [14]. There are two broad categories for constructing multiclass classifiers from binary classifiers. On the one hand, one can train, for each class, a classifier which recognizes objects only from that class while rejects objects from all other classes; this first category is called One-against-the-rest [14]. While the One-against-the-rest gives a fast classification time, it is difficult to train the classifiers. On the other hand, in One-against-one [3, 8], a set of binary classifiers which distinguish objects between two different classes are trained. The problem with this approach is the classification time; one needs a quadratic number of calls to binary classifiers.

Platt, Cristianini, and Shawe-Taylor [11] introduced a new learning architecture, called Decision Directed Acyclic Graphs (DDAG), and reduced the number of calls down to linear. However, Kijsirikul, Ussivakul, and Meknavin [7] observed that the structure of the elimination in DDAG introduces a long path of cascading calls to binary classifiers. They proposed a new structure called ADAG, whose number of cascading calls reduces to only logarithmic. For the problem with n classes, they showed that when the probability that a binary classifier gives a

wrong answer is p, DDAG successes with probability

$$\frac{1}{n}\left(\frac{1-p}{p}+(1-p)^{n-1}-\frac{(1-p)^n}{p}\right),$$

while their ADAG gives the correct answer with probability

$$\left(\frac{2n-2^{\lceil \log_2 n\rceil}}{n}\right)(1-p)^{\lceil log_2 n\rceil}+\left(\frac{2^{\lceil \log_2 n\rceil}-n}{n}\right)(1-p)^{\lceil log_2 n\rceil-1}$$

or $(1-p)^{\Theta(\log n)}$. Their experiments show that ADAG performs better than DDAG as n increases.

Phetkaew, Kijsirikul, and Rivepiboon [10] used the "internal" information on the correctness of the binary classifiers to improve ADAG. In many cases, they can get even a higher accuracy than MAX-WIN. Their idea is to avoid the noisy information from the other unrelated binary classifiers by reordering. Their experiments, however, involved only problems with at most 26 classes.

1.5 Organization

In Section 2 we present our algorithm and its analysis. We discuss some heuristic that might help increasing the success probability in Section 3. The simulation results are shown in Section 4. Finally, we give the conclusion in Section 5.

2 The algorithm and its analysis

Our algorithm, POLL, is a slight modification of MAX-WIN. We maintain a score $\hat{s}_i$ for each class i. Instead of having $\binom{n}{2}$ comparisons, for each i we performs only l comparisons, for a parameter l. Thus, we use only nl comparisons. For each i, we repeat the following procedure l times: pick j uniformly at random from $\{1,\ldots,n\}-\{i\}$, if $A_{i,j}(d)$ returns i, we increase the score $\hat{s}_i$. Finally, the algorithm says that d belongs to the class i with the highest sample score $\hat{s}_i$. For notational simplicity, we let n_1 be $n-1$.

In the subsections that follow, we analyze the POLL algorithm in two steps. First, we show that if MAX-WIN gives the result such that s_1 is much higher than s_2, the second winner, POLL gives the correct answer. In the second step, we show that under the model of Kijsirikul, Ussivakul, and Meknavin, MAX-WIN is very likely to give such an answer.

2.1 A clear-cut case

Given a data point d, MAX-WIN computes s_i for each class i. (We use s_i for the analysis only. Our algorithm does not need them.) For simplicity we reindex the classes so that $s_1 \geq s_2 \geq \cdots \geq s_n$. We assume that MAX-WIN gives a correct answer, i.e., it reports that d belongs to the 1st group.

We state rough bounds on the values of s_1 and s_2. From an averaging argument, $s_1 \geq n_1/2$, because there are $n \cdot n_1/2$ points to give, so at least one must have at least an average number. Furthermore, we also have that $s_2 \geq n_1/2-1$ (again from the same argument); thus, $s_2 \geq n_1/4$ when $n_1 > 2$. These facts will be crucial when we want to bound the probability.

The score $\hat{s}_i$ for each group i is a random variable. To compute it, we conduct l experiments, and POLL gives the correct answer when $\hat{s}_1 > \hat{s}_j$ for all $j > 1$. Thus, we want to bound the probability

$$\Pr[\hat{s}_1 \leq \hat{s}_j \text{ for some } j \neq 1].$$

We now look at the expectations. We have that $E(\hat{s}_i)$ is $s_i l/n_1$. The good thing is they always give the right answer, i.e., $E(\hat{s}_1) > E(\hat{s}_i)$ for $i \neq 1$ when the real random variables might not be so. However, if the gap between the expectation of $\hat{s}_1$ and the other $\hat{s}_i$ is large, we can hope that the 1st class always win. This motivates our definition of a separation.

For a given data point, we define a *separation* to be s_1/s_2. We say that MAX-WIN gives a *clear-cut* answer if the separation is at least some fixed constant $\alpha > 1$. We say that the instance is *α-separated*.

To prove that POLL gives the same answer we need a strong tail inequality. We use the Chernoff bound [1] in the following forms (see also [9]).

Theorem 1 (Chernoff bound) *Let $X_1, X_2, \ldots, X_l$ be independent 0-1 random variables such that, for $1 \leq i \leq l, \Pr[X_i = 1] = p_i$, where $1 < p_i < 1$. Then, for $X = \sum_{i=1}^{l} X_i$, $\mu = E(X) = \sum_{i=1}^{l} p_i$, and and any $0 < \delta \leq 1$,*

$$\Pr[X < (1-\delta)\mu] < e^{-\mu\delta^2/2}.$$

Also, for any $\delta > 0$,

$$\Pr[X > (1+\delta)\mu] < \left[\frac{e^{\delta}}{(1+\delta)^{(1+\delta)}}\right]^{\mu}.$$

We first bound the failure probability $\Pr[\hat{s}_1 \leq \hat{s}_j]$ for $j \neq 1$. With out loss of generality, we bound only $\Pr[\hat{s}_1 < \hat{s}_2]$ because $\Pr[\hat{s}_1 < \hat{s}_j] \leq \Pr[\hat{s}_1 < \hat{s}_2]$. We prove our main technical lemma.

Lemma 1 *Suppose that the problem is α-separated. For any fix constant $c > 0$, there is a constant c' and c'' such that if $l \geq c' \log n' + c''$,*

$$\Pr[\hat{s}_1 < \hat{s}_2] < n_1^{-c}.$$

Proof: Denoting $E(\hat{s}_i)$ with μ_i, we have

$$\Pr[\hat{s}_1 < \hat{s}_2] \leq \Pr[\hat{s}_1 < \frac{\mu_1 + \mu_2}{2}] + \Pr[\hat{s}_2 > \frac{\mu_1 + \mu_2}{2}].$$

We rewrite it as

$$\Pr[\hat{s}_1 < (1-\delta_1)\mu_1] + \Pr[\hat{s}_2 > (1+\delta_2)\mu_2].$$

where $\delta_1 = \frac{1}{2} - \frac{\mu_2}{2\mu_1} \geq \frac{1}{2} - \frac{1}{2\alpha}$ and $\delta_2 = \frac{\mu_1}{2\mu_2} - \frac{1}{2} \geq \frac{\alpha}{2} - \frac{1}{2}$ for an α-separated case. We bound each term separately. Specifically, we show that each event occurs with probability less than $n_1^{-c}/2$. We can use the Chernoff bound since $\hat{s}_i$ is a sum of independent random trials.

We start with the first term. We have

$$\begin{aligned} \Pr[\hat{s}_1 < (1-\delta_1)\mu_1] &< \exp(-\mu_1\delta_1^2/2) \\ &= \exp(-\frac{l s_1}{n_1} \cdot (\frac{1}{2} - \frac{1}{2\alpha})^2/2). \end{aligned}$$

Thus, if $l \geq (\log_2 e) \cdot \left(2c\left(\frac{1}{(\frac{1}{2}-\frac{1}{2\alpha})^2}\right)\frac{n_1}{s_1}\log n_1 + 1\right)$, the above probability becomes less than $n_1^{-c}/2$. Note that $s_1/n_1 > 1/2$; thus, with some rewriting, it suffices to set l to be

$$(\log_2 e) \cdot \left(4c\left(\frac{2\alpha}{\alpha-1}\right)^2 \log n_1 + 1\right).$$

Now we deal with the second term. Let γ denote the term $\left(\frac{e^{\delta_2}}{(1+\delta_2)^{(1+\delta_2)}}\right)$. From the Chernoff bound, we have

$$\begin{aligned}\Pr[\hat{s}_2 > (1+\delta_2)\mu_2] &< \left[\frac{e^{\delta_2}}{(1+\delta_2)^{(1+\delta_2)}}\right]^{\mu_2} \\ &= \gamma^{\mu_2} = \gamma^{(ls_2/n_1)}.\end{aligned}$$

It can be shown that γ is a constant less than 1 when $\delta_2 > 0$. Thus, if

$$l > (\log_2 \gamma) \cdot (c(n_1/s_2)\log n_1 + 1),$$

the failure probability in the second term becomes less than $n_1^{-c}/2$. Since we know that $s_2 \geq n_1/4$, if we set l to be at least $(\log_2 \gamma) \cdot (4c \log n_1 + 1)$, the above condition holds.

To see the dependence on α, we rewrite $\log \gamma$ as $\delta_2 \log e - (1+\delta_2)\log(1+\delta_2) = \left(\frac{\alpha-1}{2}\right)\log e - \left(\frac{1+\alpha}{2}\right)\log\left(\frac{1+\alpha}{2}\right)$. From the range of α we get that $\log \gamma \leq \frac{1}{2}\log e$.

For the possible values of α, one can show that $\frac{1}{2} < \left(\frac{2\alpha}{\alpha-1}\right)^2$. Therefore, the lemma holds if we let

$$c' = 4c \log e \cdot \left(\frac{2\alpha}{\alpha-1}\right)^2$$

and $c'' = 2 > \log e$. We note that, asymptotically, $c' = O\left(\frac{1}{(\alpha-1)^2}\right)$. ∎

The following theorem follows from the application of the union bound.

Theorem 2 *For any constant c, if the instance is α-separated, by setting $l = O((\frac{1}{\alpha-1})^2 c \log n)$, algorithm POLL gives the correct answer with probability at least $1 - \frac{1}{n^c}$.*

Proof: We apply lemma 1, i.e., we set

$$l = (4(c+1)\log e \cdot \left(\frac{2\alpha}{\alpha-1}\right)^2)\log n + 2,$$

and get that any fixed $j \neq 1$,

$$\Pr[\hat{s}_1 \leq \hat{s}_j] < \frac{1}{n^{c+1}}.$$

Using union bound, we have

$$\begin{aligned}\Pr[\hat{s}_1 \leq \hat{s}_j \text{ for some } j \neq 1] &\leq \sum_{j=2}^{n} \Pr[\hat{s}_1 \leq s_j] \\ &< n \cdot \frac{1}{n^{c+1}} = \frac{1}{n^c},\end{aligned}$$

as required. ∎

The theorem gives an upperbound on the parameter l, the sample size. For example, if the instance is 1.8-separated (when the parameter p as defined in Section 2.2 is 0.9), we need at most $234 \log n + 2$ samples. This is an over-estimate, as the simulation shows that we need only $5 \log n$ in this case (see Section 4). If a better upperbound on the failure probability is derived, a better constant can be proven.

2.2 MAX-WIN wins with high probability

In this section, we show that under the same assumption as in [7], we are very likely to be in the "clear-cut" case, i.e., MAX-WIN gives the correct answer with a good separation. We assume that each binary classifier $A_{i,j}$ can distinguish an input in class i and class j with probability at least $p > 1/2$. When an input is in class $k \notin \{i, j\}$, it reports that the input belongs to i or j each with probability $1/2$.

We randomly generate the MAX-WIN score s_i from this model; thus, each s_i becomes a random variable, with expectation $E(s_i) = n_1/2$ if $i \neq 1$ and $E(s_1) = pn_1$. The following theorem can be proved using the same technique as in lemma 1.

Theorem 3 *For any small $\epsilon > 0$, the probability that $s_1 \geq (1-\epsilon)pn_1$ and for all $j \neq 1$, $s_j \leq (1+\epsilon)n_1/2$ is at least $1 - e^{-\Omega(n)}$. Thus, the random instance from this model is $(\frac{(1-\epsilon)p}{(1+\epsilon)2})$-separated with high probability probability.*

We note that in this case, the failure probability drops exponentially with n because the sample size is $\Omega(n)$.

2.3 A more sophisticated analysis of MAX-WIN

One would be interested to analyze MAX-WIN further. We mention briefly one way of doing so. The assumption that a binary classifier $A_{i,j}$ gives random outputs when the input are not in classes i or j seems to be unrealistic. We can define the similarity measure r_i of the input for each class i such that (1) for all $1 \leq i \leq n$, $0 \leq r_i \leq 1$, (2) $\sum_i r_i = 1$, and (3) $r_1 > r_i$ for all $i \neq 1$. With this similarity measure, if $r_i > 0$ or $r_j > 0$, we can say that $A_{i,j}$ outputs i with probability $\frac{r_i}{r_i+r_j}$ and outputs j with probability $\frac{r_j}{r_i+r_j}$. When both of them is zero, $A_{i,j}$ outputs randomly between i and j. With this detailed analysis, the result resembling that in Section 2.2 can now be derived. We omit the result in this version of the paper.

3 The TOP-k heuristic

In this section, we describe a few heuristics that improve the success probability. This can be done while keeping the number of binary comparisons to still $O(n \log n)$. The idea is to pick k top score classes, and investigate them further.

Using this idea, we fail to pick the correct answer only when it is of lower rank. Clearly, this happens with smaller probability than the case that the correct answer wins. Now, when we have only k candidate classes, quite a few approaches are possible.

One can run MAX-WIN among them, resulting in an additive $O(k^2)$ comparisons. This, at first sight, seems to be a good idea. However, when some class in this set might have a very high score among the k candidates, but it scores very badly outside. We do not expect this, but it seems that another approach looks more promising. I.e., in $O(nk)$ comparisons, one can recompute the original MAX-WIN scores for these classes. The simulation shows that the latter is indeed better.

These heuristics, however, do not help much when the success probability is very close to 1. However, in other cases, it usually increases the success probability by 2-10%. We omit the analysis and the simulation results from this version of the paper.

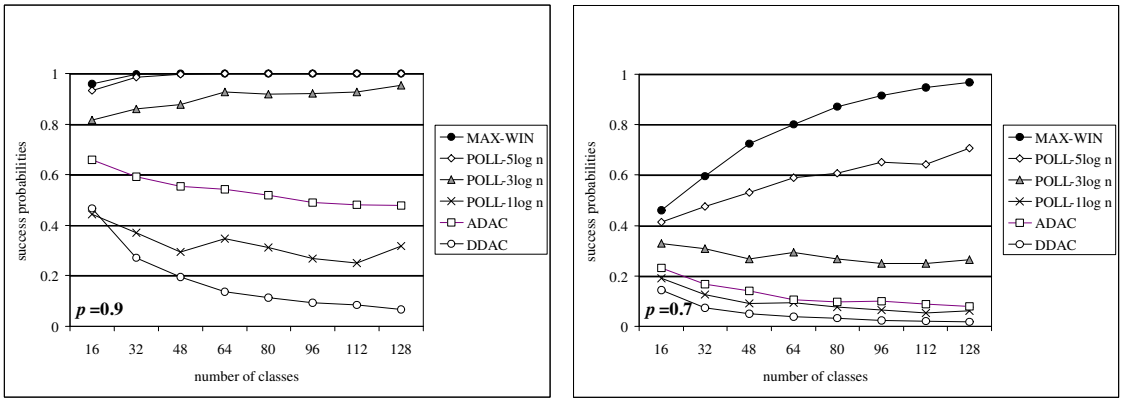

Figure 1: Small cases. The success probabilities of MAX-WIN, POLL, ADAG, and DDAG, when (a) $p = 0.9$ and (b) $p = 0.7$. POLL run with different parameters l: $\log n, 3\log n$, and $5\log n$.

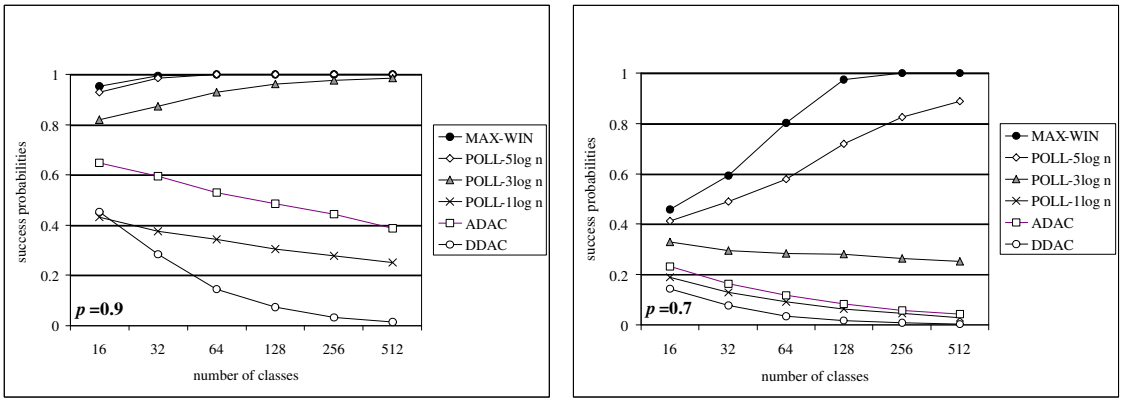

Figure 2: Large cases. The success probabilities of MAX-WIN, POLL, ADAG, and DDAG, when (a) $p = 0.9$ and (b) $p = 0.7$. POLL run with different parameters l: $\log n, 3\log n$, and $5\log n$.

4 Simulation results

We compared the performance, in terms of success probabilities, of 4 algorithms: MAX-WIN, POLL, ADAG, and DDAG, in various cases. For each parameter, we ran 10000 rounds of the experiments.

For each round, a random MAX-WIN scores are generated according to the model in Section 2.2. We simulated them in two different correctness parameter p, i.e., when $p = 0.9$ and when $p = 0.7$.

We present the result in two cases. The first case we consider is when n grows linearly from 16 to 128 (Figure 1). The other case is when n grow exponentially from 16 to 512 (Figure 2).

In general, the simulation shows that POLL, with larger sample size l, has a performance close to MAX-WIN. While in the other two faster approach, i.e., DDAG and ADAG, their performance drops as n grows (as proven in [7]). The running time of POLL is definitely slower than DDAG and ADAG by a factor of $l = O(\log n)$.

We now discuss some interesting cases. When the success probability of each binary classifier

is low (e.g., in the case that $p = 0.7$), we note that even MAX-WIN suffers from the randomness. Furthermore, the constant that we use is obviously too small for that separation, as can be seen in Figure 2(b).

5 Conclusions

We show that a sampling technique, which is widely used in combinatorial optimization, can be also used in classification problem. It improves polynomially over the previous algorithm while retaining the correctness. This technique, we hope, might be able to find other applications in other AI problems as well.

Our result seems to be only of theoretical interest since the problem with huge number of classes is very unlikely. However, if one consider problems in biology [12, 13], which has more than a hundred classes, this result might be of practical use.

6 Acknowledgment

Boonserm Kijsirikul's fascinating lecture at our department introduced us to the problem. We also thank him for various help on this paper.

References

[1] H. Chernoff. A measure of asymptotic efficiency for test of a hypothesis based on the sum of observations. *Annals of Mathematical Statistics*, 23:493–509, 1952.

[2] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[3] J. H. Friedman. Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University, 1996.

[4] T. Hastie, R. Tibshirani, and A. Buja. Flexible discriminant analysis by optimal scoring. *J. Amer. Statist. Assoc.*, 89:1255–1270, 1994.

[5] David R. Karger. Random sampling in cut, flow, and network design problems. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 648–657. ACM Press, 1994.

[6] David R. Karger and Clifford Stein. A new approach to the minimum cut problem. *Journal of the ACM*, 43(4):601–640, 1996.

[7] Boonserm Kijsirikul, Nitiwut Ussivakul, and Surapan Meknavin. Adaptive directed acyclic graphs for multiclass classification. In *PRICAI 2002*, pages 158–168, 2002.

[8] S. Kneer, L. Personnaz, and G. Dreyfus. Single-layer learning revisited: A stepwise procedure for building and training a neural network. In Fogelman-Soulie and Herault, editors, *Neurocomputing: Algorithms, Architectures and Applications*, NATO ASI Series. Springer, 1990.

[9] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[10] Thimaporn Phetkaew, Boonserm Kijsirikul, and Wanchai Rivepiboon. Reordering adaptive directed acyclic graphs for multiclass support vector machines. In *Proceedings of the Third International Conference on Intelligent Technologies (InTech 2002)*, 2002.

[11] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification. In *Advance in Neural Information Processing System*, volume 12. MIT Press, 2000.

[12] N. Rattanakronkul and K. Waiyamai. Combining association rule discovery and data classification for protein structural class prediction. In *Proceedings of the International Conference On Bio-informatics 2002 (INCOP'2002)*, 2002.

[13] N. Rattanakronkul and K. Waiyamai. Predicting protein structural class from closed protein sequences. In *Springer-verlag Lectures Notes in Artificial Intelligence*, 2003.

[14] V. Vapnik. *Statistical Learning Theory.* Wiley, 1998.