

# GEOMETRIC PROGRAMMING FOR AGGREGATION OF BINARY CLASSIFIERS

Sunho Park<sup>1</sup> and Seungjin Choi<sup>1,2</sup>

<sup>1</sup> Department of Computer Science, POSTECH, Korea

<sup>2</sup> Division of IT Convergence Engineering, POSTECH, Korea  
{titan,seungjin}@postech.ac.kr

## ABSTRACT

Multiclass classification problems are often decomposed into multiple binary problems that are solved by individual binary classifiers whose results are integrated into a final answer. We present a convex optimization-based method for aggregating results of binary classifiers in an optimal way to estimate class membership probabilities. We model the class membership probability as a softmax function whose input argument is a conic combination of discrepancies induced by individual binary classifiers. With this model, we formulate the  $\ell_1$ -regularized maximum likelihood estimation as a convex optimization that is solved by geometric programming. Numerical experiments on several UCI datasets demonstrate the high performance of our method, compared to existing methods.

**Index Terms**— Classifier aggregation, geometric programming, multiclass classification.

## 1. INTRODUCTION

Multiclass classification is an important supervised learning problem, the goal of which is to assign data points to a finite set of  $K$  classes. The most common approach to multiclass problem is to decompose it into multiple binary problems that are solved by individual binary classifiers whose results are integrated into a final answer. Reducing multiclass problems to multiple binary problems can be viewed as *encoding*, in which several methods are widely used, including all-pairs (APs), one-versus-all (OVA), and error correcting output code (ECOC) [1]. Aggregation involves combining prediction results determined by binary classifiers into a final answer to the multiclass problem. Aggregation methods can be categorized into two types: *hard decoding* and *probabilistic decoding*.

In hard decoding, one seeks a codeword which best matches a collection of labels predicted by binary classifiers, in order to determine a proper label for data  $\mathbf{x}$ . Hamming distance is often used as a discrepancy measure between a codeword and predicted labels, in the case where individual binary classifiers yield binary outputs. Various loss functions (such as exponential loss and logistic loss) are considered in [2], in the case where binary classifiers yield a score whose magnitude is a measure of confidence in the prediction. This is referred to be *loss-based decoding* [2].

In probabilistic decoding, we are given probability estimates (scores over  $[0,1]$ ) computed by binary classifiers. One couples these probability estimates to determine a set of class membership probabilities. In the case of APs, Hastie and Tibshirani [3] developed a method, *pairwise coupling*, in which pairwise class membership probability estimates are combined to form a joint probability estimates for all  $K$  classes, fitting the Bradley-Terry model [4] by minimizing a KL-divergence criterion. This was extended for arbitrary

code matrix (OVA and ECOC in addition to APs) [5], where a generalized Bradley-Terry model [6] was considered to relate probability estimates obtained by binary classifiers to class membership probability estimates. Aggregation weights were assigned to individual binary classifiers and were optimally tuned based on observed data [7]. Both aggregation weights and class membership probabilities are treated as parameters, which requires high dimensional optimization as the number of classes or the number of data points grows. Moreover, the high dimensional optimization often yields an overfitting problem, which causes the degradation of overall classification accuracy.

Most of existing methods in probabilistic decoding treat class membership probabilities as parameters, which are estimated using a model which relate them to probability estimates obtained by binary classifiers. In contrast, we directly model class membership probabilities as a softmax function whose input argument is a conic combination of discrepancies induced by binary classifiers. In this way, aggregation weights are only parameters to be tuned and have affect on a joint probability estimates for all  $K$  classes. Class membership probabilities are evaluated using the softmax function, given probability estimates computed by binary classifiers. With this model, we formulate maximum likelihood estimation with  $\ell_1$  regularization for aggregation weights, as a convex optimization that is solved by geometric programming. Numerical experiments on several UCI datasets demonstrate the high performance of our method, compared to loss-based decoding [2] and an exiting optimal aggregation method [7].

## 2. PRELIMINARIES

We are given  $N$  training examples  $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ , where  $\mathbf{x}_i$  are data vectors and  $y_i \in \mathcal{K} = \{1, \dots, K\}$  ( $K \geq 3$ ) are class labels associated with  $\mathbf{x}_i$ . Multiclass prediction involves estimating the *class membership probabilities* of  $\mathbf{x}_i$ ,

$$P_{k,i} \triangleq p(y_i = k | \mathbf{x}_i),$$

for  $k = 1, \dots, K$  such that a proper class label for  $\mathbf{x}_i$  is determined by  $\arg \max_k P_{k,i}$ . The class probability matrix  $\mathbf{P} = [\mathbf{p}_1, \dots, \mathbf{p}_N] \in \mathbb{R}^{K \times N}$  is comprised of  $P_{k,i}$  for  $k = 1, \dots, K$  and  $i = 1, \dots, N$ . Data matrix and class label vector are defined as  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N]$  and  $\mathbf{y} = [y_1, \dots, y_N]^\top$ , respectively.

Multiclass problems are decomposed into a set of binary problems that are solved by individual binary classifiers. Such decomposition can be viewed as *encoding* and various methods are widely used:

- A set of  $K$  binary functions are learned, each of which discriminates one class from the other classes, referred to as *one-versus-all* (OVA).

- A set of  $\frac{K(K-1)}{2}$  binary classifiers are learned, each of which distinguishes each pair of classes, referred to as *all-pairs* (APs).
- Error correcting output coding (ECOC) assigns a binary codeword to each class such that Hamming distances between codewords are maximized (to increase the separability) and the length of codewords determines the number of binary functions to be learned.

Aforementioned encoding methods are represented by a *coding matrix*  $\mathbf{C} = [C_{j,k}] \in \mathbb{R}^{M \times K}$  where  $M$  is the number of binary classifiers involved and  $K$  is the number of class labels. For instance, Table 1 shows the  $3 \times 3$  coding matrix for a 3-class problem in the case of APs coding. Each column in the coding matrix  $\mathbf{C}$ , denoted by  $\mathbf{c}_i$ , corresponds to *codeword*. Each row is a binary problem to be solved by a binary classifier ( $BC_i$ ). For instance,  $BC_2$  discriminates class 2 from class 3, while samples in class 1 is not used.

**Table 1.** Coding matrix in the case of APs for 3-class problem is shown, where  $BC_i$  denote binary classifiers, 1 and 0 represent positive and the negative class labels, and  $\Delta$  indicates unused class label (don't care terms).

	class 1	class 2	class 3
$BC_1$	1	0	$\Delta$
$BC_2$	$\Delta$	1	0
$BC_3$	1	$\Delta$	0

Given the coding matrix  $\mathbf{C}$ , the  $j$ th binary classifier is trained using examples  $\{(\mathbf{x}_i, C_{j,y_i})\}$ , where binary values of target are determined by the coding matrix. For instance, in the case of the 2nd binary classifier in Table 1, the binary target value for  $\mathbf{x}_i$  is  $C_{2,2} = 1$  when  $\mathbf{x}_i$  belongs to 'class 2' and is  $C_{2,3} = 0$  if  $\mathbf{x}_i$  belongs to 'class 3'.

Suppose that each binary classifier yields a probabilistic prediction, the value of which ranges between 0 and 1. For example, we can use probabilistic SVM [8]. Given  $\mathbf{x}_i$ , we define the probability estimate of the  $j$ th binary classifier as  $Q_{j,i}$ , leading to binary classifier output matrix  $\mathbf{Q} = [\mathbf{q}_1, \dots, \mathbf{q}_N] \in \mathbb{R}^{M \times N}$ . The task of multiclass problem is to estimate class membership probabilities  $\mathbf{p}_i$  using a collection of binary classifiers' probability estimates,  $\mathbf{q}_i$ .

### 3. GEOMETRIC PROGRAMMING FOR OPTIMAL AGGREGATION

In this section we present how we integrate the results of individual binary classifiers in multiclass problem. Especially we show that an optimal aggregation of binary classifiers is formulated as a convex optimization problem that is solved by geometric programming.

#### 3.1. Aggregation of Binary Classifiers

In order to predict the appropriate class label of  $\mathbf{x}_i$ , we need to evaluate which codeword  $\mathbf{c}_k$  is closest to  $\mathbf{q}_i$  (a collection of prediction results of binary classifiers, given data  $\mathbf{x}_i$ ) in the sense of a pre-specified discrepancy measure. We define this discrepancy  $\rho(\mathbf{c}_k, \mathbf{q}_i)$  as a conic combination of  $M$  discrepancies induced by binary classifiers:

$$\rho(\mathbf{c}_k, \mathbf{q}_i) = \sum_{j=1}^M w_j d(C_{j,k}, Q_{j,i}), \quad (1)$$

where

$$d(C_{j,k}, Q_{j,i}) = -C_{j,k} \log Q_{j,i} - (1 - C_{j,k}) \log(1 - Q_{j,i}), \quad (2)$$

is the *cross-entropy* error function for two classes where the model probability for membership of one class is  $Q_{j,i}$  and the corresponding true probability is  $C_{j,k}$ , while the model probability for membership of the other class is  $1 - Q_{j,i}$  and the corresponding true probability is  $1 - C_{j,k}$ . Coefficients  $w_j \geq 0$  for  $j = 1, \dots, M$  are *aggregation weights*. In the case of  $C_{j,k} = \Delta$ , we do not care what a probability estimate of the corresponding binary classifier yields, so we set  $d(\Delta, Q_{j,i}) = 0$ .

Note that the discrepancy measure is not confined to the cross-entropy only. Our method holds for any proper loss function. For instance, we can also choose the exponential loss function that was used in loss-based decoding [2]

$$d_e(C_{j,k}, Q_{j,i}) = \exp\left\{-\tilde{C}_{j,k}(Q_{j,i} - 1/2)\right\}, \quad (3)$$

where  $\tilde{C}_{j,k} = 1, -1$ , or  $0$ , when  $C_{j,k} = 1, 0$ , or  $\Delta$ , respectively.

Denote by  $\mathbf{w} \in \mathbb{R}^M$  the aggregation weight vector. Given input  $\mathbf{x}_i$  and the corresponding  $\mathbf{q}_i$  (the probability estimates of  $M$  binary classifiers), we make use of the softmax function to model the class membership probability:

$$P(y_i = k | \mathbf{w}, \mathbf{x}_i) = \frac{\exp\{-\rho(\mathbf{c}_k, \mathbf{q}_i)\}}{\sum_{j=1}^K \exp\{-\rho(\mathbf{c}_j, \mathbf{q}_i)\}}, \quad (4)$$

where  $\rho(\mathbf{c}_k, \mathbf{q}_i)$  is given in (1). Note that in the case where a class label for  $\mathbf{x}_i$  is chosen by  $\arg \max_k P(y_i = k | \mathbf{w}, \mathbf{x}_i)$  with  $w_1 = \dots = w_M = 1$  and the exponential loss (3) is chosen, our method produces the identical prediction to the loss-based decoding [2]. In other words, our method can be viewed as a probabilistic extension of the loss-based decoding, in which aggregation weights are optimally tuned using a convex optimization method.

We re-arrange the class membership probability (4) by multiplying its numerator and denominator by  $\exp\{\rho(\mathbf{c}_k, \mathbf{q}_i)\}$ :

$$\begin{aligned} P(y_i = k | \mathbf{w}, \mathbf{x}_i) &= \frac{\exp\{-\rho(\mathbf{c}_k, \mathbf{q}_i) - \rho(\mathbf{c}_k, \mathbf{q}_i)\}}{\sum_{j=1}^K \exp\{-\rho(\mathbf{c}_j, \mathbf{q}_i) - \rho(\mathbf{c}_k, \mathbf{q}_i)\}} \\ &= \frac{1}{\sum_{j=1}^K \exp\{-\mathbf{w}^\top \boldsymbol{\varphi}_i^{j,k}\}}, \end{aligned} \quad (5)$$

where the 2nd equality is derived by taking (1) into account and  $\boldsymbol{\varphi}_i^{j,k} \in \mathbb{R}^M$  are  $M$ -dimensional vectors, the  $l$ th entry of which is given by

$$[\boldsymbol{\varphi}_i^{j,k}]_l = d(C_{l,j}, Q_{l,i}) - d(C_{l,k}, Q_{l,i}). \quad (6)$$

With these relations (5) and (6), we write the likelihood as

$$p(\mathbf{y} | \mathbf{w}, \mathbf{X}) = \prod_{i=1}^N \prod_{k=1}^K \left( \frac{1}{\sum_{j=1}^K \exp\{-\mathbf{w}^\top \boldsymbol{\varphi}_i^{j,k}\}} \right)^{\delta(k, y_i)}, \quad (7)$$

where  $\delta(k, y_i)$  is Kronecker delta which equals 1 if  $y_i = k$  and otherwise 0. We estimate aggregation weights  $\mathbf{w}$  by minimizing the objective function which negative log-likelihood regularized by  $\ell_1$  norm of weight vector  $\mathbf{w}$ :

$$\begin{aligned} \mathcal{J} &= -\log p(\mathbf{y} | \mathbf{w}, \mathbf{X}) + \lambda \sum_{j=1}^M |w_j| \\ &= \sum_{i=1}^N \log \left( \sum_{j=1}^K \exp\{-\mathbf{w}^\top \boldsymbol{\varphi}_i^{j, y_i}\} \right) + \lambda^\top \mathbf{w} + \text{const} \end{aligned} \quad (8)$$

subject to the constraints  $w_j \geq 0$  for  $j = 1, \dots, M$  and  $\boldsymbol{\lambda} = [\lambda, \dots, \lambda]^\top \in \mathbb{R}^M$ . The objective function (8) and constraints  $w_j \geq 0$  for  $j = 1, \dots, M$  are convex, so the formulation becomes a convex optimization problem in which a global solution can be found. We solve this constrained convex optimization problem by converting it into an equivalent geometric programming, which is explained in Section 3.2.

### 3.2. Geometric Programming

Geometric program is an optimization problem characterized by objective function and constraint functions which have a special form involving posynomials and monomials [9]. The original geometric program is in posynomial<sup>1</sup> form but can be easily transformed to the geometric program in convex form [9, 10]. We first present a geometric program in convex form, which is equivalent to our problem (8). The corresponding geometric program in posynomial form is subsequently described, which is suited to the available software 'Mosek' [11] that was used in this paper.

We first re-write (8), leaving out the constant term, as

$$\mathcal{J} = \sum_{i=1}^N h_i(\mathbf{w}) + h_0(\mathbf{w}), \quad (9)$$

where

$$\begin{aligned} h_0(\mathbf{w}) &= \boldsymbol{\lambda}^\top \mathbf{w}, \\ h_i(\mathbf{w}) &= \log \left( \sum_{j=1}^K \exp \left\{ -\mathbf{w}^\top \boldsymbol{\varphi}_i^{j, y_i} \right\} \right), \text{ for } i = 1, \dots, N, \end{aligned}$$

with the constraints  $w_j \geq 0$  for  $j = 1, \dots, M$ . Then we introduce variables  $\mathbf{z} = [z_0, z_1, \dots, z_N]^\top$ , each of which  $z_i$  is an upper-bound on the corresponding  $h_i(\mathbf{w})$ , to write the geometric program in convex form which is equivalent to the minimization of (9):

$$\begin{aligned} &\text{minimize} && \sum_{i=0}^N z_i \\ &\text{subject to} && h_0(\mathbf{w}) - z_0 \leq 0, \\ &&& h_i(\mathbf{w}) - z_i \leq 0, \quad i = 1, \dots, N, \\ &&& w_j \geq 0, \quad j = 1, \dots, M. \end{aligned} \quad (10)$$

It is a convex problem in  $(\mathbf{w}, \mathbf{z})$  because the objective is linear and the constraint functions  $\{h_i(\mathbf{w}) - z_i\}$  are convex in  $(\mathbf{w}, \mathbf{z})$ . The equivalence between two convex problems (9) and (10) is summarized in Theorem 1, which can be proved easily by proof-by-contradiction.

**Theorem 1** *Suppose that  $(\mathbf{w}^*, \mathbf{z}^*)$  is optimal for (10) and  $\mathbf{w}^{**}$  is optimal for (9). Then  $\mathbf{w}^* = \mathbf{w}^{**}$  and  $z_i^* = h_i(\mathbf{w}^{**})$  for  $i = 0, 1, \dots, N$ .*

Then we transform the geometric program in convex form (10) to the standard form of geometric program, referred to be as the geometric program in posynomial form. To this end, we transform the objective and the constraint functions, by taking the exponential function, leading us to consider  $\exp \left\{ \sum_{i=0}^N z_i \right\}$  and  $\exp \{h_i(\mathbf{w})\}$  for  $i = 0, 1, \dots, N$ . Then, the change of variables, i.e.,  $a_j =$

<sup>1</sup>Posynomial is a sum of monomials and the exponents of a monomial can be any real numbers, whereas the exponents must be nonnegative integers in the case of polynomial.

$\exp\{w_j\}$  and  $s_i = \exp\{z_i\}$ , turns the exponential of an affine function into a monomial function

$$\begin{aligned} \prod_{l=1}^M a_l^{-[\boldsymbol{\varphi}_i^{j, y_i}]_l} &= \exp \left\{ -\mathbf{w}^\top \boldsymbol{\varphi}_i^{j, y_i} \right\}, \\ \prod_{i=0}^N s_i &= \exp \left\{ \sum_{i=0}^N z_i \right\}. \end{aligned}$$

With this change of variables and the transformation of objective and constraint functions, the geometric program in posynomial form, which corresponds to (10), is given by

$$\begin{aligned} &\text{minimize} && \prod_{i=0}^N s_i \\ &\text{subject to} && s_0^{-1} \prod_{l=1}^M a_l^\lambda \leq 1, \\ &&& s_i^{-1} \sum_{j=1}^K \left( \prod_{l=1}^M a_l^{-[\boldsymbol{\varphi}_i^{j, y_i}]_l} \right) \leq 1, \quad i = 1, \dots, N, \\ &&& a_l^{-1} \leq 1, \quad l = 1, \dots, M. \end{aligned} \quad (11)$$

Note that the transformation between two geometric programs, (10) and (11), just changes the form of the objective and constraint functions. They solve the same problems and the problem data for them are the same [9]. The optimal solution  $\mathbf{w}^*$  is determined by a change of variables,  $\mathbf{w}^* = \log \mathbf{a}^*$ , where  $\mathbf{a}^*$  is optimal for (11).

## 4. NUMERICAL EXPERIMENTS

We used 7 UCI datasets [12] to evaluate the performance of our method, compared to the loss-based decoding [2] and WMAP [7] for three different encodings (APs, OVA, ECOC). In the case of loss-based decoding, we used the exponential loss function (3). For WMAP, user parameters were manually set, choosing the values yielding the best performance after several trials were made. In our algorithm, we set  $\lambda = 0.1$ . Data descriptions are summarized in Table 2. Datasets are pre-processed such that all attributes are normalized to have unit variance, in order for attributes to reside in similar dynamic ranges. Only for 'isolet' dataset, we applied PCA to reduce the dimension down to 30.

Linear SVM was used as binary classifiers. In order to convert classifiers' scores into probabilities, we employed the sigmoid model, as used in [8], in which binary classifier's probability estimate is computed by  $Q_{j,i} = \frac{1}{1 + \exp\{-A f_j(\mathbf{x}_i) + B\}}$ , where  $f_j$  is the function learned by the  $j$ th SVM and  $A, B \in \mathbb{R}$  are parameters that are tuned, following the method in [8].

**Table 2.** Data description.

	# samples	# attributes	# classes
glass	214	9	7
satimage	6435	36	7
yeast	1484	8	10
pendigits	10992	16	10
vowel	990	10	11
isolet	7797	617	26
letter	20000	16	26

In the case of ECOC, the size of coding matrix becomes very large for  $K \geq 8$ , if the complete code is adopted. We used the complete code for  $K < 8$ , yielding  $M = 2^{K-1} - 1$  binary classifiers and generating a binary coding matrix without don't care terms ( $\Delta$ ). For  $K \geq 8$  we generated a spare random coding matrix as in [2], in which  $M = \lceil 15 \log_2 K \rceil$ , and entries of the coding matrix are chosen as  $\Delta$  with probability 1/2 and 0 or 1 with probability 1/4 for each. Table 3 summarizes the 10-fold cross-validated classification accuracy for three different methods, including loss-based decoding, WMAP, and our method. Our method shows the highest classification accuracy across most of cases. In these experiments, we used linear SVM for binary classifiers: LibSVM [13] for small-scale data sets {glass, yeast, vowel}, and Liblinear [14] for other datasets.

**Table 3.** Comparison of classification accuracy for three methods (loss-based decoding, WMAP, and our method), in which results are 10-fold cross-validated accuracy and the number in parenthesis represents the standard deviation.

		Loss [2]	WMAP [7]	Our method
glass	APs	0.662(0.104)	0.648(0.115)	<b>0.671</b> (0.096)
	OVA	0.619(0.159)	0.619(0.131)	<b>0.629</b> (0.152)
	ECOC	0.610(0.107)	0.614(0.113)	<b>0.657</b> (0.131)
sat-image	APs	0.862(0.013)	0.861(0.016)	<b>0.863</b> (0.013)
	OVA	0.770(0.014)	0.770(0.014)	<b>0.810</b> (0.010)
	ECOC	0.813(0.015)	0.812(0.015)	<b>0.852</b> (0.012)
yeast	APs	0.589(0.045)	0.588(0.047)	<b>0.595</b> (0.043)
	OVA	0.553(0.035)	0.553(0.035)	<b>0.555</b> (0.042)
	ECOC	0.520(0.052)	0.596(0.030)	<b>0.607</b> (0.021)
pen-digits	APs	0.972(0.005)	0.971(0.005)	<b>0.973</b> (0.005)
	OVA	0.848(0.007)	0.848(0.007)	<b>0.867</b> (0.006)
	ECOC	0.912(0.007)	0.912(0.005)	<b>0.943</b> (0.009)
vowel	APs	0.799(0.025)	0.783(0.035)	<b>0.805</b> (0.025)
	OVA	0.453(0.057)	0.453(0.057)	<b>0.478</b> (0.067)
	ECOC	0.526(0.048)	0.571(0.065)	<b>0.658</b> (0.041)
isolet	APs	<b>0.939</b> (0.010)	0.938(0.011)	<b>0.939</b> (0.010)
	OVA	0.780(0.009)	0.780(0.009)	<b>0.862</b> (0.011)
	ECOC	0.829(0.033)	0.853(0.015)	<b>0.875</b> (0.015)
letter	APs	0.818(0.005)	0.814(0.006)	<b>0.830</b> (0.004)
	OVA	0.534(0.014)	0.534(0.014)	<b>0.651</b> (0.009)
	ECOC	0.558(0.022)	0.567(0.028)	<b>0.611</b> (0.028)

## 5. CONCLUSIONS

We have presented a method for optimal aggregation of the binary classifiers to solve multiclass classification problems. In contrast to most of existing probabilistic decoding methods, we directly modeled the class membership probability as the softmax function whose input argument is the conic combination of discrepancies induced by binary classifiers. With this model, we formulated maximum likelihood estimation with  $\ell_1$  regularization for the aggregation weights, as the convex optimization that was solved by geometric programming. Compared to WMAP where both aggregation weights and class membership probabilities are optimized, our method is more efficient in the sense that: (1) parameters to be tuned are only the aggregation weights; (2) geometric programming formulation yields the global solution; (3) class membership probabilities for test data is easily evaluated without further optimizations. In contrast to our

recent Bayesian aggregation [15], the proposed method is based on convex aggregation, which is free from local minima problems.

**Acknowledgments:** This work was supported by NIPA Program of Software Engineering Technologies Development and Experts Education, NIPA-MSRA Creative IT/SW Research Project, and NRF WCU Program (R31-2010-000-10100-0).

## 6. REFERENCES

- [1] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes.," *Journal of Artificial Intelligence Research*, vol. 2, pp. 263–286, 1995.
- [2] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: A unifying approach for margin classifiers," *Journal of Machine Learning Research*, vol. 1, pp. 113–141, 2000.
- [3] T. Hastie and R. Tibshirani, "Classification by pairwise coupling," *The Annals of Statistics*, vol. 26, no. 2, pp. 451–471, 1998.
- [4] R. A. Bradley and M. E. Terry, "Rank analysis of incomplete block designs: The method of paired comparisons," *Biometrika*, vol. 39, pp. 324–345, 1952.
- [5] B. Zadrozny, "Reducing multiclass to binary by coupling probability estimates," in *Advances in Neural Information Processing Systems (NIPS)*. 2002, vol. 14, MIT Press.
- [6] T. K. Huang, R. C. Weng, and C. J. Lin, "Generalized Bradley-Terry models and multi-class probability estimates," *Journal of Machine Learning Research*, vol. 7, pp. 85–115, 2006.
- [7] N. Yukinawa, S. Oba, K. Kato, and S. Ishii, "Optimal aggregation of binary classifiers for multiclass cancer diagnosis using gene expression profiles," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 6, no. 2, pp. 333–343, 2009.
- [8] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *Advances in Large Margin Classifiers*, P. J. Bartlett, B. Schölkopf, D. Schuurmans, and A. J. Smola, Eds., pp. 61–74. MIT Press, 1999.
- [9] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.
- [10] S. Boyd, S. J. Kim, L. Vandenberghe, and A. Hassibi, "A tutorial on geometric programming," *Optimization and Engineering*, vol. 8, no. 1, pp. 67–127, 2007.
- [11] MOSEK ApS, *The MOSEK Optimization Tools Version 6.0, User's Manual and Reference*, 2010, Available from <http://www.mosek.com>.
- [12] A. Frank and A. Asuncion, "UCI machine learning repository," 2010.
- [13] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines," 2001.
- [14] R. E. Fan, K. W. Chang, C. J. Hsieh, X. R. Wang, and C. J. Lin, "LIBLINEAR: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.
- [15] S. Park and S. Choi, "Bayesian aggregation of binary classifiers," in *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, Sydney, Australia, 2010.