Journal of Machine Learning Research 5 (2004) 975–1005 Submitted 11/03; Revised 05/04; Published 8/04

Probability Estimates for Multi-class Classification by Pairwise Coupling

Ting-Fan Wu b89098@csie.ntu.edu.tw
Chih-Jen Lin cjlin@csie.ntu.edu.tw
Department of Computer Science
National Taiwan University
Taipei 106, Taiwan

Ruby C. Weng chweng@nccu.edu.tw
Department of Statistics
National Chengchi University
Taipei 116, Taiwan

Editor: Yoram Singer

Abstract

Pairwise coupling is a popular multi-class classification method that combines all comparisons for each pair of classes. This paper presents two approaches for obtaining class probabilities. Both methods can be reduced to linear systems and are easy to implement. We show conceptually and experimentally that the proposed approaches are more stable than the two existing popular methods: voting and the method by Hastie and Tibshirani (1998).

Keywords: pairwise coupling, probability estimates, random forest, support vector machines

1. Introduction

The multi-class classification problem refers to assigning each of the observations into one of k classes. As two-class problems are much easier to solve, many authors propose to use two-class classifiers for multi-class classification. In this paper we focus on techniques that provide a multi-class probability estimate by combining all pairwise comparisons.

A common way to combine pairwise comparisons is by voting (Knerr et al., 1990; Friedman, 1996). It constructs a rule for discriminating between every pair of classes and then selecting the class with the most winning two-class decisions. Though the voting procedure requires just pairwise decisions, it only predicts a class label. In many scenarios, however, probability estimates are desired. As numerous (pairwise) classifiers do provide class probabilities, several authors (Refregier and Vallet, 1991; Price et al., 1995; Hastie and Tibshirani, 1998) have proposed probability estimates by combining the pairwise class probabilities.

Given the observation $\mathbf{x}$ and the class label y, we assume that the estimated pairwise class probabilities r_{ij} of $\mu_{ij} = P(y = i \mid y = i \text{ or } j, \mathbf{x})$ are available. From the ith and jth classes of a training set, we obtain a model which, for any new $\mathbf{x}$, calculates r_{ij} as an approximation of μ_{ij}. Then, using all r_{ij}, the goal is to estimate $p_i = P(y = i \mid \mathbf{x}), i = 1, \ldots, k$. In this paper, we first propose a method for obtaining probability estimates via an approximation solution to an identity. The existence of the solution is guaranteed by theory in finite Markov Chains. Motivated by the optimization formulation of this method, we propose a second approach. Interestingly, it can also be regarded as an improved version of the coupling approach given by Refregier and Vallet (1991). Both of the proposed methods can be reduced to solving linear systems and are simple in practical implementation. Furthermore, from conceptual and experimental points of view, we show that the two proposed methods are more stable than voting and the method by Hastie and Tibshirani (1998).

We organize the paper as follows. In Section 2, we review several existing methods. Sections 3 and 4 detail the two proposed approaches. Section 5 presents the relationship between different methods through their corresponding optimization formulas. In Section 6, we compare these methods using simulated data. In Section 7, we conduct experiments using real data. The classifiers considered are support vector machines and random forest. A preliminary version of this paper was presented previously (Wu et al., 2004).

2. Survey of Existing Methods

For methods surveyed in this section and those proposed later, each provides a vector of multi-class probability estimates. We denote it as $\mathbf{p}^*$ according to method $*$. Similarly, there is an associated rule $\arg\max_i[p_i^*]$ for prediction and we denote the rule as δ_*.

2.1 Voting

Let r_{ij} be the estimates of $\mu_{ij} \equiv P(y = i \mid y = i \text{ or } j, \mathbf{x})$ and assume $r_{ij} + r_{ji} = 1$. The voting rule (Knerr et al., 1990; Friedman, 1996) is

$$\delta_V = \arg\max_i [\sum_{j:j\neq i} I_{\{r_{ij}>r_{ji}\}}], \tag{1}$$

where I is the indicator function: $I\{x\} = 1$ if x is true, and 0 otherwise. A simple estimate of probabilities can be derived as

$$p_i^v = 2\sum_{j:j\neq i} I_{\{r_{ij}>r_{ji}\}}/(k(k-1)).$$

2.2 Method by Refregier and Vallet

With $\mu_{ij} = p_i/(p_i + p_j)$, Refregier and Vallet (1991) consider that

$$\frac{r_{ij}}{r_{ji}} \approx \frac{\mu_{ij}}{\mu_{ji}} = \frac{p_i}{p_j}. \tag{2}$$

Thus, making (2) an equality may be a way to solve p_i. However, the number of equations, $k(k-1)/2$, is more than the number of unknowns k, so Refregier and Vallet (1991) propose to choose any $k-1$ r_{ij}. Then, with the condition $\sum_{i=1}^k p_i = 1$, $\mathbf{p}^{RV}$ can be obtained by solving a linear system. However, as pointed out previously by Price et al. (1995), the results depend strongly on the selection of $k-1$ r_{ij}.

In Section 4, by considering (2) as well, we propose a method which remedies this problem.

2.3 Method by Price, Knerr, Personnaz, and Dreyfus

Price et al. (1995) consider that

$$\left(\sum_{j:j\neq i} P(y=i \text{ or } j \mid \mathbf{x})\right) - (k-2)P(y=i \mid \mathbf{x}) = \sum_{j=1}^{k} P(y=j \mid \mathbf{x}) = 1.$$

Using

$$r_{ij} \approx \mu_{ij} = \frac{P(y=i \mid \mathbf{x})}{P(y=i \text{ or } j \mid \mathbf{x})},$$

one obtains

$$p_i^{PKPD} = \frac{1}{\sum_{j:j\neq i} \frac{1}{r_{ij}} - (k-2)}. \tag{3}$$

As $\sum_{i=1}^{k} p_i = 1$ does not hold, we must normalize $\mathbf{p}^{PKPD}$. This approach is very simple and easy to implement. In the rest of this paper, we refer to this method as PKPD.

2.4 Method by Hastie and Tibshirani

Hastie and Tibshirani (1998) propose to minimize the Kullback-Leibler (KL) distance between r_{ij} and μ_{ij}:

$$\begin{aligned} l(\mathbf{p}) &= \sum_{i\neq j} n_{ij} r_{ij} \log \frac{r_{ij}}{\mu_{ij}}, \qquad (4) \\ &= \sum_{i<j} n_{ij} \left(r_{ij} \log \frac{r_{ij}}{\mu_{ij}} + (1-r_{ij}) \log \frac{1-r_{ij}}{1-\mu_{ij}} \right), \end{aligned}$$

where $\mu_{ij} = p_i/(p_i+p_j)$, $r_{ji} = 1-r_{ij}$, and n_{ij} is the number of training data in the ith and jth classes.

To minimize (4), they first calculate

$$\frac{\partial l(\mathbf{p})}{\partial p_i} = \sum_{j:j\neq i} n_{ij} \left(-\frac{r_{ij}}{p_i} + \frac{1}{p_i+p_j}\right).$$

Thus, letting $\partial l(\mathbf{p})/\partial p_i = 0, i = 1,\ldots,k$ and multiplying p_i on each term, Hastie and Tibshirani (1998) propose finding a point that satisfies

$$\sum_{j:j\neq i} n_{ij}\mu_{ij} = \sum_{j:j\neq i} n_{ij} r_{ij}, \qquad \sum_{i=1}^{k} p_i = 1, \text{and} \qquad p_i > 0, i = 1,\ldots,k. \tag{6}$$

Such a point is obtained by the following algorithm:

Algorithm 1

1. *Start with some initial* $p_j > 0, \forall j$ *and corresponding* $\mu_{ij} = p_i/(p_i+p_j)$.

2. *Repeat* $(i = 1, \ldots, k, 1, \ldots)$

$$\alpha = \frac{\sum_{j:j \neq i} n_{ij} r_{ij}}{\sum_{j:j \neq i} n_{ij} \mu_{ij}} \tag{7}$$

$$\mu_{ij} \leftarrow \frac{\alpha \mu_{ij}}{\alpha \mu_{ij} + \mu_{ji}}, \quad \mu_{ji} \leftarrow 1 - \mu_{ij}, \ \textit{for all } j \neq i \tag{8}$$

$$p_i \leftarrow \alpha p_i \tag{9}$$

$$\textit{normalize } \mathbf{p} \ (\textit{optional}) \tag{10}$$

until k consecutive α are all close to ones.

3. $\mathbf{p} \leftarrow \mathbf{p} / \sum_{i=1}^{k} p_i$

(9) implies that in each iteration, only the ith component is updated and all others remain the same. There are several remarks about this algorithm. First, the initial $\mathbf{p}$ must be positive so that all later $\mathbf{p}$ are positive and α is well defined (i.e., no zero denominator in (7)). Second, (10) is an optional operation because whether we normalize $\mathbf{p}$ or not does not affect the values of μ_{ij} and α in (7) and (8).

Hastie and Tibshirani (1998) prove that Algorithm 1 generates a sequence of points at which the KL distance is strictly decreasing. However, Hunter (2004) indicates that the strict decrease in $l(\mathbf{p})$ does not guarantee that any limit point satisfies (6). Hunter (2004) discusses the convergence of algorithms for generalized Bradley-Terry models where Algorithm 1 is a special case. It points out that Zermelo (1929) has proved that, if $r_{ij} > 0, \forall i \neq j$, for any initial point, the whole sequence generated by Algorithm 1 converges to a point satisfying (6). Furthermore, this point is the unique global minimum of $l(\mathbf{p})$ under the constraints $\sum_{i=1}^{k} p_i = 1$ and $0 \leq p_i \leq 1, i = 1, \ldots, k$.

Let $\mathbf{p}^{HT}$ denote the global minimum of $l(\mathbf{p})$. It is shown in Zermelo (1929) and Theorem 1 of (Hastie and Tibshirani, 1998) that if weights n_{ij} in (4) are considered equal, then $\mathbf{p}^{HT}$ satisfies

$$p_i^{HT} > p_j^{HT} \text{ if and only if } \tilde{p}_i^{HT} \equiv \frac{2 \sum_{s:i \neq s} r_{is}}{k(k-1)} > \tilde{p}_j^{HT} \equiv \frac{2 \sum_{s:j \neq s} r_{js}}{k(k-1)}. \tag{11}$$

Therefore, $\tilde{\mathbf{p}}^{HT}$ is sufficient if one only requires the classification rule. In fact, $\tilde{\mathbf{p}}^{HT}$ can be derived as an approximation to the identity

$$p_i = \sum_{j:j \neq i} \left(\frac{p_i + p_j}{k-1} \right) \left(\frac{p_i}{p_i + p_j} \right) = \sum_{j:j \neq i} \left(\frac{p_i + p_j}{k-1} \right) \mu_{ij} \tag{12}$$

by replacing $p_i + p_j$ with $2/k$, and μ_{ij} with r_{ij} in (12). We refer to the decision rule as δ_{HT}, which is essentially

$$\arg\max_i [\tilde{p}_i^{HT}]. \tag{13}$$

In the next two sections, we propose two methods which are simpler in both practical implementation and algorithmic analysis.

If the multi-class data are balanced, it is reasonable to assume equal weighting (i.e., $n_{ij} = 1$) as the above. In the rest of this paper, we restrict our discussion under such an assumption.

3. Our First Approach

As δ_{HT} relies on $p_i + p_j \approx 2/k$, in Section 6 we use two examples to illustrate possible problems with this rule. In this section, instead of replacing $p_i + p_j$ by $2/k$ in (12), we propose to solve the system:

$$p_i = \sum_{j:j\neq i} (\frac{p_i + p_j}{k-1}) r_{ij}, \forall i, \qquad \text{subject to } \sum_{i=1}^{k} p_i = 1, p_i \geq 0, \forall i. \tag{14}$$

Let $\mathbf{p}^1$ denote the solution to (14). Then the resulting decision rule is

$$\delta_1 = \arg\max_i [p_i^1].$$

3.1 Solving (14)

To solve (14), we rewrite it as

$$Q\mathbf{p} = \mathbf{p}, \quad \sum_{i=1}^{k} p_i = 1, \quad p_i \geq 0, \forall i, \text{ where } Q_{ij} \equiv \begin{cases} r_{ij}/(k-1) & \text{if } i \neq j, \\ \sum_{s:s\neq i} r_{is}/(k-1) & \text{if } i = j. \end{cases} \tag{15}$$

Observe that $\sum_{i=1}^{k} Q_{ij} = 1$ for $j = 1, \ldots, k$ and $0 \leq Q_{ij} \leq 1$ for $i, j = 1, \ldots, k$, so there exists a finite Markov Chain whose transition matrix is Q. Moreover, if $r_{ij} > 0$ for all $i \neq j$, then $Q_{ij} > 0$, which implies that this Markov Chain is irreducible and aperiodic. From Theorem 4.3.3 of Ross (1996), these conditions guarantee the existence of a unique stationary probability and all states being positive recurrent. Hence, we have the following theorem:

Theorem 1 *If $r_{ij} > 0$, $i \neq j$, then (15) has a unique solution $\mathbf{p}$ with $0 < p_i < 1$ $\forall i$.*

Assume the solution from Theorem 1 is $\mathbf{p}^*$. We claim that without the constraints $p_i \geq 0, \forall i$, the linear system

$$Q\mathbf{p} = \mathbf{p}, \qquad \sum_{i=1}^{k} p_i = 1 \tag{16}$$

still has the same unique solution $\mathbf{p}^*$. Otherwise, there is another solution $\bar{\mathbf{p}}^* (\neq \mathbf{p}^*)$. Then for any $0 \leq \lambda \leq 1$, $\lambda \mathbf{p}^* + (1-\lambda)\bar{\mathbf{p}}^*$ satisfies (16) as well. As $p_i^* > 0, \forall i$, when λ is sufficiently close to 1, $\lambda p_i^* + (1-\lambda)\bar{p}_i^* > 0, i = 1, \ldots, k$. This violates the uniqueness property in Theorem 1.

Therefore, unlike the method in Section 2.4 where a special iterative procedure has to be implemented, here we only solve a simple linear system. As (16) has $k+1$ equalities but only k variables, practically we remove any one equality from $Q\mathbf{p} = \mathbf{p}$ and obtain a square system. Since the column sum of Q is the vector of all ones, the removed equality is a linear combination of all remaining equalities. Thus, any solution of the square system satisfies (16) and vice versa. Therefore, this square system has the same unique solution as (16) and hence can be solved by standard Gaussian elimination.

Instead of Gaussian elimination, as the stationary solution of a Markov Chain can be derived by the limit of the n-step transition probability matrix Q^n, we can solve (14) by repeatedly multiplying Q with any initial probability vector.

3.2 Another Look at (14)

The following arguments show that the solution to (14) is a global minimum of a meaningful optimization problem. To begin, using the property that $r_{ij}+r_{ji}=1, \forall i \neq j$, we re-express $p_i = \sum_{j:j\neq i}(\frac{p_i+p_j}{k-1})r_{ij}$ of (15) (i.e., $Q\mathbf{p}=\mathbf{p}$ of (16)) as

$$\sum_{j:j\neq i} r_{ji}p_i - \sum_{j:j\neq i} r_{ij}p_j = 0, i=1,\ldots,k.$$

Therefore, a solution of (15) is in fact the unique global minimum of the following convex problem:

$$\begin{aligned} \min_{\mathbf{p}} \quad & \sum_{i=1}^{k}(\sum_{j:j\neq i} r_{ji}p_i - \sum_{j:j\neq i} r_{ij}p_j)^2 \\ \text{subject to} \quad & \sum_{i=1}^{k} p_i = 1, \qquad p_i \geq 0, i=1,\ldots,k. \end{aligned} \tag{18}$$

The reason is that the object function is always nonnegative, and it attains zero under (15). Note that the constraints $p_i \geq 0\ \forall i$ are not redundant following the discussion around Equation (16).

4. Our Second Approach

Note that both approaches in Sections 2.4 and 3 involve solving optimization problems using relations like $p_i/(p_i+p_j) \approx r_{ij}$ or $\sum_{j:j\neq i} r_{ji}p_i \approx \sum_{j:j\neq i} r_{ij}p_j$. Motivated by (18), we suggest another optimization formulation as follows:

$$\min_{\mathbf{p}} \sum_{i=1}^{k}\sum_{j:j\neq i}(r_{ji}p_i - r_{ij}p_j)^2 \qquad \text{subject to } \sum_{i=1}^{k} p_i = 1, p_i \geq 0, \forall i. \tag{19}$$

Note that the method (Refregier and Vallet, 1991) described in Section 2.2 considers a random selection of $k-1$ equations of the form $r_{ji}p_i = r_{ij}p_j$. As (19) considers all $r_{ij}p_j - r_{ji}p_i$, not just $k-1$ of them, it can be viewed as an improved version of the coupling approach by Refregier and Vallet (1991).

Let $\mathbf{p}^2$ denote the corresponding solution. We then define the classification rule as

$$\delta_2 = \arg\max_i [p_i^2].$$

4.1 A Linear System from (19)

Since (18) has a unique solution, which can be obtained by solving a simple linear system, it is desirable to see whether the minimization problem (19) has these nice properties. In this subsection, we show that (19) has a unique solution and can be solved by a simple linear system.

First, the following theorem shows that the nonnegative constraints in (19) are redundant.

Theorem 2 *Problem* (19) *is equivalent to*

$$\min_{\mathbf{p}} \sum_{i=1}^{k} \sum_{j:j\neq i} (r_{ji}p_i - r_{ij}p_j)^2 \qquad \text{subject to} \sum_{i=1}^{k} p_i = 1. \tag{20}$$

The proof is in Appendix A. Note that we can rewrite the objective function of (20) as

$$\min_{\mathbf{p}} \;\; 2\mathbf{p}^T Q\mathbf{p} \equiv \min_{\mathbf{p}} \;\; \frac{1}{2}\mathbf{p}^T Q\mathbf{p}, \tag{21}$$

where

$$Q_{ij} = \begin{cases} \sum_{s:s\neq i} r_{si}^2 & \text{if } i = j, \\ -r_{ji}r_{ij} & \text{if } i \neq j. \end{cases} \tag{22}$$

We divide the objective function by a positive factor of four so its derivative is a simple form $Q\mathbf{p}$. From (22), Q is positive semi-definite as for any $\mathbf{v} \neq \mathbf{0}, \mathbf{v}^T Q\mathbf{v} = 1/2 \sum_{i=1}^{k}\sum_{j=1}^{k}(r_{ji}v_i - r_{ij}v_j)^2 \geq 0$. Therefore, without constraints $p_i \geq 0, \forall i$, (21) is a linear-equality-constrained convex quadratic programming problem. Consequently, a point $\mathbf{p}$ is a global minimum if and only if it satisfies the optimality condition: There is a scalar b such that

$$\begin{bmatrix} Q & \mathbf{e} \\ \mathbf{e}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ b \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix}. \tag{23}$$

Here $Q\mathbf{p}$ is the derivative of (21), b is the Lagrangian multiplier of the equality constraint $\sum_{i=1}^{k} p_i = 1$, $\mathbf{e}$ is the $k \times 1$ vector of all ones, and $\mathbf{0}$ is the $k \times 1$ vector of all zeros. Thus, the solution to (19) can be obtained by solving the simple linear system (23).

4.2 Solving (23)

Equation (23) can be solved by some direct methods in numerical linear algebra. Theorem 3(i) below shows that the matrix in (23) is invertible; therefore, Gaussian elimination can be easily applied.

For symmetric positive definite systems, Cholesky factorization reduces the time for Gaussian elimination by half. Though (23) is symmetric but not positive definite, if Q is positive definite, Cholesky factorization can be used to obtain $b = -1/(\mathbf{e}^T Q^{-1}\mathbf{e})$ first and then $\mathbf{p} = -bQ^{-1}\mathbf{e}$. Theorem 3(ii) shows that Q is positive definite under quite general conditions. Moreover, even if Q is only positive semi-definite, Theorem 3(i) proves that $Q + \Delta\mathbf{e}\mathbf{e}^T$ is positive definite for any constant $\Delta > 0$. Along with the fact that (23) is equivalent to

$$\begin{bmatrix} Q + \Delta\mathbf{e}\mathbf{e}^T & \mathbf{e} \\ \mathbf{e}^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p} \\ b \end{bmatrix} = \begin{bmatrix} \Delta\mathbf{e} \\ 1 \end{bmatrix},$$

we can do Cholesky factorization on $Q + \Delta\mathbf{e}\mathbf{e}^T$ and solve b and $\mathbf{p}$ similarly, regardless whether Q is positive definite or not.

Theorem 3 *If $r_{tu} > 0 \; \forall t \neq u$, we have* *(i) For any $\Delta > 0$, $Q + \Delta \mathbf{e}\mathbf{e}^T$ is positive definite. In addition, $\begin{bmatrix} Q & \mathbf{e} \\ \mathbf{e}^T & 0 \end{bmatrix}$ is invertible, and hence (19) has a unique global minimum.*

(ii) If for any $i = 1, \ldots, k$, there are $s \neq j$ for which $s \neq i$, $j \neq i$, and

$$\frac{r_{si} r_{sj}}{r_{is}} \neq \frac{r_{ji} r_{js}}{r_{ij}}, \tag{25}$$

then Q is positive definite.

We leave the proof in Appendix B.

In addition to direct methods, next we propose a simple iterative method for solving (23):

Algorithm 2

1. *Start with some initial $p_i \geq 0, \forall i$ and $\sum_{i=1}^k p_i = 1$.*

2. *Repeat ($t = 1, \ldots, k, 1, \ldots$)*

$$p_t \leftarrow \frac{1}{Q_{tt}}[-\sum_{j:j \neq t} Q_{tj} p_j + \mathbf{p}^T Q \mathbf{p}] \tag{26}$$

$$\textit{normalize } \mathbf{p} \tag{27}$$

until (23) is satisfied.

Equation (22) and the assumption $r_{ij} > 0, \forall i \neq j$, ensure that the right-hand side of (26) is always nonnegative. For (27) to be well defined, we must ensure that $\sum_{i=1}^k p_i > 0$ after the operation in (26). This property holds (see (43) for more explanation). With $b = -\mathbf{p}^T Q \mathbf{p}$ obtained from (23), (26) is motivated from the tth equality in (23) with b replaced by $-\mathbf{p}^T Q \mathbf{p}$. The convergence of Algorithm 2 is established in the following theorem:

Theorem 4 *If $r_{sj} > 0, \forall s \neq j$, then $\{\mathbf{p}^i\}_{i=1}^\infty$, the sequence generated by Algorithm 2, converges globally to the unique minimum of (19).*

The proof is in Appendix C. Algorithm 2 is implemented in the software LIBSVM developed by Chang and Lin (2001) for multi-class probability estimates. We discuss some implementation issues of Algorithm 2 in Appendix D.

5. Relations Between Different Methods

Among the methods discussed in this paper, the four decision rules δ_{HT}, δ_1, δ_2, and δ_V can be written as $\arg\max_i [p_i]$, where $\mathbf{p}$ is derived by the following four optimization formulations under the constraints $\sum_{i=1}^{k} p_i = 1$ and $p_i \geq 0, \forall i$:

$$\delta_{HT} : \min_{\mathbf{p}} \sum_{i=1}^{k} [\sum_{j:j \neq i} (r_{ij} \frac{1}{k} - \frac{1}{2} p_i)]^2, \tag{28}$$

$$\delta_1 : \quad \min_{\mathbf{p}} \sum_{i=1}^{k} [\sum_{j:j \neq i} (r_{ij} p_j - r_{ji} p_i)]^2, \tag{29}$$

$$\delta_2 : \quad \min_{\mathbf{p}} \sum_{i=1}^{k} \sum_{j:j \neq i} (r_{ij} p_j - r_{ji} p_i)^2, \tag{30}$$

$$\delta_V : \quad \min_{\mathbf{p}} \sum_{i=1}^{k} \sum_{j:j \neq i} (I_{\{r_{ij} > r_{ji}\}} p_j - I_{\{r_{ji} > r_{ij}\}} p_i)^2. \tag{31}$$

Note that (28) can be easily verified from (11), and that (29) and (30) have been explained in Sections 3 and 4. For (31), its solution is

$$p_i = \frac{c}{\sum_{j:j \neq i} I_{\{r_{ji} > r_{ij}\}}}, \tag{32}$$

where c is the normalizing constant; and therefore, $\arg\max_i [p_i]$ is the same as (1).[1] Detailed derivation of (32) is in Appendix E.

Clearly, (28) can be obtained from (29) by letting $p_j = 1/k$ and $r_{ji} = 1/2$. Such approximations ignore the differences between p_i. Next, (31) is from (30) with r_{ij} replaced by $I_{\{r_{ij} > r_{ji}\}}$, and hence, (31) may enlarge the differences between p_i. Moreover, compared with (30), (29) allows the difference between $r_{ij} p_j$ and $r_{ji} p_i$ to be canceled first, so (29) may tend to underestimate the differences between p_i. In conclusion, conceptually, (28) and (31) are more extreme – the former tends to underestimate the differences between p_i, while the latter overestimates them. These arguments will be supported by simulated and real data in the next two sections.

For PKPD approach (3), the decision rule can be written as:

$$\delta_{PKPD} = \arg\min_i [\sum_{j:j \neq i} \frac{1}{r_{ij}}].$$

This form looks similar to $\delta_{HT} = \arg\max_i [\sum_{j:j \neq i} r_{ij}]$, which can be obtained from (11) and (13). Notice that the differences among $\sum_{j:j \neq i} r_{ij}$ tend to be larger than those among $\sum_{j:j \neq i} \frac{1}{r_{ij}}$, because $1/r_{ij} > 1 > r_{ij}$. More discussion on these two rules will be given in Section 6.

6. Experiments on Synthetic Data

In this section, we use synthetic data to compare the performance of existing methods described in Section 2 as well as two new approaches proposed in Sections 3 and 4. Here we

1. For $I_{\{r_{ij} > r_{ji}\}}$ to be well defined, we consider $r_{ij} \neq r_{ji}$, which is generally true. In addition, if there is an i for which $\sum_{j:j \neq i} I_{\{r_{ji} > r_{ij}\}} = 0$, an optimal solution of (31) is $p_i = 1$, and $p_j = 0, \forall j \neq i$. The resulting decision is the same as that of (1).

do not include the method in Section 2.2 because its results depend strongly on the choice of $k-1$ r_{ij} and our second method is an improved version of it.

Hastie and Tibshirani (1998) design a simple experiment in which all p_i are fairly close and their method δ_{HT} outperforms the voting strategy δ_V. We conduct this experiment first to assess the performance of our proposed methods. Following their settings, we define class probabilities

(a) $p_1 = 1.5/k$, $p_j = (1-p_1)/(k-1)$, $j = 2, \ldots, k$,

and then set

$$r_{ij} = \frac{p_i}{p_i + p_j} + 0.1 z_{ij} \quad \text{if } i > j, \tag{33}$$

$$r_{ji} = \frac{p_j}{p_i + p_j} + 0.1 z_{ji} = 1 - r_{ij} \text{ if } j > i, \tag{34}$$

where z_{ij} are standard normal variates and $z_{ji} = -z_{ij}$. Since r_{ij} are required to be within (0,1), we truncate r_{ij} at ϵ below and $1-\epsilon$ above, with $\epsilon = 10^{-7}$. In this example, class 1 has the highest probability and hence is the correct class.

Figure 2(a) shows accuracy rates for each of the five methods when $k = 2^2, \lceil 2^{2.5} \rceil, 2^3, \ldots, 2^7$, where $\lceil x \rceil$ denotes the largest integer not exceeding x. The accuracy rates are averaged over 1,000 replicates. Note that in this experiment all classes are quite competitive, so, when using δ_V, sometimes the highest vote occurs at two or more different classes. We handle this problem by randomly selecting one class from the ties. This partly explains the poor performance of δ_V. Another explanation is that the r_{ij} here are all close to 1/2, but (31) uses 1 or 0 instead, as stated in the previous section; therefore, the solution may be severely biased. Besides δ_V, the other four rules have good performance in this example.

Since δ_{HT} relies on the approximation $p_i + p_j \approx k/2$, this rule may suffer some losses if the class probabilities are not highly balanced. To examine this point, we consider the following two sets of class probabilities:

(b) We let $k_1 = k/2$ if k is even, and $(k+1)/2$ if k is odd; then we define $p_1 = 0.95 \times 1.5/k_1$, $p_i = (0.95 - p_1)/(k_1 - 1)$ for $i = 2, \ldots, k_1$, and $p_i = 0.05/(k - k_1)$ for $i = k_1 + 1, \ldots, k$.

(c) We define $p_1 = 0.95 \times 1.5/2$, $p_2 = 0.95 - p_1$, and $p_i = 0.05/(k-2), i = 3, \ldots, k$.

An illustration of these three sets of class probabilities is in Figure 1.

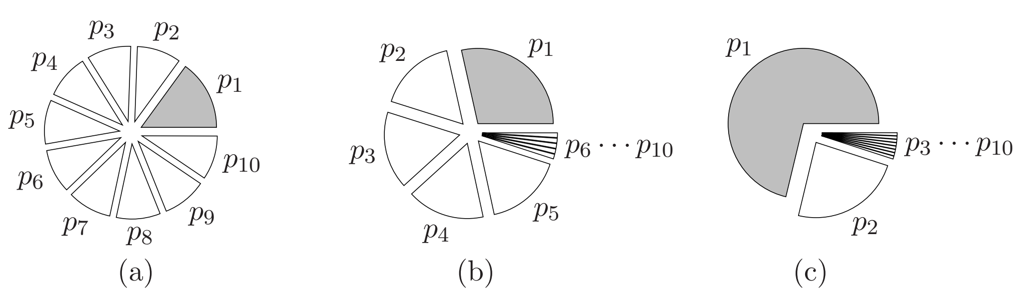


Figure 1: Three sets of class probabilities

After setting p_i, we define the pairwise comparisons r_{ij} as in (33)-(34). Both experiments are repeated for 1,000 times. The accuracy rates are shown in Figures 2(b) and 2(c). In both scenarios, p_i are not balanced. As expected, δ_{HT} is quite sensitive to the imbalance of p_i. The situation is much worse in Figure 2(c) because the approximation $p_i + p_j \approx k/2$ is more seriously violated, especially when k is large.

A further analysis of Figure 2(c) shows that when k is large,

$$\begin{aligned}
&r_{12} = \frac{3}{4} + 0.1z_{12}, r_{1j} \approx 1 + 0.1z_{1j}, j \geq 3, \\
&r_{21} = \frac{1}{4} + 0.1z_{21}, r_{2j} \approx 1 + 0.1z_{2j}, j \geq 3, \\
&r_{ij} \approx 0 + 0.1z_{ij}, i \neq j, i \geq 3,
\end{aligned}$$

where $z_{ji} = -z_{ij}$ are standard normal variates. From (11), the decision rule δ_{HT} in this case is mainly on comparing $\sum_{j:j\neq 1} r_{1j}$ and $\sum_{j:j\neq 2} r_{2j}$. The difference between these two sums is $\frac{1}{2} + 0.1(\sum_{j:j\neq 1} z_{1j} - \sum_{j:j\neq 2} z_{2j})$, where the second term has zero mean and, when k is large, high variance. Therefore, for large k, the decision depends strongly on these normal variates, and the probability of choosing the first class is approaching half. On the other hand, δ_{PKPD} relies on comparing $\sum_{j:j\neq 1} 1/r_{1j}$ and $\sum_{j:j\neq 2} 1/r_{2j}$. As the difference between $1/r_{12}$ and $1/r_{21}$ is larger than that between r_{12} and r_{21}, though the accuracy rates decline when k increases, the situation is less serious.

We also analyze the mean square error (MSE) in Figure 3:

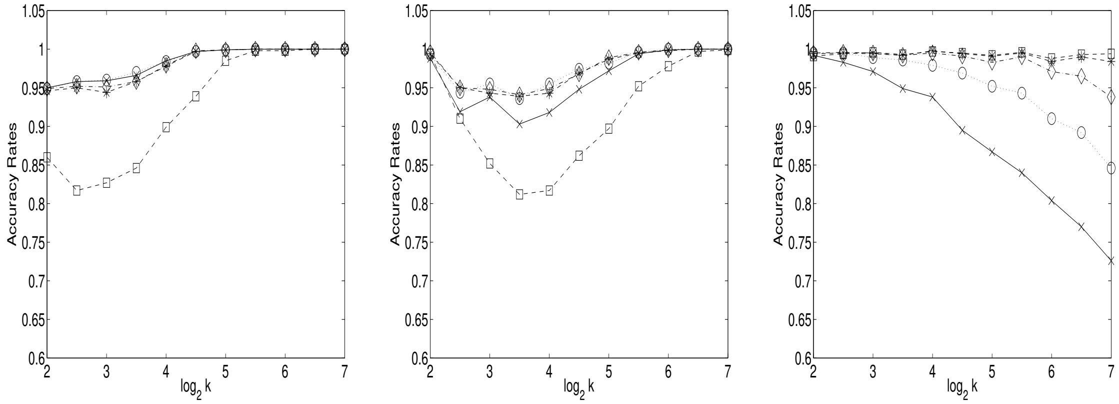
$$\text{MSE} = \frac{1}{1000} \sum_{j=1}^{1000} \frac{1}{k} \sum_{i=1}^{k} (\hat{p}_i^j - p_i)^2, \tag{35}$$

where $\hat{\mathbf{p}}^j$ is the probability estimate obtained in the jth of the 1,000 replicates. Overall, δ_{HT} and δ_V have higher MSE, confirming again that they are less stable. Note that Algorithm 1 and (11) give the same prediction for δ_{HT}, but their MSE are different. Here we consider (11) as it is the one analyzed and compared in Section 5.

In summary, δ_1 and δ_2 are less sensitive to p_i, and their overall performance are fairly stable. All observations about δ_{HT}, δ_1, δ_2, and δ_V here agree with our analysis in Section 5. Despite some similarity to δ_{HT}, δ_{PKPD} outperforms δ_{HT} in general. Experiments in this study are conducted using MATLAB.

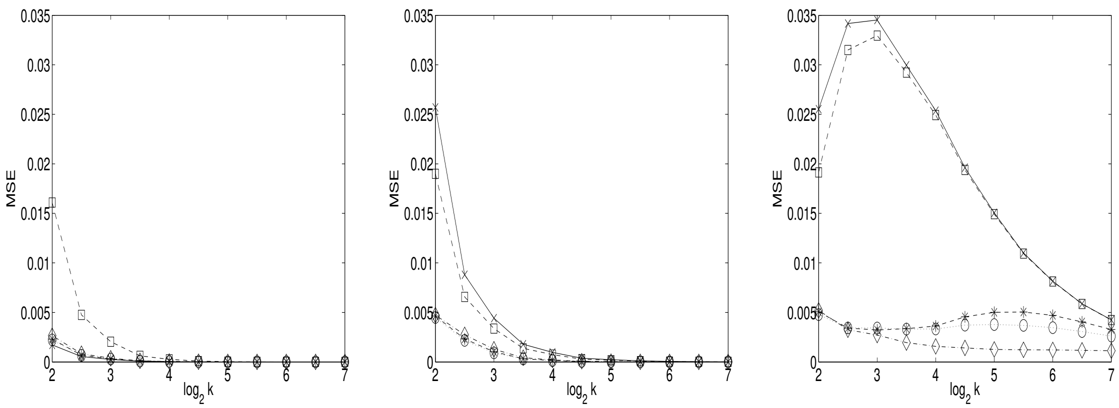
7. Experiments on Real Data

In this section we present experimental results on several multi-class problems: dna, satimage, segment, and letter from the Statlog collection (Michie et al., 1994), waveform from UCI Machine Learning Repository (Blake and Merz, 1998), USPS (Hull, 1994), and MNIST (LeCun et al., 1998). The numbers of classes and features are reported in Table 7. Except dna, which takes two possible values 0 and 1, each attribute of all other data is linearly scaled to $[-1, 1]$. In each scaled data, we randomly select 300 training and 500 testing instances from thousands of data points. 20 such selections are generated and the testing error rates are averaged. Similarly, we do experiments on larger sets (800 training and 1,000 testing). All training and testing sets used are available at `http://`



(a) (b) (c)

Figure 2: Accuracy of predicting the true class by the methods: δ_{HT} (solid line, cross marked), δ_V (dashed line, square marked), δ_1 (dotted line, circle marked), δ_2 (dashed line, asterisk marked), and δ_{PKPD} (dashdot line, diamond marked).



(a) (b) (c)

Figure 3: MSE by the methods: δ_{HT} via (11) (solid line, cross marked), δ_V (dashed line, square marked), δ_1 (dotted line, circle marked), δ_2 (dashed line, asterisk marked), and δ_{PKPD} (dashdot line, diamond marked).

`www.csie.ntu.edu.tw/~cjlin/papers/svmprob/data` and the code is available at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/svmprob/svmprob-1.0.tgz`.

For the implementation of the four probability estimates, δ_1 and δ_2 are via solving linear systems. For δ_{HT}, we implement Algorithm 1 with the following stopping condition

$$\sum_{i=1}^{k} \left| \frac{\sum_{j:j\neq i} r_{ij}}{\sum_{j:j\neq i} \mu_{ij}} - 1 \right| \leq 10^{-3}.$$

We observe that the performance of δ_{HT} may downgrade if the stopping condition is too loose.

dataset	dna	waveform	satimage	segment	USPS	MNIST	letter
#class	3	3	6	7	10	10	26
#attribute	180	21	36	19	256	784	16

Table 1: Data set Statistics

7.1 SVM as the Binary Classifier

We first consider support vector machines (SVM) (Boser et al., 1992; Cortes and Vapnik, 1995) with the RBF kernel $e^{-\gamma\|\mathbf{x}_i-\mathbf{x}_j\|^2}$ as the binary classifier. The regularization parameter C and the kernel parameter γ are selected by cross-validation (CV). To begin, for each training set, a five-fold cross-validation is conducted on the following points of (C,γ): $[2^{-5}, 2^{-3}, \ldots, 2^{15}] \times [2^{-5}, 2^{-3}, \ldots, 2^{15}]$. This is done by modifying LIBSVM (Chang and Lin, 2001), a library for SVM. At each (C,γ), sequentially four folds are used as the training set while one fold as the validation set. The training of the four folds consists of $k(k-1)/2$ binary SVMs. For the binary SVM of the ith and jth classes, we employ an improved implementation (Lin et al., 2003) of Platt's posterior probabilities (Platt, 2000) to estimate r_{ij}:

$$r_{ij} = P(i \mid i \text{ or } j, \mathbf{x}) = \frac{1}{1+e^{A\hat{f}+B}}, \tag{36}$$

where A and B are estimated by minimizing the negative log-likelihood function, and $\hat{f}$ are the decision values of training data. Platt (2000) and Zhang (2004) observe that SVM decision values are easily clustered at ± 1, so the probability estimate (36) may be inaccurate. Thus, it is better to use CV decision values as we less overfit the model and values are not so close to ± 1. In our experiments here, this requires a further CV on the four-fold data (i.e., a second level CV).

Next, for each instance in the validation set, we apply the pairwise coupling methods to obtain classification decisions. The error of the five validation sets is thus the cross-validation error at (C,γ). From this, each rule obtains its best (C,γ).[2] Then, the decision values from the five-fold cross-validation at the best (C,γ) are employed in (36) to find the final A and B for future use. These two values and the model via applying the best parameters on the whole training set are then used to predict testing data. Figure 4 summarizes the procedure of getting validation accuracy at each given (C,γ).

2. If more than one parameter sets return the smallest cross-validation error, we simply choose the one with the smallest C.

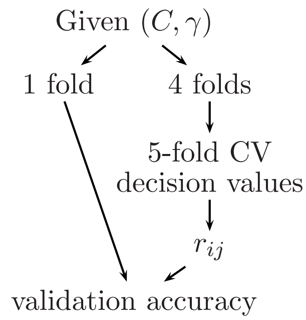


Figure 4: Parameter selection when using SVM as the binary classifier

The average of 20 MSEs are presented on the left panel of Figure 5, where the solid line represents results of small sets (300 training/500 testing), and the dashed line of large sets (800 training/1,000 testing). The definition of MSE here is similar to (35), but as there is no correct p_i for these problems, we let $p_i = 1$ if the data is in the ith class, and 0 otherwise. This measurement is called Brier Score (Brier, 1950), which is popular in meteorology. The figures show that for smaller k, δ_{HT}, δ_1, δ_2 and δ_{PKPD} have similar MSEs, but for larger k, δ_{HT} has the largest MSE. The MSEs of δ_V are much larger than those by all other methods, so they are not included in the figures. In summary, the two proposed approaches, δ_1 and δ_2, are fairly insensitive to the values of k, and all above observations agree well with previous findings in Sections 5 and 6.

Next, left panels of Figures 6 and 7 present the average of 20 test errors for problems with small size (300 training/500 testing) and large size (800 training/1,000 testing), respectively. The caption of each sub-figure also shows the average of 20 test errors of the multi-class implementation in LIBSVM. This rule is voting using merely pairwise SVM decision values, and is denoted as δ_{DV} for later discussion. The figures show that the errors of the five methods are fairly close for smaller k, but quite different for larger k. Notice that for smaller k (Figures 6 and 7 (a), (c), (e), and (g)) the differences of the averaged errors among the five methods are small, and there is no particular trend in these figures. However, for problems with larger k (Figures 6 and 7 (i), (k), and (m)), the differences are bigger and δ_{HT} is less competitive. In particular, for letter problem (Figure 6 (m), k =26), δ_2 and δ_V outperform δ_{HT} by more than 4%. The test errors along with MSE seems to indicate that, for problems with larger k, the posterior probabilities p_i are closer to the setting of Figure 2(c), rather than that of Figure 2(a). Another feature consistent with earlier findings is that when k is larger the results of δ_2 are closer to those of δ_V, and δ_1 closer to δ_{HT}, for both small and large training/testing sets. As for δ_{PKPD}, its overall performance is competitive, but we are not clear about its relationships to the other methods.

Finally, we consider another criterion on evaluating the probability estimates: the likelihood.

$$\prod_{j=1}^{l} p_{y_j}^j$$

In practice, we use its log likelihood and divide the value by a scaling factor l:

$$\frac{1}{l}\sum_{j=1}^{l}\log p^j_{y_j}, \tag{37}$$

where l is the number of test data, $\mathbf{p}^j$ is the probability estimates for the jth data, and y_j is its actual class label.

A larger value implies a possibly better estimate. The left panel of Figure 8 presents the results of using SVM as the binary classifier. Clearly the trend is the same as MSE and accuracy. When k is larger, δ_2 and δ_V have larger values and hence better performance. Similar to MSE, values of δ_V are not presented as they are too small.

7.2 Random Forest as the Binary Classifier

In this subsection we consider random forest (Breiman, 2001) as the binary classifier and conduct experiments on the same data sets. As random forest itself can provide multi-class probability estimates, we denote the corresponding rule as δ_{RF} and also compare it with the coupling methods.

For each two classes of data, we construct 500 trees as the random forest classifiers. Using m_{try} randomly selected features, a bootstrap sample (around two thirds) of training data are employed to generate a full tree without pruning. For each test instance, r_{ij} is simply the proportion out of the 500 trees that class i wins over class j. As we set the number of trees to be fixed at 500, the only parameter left for tuning is m_{try}. Similar to (Sventnik et al., 2003), we select m_{try} from $\{1, \sqrt{m}, m/3, m/2, m\}$ by five-fold cross validation, where m is the number of attributes. The cross validation procedure first sequentially uses four folds as the training set to construct $k(k-1)/2$ pairwise random forests, next obtains the decision for each instance in the validation set by the pairwise coupling methods, and then calculates the cross validation error at the given m_{try} by the error of five validation sets. This is similar to the procedure in Section 7.1, but we do not need a second-level CV for obtaining accurate two-class probabilistic estimates (i.e., r_{ij}). Instead of CV, a more efficient "out of bag" validation can be used for random forest, but here we keep using CV for consistency. Experiments are conducted using an R-interface (Liaw and Wiener, 2002) to the code from (Breiman, 2001).

The MSE presented in the right panel of Figure 5 shows that δ_1 and δ_2 yield more stable results than δ_{HT} and δ_V for both small and large sets. The right panels of Figures 6 and 7 give the average of 20 test errors. The caption of each sub-figure also shows the averaged error when using random forest as a multi-class classifier (δ_{RF}). Notice that random forest bears a resemblance to SVM: the errors are only slightly different among the five methods for smaller k, but δ_V and δ_2 tend to outperform δ_{HT} and δ_1 for larger k. The right panel of Figure 8 presents the log likelihood value (37). The trend is again the same. In summary, the results by using random forest as the binary classifier strongly support previous findings regarding the four methods.

7.3 Miscellaneous Observations and Discussion

Recall that in Section 7.1 we consider δ_{DV}, which does not use Platt's posterior probabilities. Experimental results in Figure 6 show that δ_{DV} is quite competitive (in particular, 3% better for letter), but is about 2% worse than all probability-based methods for waveform. Similar observations on waveform are also reported in (Duan and Keerthi, 2003), where the comparison is between δ_{DV} and δ_{HT}. We explain why the results by probability-based and decision-value-based methods can be so distinct. For some problems, the parameters selected by δ_{DV} are quite different from those by the other five rules. In waveform, at some parameters all probability-based methods gives much higher cross validation accuracy than δ_{DV}. We observe, for example, the decision values of validation sets are in [0.73, 0.97] and [0.93, 1.02] for data in two classes; hence, all data in the validation sets are classified as in one class and the error is high. On the contrary, the probability-based methods fit the decision values by a sigmoid function, which can better separate the two classes by cutting at a decision value around 0.95. This observation shed some light on the difference between probability-based and decision-value based methods.

Though the main purpose of this section is to compare different probability estimates, here we check the accuracy of another multi-class classification method: exponential loss-based decoding by Allwein et al. (2001). In the pairwise setting, if $\hat{f}_{ij} \in R$ is the two-class hypothesis so that $\hat{f}_{ij} > 0$ (< 0) predicts the data to be in the ith (jth) class, then

$$\text{predicted label} = \arg\min_i \left(\sum_{j:j<i} e^{\hat{f}_{ji}} + \sum_{j:j>i} e^{-\hat{f}_{ij}} \right). \tag{38}$$

For SVM, we can simply use decision values as $\hat{f}_{ij}$. On the other hand, $r_{ij} - 1/2$ is another choice. Table 2 presents the error of the seven problem using these two options. Results indicate that using decision values is worse than $r_{ij} - 1/2$ when k is large (USPS,MNIST, and letter). This observation seems to indicate that large numerical ranges of $\hat{f}_{ij}$ may cause (38) to have more erroneous results ($r_{ij} - 1/2$ is always in $[-1/2, 1/2]$). The results of using $r_{ij} - 1/2$ is competitive with those in Figures 6 and 7 when k is small. However, for larger k (e.g., letter), it is slightly worse than δ_2 and δ_V. We think this result is due to the similarity between (38) and δ_{HT}. When $\hat{f}_{ij}$ is close to zero, $e^{\hat{f}_{ij}} \approx 1 + \hat{f}_{ij}$, so (38) reduces to a "linear loss-based encoding." When $r_{ij} - 1/2$ is used, $\hat{f}_{ji} = r_{ji} - 1/2 = 1/2 - r_{ij}$. Thus, the linear encoding is $\arg\min_i[\sum_{j:j\neq i} -r_{ij}] \equiv \arg\max_i[\sum_{j:j\neq i} r_{ij}]$, exactly the same as (11) of δ_{HT}.

training/testing ($\hat{f}_{ij}$)	dna	waveform	satimage	segment	USPS	MNIST	letter
300/500 (dec. values)	10.47	16.23	14.12	6.21	11.57	14.99	38.59
300/500 ($r_{ij} - 1/2$)	10.47	15.11	14.45	6.03	11.08	13.58	38.27
800/1000 (dec. values)	6.36	14.20	11.55	3.35	8.47	8.97	22.54
800/1000 ($r_{ij} - 1/2$)	6.22	13.45	11.6	3.19	7.71	7.95	20.29

Table 2: Average of 20 test errors using exponential loss-based decoding (in percentage)

Regarding the accuracy of pairwise (i.e., δ_{DV}) and non-pairwise (e.g., "one-against-the-rest") multi-class classification methods, there are already excellent comparisons. As δ_V

and δ_2 have similar accuracy to δ_{DV}, roughly how non-pairwise methods compared to δ_{DV} is the same as compared to δ_V and δ_2.

The results of random forest as a multi-class classifier (i.e., δ_{RF}) are reported in the caption of each sub-figure in Figures 6 and 7. We observe from the figures that, when the number of classes is larger, using random forest as a multi-class classifier is better than coupling binary random forests. However, for `dna` ($k = 3$) the result is the other way around. This observation for random forest is left as a future research issue.

Acknowledgments

The authors thank S. Sathiya Keerthi for helpful comments. They also thank the associate editor and anonymous referees for useful comments. This work was supported in part by the National Science Council of Taiwan via the grants NSC 90-2213-E-002-111 and NSC 91-2118-M-004-003.

Appendix A. Proof of Theorem 2

It suffices to prove that any optimal solution $\mathbf{p}$ of (20) satisfies $p_i \geq 0, i = 1, \ldots, k$. If this is not true, without loss of generality, we assume

$$p_1 < 0, \ldots, p_r < 0, p_{r+1} \geq 0, \ldots, p_k \geq 0,$$

where $r < k$ because $\sum_{i=1}^k p_i = 1$. We can then define a new feasible solution of (20):

$$p'_1 = 0, \ldots, p'_r = 0, p'_{r+1} = p_{r+1}/\alpha, \ldots, p'_k = p_k/\alpha,$$

where $\alpha = 1 - \sum_{i=1}^r p_i > 1$.

With $r_{ij} > 0$ and $r_{ji} > 0$, we obtain

$$\begin{aligned}
&(r_{ji}p_i - r_{ij}p_j)^2 \geq 0 = (r_{ji}p'_i - r_{ij}p'_j)^2, \text{ if } 1 \leq i, j \leq r, \\
&(r_{ji}p_i - r_{ij}p_j)^2 > \frac{(r_{ij}p_j)^2}{\alpha^2} = (r_{ji}p'_i - r_{ij}p'_j)^2, \text{ if } 1 \leq i \leq r, r+1 \leq j \leq k, \\
&(r_{ji}p_i - r_{ij}p_j)^2 \geq \frac{(r_{ji}p_i - r_{ij}p_j)^2}{\alpha^2} = (r_{ji}p'_i - r_{ij}p'_j)^2, \text{ if } r+1 \leq i, j \leq k.
\end{aligned}$$

Therefore,

$$\sum_{i=1}^k \sum_{j:j \neq i} (r_{ij}p_i - r_{ji}p_j)^2 > \sum_{i=1}^k \sum_{j:j \neq i} (r_{ij}p'_i - r_{ji}p'_j)^2.$$

This contradicts the assumption that $\mathbf{p}$ is an optimal solution of (20).

Appendix B. Proof of Theorem 3

(i) If $Q + \Delta \mathbf{e}\mathbf{e}^T$ is not positive definite, there is a vector $\mathbf{v}$ with $v_i \neq 0$ such that

$$\mathbf{v}^T(Q + \Delta \mathbf{e}\mathbf{e}^T)\mathbf{v} = \frac{1}{2} \sum_{t=1}^k \sum_{u:u \neq t} (r_{ut}v_t - r_{tu}v_u)^2 + \Delta(\sum_{t=1}^k v_t)^2 = 0. \tag{39}$$

For all $t \neq i$, $r_{it}v_t - r_{ti}v_i = 0$, so

$$v_t = \frac{r_{ti}}{r_{it}} v_i \neq 0.$$

Thus,

$$\sum_{t=1}^{k} v_t = (1 + \sum_{t:t\neq i} \frac{r_{ti}}{r_{it}}) v_i \neq 0,$$

which contradicts (39).

The positive definiteness of $Q + \Delta \mathbf{e}\mathbf{e}^T$ implies that $\begin{bmatrix} Q+\Delta\mathbf{e}\mathbf{e}^T & \mathbf{e} \\ \mathbf{e}^T & 0 \end{bmatrix}$ is invertible. As $\begin{bmatrix} Q+\Delta\mathbf{e}\mathbf{e}^T & \mathbf{e} \\ \mathbf{e}^T & 0 \end{bmatrix}$ is from adding the last row of $\begin{bmatrix} Q & \mathbf{e} \\ \mathbf{e}^T & 0 \end{bmatrix}$ to its first k rows (with a scaling factor Δ), the two matrices have the same rank. Thus, $\begin{bmatrix} Q & \mathbf{e} \\ \mathbf{e}^T & 0 \end{bmatrix}$ is invertible as well. Then (23) has a unique solution, and so does (19).

(ii) If Q is not positive definite, there is a vector $\mathbf{v}$ with $v_i \neq 0$ such that

$$\mathbf{v}^T Q \mathbf{v} = \frac{1}{2} \sum_{t=1}^{k} \sum_{u:u\neq t} (r_{ut}v_t - r_{tu}v_u)^2 = 0.$$

Therefore,

$$(r_{ut}v_t - r_{tu}v_u)^2 = 0, \forall t \neq u.$$

As $r_{tu} > 0, \forall t \neq u$, for any $s \neq j$ for which $s \neq i$ and $j \neq i$, we have

$$v_s = \frac{r_{si}}{r_{is}} v_i, \qquad v_j = \frac{r_{ji}}{r_{ij}} v_i, \qquad v_s = \frac{r_{sj}}{r_{js}} v_j. \tag{40}$$

As $v_i \neq 0$, (40) implies

$$\frac{r_{si}r_{sj}}{r_{is}} = \frac{r_{ji}r_{js}}{r_{ij}},$$

which contradicts (25).

Appendix C. Proof of Theorem 4

First we need a lemma to show the strict decrease of the objective function:

Lemma 5 *If $r_{ij} > 0, \forall i \neq j$, $\mathbf{p}$ and $\mathbf{p}^n$ are from two consecutive iterations of Algorithm 2, and $\mathbf{p}^n \neq \mathbf{p}$, then*

$$\frac{1}{2}(\mathbf{p}^n)^T Q \mathbf{p}^n < \frac{1}{2} \mathbf{p}^T Q \mathbf{p}. \tag{41}$$

Proof. Assume that p_t is the component to be updated. Then, $\mathbf{p}^n$ is obtained through the following calculation:

$$\bar{p}_i = \begin{cases} p_i & \text{if } i \neq t, \\ \frac{1}{Q_{tt}}(-\sum_{j:j\neq t} Q_{tj}p_j + \mathbf{p}^T Q \mathbf{p}) & \text{if } i = t, \end{cases} \tag{42}$$

and

$$\mathbf{p}^n = \frac{\bar{\mathbf{p}}}{\sum_{i=1}^k \bar{p}_i}. \tag{43}$$

For (43) to be a valid operation, $\sum_{i=1}^k \bar{p}_i$ must be strictly positive. To show this, we first suppose that the current solution $\mathbf{p}$ satisfies $p_i \geq 0, i = 1, \ldots, l$, but the next solution $\bar{\mathbf{p}}$ has $\sum_{i=1}^k \bar{p}_i = 0$. In Section 4.2, we have shown that $\bar{p}_t \geq 0$, so with $\bar{p}_i = p_i \geq 0, \forall i \neq t$, $\bar{p}_i = 0$ for all i. Next, from (42), $p_i = \bar{p}_i = 0$ for $i \neq t$, which, together with the equality $\sum_{i=1}^k p_i = 1$ implies that $p_t = 1$. However, if $p_t = 1$ and $p_i = 0$ for $i \neq t$, then $\bar{p}_t = 1$ from (42). This contradicts the situation that $\bar{p}_i = 0$ for all i. Therefore, by induction, the only requirement is to have nonnegative initial $\mathbf{p}$.

To prove (41), first we rewrite the update rule (42) as

$$\begin{aligned} \bar{p}_t &= p_t + \frac{1}{Q_{tt}}(-(Q\mathbf{p})_t + \mathbf{p}^T Q\mathbf{p}) \qquad (44) \\ &= p_t + \Delta. \end{aligned}$$

Since we keep $\sum_{i=1}^k p_i = 1$, $\sum_{i=1}^k \bar{p}_i = 1 + \Delta$. Then

$$\begin{aligned} & \bar{\mathbf{p}}^T Q\bar{\mathbf{p}} - (\sum_{i=1}^k \bar{p}_i)^2 \mathbf{p}^T Q\mathbf{p} \\ = \; & \mathbf{p}^T Q\mathbf{p} + 2\Delta(Q\mathbf{p})_t + Q_{tt}\Delta^2 - (1+\Delta)^2 \mathbf{p}^T Q\mathbf{p} \\ = \; & 2\Delta(Q\mathbf{p})_t + Q_{tt}\Delta^2 - (2\Delta + \Delta^2)\mathbf{p}^T Q\mathbf{p} \\ = \; & \Delta\left(2(Q\mathbf{p})_t - 2\mathbf{p}^T Q\mathbf{p} + Q_{tt}\Delta - \Delta\mathbf{p}^T Q\mathbf{p}\right) \\ = \; & \Delta(-Q_{tt}\Delta - \Delta\mathbf{p}^T Q\mathbf{p}) \qquad (45) \\ = \; & -\Delta^2(Q_{tt} + \mathbf{p}^T Q\mathbf{p}) < 0. \qquad (46) \end{aligned}$$

(45) follows from the definition of Δ in (44). For (46), it uses $Q_{tt} = \sum_{j:j\neq t} r_{jt}^2 > 0$ and $\Delta \neq 0$, which comes from the assumption $\mathbf{p}^n \neq \mathbf{p}$. □

Now we are ready to prove the theorem. If this result does not hold, there is a convergent sub-sequence $\{\mathbf{p}^i\}_{i\in K}$ such that $\mathbf{p}^* = \lim_{i\in K, i\to\infty} \mathbf{p}^i$ is not optimal for (19). Note that there is at least one index of $\{1, \ldots, k\}$ which is considered in infinitely many iterations. Without loss of generality, we assume that for all $\mathbf{p}^i, i \in K$, p_t^i is updated to generate the next iteration $\mathbf{p}^{i+1}$. As $\mathbf{p}^*$ is not optimal for (19), starting from $t, t+1, \ldots, k, 1, \ldots, t-1$, there is a first component $\bar{t}$ for which

$$\sum_{j=1}^k Q_{\bar{t}j} p_j^* - (\mathbf{p}^*)^T Q\mathbf{p}^* \neq 0.$$

By applying one iteration of Algorithm 2 on $p_{\bar{t}}^*$, from an explanation similar to the proof of Lemma 5, we obtain $\mathbf{p}^{*,n}$ satisfying $p_{\bar{t}}^{*,n} \neq p_{\bar{t}}^*$. Then by Lemma 5,

$$\frac{1}{2}(\mathbf{p}^{*,n})^T Q\mathbf{p}^{*,n} < \frac{1}{2}(\mathbf{p}^*)^T Q\mathbf{p}^*.$$

Assume it takes $\bar{i}$ steps from t to $\bar{t}$ and $\bar{i} > 1$,

$$\begin{aligned}\lim_{i\in K, i\to\infty} p_t^{i+1} &= \lim_{i\in K, i\to\infty} \frac{\frac{1}{Q_{tt}}(-\sum_{j:j\neq t} Q_{tj}p_t^i + (\mathbf{p}^i)^T Q\mathbf{p}^i)}{\frac{1}{Q_{tt}}(-\sum_{j:j\neq t} Q_{tj}p_t^i + (\mathbf{p}^i)^T Q\mathbf{p}^i) + \sum_{j:j\neq t} p_j^i} \\ &= \frac{\frac{1}{Q_{tt}}(-\sum_{j:j\neq t} Q_{tj}p_t^* + (\mathbf{p}^*)^T Q\mathbf{p}^*)}{\frac{1}{Q_{tt}}(-\sum_{j:j\neq t} Q_{tj}p_t^* + (\mathbf{p}^*)^T Q\mathbf{p}^*) + \sum_{j:j\neq t} p_j^*} \\ &= \frac{p_t^*}{\sum_{j=1}^k p_j^*} = p_t^*,\end{aligned}$$

we have

$$\lim_{i\in K, i\to\infty} \mathbf{p}^i = \lim_{i\in K, i\to\infty} \mathbf{p}^{i+1} = \cdots = \lim_{i\in K, i\to\infty} \mathbf{p}^{i+\bar{i}-1} = \mathbf{p}^*.$$

Moreover,

$$\lim_{i\in K, i\to\infty} \mathbf{p}^{i+\bar{i}} = \mathbf{p}^{*,n}$$

and

$$\begin{aligned}\lim_{i\in K, i\to\infty} \frac{1}{2}(\mathbf{p}^{i+\bar{i}})^T Q\mathbf{p}^{i+\bar{i}} &= \frac{1}{2}(\mathbf{p}^{*,n})^T Q\mathbf{p}^{*,n} \\ &< \frac{1}{2}(\mathbf{p}^*)^T Q\mathbf{p}^* \\ &= \lim_{i\in K, i\to\infty} \frac{1}{2}(\mathbf{p}^i)^T Q\mathbf{p}^i.\end{aligned}$$

This contradicts the fact from Lemma 5:

$$\frac{1}{2}(\mathbf{p}^1)^T Q\mathbf{p}^1 \geq \frac{1}{2}(\mathbf{p}^2)^T Q\mathbf{p}^2 \geq \cdots \geq \frac{1}{2}(\mathbf{p}^*)^T Q\mathbf{p}^*.$$

Therefore, $\mathbf{p}^*$ must be optimal for (19).

Appendix D. Implementation Notes of Algorithm 2

From Algorithm 2, the main operation of each iteration is on calculating $-\sum_{j:j\neq t} Q_{tj}p_j$ and $\mathbf{p}^T Q\mathbf{p}$, both $O(k^2)$ procedures. In the following, we show how to easily reduce the cost per iteration to $O(k)$.

Following the notation in Lemma 5 of Appendix D, we consider $\mathbf{p}$ the current solution. Assume p_t is the component to be updated, we generate $\bar{\mathbf{p}}$ according to (42) and normalize $\bar{\mathbf{p}}$ to the next iterate $\mathbf{p}^n$. Note that $\bar{\mathbf{p}}$ is the same as $\mathbf{p}$ except the tth component and we consider the form (44). Since $\sum_{i=1}^k p_i = 1$, (43) is $\mathbf{p}^n = \bar{\mathbf{p}}/(1+\Delta)$. Throughout iterations, we keep the current $Q\mathbf{p}$ and $\mathbf{p}^T Q\mathbf{p}$, so Δ can be easily calculated. To obtain $Q\mathbf{p}^n$ and $(\mathbf{p}^n)^T Q\mathbf{p}^n$, we use

$$\begin{aligned}(Q\mathbf{p}^n)_j &= \frac{(Q\bar{\mathbf{p}})_j}{1+\Delta} \\ &= \frac{(Q\mathbf{p})_j + Q_{jt}\Delta}{1+\Delta}, j = 1, \ldots, k, \qquad (47)\end{aligned}$$

and

$$
\begin{aligned}
(\mathbf{p}^n)^T Q(\mathbf{p}^n) &= \frac{\bar{\mathbf{p}}^T Q \bar{\mathbf{p}}}{(1+\Delta)^2} \\
&= \frac{\mathbf{p}^T Q \mathbf{p} + 2\Delta \sum_{j=1}^k (Q\mathbf{p})_j + Q_{tt}\Delta^2}{(1+\Delta)^2}.
\end{aligned} \tag{48}
$$

Both (and hence the whole iteration) takes $O(k)$ operations.

Numerical inaccuracy may accumulate through iterations, so gradually (47) and (48) may be away from values directly calculated using $\mathbf{p}$. An easy prevention of this problem is to recalculate $Q\mathbf{p}$ and $\mathbf{p}^T Q\mathbf{p}$ directly using $\mathbf{p}$ after several iterations (e.g., k iterations). Then, the average cost per iteration is still $O(k) + O(k^2)/k = O(k)$.

Appendix E. Derivation of (32)

$$
\begin{aligned}
& \sum_{i=1}^k \sum_{j:j\neq i} (I_{\{r_{ij}>r_{ji}\}} p_j - I_{\{r_{ji}>r_{ij}\}} p_i)^2 \\
= & \sum_{i=1}^k \sum_{j:j\neq i} (I_{\{r_{ij}>r_{ji}\}} p_j^2 + I_{\{r_{ji}>r_{ij}\}} p_i^2) \\
= & \; 2\sum_{i=1}^k (\sum_{j:j\neq i} I_{\{r_{ji}>r_{ij}\}}) p_i^2.
\end{aligned}
$$

If $\sum_{j:j\neq i} I_{\{r_{ji}>r_{ij}\}} \neq 0, \forall i$, then, under the constraint $\sum_{i=1}^k p_i = 1$, the optimal solution satisfies

$$
\frac{p_1}{\sum_{j:j\neq 1} I_{\{r_{j1}>r_{1j}\}}} = \cdots = \frac{p_k}{\sum_{j:j\neq k} I_{\{r_{jk}>r_{kj}\}}}.
$$

Thus, (32) is the optimal solution of (31).

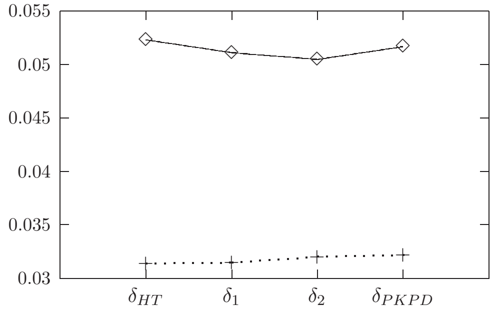
References

E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2001. ISSN 1533-7928.

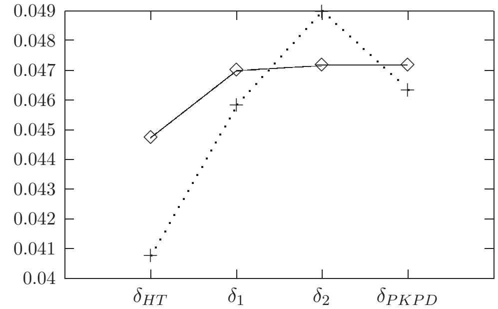
C. L. Blake and C. J. Merz. UCI repository of machine learning databases. Technical report, University of California, Department of Information and Computer Science, Irvine, CA, 1998. Available at `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

B. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 1992.

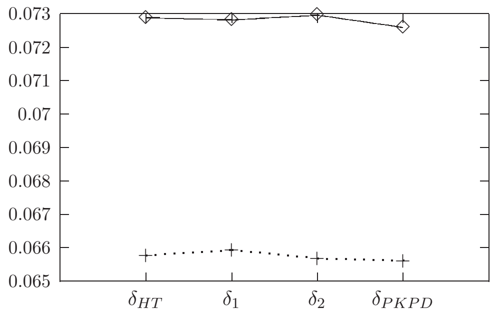
L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. URL `citeseer.nj.nec.com/breiman01random.html`.



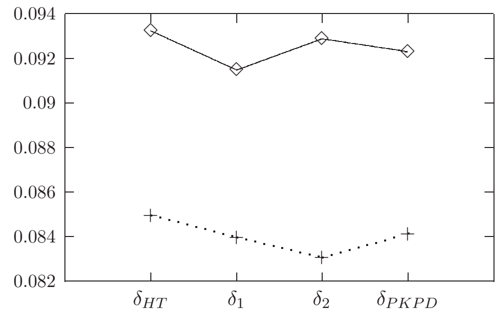
(a) dna ($k = 3$) by binary SVMs



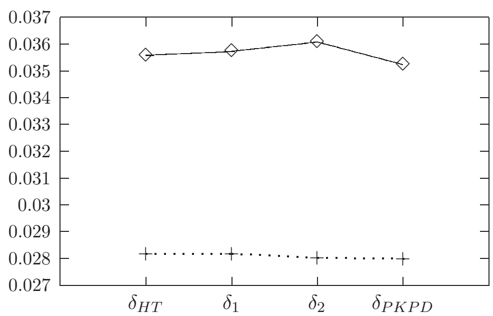
(b) dna ($k = 3$) by binary random forests



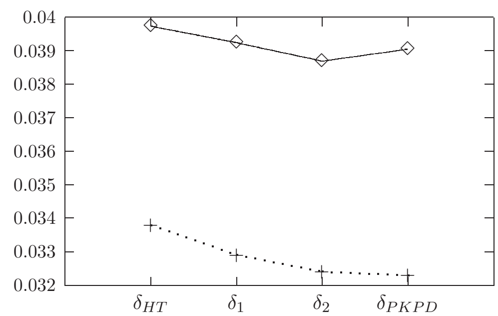
(c) waveform ($k = 3$) by binary SVMs



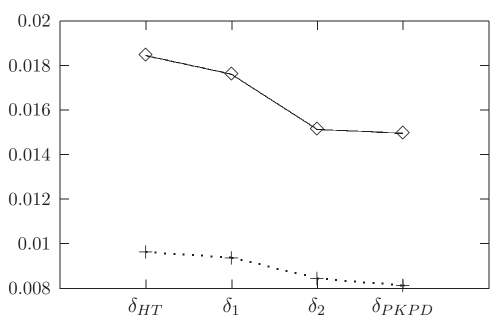
(d) waveform ($k = 3$) by binary random forests



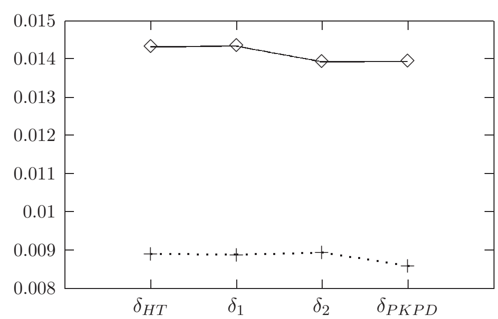
(e) satimage ($k = 6$) by binary SVMs



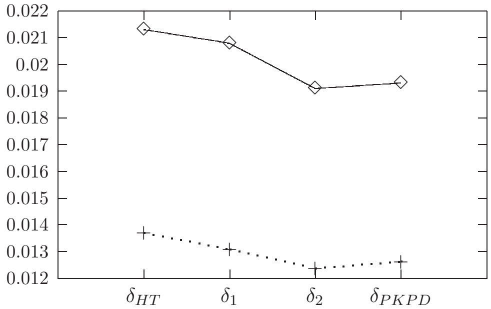
(f) satimage ($k = 6$) by binary random forests



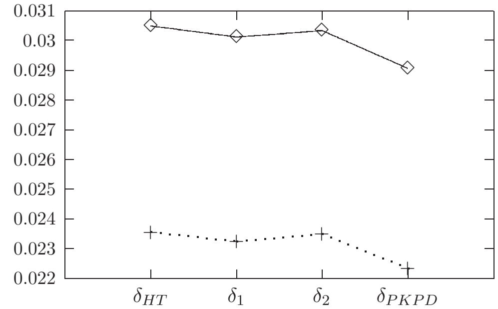
(g) segment ($k = 7$) by binary SVMs



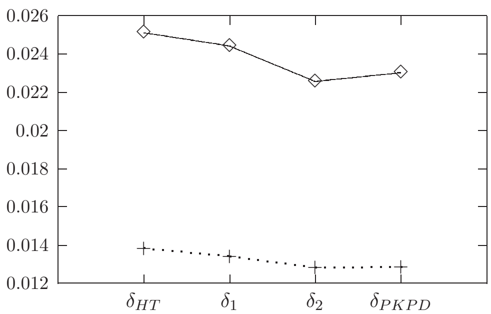
(h) segment ($k = 7$) by binary random forests



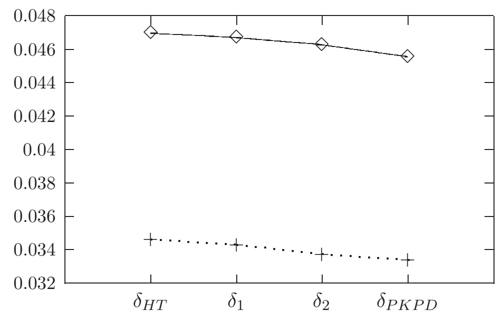
(i) USPS ($k = 10$) by binary SVMs



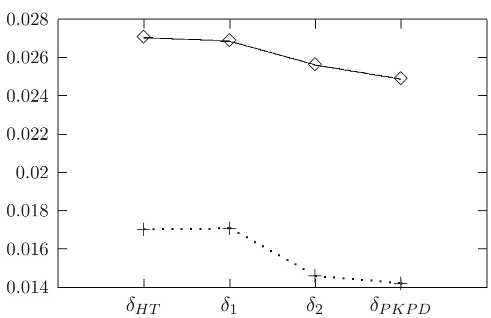
(j) USPS ($k = 10$) by binary random forests



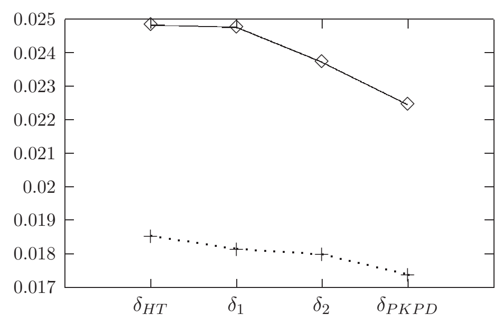
(k) MNIST ($k = 10$) by binary SVMs



(l) MNIST ($k = 10$) by binary random forests

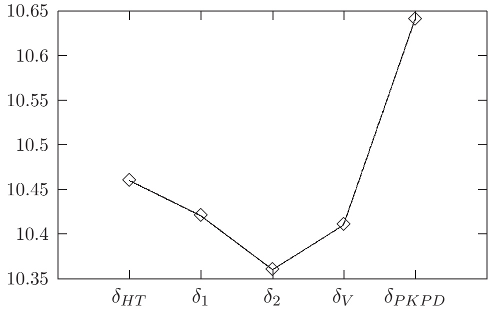


(m) letter ($k = 26$) by binary SVMs

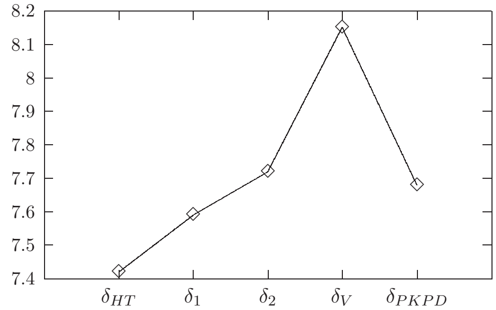


(n) letter ($k = 26$) by binary random forests

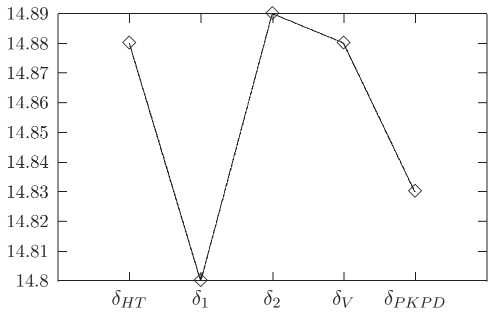
Figure 5: MSE by using four probability estimates methods based on binary SVMs (left) and binary random forests (right). MSE of δ_V is too large and is not presented. solid line: 300 training/500 testing points; dotted line: 800 training/1,000 testing points.



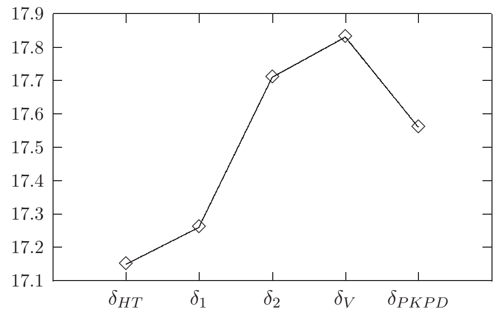
(a) dna ($k = 3$) by binary SVMs; $\delta_{DV} = 10.82\%$



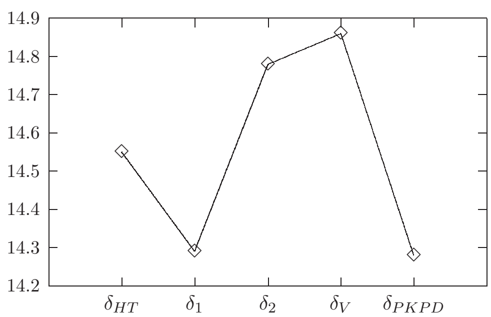
(b) dna ($k = 3$) by binary random forests; $\delta_{RF} = 8.74\%$



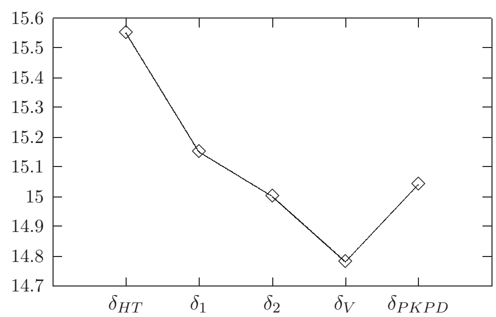
(c) waveform ($k = 3$) by binary SVMs; $\delta_{DV} = 16.47\%$



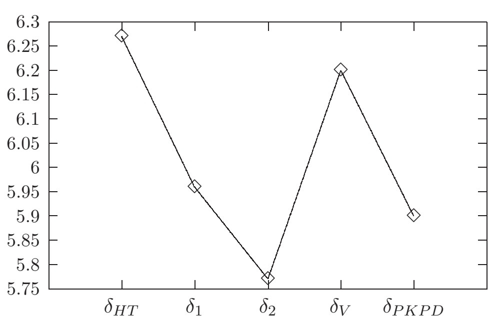
(d) waveform ($k = 3$) by binary random forests; $\delta_{RF} = 17.39\%$



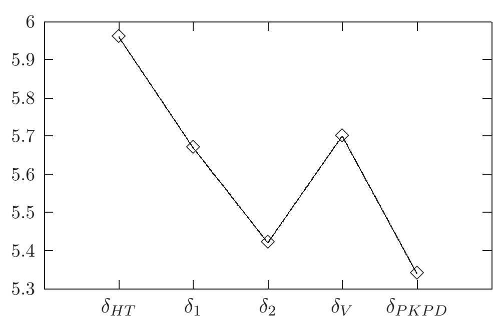
(e) satimage ($k = 6$) by binary SVMs; $\delta_{DV} = 14.88\%$



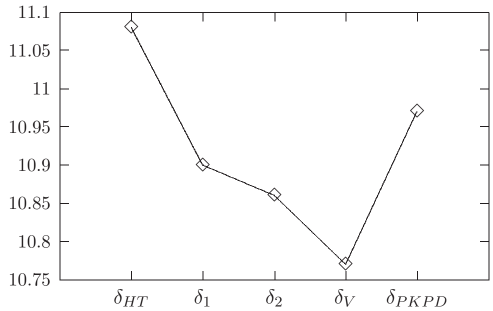
(f) satimage ($k = 6$) by binary random forests;$\delta_{RF} = 14.74\%$



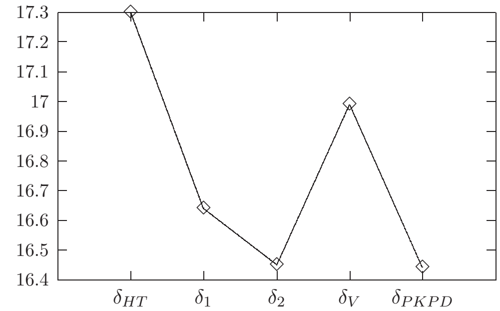
(g) segment ($k = 7$) by binary SVMs; $\delta_{DV} = 5.82\%$



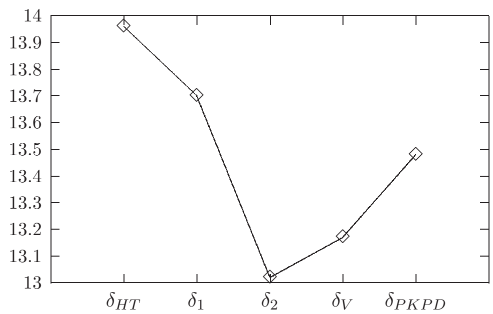
(h) segment ($k = 7$) by binary random forests; $\delta_{RF} = 5.23\%$



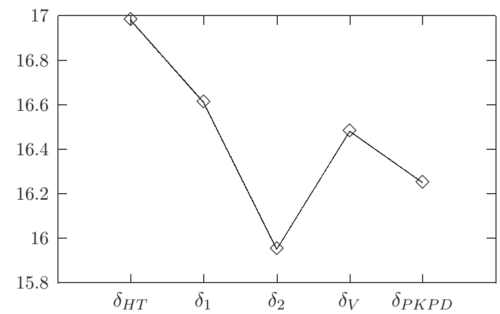
(i) USPS ($k = 10$) by binary SVMs; $\delta_{DV} = 10.71\%$



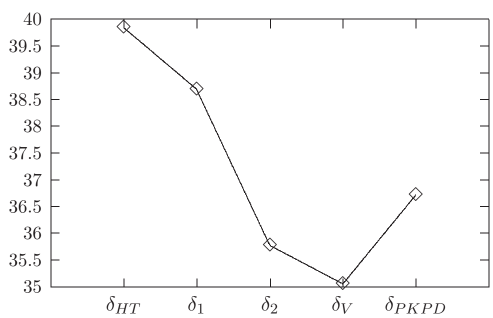
(j) USPS ($k = 10$) by binary random forests; $\delta_{RF} = 14.28\%$



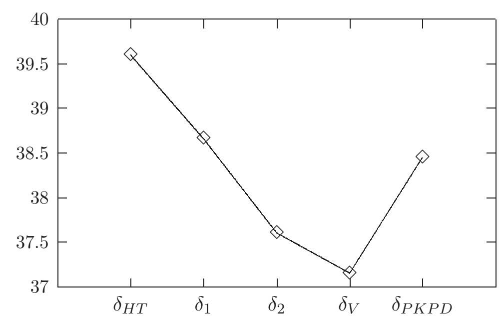
(k) MNIST ($k = 10$) by binary SVMs; $\delta_{DV} = 13.02\%$



(l) MNIST ($k = 10$) by binary random forests; $\delta_{RF} = 16.18\%$

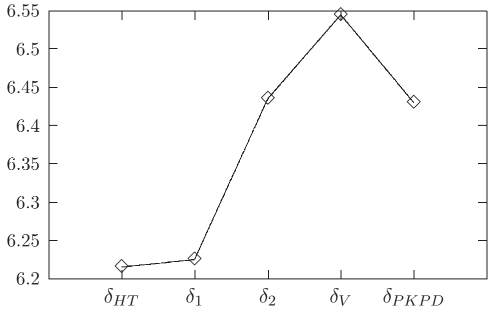


(m) letter ($k = 26$) by binary SVMs; $\delta_{DV} = 32.29\%$

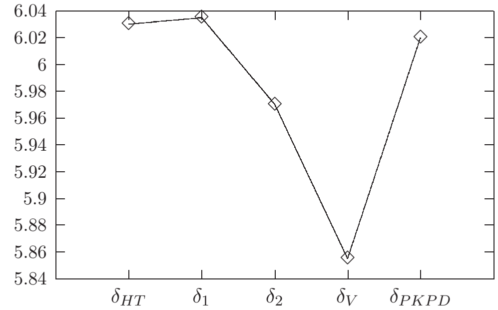


(n) letter ($k = 26$) by binary random forests; $\delta_{RF} = 32.55\%$

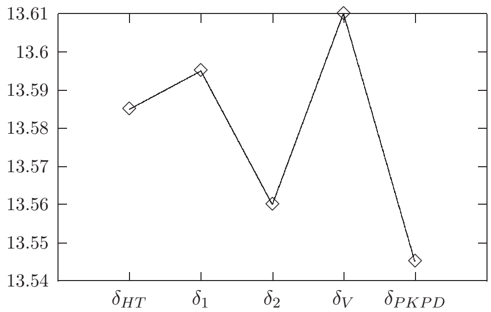
Figure 6: Average of 20 test errors by five probability estimates methods based on binary SVMs (left) and binary random forests (right). Each of the 20 test errors is by 300 training/500 testing points. Caption of each sub-figure shows the averaged error by voting using pairwise SVM decision values (δ_{DV}) and the multi-class random forest (δ_{RF}).



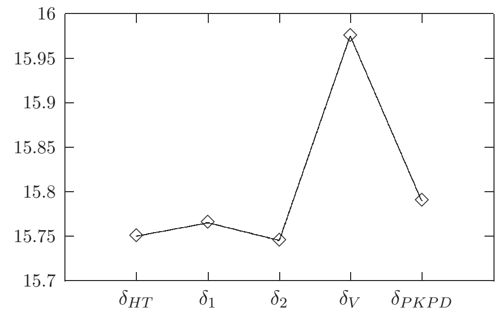
(a) dna ($k = 3$) by binary SVMs; $\delta_{DV} = 6.31\%$



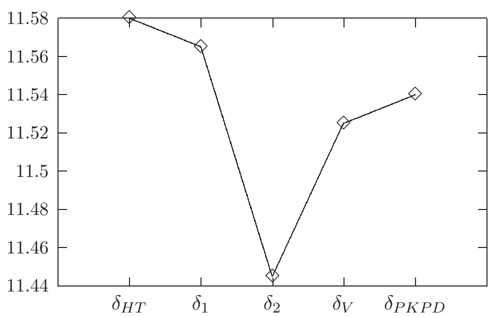
(b) dna ($k = 3$) by binary random forests; $\delta_{RF} = 6.91\%$



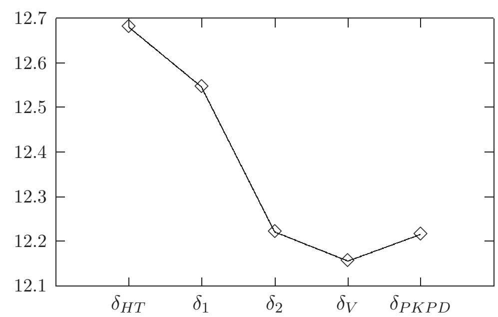
(c) waveform ($k = 3$) by binary SVMs; $\delta_{DV} = 14.33\%$



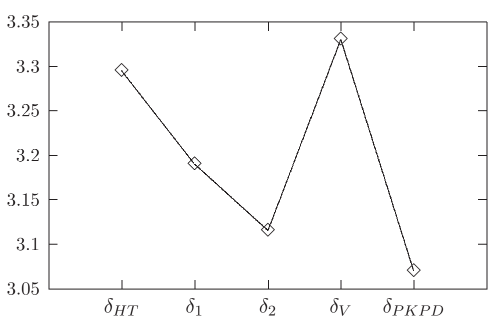
(d) waveform ($k = 3$) by binary random forests; $\delta_{RF} = 15.66\%$



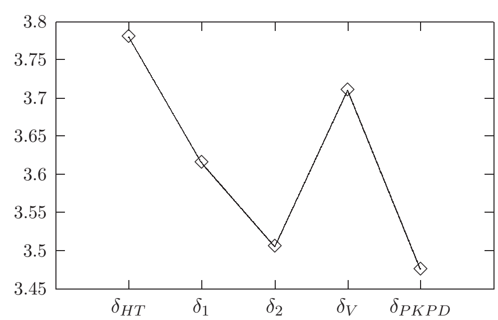
(e) satimage ($k = 6$) by binary SVMs; $\delta_{DV} = 11.54\%$



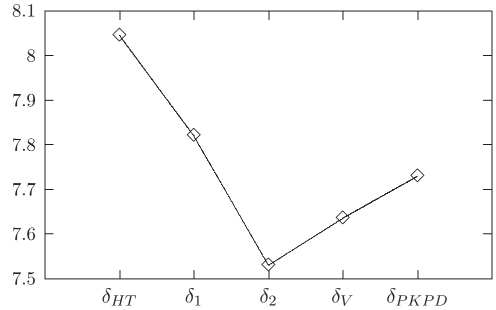
(f) satimage ($k = 6$) by binary random forests; $\delta_{RF} = 11.92\%$



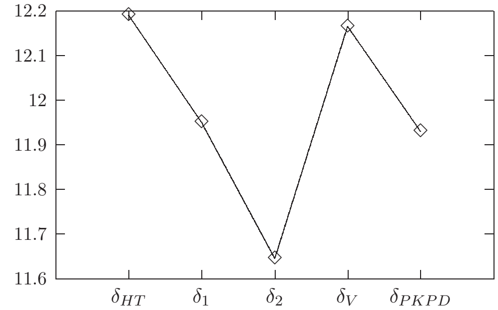
(g) segment ($k = 7$) by binary SVMs; $\delta_{DV} = 3.30\%$



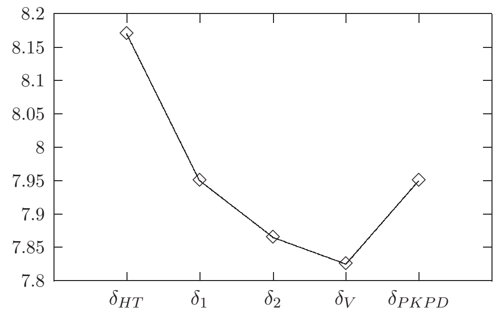
(h) segment ($k = 7$) by binary random forests; $\delta_{RF} = 2.77\%$



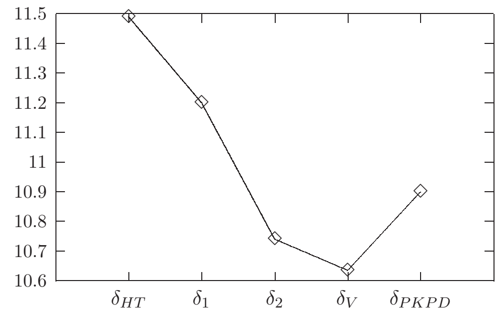
(i) USPS ($k = 10$) by binary SVMs; $\delta_{DV} = 7.54\%$



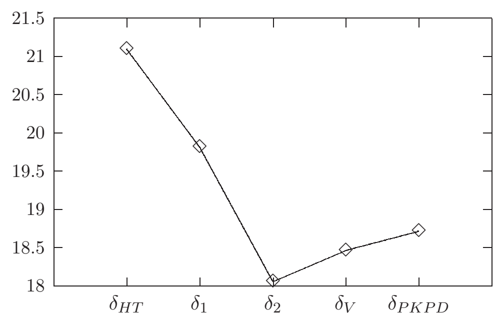
(j) USPS ($k = 10$) by binary random forests; $\delta_{RF} = 10.23\%$



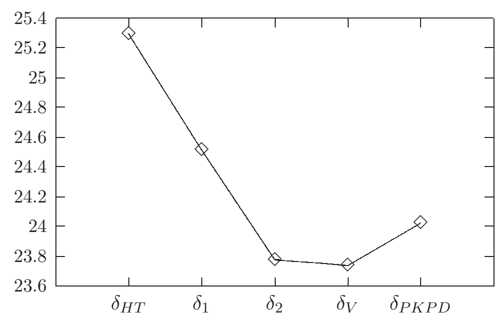
(k) MNIST ($k = 10$) by binary SVMs; $\delta_{DV} = 7.77\%$



(l) MNIST ($k = 10$) by binary random forests; $\delta_{RF} = 10.10\%$

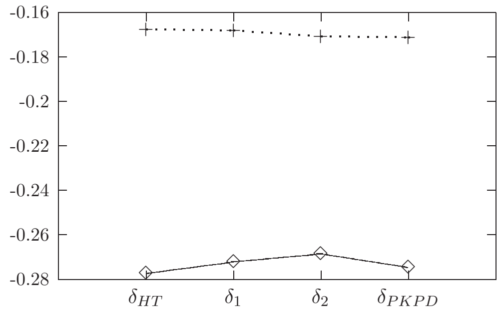


(m) letter ($k = 26$) by binary SVMs; $\delta_{DV} = 18.61\%$

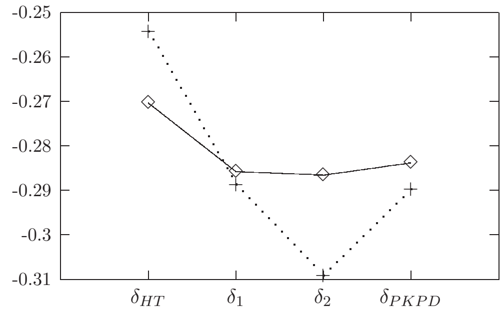


(n) letter ($k = 26$) by binary random forests; $\delta_{RF} = 20.25\%$

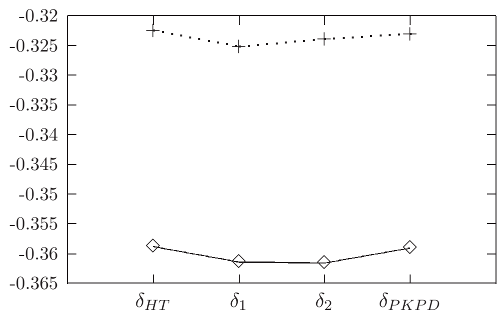
Figure 7: Average of 20 test errors by five probability estimates methods based on binary SVMs (left) and binary random forests (right). Each of the 20 test errors is by 800 training/1,000 testing points. Caption of each sub-figure shows the averaged error by voting using pairwise SVM decision values (δ_{DV}) and the multi-class random forest (δ_{RF}).



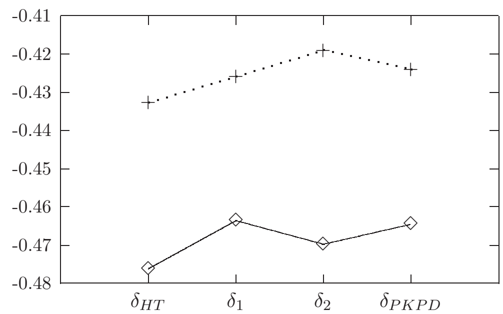
(a) dna ($k = 3$) by binary SVMs



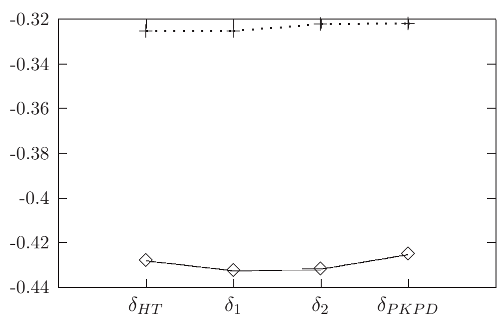
(b) dna ($k = 3$) by binary random forests



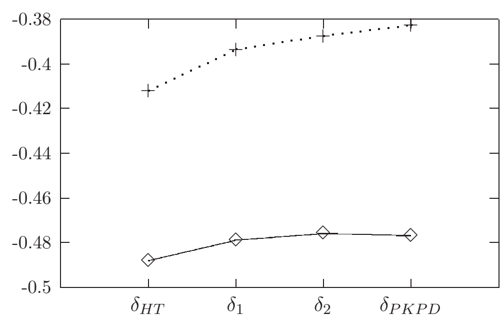
(c) waveform ($k = 3$) by binary SVMs



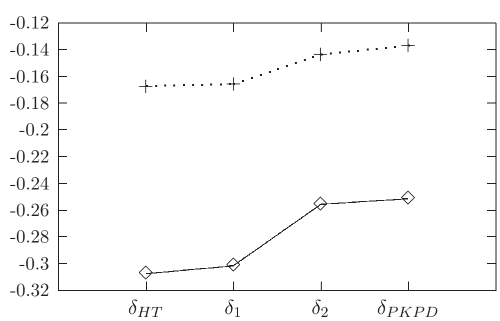
(d) waveform ($k = 3$) by binary random forests



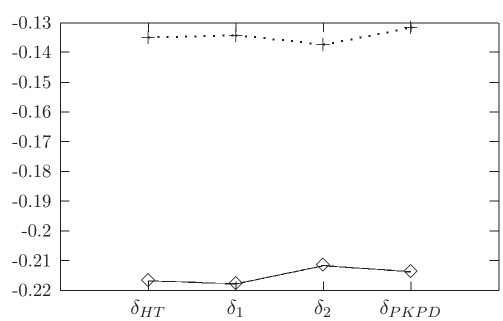
(e) satimage ($k = 6$) by binary SVMs



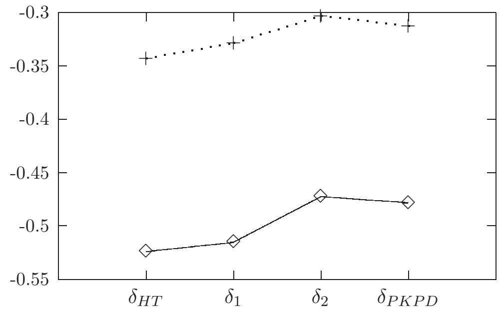
(f) satimage ($k = 6$) by binary random forests



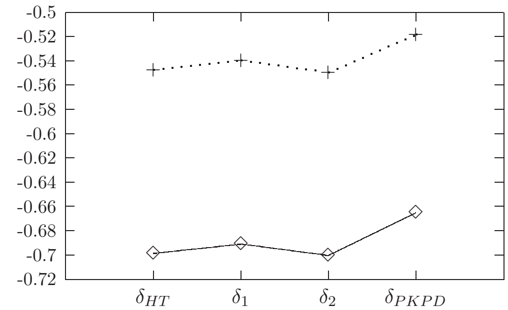
(g) segment ($k = 7$) by binary SVMs



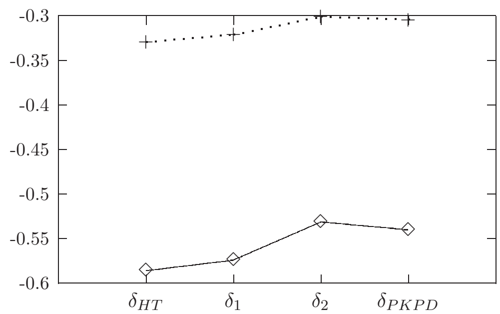
(h) segment ($k = 7$) by binary random forests



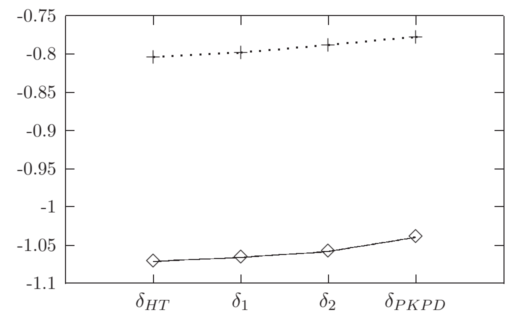
(i) USPS ($k = 10$) by binary SVMs



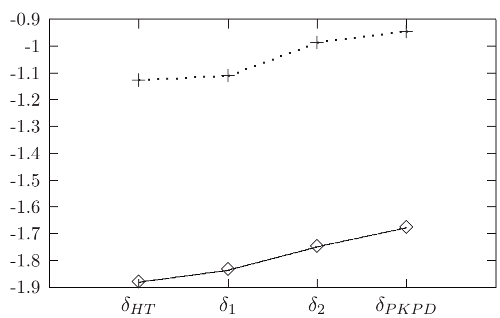
(j) USPS ($k = 10$) by binary random forests



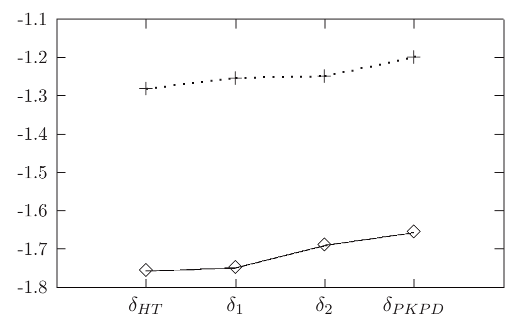
(k) MNIST ($k = 10$) by binary SVMs



(l) MNIST ($k = 10$) by binary random forests



(m) letter ($k = 26$) by binary SVMs



(n) letter ($k = 26$) by binary random forests

Figure 8: Log likelihood (37) by using four probability estimates methods based on binary SVMs (left) and binary random forests (right). MSE of δ_V is too small and is not presented. solid line: 300 training/500 testing points; dotted line: 800 training/1,000 testing points.

G. W. Brier. Verification of forecasts expressed in probabilities. *Monthly Weather Review*, 78:1–3, 1950.

C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

C. Cortes and V. Vapnik. Support-vector network. *Machine Learning*, 20:273–297, 1995.

K. Duan and S. S. Keerthi. Which is the best multiclass SVM method? An empirical study. Technical Report CD-03-12, Control Division, Department of Mechanical Engineering, National University of Singapore, 2003.

J. Friedman. Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University, 1996. Available at http://www-stat.stanford.edu/reports/friedman/poly.ps.Z.

T. Hastie and R. Tibshirani. Classification by pairwise coupling. *The Annals of Statistics*, 26(1):451–471, 1998.

J. J. Hull. A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, May 1994.

D. R. Hunter. MM algorithms for generalized Bradley-Terry models. *The Annals of Statistics*, 32:386–408, 2004.

S. Knerr, L. Personnaz, and G. Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In J. Fogelman, editor, *Neurocomputing: Algorithms, Architectures and Applications*. Springer-Verlag, 1990.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. MNIST database available at http://yann.lecun.com/exdb/mnist/.

A. Liaw and M. Wiener. Classification and regression by randomForest. *R News*, 2/3:18–22, December 2002. URL http://cran.r-project.org/doc/Rnews/Rnews_2002-3.pdf.

H.-T. Lin, C.-J. Lin, and R. C. Weng. A note on Platt's probabilistic outputs for support vector machines. Technical report, Department of Computer Science, National Taiwan University, 2003. URL http://www.csie.ntu.edu.tw/~cjlin/papers/plattprob.ps.

D. Michie, D. J. Spiegelhalter, and C. C. Taylor. *Machine Learning, Neural and Statistical Classification*. Prentice Hall, Englewood Cliffs, N.J., 1994. Data available at http://www.ncc.up.pt/liacc/ML/statlog/datasets.html.

J. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, Cambridge, MA, 2000. MIT Press. URL citeseer.nj.nec.com/platt99probabilistic.html.

D. Price, S. Knerr, L. Personnaz, and G. Dreyfus. Pairwise nerual network classifiers with probabilistic outputs. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Neural Information Processing Systems*, volume 7, pages 1109–1116. The MIT Press, 1995.

P. Refregier and F. Vallet. Probabilistic approach for multiclass classification with neural networks. In *Proceedings of International Conference on Artificial Networks*, pages 1003–1007, 1991.

S. Ross. *Stochastic Processes*. John Wiley & Sons, Inc., second edition, 1996.

V. Sventnik, A. Liaw, C. Tong, J. C. Culberson, R. P. Sheridan, and B. P. Feuston. Random forest: a tool for classification and regression in compound classification and QSAR modeling. *Journal of Chemical Information and Computer Science*, 43(6):1947–1958, 2003.

T.-F. Wu, C.-J. Lin, and R. C. Weng. Probability estimates for multi-class classification by pairwise coupling. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/svmprob.pdf`.

E. Zermelo. Die berechnung der turnier-ergebnisse als ein maximumproblem der wahrscheinlichkeitsrechnung. *Mathematische Zeitschrift*, 29:436–460, 1929.

T. Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, 32(1):56–134, 2004.