

Multi-Class Classifiers and Their Underlying Shared Structure

Volkan Vural ^{*†}, Glenn Fung [‡], Romer Rosales [‡] and Jennifer G. Dy [†]

[†] Department of Electrical and Computer Engineering, Northeastern University, Boston, MA, USA

[‡] Computer Aided Diagnosis and Therapy, Siemens Medical Solutions, Malvern, PA, USA

Abstract

Multi-class problems have a richer structure than binary classification problems. Thus, they can potentially improve their performance by exploiting the relationship among class labels. While for the purposes of providing an automated classification result this class structure does not need to be explicitly unveiled, for human level analysis or interpretation this is valuable. We develop a multi-class large margin classifier that extracts and takes advantage of class relationships. We provide a bi-convex formulation that explicitly learns a matrix that captures these class relationships and is de-coupled from the feature weights. Our representation can take advantage of the class structure to compress the model by reducing the number of classifiers employed, maintaining high accuracy even with large compression. In addition, we present an efficient formulation in terms of speed and memory.

1 Introduction

Multi-class problems often have a much richer structure than binary classification problems. This (expected) property is a direct consequence of the varying levels of relationships that may exist among the different classes, not available by nature in binary classification. Thus, it is of no surprise that multi-class classifiers could improve their performance by exploiting the relationship among class labels. While for the purposes of providing an automated classification result this class structure does not need to be explicitly unveiled, for (human level) analysis or interpretation purposes, an explicit representation of this structure or these class relationships can be extremely valuable.

As a simple example, consider the case for a given problem, when inputs that are likely to be in class k_1 are also likely to be in class k_2 but very unlikely to be in class k_3 . The ability to provide such explicit relationship information can be helpful in many domains. Similarly, the decision function for determining class k_1 can benefit from the decision functions learned from k_2 and k_3 - a form of transfer learning. This is helpful specially when training samples are limited.

*Corresponding author.Email: vvural@ece.neu.edu

We summarize the contributions of this paper as follows:

- We develop a multi-class large margin classifier that takes advantage of class relationships and at the same time automatically extracts these relationships. We provide a formulation that explicitly learns a matrix that captures these class relationships de-coupled from the feature weights. This provides a flexible model that can take advantage of the available knowledge.
- We provide a formulation to the above problem that leads to a bi-convex optimization problem that rapidly converges to good local solutions [Al-Khayyal and Falk, 1983; Bezdek and Hathaway, 2003]. Furthermore, we present a modified version of our general formulation that uses a novel regularization term and that is efficient in both time and memory. This approach is comparable to the fastest multi-class learning methods, and is much faster than approaches that attempt to learn a comparable number of parameters for multi-class classification.
- Our representation can take advantage of the class structure to compress the model by reducing the number of classifiers (weight vectors) and, as shown in the experimental results, maintain accuracy even with large model compression (controllable by the user).
- Our experiments show that our method is competitive and more consistent than state-of-the-art methods. It often has a lower test error than competing approaches, in particular when training data is scarce.

2 Related Work

Two types of approaches are generally followed for addressing multi-class problems: one type reduces the multi-class problem to a set of binary classification problems, while the second type re-casts the binary objective function to a multi-category problem. Variations of the first approach include: (a) constructing binary classifiers for every pair of classes (one-vs-one method), and (b) constructing binary classifiers for differentiating every class against the rest (one-vs-rest method) [Friedman, 1996; Platt *et al.*, 2000]. The results of these binary classifiers can be combined in various ways (see [Allwein *et al.*, 2000]).

We focus on the second, where the multi-class problem is cast as a single optimization problem [Weston and Watkins,

1999; Bredensteiner and Bennett, 1999]. This is appealing because the problem can be represented compactly, and it more clearly defines the optimal classifier for the K -class problem; basically, all classes are handled simultaneously. One of its main drawbacks is that the problem size/time complexity grows *impractically* fast. Thus, current research has focused on algorithm efficiency [Crammer and Singer, 2001; Tsochantaridis *et al.*, 2005; Bordes *et al.*, 2007].

The approach taken in [Crammer and Singer, 2001] consisted of breaking the large resulting optimization problem into smaller ones, taking advantage of sequential optimization ideas [Platt, 1999]. However, in general the derivatives are of size $O(m^2K^2)$, for m data points and K classes, and thus this approach does not scale well. In [Tsochantaridis *et al.*, 2005] this problem was circumvented by employing the *cutting plane algorithm* requiring only a partial gradient computation. A stochastic gradient step is further employed in [Bordes *et al.*, 2007], where various implementations were compared with the following results: for the problem sizes/implementations considered, the method in [Crammer and Singer, 2001] was the fastest, but it had by far the most memory requirements. The two other methods [Tsochantaridis *et al.*, 2005; Bordes *et al.*, 2007] had similar memory requirements.

Despite these important developments, we note two important shortcomings: (1) the solution does not provide the means to uncover the relationship between the classes or more importantly to incorporate domain knowledge about these relationships, and (2) there is no clear way to more directly relate the weight vectors associated with each class. The first shortcoming is more related to model interpretability and knowledge incorporation, while the second is related to the flexibility of the approach (*e.g.*, to learn weight vectors with specific, desirable properties).

The first issue was partly addressed in [Amit *et al.*, 2007] where a class relationship matrix is explicitly incorporated into the problem, in addition to the usual weight matrix. However, this method is limited to regularization based on the sum of the Frobenious norms of these matrices. As shown in their work, this is equivalent to a trace-norm regularization of the product of these two matrices (a convex problem). As a consequence of the above limitation, it is not clear how additional information can be incorporated into the relationship and weight matrix. In this paper, we address these shortcomings and present a formulation that, in addition to being efficient, allows for a better control of the solution space both in terms of the class relationship matrix and the weight matrix. We note that the same notion of *hidden features* motivating [Amit *et al.*, 2007] is handled by the current formulation; however we concentrate on viewing this as representing the class structure.

3 General Formulation

Before we introduce our method, let us define the notation we use in this paper. The notation $A \in R^{m \times n}$ signifies a real $m \times n$ matrix. For such a matrix, A' denotes the transpose of A and A_i the i -th row of A . All vectors are column vectors. For $x \in R^n$, $\|x\|_p$ denotes the p -norm, $p = 1, 2, \infty$. A vec-

tor of ones and a vector of zeros in a real space of arbitrary dimensionality are denoted by e and 0 respectively. Thus, for $e \in R^m$ and $y \in R^m$, $e'y$ is the sum of the components of y . We can change the notation of a *separating hyperplane* from $f(\tilde{x}) = \tilde{w}'\tilde{x} - \gamma$ to $f(x) = x'w$, by concatenating a -1 component to the end of the datapoint vector, *i.e.* $x = [\tilde{x}, -1]$. Here the new parameter w will be $w = [\tilde{w}, \gamma]$.

Let m be the total number of training points and K be the number of classes. The matrix M will be used to represent numerical relationships (linear combinations) among the different classifiers used for a given problem. In some cases (as shown later), M can be seen as an indicative of the relation among classes which are not usually considered when learning large margin multiclass classifiers (one-vs-rest, one-vs-one, etc.). In more recent papers, a simplified version of the inter-class correlation matrix M is predefined by the user [Lee *et al.*, 2004] or not considered explicitly [Crammer and Singer, 2001; Tsochantaridis *et al.*, 2005; Bordes *et al.*, 2007], separated from the class weight vector/matrix.

The idea behind our algorithm is to learn and consider the implicit underlying relationships among the b base classifiers we want to learn (depending on the chosen model) in order to improve performance. Note that the dimensions of the matrix M ($b \times p$, number of base classifiers by number of final classifiers) are strongly related to the chosen model for multiclass classification. For example, when a paradigm similar to the one-vs-rest method is considered, $p = K$ and $b < p$, if we chose $b = p = K$, M is a $K \times K$ square matrix. This is, we are going to learn K classifiers, each one trying to separate each class from the rest, but they are going to be learned jointly (interacting with each other and sharing information through M) instead of separately as is the case with the standard one-vs-rest method. Similarly, when the one-vs-one method is applied, $p = K(K-1)/2$ and b the number of base classifiers can be any number large enough $b \leq p$. If $b = p = K(K-1)/2$ and $M = I$ (fixed), the result is the exact one-vs-one method, however learning M will enforce sharing the information among classifiers and it will help improve performance. One of the main advantages of this proposed method is the possibility of using a smaller number of base classifiers than final classifiers (case $b < p$). This will result in a more compact representation of the models with less parameters to learn and hence less prone to overfitting, specially when the data is scarce.

Let A^l represent the training data used to train the l^{th} classifier where $l = 1, \dots, p$. For one-vs-rest, $A^l = A$ for every l . However, for the one-vs-one approach A^l will represent the training data for one of the p combinations, *i.e.* a matrix that consists of the training points from the i^{th} and the j^{th} classes $i, j \in (1, \dots, K)$. D^l will signify a diagonal matrix with labels (1 or -1) on the diagonals. The way we assign labels to training points depends on the approach we implement. For instance, for a one-vs-rest setting the labels corresponding to the data points in class l will be 1, whereas the labels for the rest of the data points will be -1 . Let $W \in R^{n \times b}$ be a matrix containing the b base classifiers and M^l be the column l of the matrix $M \in R^{b \times p}$. Similarly to [Fung and Mangasarian, 2001], we can define the set of constraints to separate A^l with

respect to the labeling defined in D^l as:

$$D^l A^l W M^l + y^l = e$$

Using these constraints, the problem becomes:

$$\min_{(W, M)} \sum_{l=1}^p \mu \|e - D^l A^l W M^l\|_2^2 + \|W\|_F^2 + \nu \|M\|_F^2, \quad (1)$$

where $\|W\|_F^2$ and $\|M\|_F^2$ are the regularization terms for W and M respectively, and the pair (μ, ν) control the trade-off between accuracy and generalization.

We can solve Eq. (1) in a bi-convex fashion such that when W is given then we solve the following problem to obtain M :

$$\min_{(M)} \sum_{l=1}^p \mu \|e - D^l A^l W M^l\|_2^2 + \nu \|M\|_F^2 \quad (2)$$

When M is given, on the other hand, just note that $A^l W M^l = \hat{A}^l \hat{W}$, where $\hat{A}^l = [M_{1,l} A^l, \dots, M_{b,l} A^l]$ and $\hat{W} = [W_1, \dots, W_b]^l$. Using the new notations, we can solve the following problem to obtain W :

$$\min_{(W)} \sum_{l=1}^p \mu \|e - D^l \hat{A}^l \hat{W}\|_2^2 + \|\hat{W}\|_2^2 \quad (3)$$

We can summarize the bi-convex solution to Eq. (1) as follows:

0. *initialization*: if M is a square matrix then $M^0 = I$ otherwise initialize M by setting the components to 0 or 1 randomly.
1. At iteration t , if the stopping criterion is not met, solve Eq. (3) to obtain W^{t+1} given $M = M^t$.
2. For the obtained W^{t+1} , solve Eq. (2) to obtain M^{t+1} :

In contrast with [Amit *et al.*, 2007], that recently proposed implicitly modeling the hidden class relationships to improve multi-class classification performance, our approach produces an explicit representation of the inter-class relationships learned. In [Amit *et al.*, 2007], they learn a matrix W to account for both class relationships and weight vectors; whereas, in our formulation we explicitly de-coupled the effect of the class relationships from the weight vectors captured in M and W respectively. Another important difference with [Amit *et al.*, 2007] is that in our formulation, additional constraints on M can be added to formulation (1) to enforce desired conditions or prior knowledge about the problem. As far as we know, this is not possible with any previous related approaches. Examples of conditions that can be easily imposed on M without significantly altering the complexity of the problem are: symmetry, positive definiteness, constraint bounds on the elements of M , a prior known relation between two or more of the classes, etc. Experiments regarding different types of constraints in M is currently being explored and it will be a part of future work. Since we are also explicitly modeling the classifiers weight vectors W , separately from the matrix M , we can also impose extra conditions on the hyperplanes W (alone). For example: we could include regularization-based feature selection, we could also enforce uniform feature selection across all the K classifiers by using a block regularization on all the i components of the K hyperplane classifiers, or simply incorporate *set* and *block* feature selection [Yuan and Lin, 2006] to formulation (1).

4 Practical Formulations

4.1 The one-vs-rest case

In order to simplify notation, when the one-vs-rest formulation is considered, formulation (1) can be indexed by training datapoint and rewritten as below. Let us assume that

$f^k(x) = x' w^k$ represents the k th classifier and \tilde{f}_i^k represents the output of the k th classifier for the i^{th} datapoint, A_i . i.e. $\tilde{f}_i^k = A_i w^k$. Let us define a vector f_i such that $f_i = [\tilde{f}_i^1, \dots, \tilde{f}_i^K]^l$. We also define a vector, \mathbf{y}_i , of slack variables such that $\mathbf{y}_i = [y_i^1, \dots, y_i^K]^l$ and a matrix W such that $W = [w^1, \dots, w^K]$.

$$\min_{(W)} \mu \sum_{i=1}^m \|e - D^i M f_i\|_2^2 + \sum_{k=1}^K \|w^k\|_2^2 + \nu \|M\|_F^2 \quad (4)$$

This formulation can be solved efficiently by exploiting the bi-convexity property of the problem similar to [Fung *et al.*, 2004]. Subproblem (3) is an unconstrained convex quadratic problem that is independent of any restrictions on M and it is easily obtained by solving a system of linear equations. When $M = R^{K \times K}$, this is also an unconstrained convex quadratic programming problem and its solution can also be found by solving a simple system of linear equations. The complexity of formulation (2) depends heavily on the conditions imposed on M . In what is left of the paper, we impose conditions on M , such that its elements are bounded in the set $[-1, 1]$. By doing this, M can be associated to correlations among the classes (although symmetry was not enforced in our experiments to allow more general asymmetrical relationships). This is achieved by adding simple box constraints in the elements of M , modifying formulation (2) into a constrained quadratic programming problem.

4.2 Efficient formulation for the one-vs-rest case

In this section, we propose a slightly modified version of our formulation presented in section 4.1. This modified formulation is time and memory efficient and requires neither building large sized matrices nor solving large sized systems of linear equations.

Instead of using the 2-norm regularization term in formulation (4) that corresponds to a Gaussian prior on each one of the hyperplane weight vectors w^k , we use a joint prior on the set of w^k 's, inspired by the work presented in [Lee *et al.*, 2004]. In that work, they considered the following vector-valued class codes:

$$D_{kk}^i = \begin{cases} 1 & A_i \text{ belongs to class } k \\ \frac{-1}{k-1} & \text{otherwise} \end{cases} \quad (5)$$

They showed (inspired by the Bayes rule) that enforcing, for each data point, that the conditional probabilities of the k classes add to 1 is equivalent to the sum-to-zero constraint on the separating functions (classifiers) f_k (for all x): $\sum_{k=1}^K f_k(x) = 0$. However, when considering linear classifiers, we have that:

$$\forall x, \sum_{k=1}^K f_k(x) = 0 \Rightarrow \left(\sum_{k=1}^K w^k \right) = 0 \quad (6)$$

Inspired by this constraint, instead of the standard 2-norm regularization term $\sum_{k=1}^K \|w^k\|_2^2$, we propose to apply a new

regularization term inspired by the sum-to-zero constraint: $\left\| \sum_{k=1}^K w^k \right\|_2^2$. By minimizing this term, we are favoring solutions for which equation (6) is satisfied (or close to be satisfied). As stated in [Lee *et al.*, 2004], by using this regularization term the discriminating power of the k classifiers is improved, especially where there is not a dominating class (which is usually the case). With the new regularization term we obtain the following formulation, which is very similar to formulation (4):

$$\min_{(W, M)} \mu \sum_{i=1}^m \|e - D^i M f_i\|_2^2 + \left\| \sum_{k=1}^K w^k \right\|_2^2 + \nu \|M\|_F^2 \quad (7)$$

The solution to formulation(7) can be obtained by using an alternating optimization algorithm similar to (1), but with some added computational benefits that make this formulation much faster and scalable. The main difference is that in the second iterative step of the alternating optimization, instead of (3), the problem to solve becomes:

$$\min_{(W)} \mu \sum_{i=1}^m \|e - D^i M f_i\|_2^2 + \left\| \sum_{k=1}^K w^k \right\|_2^2 \quad (8)$$

whose solution can be obtained by solving a system of linear equations such that $w^* = [w^1, \dots, w^{K'}]' = -P^{-1}q$, where $P = \Gamma \otimes \bar{A}$ (the standard Kronecker product) with $\Gamma = M' M$ and $\bar{A} = \sum_{i=1}^m (A_i' A_i) + (1/\mu)I$.

Here, $q = -2 \sum_{i=1}^m B^i e$, where

$$B^i = \begin{bmatrix} D_{11}^i M_{11} A_i & \dots & D_{1K}^i M_{1K} A_i \\ \vdots & & \vdots \\ D_{KK}^i M_{K1} A_i & \dots & D_{KK}^i M_{KK} A_i \end{bmatrix} \quad (9)$$

Note that P is a matrix of size $K(n+1)$ by $K(n+1)$. Recall that K is the number of classes and n is the number of features. As we presented above, the solution to (7) is $w^* = -P^{-1}q$ and can be calculated efficiently (both memory-wise and time-wise) by using a basic property of the Kronecker product:

$$P^{-1} = \Gamma^{-1} \otimes \bar{A}^{-1}$$

We can draw the following conclusions from the equations shown above:

- 1) Calculating $\bar{A}^{-1} \in R^{(n+1) \times (n+1)}$ and $\Gamma^{-1} \in R^{K \times K}$ separately is enough to obtain $P^{-1} \in R^{K(n+1) \times K(n+1)}$.
- 2) At every iteration of our method, only the inverse of $\Gamma \in R^{K \times K}$ is required to be calculated where K is around ten for most problems.
- 3) We can calculate any row, column or component of P^{-1} without actually building or storing P^{-1} itself or even P .
- 4) Consequently, no matter how big the P^{-1} matrix is, we can still perform basic matrix operations with P^{-1} such as $w^* = -P^{-1}q$.
- 5) Finally, there is a relatively low computational cost involved on finding the solution W^* , regardless of the size of P^{-1} .

The time complexity for this efficient method can be presented as $O(n^3 + iK^3)$, where i is the number of iterations it takes for the alternating optimization problem to converge. In our experiments, it took five iterations on average for our methods to converge. The memory complexity of our methods are also very efficient and can be presented as $O(n^2 + K^2)$.

5 Exploring the Benefits of the M Matrix

There are two main advantages of incorporating the M matrix explicitly in our proposed formulation.

One of them is to discover useful information about the relationships among the classes involved in the classification problem. In order to present an illustrative toy example, we generated two-dimensional synthetic data with five classes as depicted in Figure 1a. As shown in the Figure, there are five classes where Class 1 is spatially related to Class 2 (the classes are adjacent and similarly distributed) and similarly Class 3 is related to Class 4. On the other hand, Class 5 is relatively more distant to the rest of the classes. When we train our algorithm using an RBF kernel, we obtain the classifier boundaries in Figure 1b and the M matrix shown in Figure 1c. For this toy example, in the M matrix, similarity relations would appear as relatively large absolute M matrix coefficients. The block structure of the matrix suggests the similarities among some of the classes. Note that for similar classes the corresponding M coefficient has opposite signs. For example, let us explore the result for $f_1(x)$ and $f_2(x)$. Note if x_k belongs to class 1 we must have that $f_1(x_k) > 0$ but because class 1 and 2 are similar, $f_2(x_k)$ is likely to be greater than zero as well. This potential problem is compensated by M such that the final classifier $\tilde{f}_1(x_k)$ for class 1 is approximately $0.9f_1(x_k) - 0.1f_2(x_k)$. This means that in case of confusion (both classifiers are positive for x_k) a large magnitude of $f_2(x_k)$ (classifier 2 is very certain that x_k belong to class 2) could swing the value of $\tilde{f}_1(x_k)$ to be negative (or to change its prediction). Note that the M matrix in Figure 1c, learned the expected relationships among the five classes. Another property of the M matrix is that by using a rectangular M , we can decrease the number of classifiers required to classify a multiclass problem by slightly modifying our original formulation. In order to exemplify this M property, let us consider the one-vs-one approach where we are required to train $p = K(K-1)/2$ classifiers. Our goal is to train $b < p$ number of classifiers that can achieve comparable accuracies. This property has the potential to create compact multiclass classification models that require considerably less parameters to learn (hyperplane coefficients) especially when non-linear kernels are used. For example, let us consider the case where a kernel with 1000 basis functions is used and the one-vs-one method is considered for a 10-class problem. Using the standard one-vs-one method, $p = 45$ classifiers have to be learned with 1000 coefficients each (45000 coefficients). However, by considering $M \in R^{10 \times 45}$ we only have to learn 10×1000 plane coefficients plus M (450 coefficients) for a total of 10450 coefficients. This compact representation might benefit from Occam's razor (by avoiding overfitting) and hence resulting in more robust classifiers.

6 Numerical Experiments

In these experiments, we investigate whether we can improve the accuracy of one-vs-rest by using the information that we learn from inter-class relationships, how the performance of our classifiers compares with related state-of-the-art methods, and if we can achieve the accuracy of one-vs-one by using less number of classifiers than what one-vs-one re-

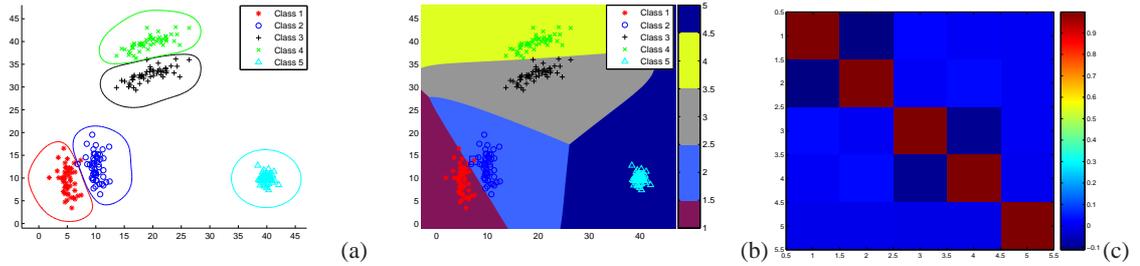


Figure 1: a: Synthetic Data-1 with five classes. b: Class boundaries for Synthetic Data-1. c: The M matrix obtained for Synthetic Data-1.

quires. We test on five data sets: ICD (978,520,35)¹, Vehicle (846, 18, 4), USPS (9298,256,10), Letter (20000,16,26)², and HRCT (500,108,5)³, with (a, b, c) representing number of samples, features, and classes respectively. We implemented the following algorithms: *original*, our original formulation in Eq. 4; *fast* our efficient formulation (7) from Sec. 4.2; *traceNorm* from [Amit *et al.*, 2007]; and *lvsrest* a standard one-vs-rest formulation. While solving (2), we imposed the conditions on M such that $M_{ij} \in [-1, 1], \forall i, j$. We used conjugate gradient descent to solve *traceNorm* as indicated in [Amit *et al.*, 2007]. We also compared our methods against popular multi-class algorithms such as *LaRank* and *SVMstruct* and used the implementations distributed by the authors. After randomly selecting the training set, we separate 10% of the remaining data as a validation set to tune the parameters. The remaining data is used for testing. We tuned our parameters (μ and ν) and the trade-off parameter (C) for all the methods over 11 values ranging from 10^{-5} to 10^5 . All of the classifiers use a linear kernel for ICD, *Vehicle* and *HRCT*, and the Gaussian Kernel with $\xi = 0.1$ for the *USPS* and *Letter* data.

Figures 2 (a) – (f) plot the performance (y -axis) of the six methods for varying amounts of training data (x -axis). Each point on the plots is an average of ten random trials. For the *USPS* and *Letter* data, the competing method [Amit *et al.*, 2007] was too demanding computationally to run for training set sizes of more than 12%, thus we limited comparisons to a lower range. For the *ICD* data, we observe that *fast* and *original* improve the one-vs-rest coding and outperforms the other algorithms for every amount of training data. *traceNorm* exceeded our memory capacity for this data set and thus results are not available. For the *HRCT* data, we observe that the four methods (*lvsrest*, *fast*, *original* and *traceNorm*) are comparable. However, *LaRank* and *SVMstruct* performed poorly. For the *Vehicle* data, we see that *fast* and *original* are better than others for the first one hundred training samples. For more than one hundred training samples, the performances of the algorithms are comparable except for *LaRank* (which performed poorly). For the *USPS* data, we observe the improvement due to the sharing of in-

formation among the classes as demonstrated by the significantly higher accuracies obtained by the *fast*, *original*, *traceNorm*, and *SVMstruct* methods, which take advantage of the shared class structure, compared to *lvsrest* and *LaRank*. *In summary, our proposed algorithms were consistently ranked among the best. SVMstruct also did well for most datasets; however, it cannot extract explicit class relationships. The main competing approach, traceNorm did not scale well to large datasets. Another advantage of our methods over traceNorm is that we can explicitly learn the inter-class relations and represent it in a matrix form, M.*

In Figure 2(f), we present the M matrix that we learned for *USPS* by using *fast* with 1000 data points. *USPS* is a good example where we can relate the relationships of classes given in M to the physical domain, since the classes correspond to digits and we can intuitively say that some digits are similar, such as 3 and 8. Looking at the 5th row of M in Fig. 2(f), we observe that the 5th classifier, $f_5(x)$, is related to digits 3, 6 and 8, which looks similar in the physical domain. Similar relations can be observed in rows 6, 7 and 8, where digit 7 is similar to 1, 6 to 5, and 9 to 8.

We also ran experiments comparing the performance of one-vs-one versus our approach with a reduced set of base classifiers, *reduced*. Due to space limitations, we only report the results on the *HRCT* and *USPS* data. Figures 2 (g) and (h) show the results. For these data, *one-vs-one* requires 10 and 50 classifiers, whereas *reduced* only trained with 5 and 10 classifiers for the *HRCT* and *USPS* respectively. In order to combine the outputs of the classifiers, we applied *voting* for both approaches. The figures demonstrate that we can achieve similar performance in terms of accuracy even with fewer base classifiers (half and one-fifth in these cases) compared to *one-vs-one*, because we take advantage of inter-class relationships.

7 Conclusions

We developed a general formulation that allows us to learn the classifier and uncover the underlying class relationships. This provides an advantage over methods that solely obtain decision boundaries since the results of our approach are more amenable to (*e.g.*, human-level) analysis. A closely related work is *traceNorm* [Amit *et al.*, 2007]. However in [Amit *et al.*, 2007], they require further work (solving another optimization problem after learning the decision boundaries) to recover the class relationships; whereas, our approach learns

¹A medical dataset from www.computationalmedicine.org

²<http://www.ics.uci.edu/~mllearn/MLRepository.html>

³A real-world high-resolution computed tomography image medical data set for lung disease classification.

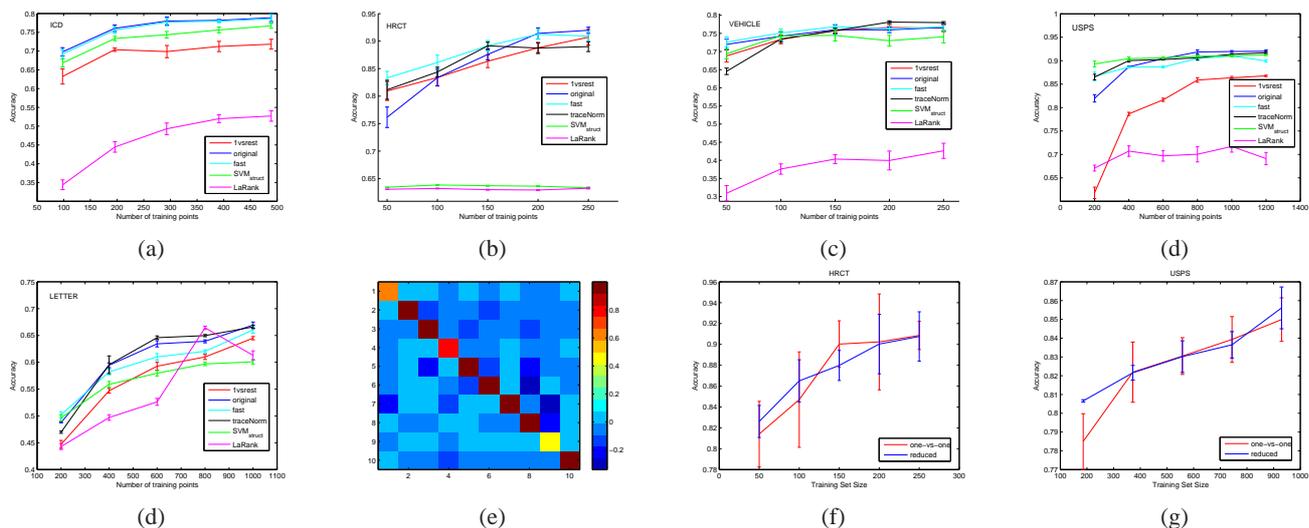


Figure 2: a-d: Experimental results for various datasets comparing one-against-rest vs. our original method (original), the fast method (fast), LaRank, SVMstruct and traceNorm. e: M matrix representing the inter-class correlations learned for USPS data. Indices correspond to digits 1 . . . 9 and 0 respectively. f-g: Comparing one-vs-one vs. our method, reduced, on the HRCT and USPS data.

the boundaries and the relationships at the same time. In addition, we are able to achieve this efficiently at faster speeds and less memory compared to the traceNorm formulation. Our fast formulation has speeds comparable to the standard one-vs-rest approach. In terms of accuracy, the performance of our formulation is consistently ranked among the best and the results confirm that taking class relationships into account improves class accuracy compared to the standard one-vs-rest approach. Moreover, experiments reveal that our approach can perform as well as standard one-vs-one with fewer base classifiers by taking advantage of shared class information.

This work provides a flexible model that is amenable to incorporating prior knowledge of class relationships. For future work, by using different norms, the model can be extended to allow regularization-based feature selection (1-norm-based). We are not aware of any method that compares with our proposed family of large margin formulations in providing the flexibility of incorporating prior knowledge to the class structure (explicit constraints on M) and adding constraints to feature weights.

Acknowledgements Thanks to Siemens Medical Solutions USA, Inc. and NSF CAREER IIS-0347532.

References

- [Al-Khayyal and Falk, 1983] F. A. Al-Khayyal and J. E. Falk. Jointly constrained biconvex programming. *MOR*, 8(2):273–286, 1983.
- [Allwein *et al.*, 2000] E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *JMLR*, pages 113–141, 2000.
- [Amit *et al.*, 2007] Y. Amit, M. Fink, N. Srebro, and S. Ullman. Uncovering shared structures in multiclass classification. In *ICML 2007*, pages 17–24, 2007.
- [Bezdek and Hathaway, 2003] J. Bezdek and R. Hathaway. Convergence of alternating optimization. *Neural, Parallel Sci. Comput.*, 11(4):351–368, 2003.
- [Bordes *et al.*, 2007] A. Bordes, L. Bottou, P. Gallinari, and J. Weston. Solving multiclass support vector machines with larank. In *ICML 07*, pages 89–96, 2007.
- [Bredensteiner and Bennett, 1999] E. Bredensteiner and K. Bennett. Multicategory classification by support vector machines. *Computational Optimization and Applications*, 12:53–79, 1999.
- [Crammer and Singer, 2001] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *JMLR*, 2:265–292, 2001.
- [Friedman, 1996] J. Friedman. Another approach to polychotomous classification. Technical report, Stanford University, Department of Statistics, 1996.
- [Fung and Mangasarian, 2001] Glenn Fung and O. L. Mangasarian. Proximal support vector machine classifiers. In *Knowledge Discovery and Data Mining*, pages 77–86, 2001.
- [Fung *et al.*, 2004] Glenn Fung, Murat Dunder, Jinbo Bi, and Bharat Rao. A fast iterative algorithm for fisher discriminant using heterogeneous kernels. In *ICML '04*, page 40, 2004.
- [Lee *et al.*, 2004] Y. Lee, Y. Lin, and G. Wahba. Multicategory support vector machines, theory, and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association*, 99(465), 2004.
- [Platt *et al.*, 2000] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin dags for multiclass classification. In S.A. Solla, T.K. Leen, and K.-R. Mueller, editors, *Advances in Neural Information Processing Systems 12*, pages 547–553, 2000.
- [Platt, 1999] John C. Platt. *Fast training of support vector machines using sequential minimal optimization*. MIT Press, 1999.
- [Tsochantaridis *et al.*, 2005] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 6:1453–1484, 2005.
- [Weston and Watkins, 1999] J. Weston and C. Watkins. Support vector machines for multiclass pattern recognition. In *Proceedings of the Seventh European Symposium On Artificial Neural Networks*, pages 219–224, 1999.
- [Yuan and Lin, 2006] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal Of The Royal Statistical Society Series B*, 68(1):49–67, 2006.