IEEE Copyright Notice

MAXIMUM DISCRIMINANT MARGIN TRANSFORM OF DISCRIMINANT FUNCTIONS

Fabian Schmieder and Bin Yang

Institute of Signal Processing and System Theory, University of Stuttgart
email: {fabian.schmieder, bin.yang}@iss.uni-stuttgart.de

ABSTRACT

Many current multiclass classification approaches can be described by a set of discriminant functions, where the label of the class with the largest discriminant function is chosen as the best prediction. We develop an affine transform called maximum discriminant margin (MDM), which can use independently estimated discriminant functions to solve multiclass classification problems.

***Index Terms*—** Multiclass classification, Discriminant functions, Maximal margin, Cutting planes

1. INTRODUCTION

In this paper we examine the problem of multiclass classification. The task is to estimate an unknown class label $\omega \in \Omega$ out of K possible classes $\Omega = \{1, \ldots, K\}$ based on an observed sample $\boldsymbol{x}$. A framework used by many current classifiers, such as neural network, support vector machine (SVM) and Bayes-plug-in classifier, is to derive K discriminant functions $f_1(\boldsymbol{x}), \ldots, f_K(\boldsymbol{x})$ using a training set containing both the measured features $\boldsymbol{x}^{(i)}$ and true class labels $y^{(i)}$. The discriminant function $f_k(\boldsymbol{x})$ is large if the sample $\boldsymbol{x}$ belongs to class i and small otherwise. This results in the decision rule

$$\hat{\omega}(\boldsymbol{x}) = \arg\max_{k \in \Omega} f_k(\boldsymbol{x}). \tag{1}$$

The common approach in machine learning is to estimate the K discriminant functions f_k jointly. For example, neural networks with K output neurons for f_k are often trained using the back-propagation algorithm, see [1]. Since the original SVM [2] only solves a binary classification problem, a number of different extensions were proposed to solve the multiclass problem, see [3–5] for a comparison. A well known approach is the one-vs-one approach. Here the K-class problem is solved by training $\frac{1}{2}K(K-1)$ binary SVMs on all possible pairs of classes, and the values of the discriminant functions f_k are given by the number of times class k is chosen among all binary classifications.

Other approaches do not estimate the K discriminant functions together, rather each discriminant function f_k is estimated on its own. For this to work, the different f_k have to share a common scale. Approaches of this group are the Bayes-plug-in classifiers. They estimate the probabilistic model for each class independently by assuming a certain likelihood like a naive Bayes model or a Gaussian mixture model (GMM). The discriminant function f_k is then determined by the likelihood and a priori probability of the class. Because these discriminant functions are all derived using the Bayes theory, they share the common probabilistic scale and are therefore still directly comparable.

However, some approaches do not have the advantage of a probabilistic scale and rely on other ideas to find a common scale. One example is the one-vs-all SVM approach, where each f_k is the discriminant function of a binary SVM which considers the samples from class k as samples of the binary class 1 and the samples from all $K-1$ remaining classes as samples of the binary class -1. The common scale in this case is based on the implicit scaling of the SVM formulation. This scaling ensures that all free support vectors $\boldsymbol{x}_{k,sv}$ of all K binary SVMs f_k have the same the discriminant value of $|f_k(\boldsymbol{x}_{k,sv})| = 1$. In practice this approach performs quite well [5].

All methods considered to this point share some kind of common scale for the discriminant functions. But in some approaches the f_k may be estimated independently without any common scale. For example, Sun and Huang [6] and Sachs et al. [7] estimated the discriminant functions f_k by training a one-class SVM for each class independently. The used one-class SVM searches for the smallest hypersphere enclosing the samples in the Hilbert space induced by the kernel function $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. This creates a discriminant function f_k which has a large value in regions with a high density of samples, see [8] and [9]. However, if the derived f_k do not share a common scale, a direct comparison according to (1) is in general difficult. Sun and Huang solved this problem by not using f_k directly. Instead they used these f_k as inputs of a neural network whose K output neurons should create a set of comparable discriminant functions $\tilde{f}_k$.

In this paper, we examine the general problem of multiclass classification based on K given discriminant functions f_k which have been estimated independently without a common scale.

2. DISCRIMINANT TRANSFORM

2.1. Basic idea

If the discriminant functions are estimated independently, it may not be optimal to estimate the class using (1) directly. We propose to transform each discriminant function prior to the decision using a discriminant transform $g_k(\boldsymbol{x}) = h_k(f_k(\boldsymbol{x}))$. Finding general optimal transforms h_k is a difficult problem.

However, a solution can be found by only considering the set of affine transforms

$$g_k(\boldsymbol{x}) = a_k f_k(\boldsymbol{x}) + b_k, \quad k = 1, \dots, K \tag{2}$$

with $a_k \geq 0$. The new decision rule is then

$$\hat{\omega}(\boldsymbol{x}) = \arg\max_{k \in \Omega} g_k(\boldsymbol{x}). \tag{3}$$

These transforms can compensate for different scales and offsets in the original discriminant functions f_k. A training set $\mathcal{S}_x = \{(\boldsymbol{x}^{(1)}, y^{(1)}), \dots, (\boldsymbol{x}^{(l)}, y^{(l)})\}$ is used to estimate the unknown parameters a_k and b_k. For a simpler notation, the new set $\mathcal{S}_f = \{(\boldsymbol{f}^{(1)}, y^{(1)}), \dots, (\boldsymbol{f}^{(l)}, y^{(l)})\} \in \left(\mathbb{R}^K \times \Omega\right)^l$ containing the discriminant values $\boldsymbol{f}^{(i)} = [f_1(\boldsymbol{x}^{(i)}), \dots, f_K(\boldsymbol{x}^{(i)})]^T$ is used.

Finding an optimal choice of a_k and b_k can be achieved by considering the error rate of the training set. To correctly classify a sample $\boldsymbol{f}^{(i)}$ of class k, the corresponding discriminant function g_k has to be the largest, i.e

$$a_k f_k^{(i)} + b_k \geq a_q f_q^{(i)} + b_q, \quad q \neq k. \tag{4}$$

2.2. Maximum discriminant margin (MDM)

Under the assumption of a separable training set $\mathcal{S}_f$, a good choice for the unknown parameters a_k and b_k should maximize the difference between the correct discriminant value g_k and the next largest discriminant value. A similar idea was explored by Crammer and Singer in [10] where they also maximized the discriminant difference to estimate a continuous error code matrix combining multiple binary SVMs with different class assignments to solve the multiclass problem. Following the above idea the problem can be stated as a linear program (LP)

$$\max_{\boldsymbol{a}, \boldsymbol{b}, \gamma} \gamma \quad \text{s.t. } \boldsymbol{F}^{(i)} \boldsymbol{a} + \boldsymbol{S}^{(i)} \boldsymbol{b} \geq \boldsymbol{1}\, \gamma, \quad i = 1, \dots, l \tag{5}$$

where $\boldsymbol{a} = [a_1, \dots, a_K]^T$, $\boldsymbol{b} = [b_1, \dots, b_K]^T$ and $\boldsymbol{F}^{(i)}$, $\boldsymbol{S}^{(i)}$ are matrices of dimension $(K-1) \times K$ defined in the appendix A. $\gamma > 0$ is the discriminant margin. However, the problem in (5) is unbounded as a scaling of γ with a positive number is always possible by a scaling of $\boldsymbol{a}$ and $\boldsymbol{b}$. Therefore, further constraints are required to force a unique solution for $\boldsymbol{a}$ and $\boldsymbol{b}$. A natural choice is the range r of the transformed discriminant functions g_k, i.e.

$$\begin{aligned} r &= \max_{i,k} g_k(\boldsymbol{x}^{(i)}) - \min_{i,k} g_k(\boldsymbol{x}^{(i)}) \\ &= \max_{i,k} \left(a_k f_k^{(i)} + b_k\right) - \min_{i,k} \left(a_k f_k^{(i)} + b_k\right). \end{aligned} \tag{6}$$

As the affine transforms h_k in (2) are monotonically increasing, the maximum and minimum over the samples can be precomputed by $f_{k,\max} = \max_i f_k^{(i)}$ and $f_{k,\min} = \min_i f_k^{(i)}$. Then the range is

$$\begin{aligned} r &= \max_k \left(a_k f_{k,\max} + b_k\right) - \min_q \left(a_q f_{q,\min} + b_q\right) \\ &= \max_{k,q} \left(a_k f_{k,\max} + b_k - a_q f_{q,\min} - b_q\right) \\ &= \max \left(\boldsymbol{F}_r \boldsymbol{a} + \boldsymbol{S}_r \boldsymbol{b}\right), \end{aligned}$$

see appendix A for the definition of the $K^2 \times K$ matrices $\boldsymbol{F}_r$ and $\boldsymbol{S}_r$.

Instead of maximizing the margin γ, we now try to minimize the range r while enforcing a constant margin $\gamma = 1$. To incorporate possible training errors for non-separable data, we use the same approach as the soft-margin SVM. We introduce a slack variable $\xi_i \geq 0$ for each sample $\boldsymbol{f}^{(i)}$. $\xi_i > 0$ means that sample $\boldsymbol{f}^{(i)}$ does not fulfill the original hard margin inequality with margin 1. The resulting LP is now given by

$$\min_{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{\xi}, r} \quad r + \frac{C}{l} \boldsymbol{1}^T \boldsymbol{\xi} \tag{7a}$$

$$\text{s.t.} \quad \boldsymbol{F}^{(i)} \boldsymbol{a} + \boldsymbol{S}^{(i)} \boldsymbol{b} \geq \boldsymbol{1}\,(1 - \xi_i), \quad i = 1, \dots, l, \tag{7b}$$

$$\boldsymbol{F}_r \boldsymbol{a} + \boldsymbol{S}_r \boldsymbol{b} \leq \boldsymbol{1} r, \tag{7c}$$

$$\boldsymbol{a} \geq \boldsymbol{0}, \tag{7d}$$

$$\boldsymbol{\xi} \geq \boldsymbol{0} \tag{7e}$$

with $\boldsymbol{\xi} = [\xi_1, \dots, \xi_l]^T$. The parameter C controls the trade-off between minimizing the range r and the average value of the slack variables $\frac{1}{l} \boldsymbol{1}^T \boldsymbol{\xi}$. Note that $\frac{1}{l} \boldsymbol{1}^T \boldsymbol{\xi}$ also defines a simple upper bound on the training error rate. The problem (7) is bounded, but the solution for $\boldsymbol{b}$ is still not unique, as only the differences $b_k - b_q$ appear in the problem. Therefore, a constant offset $\tilde{\boldsymbol{b}} = \boldsymbol{b} + c\,\boldsymbol{1}$ yields no change. Below $b_1 = 0$ is assumed to enforce a unique solution.

3. ALGORITHM

The problem in (7) could be solved directly by many current LP solvers. However, as the number of slack variables and the number of constraints in (7b) and (7e) grow linearly with the number of training samples l, an alternative formulation which scales better for large datasets may be more appropriate. Our algorithm to solve (7) is an adaption of the cutting plane algorithm for solving a linear SVM proposed by Joachims [11]. To apply the algorithm, we first move the constraints (7b) and (7e) into a new risk function

$$R(\boldsymbol{a}, \boldsymbol{b}) = \frac{1}{l} \sum_{i=1}^{l} \xi_i = \frac{1}{l} \sum_{i=1}^{l} \max\left\{0, \max_q v_q^{(i)}\right\} \tag{8}$$

with $\boldsymbol{v}^{(i)} = \boldsymbol{1} - \boldsymbol{F}^{(i)} \boldsymbol{a} - \boldsymbol{S}^{(i)} \boldsymbol{b} = [v_1^{(i)}, \dots, v_{K-1}^{(i)}]^T$. The risk $R(\boldsymbol{a}, \boldsymbol{b})$ is also called the hinge loss for the original SVM problem. Then (7) can be reformulated to

$$\begin{aligned} \min_{\boldsymbol{a}, \boldsymbol{b}, r} \; & r + C\, R(\boldsymbol{a}, \boldsymbol{b}) \\ \text{s.t. } & \boldsymbol{F}_r \boldsymbol{a} + \boldsymbol{S}_r \boldsymbol{b} \leq \boldsymbol{1} r, \\ & \boldsymbol{a} \geq \boldsymbol{0}. \end{aligned} \tag{9}$$

This problem formulation only depends on the number of classes K and is independent of the number of samples l. However, the risk $R(\boldsymbol{a}, \boldsymbol{b})$ is nonlinear and not continuously differentiable. This makes a direct optimization of (9) difficult. Instead, we solve a series of problems which lower bound the original risk $R(\boldsymbol{a}, \boldsymbol{b}) \geq \hat{R}_t(\boldsymbol{a}, \boldsymbol{b})$ by using a set of cutting planes, i.e

$$\hat{R}_t(\boldsymbol{a}, \boldsymbol{b}) = \max \left\{ 0, \max_{i=1,\dots,t} \left(\boldsymbol{p}_i^T \boldsymbol{a} + \boldsymbol{q}_i^T \boldsymbol{b} + r_i \right) \right\}, \quad (10)$$

where $\boldsymbol{p}_i^T \boldsymbol{a} + \boldsymbol{q}_i^T \boldsymbol{b} + r_i$ are tangent hyperplanes of $R(\boldsymbol{a}, \boldsymbol{b})$ given by

$$\boldsymbol{p} = \boldsymbol{\nabla_a} R(\boldsymbol{a}, \boldsymbol{b}) = -\frac{1}{l} \sum_{i=1}^{l} \boldsymbol{F}^{(i)T} \boldsymbol{c}^{(i)}, \quad (11)$$

$$\boldsymbol{q} = \boldsymbol{\nabla_b} R(\boldsymbol{a}, \boldsymbol{b}) = -\frac{1}{l} \sum_{i=1}^{l} \boldsymbol{S}^{(i)T} \boldsymbol{c}^{(i)}, \quad (12)$$

$$r = R(\boldsymbol{a}, \boldsymbol{b}) - \boldsymbol{p}^T \boldsymbol{a} - \boldsymbol{q}^T \boldsymbol{b} \quad (13)$$

with $\boldsymbol{c}^{(i)} = [c_1^{(i)}, \dots, c_{K-1}^{(i)}]^T$ and

$$c_q^{(i)} = \begin{cases} 1 & \text{if} \quad v_q^{(i)} = \max \left\{ 0, \max\limits_{q'} v_{q'}^{(i)} \right\} \\ 0 & \text{if} \quad v_q^{(i)} < \max \left\{ 0, \max\limits_{q'} v_{q'}^{(i)} \right\} \end{cases}. \quad (14)$$

The LP derived from this lower bound is given by

$$\begin{aligned} \min_{\boldsymbol{a},\boldsymbol{b},\xi_{cp},r} \quad & r + C\xi_{cp} & (15) \\ \text{s.t.} \quad & \boldsymbol{P}_t \boldsymbol{a} + \boldsymbol{Q}_t \boldsymbol{b} + \boldsymbol{r}_t \leq \boldsymbol{1}\xi_{cp}, \\ & \boldsymbol{F}_r \boldsymbol{a} + \boldsymbol{S}_r \boldsymbol{b} \leq \boldsymbol{1} r, \\ & \boldsymbol{a} \geq \boldsymbol{0}, \\ & \xi_{cp} \geq 0, \end{aligned}$$

where $\boldsymbol{P}_t = [\boldsymbol{p}_1, \dots, \boldsymbol{p}_t]^T$, $\boldsymbol{Q}_t = [\boldsymbol{q}_1, \dots, \boldsymbol{q}_t]^T$, $\boldsymbol{r}_t = [r_1, \dots, r_t]^T$ and $\xi_{cp} = \hat{R}_t$. At each iteration t of the algorithm, we first solve (15) using the GLPK solver [12] and then add a new tangent hyperplane at the current solution $\hat{\boldsymbol{a}}_t, \hat{\boldsymbol{b}}_t$ to the previous set of cutting planes. This creates a series of monotonically increasing lower bounds $\hat{R}_t(\boldsymbol{a}, \boldsymbol{b})$ on the original risk $R(\boldsymbol{a}, \boldsymbol{b})$. The algorithm terminates when the lower bound $\hat{R}_t(\boldsymbol{a}, \boldsymbol{b})$ is sufficiently close to $R(\boldsymbol{a}, \boldsymbol{b})$, see Algorithm 1. Sometimes the proposed algorithm may increase the error rate on the given dataset, as it only minimizes an upper bound. In this case we use the original discriminant functions with $\hat{\boldsymbol{a}} = \boldsymbol{1}$ and $\hat{\boldsymbol{b}} = \boldsymbol{0}$.

Algorithm 1 Cutting Plane Algorithm

```
$\hat{\boldsymbol{a}}_0 = \boldsymbol{1}$, $\hat{\boldsymbol{b}}_0 = \boldsymbol{0}$, $t = 0$.
repeat
    $t \leftarrow t + 1$
    Compute a new cutting plane at $\hat{\boldsymbol{a}}_{t-1}$, $\hat{\boldsymbol{b}}_{t-1}$ with
        $\boldsymbol{p}_t = -\sum_{i=1}^{l} \boldsymbol{F}^{(i)T} \boldsymbol{c}^{(i)}$,
        $\boldsymbol{q}_t = -\sum_{i=1}^{l} \boldsymbol{S}^{(i)T} \boldsymbol{c}^{(i)}$,
        $\boldsymbol{r}_t = R(\hat{\boldsymbol{a}}_{t-1}, \hat{\boldsymbol{b}}_{t-1}) - \boldsymbol{p}_t^T \hat{\boldsymbol{a}}_{t-1} - \boldsymbol{q}_t^T \hat{\boldsymbol{b}}_{t-1}$.
    Find $\hat{\boldsymbol{a}}_t$, $\hat{\boldsymbol{b}}_t$ by solving (15).
until $R(\hat{\boldsymbol{a}}_t, \hat{\boldsymbol{b}}_t) - \hat{R}_t(\hat{\boldsymbol{a}}_t, \hat{\boldsymbol{b}}_t) < \epsilon \hat{R}_t(\hat{\boldsymbol{a}}_t, \hat{\boldsymbol{b}}_t)$
```

4. SIMULATION RESULTS

In the first experiment, we test our MDM transform with 20 setups of simulated data. In all setups, $K = 8$ classes with 30 dimensional Gaussian likelihoods and equal class probabilities are used. The mean and covariance matrix for the class likelihoods are randomly drawn for each of the 20 setups. To simulate label errors or outliers, the correct class labels are randomly changed with a probability of 2%. In this case, the optimum maximum a'posteriori discriminant functions are known and given by the log-likelihoods $\log p_k(\boldsymbol{x})$ of the Gaussians. They can be precomputed and are used to test the MDM. In order to simulate independently estimated discriminant functions f_k, we disturb each optimum discriminant function by a random scaling and offset, i.e $f_k(\boldsymbol{x}) = \tilde{a}_k \log p_k(\boldsymbol{x}) + \tilde{b}_k$. $\tilde{a}_k$ is drawn from a uniform distribution in $[1, 9]$ and $\tilde{b}_k \sim \mathrm{N}(0, 100)$.

For each of the 20 setups, a total number of 50 independent training sets, each with $l = 200$ samples, are generated. For each training set, the MDM algorithm with $C = 10^5$ and $\epsilon = 10^{-4}$ estimates the optimal transform coefficients a_k and b_k. Finally, the generalization error rate for all 50 solutions is computed using 10^4 previously unseen test samples. Table 1 compares the Bayes error rate by using the optimum discriminant functions $\log p_k(\boldsymbol{x})$, the raw error rate by using the randomly modified discriminant functions f_k and the error rate by using the corrected discriminant functions g_k over the 50 training sets for 5 exemplary data setups of all 20 data setups. The last row shows the mean difference between the

		MDM Percentiles		
Bayes	Raw	25th	50th	75th
2.08%	67.65%	2.33%	2.56%	3.21%
2.12%	36.15%	2.31%	2.38%	2.66%
9.24%	86.67%	10.19%	10.52%	10.79%
10.93%	87.17%	11.82%	12.10%	12.49%
11.96%	86.84%	13.17%	13.50%	13.87%
mean difference		0.67%	0.97%	1.58%

Table 1. Bayes error rate, raw error rate and error rate percentiles after MDM over 50 training sets for 5 exemplary setups.

Bayes error rate and the error rate after MDM over all 20 data setups for the three given percentiles. Clearly, MDM is able to compensate for individual scalings and offsets of independently estimated discriminant functions and achieves almost the optimum Bayes error rate.

The second experiment applies the MDM transform on the datasets listed in Table 2. The *dna* and *letter* sets are from the UCI [13], the *satimage* set from the StatLog Project and for the *usps* dataset, see [14]. For all datasets the discriminant functions f_k are estimated by two different approaches. The first approach estimates f_k using the one-vs-all SVM and the second approach uses a one-class SVM [8] for each class independently. Both SVM approaches use the radial basis function (rbf) kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\gamma \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2\right)$ and 75% of the training data to estimate the discriminant functions f_k. The MDM uses the remaining 25% of the training data to estimate the optimal transforms. The hyper-parameters for the one-vs-all and one-class SVM are estimated by a grid search minimizing the error rate on the 25% of the training data used by MDM. The resulting generalization error rates on the test set with and without MDM are shown in Table 3.

	#classes K	#training	#test	#features
dna	3	2000	1186	180
letter	26	15000	5000	16
satimage	6	4435	2000	36
usps	10	7291	2007	256

Table 2. Datasets used in the second experiment with the number of classes, training and test samples, and features.

	one-vs-all SVM		one-class SVM	
	Raw	MDM	Raw	MDM
dna	4.97%	4.97%	30.69%	7.08%
letter	2.60%	2.60%	5.48%	3.50%
satimage	8.15%	8.15%	13.35%	8.95%
usps	4.40%	4.40%	9.12%	4.90%

Table 3. Generalization error for all datasets without and with using the MDM.

In the case of one-class SVM, MDM considerably reduces the error rate for all datasets. The missing common scale for the different one-class discriminant functions is successfully restored by MDM. For one-vs-all SVM, MDM does not improve the classification performance. One reason is that each one-vs-all SVM uses samples of all classes. Therefore the binary SVMs are not truly independently estimated and can use the samples of the other classes to find an implicit common scale. Also Rifkin showed in [5] that one-vs-all SVM performs well for many different problems. However, a closer look at the error rates for different kernels shows that MDM is able to improve the performance of the one-vs-all SVM for narrower rbf (larger γ). This is shown in Figure 1 for the *usps* dataset. The other datasets show a similar behavior. One reason is that for larger γ SVM tends to overfitting and the number of free support vectors grows. This also means that the scaling based on these support vector is less reliable. MDM is able to compensate for this overfitting effect, as it uses samples previously unseen by the SVM training to estimate the transform. In other words, MDM makes one-vs-all SVM more robust against a bad choice of γ.

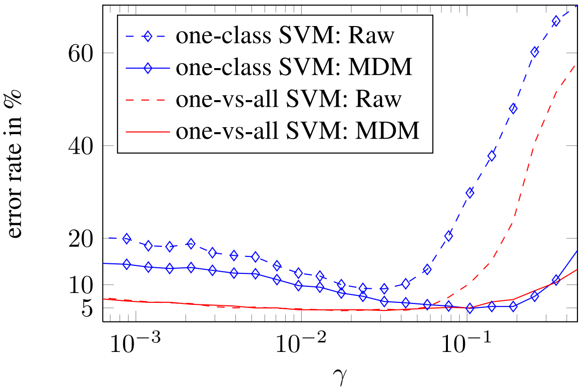

Fig. 1. Generalization error rates for different γ of the rbf kernel for the usps dataset.

5. CONCLUSION

We introduced a new approach called MDM to transform a set of discriminant functions. MDM can use independently estimated discriminant functions to solve a multiclass classification problem by using a small set of labeled examples to estimate an affine transform. The results show that MDM can find a solution close to the Bayes error rate even for a small number of labeled examples. Using this approach the one-class classification approach could be improved for all considered datasets by only increasing the classification complexity by an additional multiplication and addition.

A. APPENDIX

$\boldsymbol{F}^{(i)}$ and $\boldsymbol{S}^{(i)}$ are created by removing the $y^{(i)}$-th row of the $K \times K$ matrices $\tilde{\boldsymbol{F}}^{(i)}$ and $\tilde{\boldsymbol{S}}^{(i)}$ given by

$$\begin{aligned} \tilde{\boldsymbol{F}}^{(i)} &= f^{(i)}_{y^{(i)}} \, \mathbf{1} \, \boldsymbol{e}^T_{y^{(i)}} - \operatorname{diag}\left(\boldsymbol{f}^{(i)}\right), \\ \tilde{\boldsymbol{S}}^{(i)} &= \qquad \mathbf{1} \, \boldsymbol{e}^T_{y^{(i)}} - \boldsymbol{I} \end{aligned}$$

where $\boldsymbol{I}$ is the $K \times K$ identity matrix, $\boldsymbol{e}_i$ is the i-th unit column vector and diag $(.)$ creates a diagonal matrix using the given vector as the diagonal elements.

The range matrices $\boldsymbol{F}_r$ and $\boldsymbol{S}_r$ are created by using all K^2 combinations of $f_{k,\text{max}}$ and $f_{q,\text{min}}$ with $k, q = 1, \ldots, K$. Using the Kronecker tensor product $\otimes$, they are given by

$$\begin{aligned} \boldsymbol{F}_r &= \operatorname{diag}\left(\boldsymbol{f}_{\text{max}}\right) \otimes \mathbf{1} - \mathbf{1} \otimes \operatorname{diag}\left(\boldsymbol{f}_{\text{min}}\right), \\ \boldsymbol{S}_r &= \boldsymbol{I} \otimes \mathbf{1} - \mathbf{1} \otimes \boldsymbol{I} \end{aligned}$$

with $\boldsymbol{f}_{\text{min}} = [f_{1,\text{min}}, \ldots, f_{K,\text{min}}]^T$ and $\boldsymbol{f}_{\text{max}} = [f_{1,\text{max}}, \ldots, f_{K,\text{max}}]^T$.

B. REFERENCES

[1] Raul Rojas, *Neural networks: A systematic introduction*, Springer, 1. edition, July 1996.

[2] Vladimir N. Vapnik, *Statistical learning theory*, John Wiley and Sons, New York, 1. edition, Sept. 1998.

[3] Kai bo Duan and S. Sathiya Keerthi, "Which is the best multiclass SVM method? An empirical study," in *Proceedings of the Sixth International Workshop on Multiple Classifier Systems*, 2005, pp. 278–285.

[4] Chih-Wei Hsu and Chih-Jen Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, Mar. 2002.

[5] Ryan Rifkin and Aldebaro Klautau, "In defense of one-vs-all classification," *Journal of Machine Learning Research*, vol. 5, pp. 101–141, Dec. 2004.

[6] Bing-Yu Sun and De-Shuang Huang, "Support vector clustering for multiclass classification problems," in *Evolutionary Computation*, Dec. 2003, vol. 2, pp. 1480–1485.

[7] Andreas Sachs, Christian Thiel, and Friedhelm Schwenker, "One-class support-vector machines for the classification of bioacoustic time series," in *International Journal on Artificial Intelligence and Machine Learning*, 2006, vol. 6, pp. 29–34.

[8] Bernhard Schölkopf, John C. Platt, John C. Shawe-Taylor, Alex J. Smola, and Robert C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computing*, vol. 13, pp. 1443–1471, July 2001.

[9] Asa Ben-Hur, David Horn, Hava T. Siegelmann, and Vladimir Vapnik, "Support vector clustering," *Journal of Machine Learning Research*, vol. 2, pp. 125–137, 2001.

[10] Koby Crammer and Yoram Singer, "On the learnability and design of output codes for multiclass problems," *Machine Learning*, vol. 47, no. 2-3, pp. 201–233, May 2002.

[11] T. Joachims, "Training linear SVMs in linear time," in *International Conference on Knowledge Discovery and Data Mining (KDD)*, 2006, pp. 217–226.

[12] "GLPK (GNU linear programming kit)," 2009.

[13] A. Asuncion and D.J. Newman, "UCI machine learning repository," Tech. Rep., University of California, Irvine, School of Information and Computer Sciences, 2007.

[14] J.J. Hull, "A database for handwritten text recognition research," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 16, no. 5, pp. 550 –554, may 1994.