Maximizing All Margins: Pushing Face Recognition with Kernel Plurality

Ritwik Kumar
IBM Research - Almaden
rkkumar@us.ibm.com

Arunava Banerjee, Baba C. Vemuri
University of Florida
{arunava,vemuri}@cise.ufl.edu

Hanspeter Pfister
Harvard University
pfister@seas.harvard.edu

Abstract

We present two theses in this paper: First, performance of most existing face recognition algorithms improves if instead of the whole image, smaller patches are individually classified followed by label aggregation using voting. Second, weighted plurality[1] voting outperforms other popular voting methods if the weights are set such that they maximize the victory margin for the winner with respect to each of the losers. Moreover, this can be done while taking higher order relationships among patches into account using kernels. We call this scheme Kernel Plurality.

We verify our proposals with detailed experimental results and show that our framework with Kernel Plurality improves the performance of various face recognition algorithms beyond what has been previously reported in the literature. Furthermore, on five different benchmark datasets - Yale A, CMU PIE, MERL Dome, Extended Yale B and Multi-PIE, we show that Kernel Plurality in conjunction with recent face recognition algorithms can provide state-of-the-art results in terms of face recognition rates.

1. Introduction

There is little debate that today we live in an abundance of face recognition (FR) methods [24, 12, 2, 3, 14]. Some of the methods do well on concrete measures like classification accuracy and computational efficiency while others score high on subjective measures like ease of implementation and public domain availability. Here we intend to revisit existing FR methods, from the rusty old Eigenfaces [24] to the more recent Volterrafaces [14], in order to explore the possibility of squeezing more performance from them, while maintaining their existing advantages.

We begin by noting that FR as a classification problem is characterized by high data dimensionality and data sparsity. These are the textbook conditions that lead classifiers to overfit the data. We believe that this is one of the reasons performance of many FR algorithms has been limited to a much lower level than what could be achieved by them if this issue is addressed. Our simple yet effective solution to this problem is to divide images into patches and to train classifiers per patch location. During the testing stage, single label for an image is obtained by weighted plurality voting by the patch locations. Note that use of patches has been explored from time to time in FR, but our proposal is broader in the sense that it calls upon all FR methods to be used in this manner.

Next, we make the observation that in a weighted voting scheme, the manner in which weights are selected is critical. There is a large body of literature which has tried to address this problem with a few significant methods like Log-Odds Weighted Voting [16], Weighted Majority Voting [17], Bagging [4], Boosting [21, 9], Stacking [26]. It has been shown that most of the supervised weighted voting methods learn weights based on maximization of the margin of victory [22, 13] in a two class scenario. In the case of plurality voting (multiclass), there is a margin of victory with respect to each of the losers. Interestingly even the more recent multiclass Boosting methods do not take advantage of this and only maximize the minimum margin of victory [21]. We propose to learn plurality voting weights such that all the margins of victory are maximized simultaneously. We call our scheme Kernel Plurality since in addition to maximizing all margins, it also allows for higher order relations among various patch labels to be taken into account while weight computation via the use of kernels.

We corroborate our proposals with extensive experimental results using five different benchmark face datasets and five different FR algorithms. We show that: (1) FR algorithms when used within our framework significantly exceed their own performance without our framework. (2) Kernel Plurality outperforms simple Plurality, Log-Odds Weighted Plurality [16] and Stacking [26] implemented with SVMs. Note that different FR methods perform differently on various datasets and though the absolute performance of FR methods is important as shown in Fig. 4, it is more enlightening to look at the percentage improvement in performance of various FR methods (Fig. 5). That said, in

[1] Plurality [19] is a form of voting where in a multi-class contest, the class with the maximum votes wins. This is in contrast to Majority, where the a winner must get at least half of all the votes.

Symbol	Meaning
$\mathbb{D}, x_i$	Feature/Input/Data space, i^{th} vector in it.
$\mathbb{L}, l_j$	Label space, j^{th} label in it.
$\mathbb{C}, c_k$	Classifier ensemble, k^{th} classifier in it.
w_k	weight associated with classifier c_k.
$c_k(x)$	Label assigned by c_k to $x \in \mathbb{D}$
$\mathbb{R}, \mathbb{R}_+$	Set of Real numbers, positive Real numbers
$\mathbb{I}_A(x)$	Indicator function, 1 if $x \in A$ else 0
$\mathbb{P}$	Prediction subspace, $\mathbb{P} = \{-1, 0, 1\}^{\lvert\mathbb{C}\rvert}$
$\delta_{i,j}$	Kronecker delta function, 1 if $i = j$, else 0
$\mathbb{K}, \phi, K(\cdot,\cdot)$	Kernel space, Mapping, Matrix
$\mathbb{T}$	Training set, $\mathbb{T} \subset \mathbb{D}$

Table 1. Symbols and their meaning

conjunction with the recently proposed Volterrafaces [14] and LBP [2], Kernel Plurality does provide the state-of-the-art results.

To summarize, the key points made in this paper are: (1) Patch based voting outperforms holistic classification for various algorithms across databases. (2) Using off-the-shelf classifiers (e.g. SVMs) for label aggregation is not optimal. (3) Kernel Plurality outperforms existing voting methods across most databases and training set sizes indicating utility of all-margin maximization. (4) On average, Kernel Plurality improves accuracy over Plurality by $3-21\%$ while for a state-of-the-art method like Volterrafaces improvement ranges from $5-66\%$.

2. Kernel Plurality

Kernel Plurality is a new kernel based voting method. In the next subsection we describe the process through which the optimal weights are obtained for a given kernel using a training set of feature vectors. Following that we will outline the process by which a winning label is selected for a test feature vector using a given kernel and computed weights.

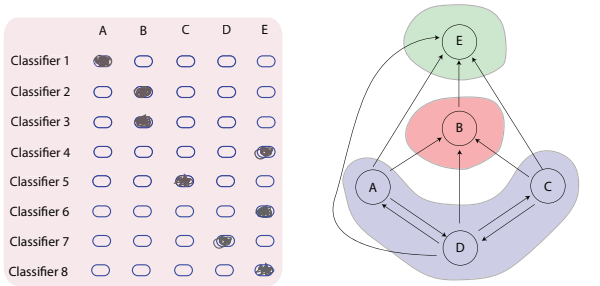
2.1. Weight Computation

The meaning of various symbols and functions used in this discussion is summarized in Table 1. According to weighted Plurality, if we ignore ties for the moment, x_i is assigned a label l according to following criteria

$$l = \arg\max_j \{\sum_{k=1}^{|\mathbb{C}|} w_k \delta_{c_k(x_i),j} | j \in \mathbb{L}\} \quad (1)$$

where δ is the Kronecker delta function and $w_k \in \mathbb{R}$ is the weight associated with the classifier c_k. Another way to express the criteria in Eq. 1 is to say that x_i should be assigned the label l such that

$$\prod_{m \in \mathbb{L}, m \neq l} \mathbb{I}_{\mathbb{R}_+}(A_{lm}(x_i)) > 0, \quad (2)$$



(a) Classifier Voting Pattern

(b) Voting Digraph and its Strongly Connected Components Graph

Figure 1. **Plurality as a set of pair-wise contests**: (a) Votes cast by 8 classifiers toward classes A to E. (b) The corresponding voting digraph (in black) showing pair-wise contests and its Strongly Connected Components graph (in color).

where $A_{lm}(x_i) = \sum_{k=1}^{|\mathbb{C}|}(\delta_{c_k(x_i),l} - \delta_{c_k(x_i),m})w_k$ and $\mathbb{I}$ is the indicator function (Table 1). Eq. 2 encodes that the winner label l must have more weighted votes than each of the other losing labels. We can rewrite this in dot product form as

$$\prod_{m \in \mathbb{L}, m \neq l} \mathbb{I}_{\mathbb{R}_+}(\langle \overrightarrow{p}_{lm}(x_i), \overrightarrow{w} \rangle) > 0, \quad (3)$$

where

$$\overrightarrow{p}_{lm}(x_i) = (\delta_{c_1(x_i),l} - \delta_{c_1(x_i),m}, \cdots, \delta_{c_{|\mathbb{C}|}(x_i),l} - \delta_{c_{|\mathbb{C}|}(x_i),m})^T \quad (4)$$

is the prediction vector and $\overrightarrow{w} = (w_1, w_2, \cdots, w_{|\mathbb{C}|})^T \in \mathbb{R}^{|\mathbb{C}|}$ is the weight vector. Note that $\overrightarrow{p}_{lm}(x_i) \in \{-1, 0, 1\}^{|\mathbb{C}|} = \mathbb{P}$, the prediction subspace.

The transformation of the decision criteria from Eq. 1 to Eq. 3 brings out the fact that a Plurality contest among multiple classes can be fully described by a set of multiple pair-wise contests. To understand it more clearly, consider the example outlined in Fig. 1. There are eight classifiers that vote for five classes (A-E) as shown in Fig. 1(a). In this example, Eq. 1 selects class E as the winner of the Plurality contest. The same conclusion can also be reached if we consider all binary contests between the classes A-E, which we represent using a digraph (directed graph) in Fig. 1(b) with an edge from label l_i to l_j if

$$\mathbb{I}_{\mathbb{R}_+}(\langle \overrightarrow{p}_{l_j l_i}(y_i), \overrightarrow{w} \rangle) > 0. \quad (5)$$

If there is a tie, edges pointing to both labels are added. Given such a digraph, the winner of the Plurality contest is the root of the corresponding Strongly Connected Components (SCC) graph [6, 18]. The SCC graph is shown in Fig. 1(b) using colored overlays where class E, the correct winner, is also the root of the SCC graph. In case of a tie for the win, the SCC root will correspond to multiple voting digraph nodes (i.e. Eq. 3 will be set to zero for multiple l) and a strategy must be chosen to resolve the tie. We will revisit this graph formulation of voting while using Kernel Plurality on test feature vectors.

At this stage we introduce the first of the two key ideas behind Kernel Plurality. Note that the ensembles we are considering have fixed size and the classifiers are learned independently using different patches. In such a setting, the linear relation in Eq. 3 implies that the elements of $\overrightarrow{p}_{lm}(x_i)$ act independently as they contribute their votes toward a decision. For instance, conditions such as 'The winner should be the label that is picked by both classifier 1 and classifier 2' cannot be encoded using a linear equation like Eq. 3. We would like to take such higher order interactions among classifiers into account while deciding a winner of a Plurality contest. Mathematically, this translates to transforming the prediction vector $\overrightarrow{p}_{lm}(x_i)$ and the weight vector $\overrightarrow{w}$ using some mapping ϕ to a kernel space $\mathbb{K}$. The winner label l must be now be chosen such that

$$\prod_{m \in \mathbb{L}, m \neq l} \mathbb{I}_{\mathbb{R}_+}(\langle \phi(\overrightarrow{p}_{lm}(x_i)), \phi(\overrightarrow{w}) \rangle) > 0. \tag{6}$$

For a given ensemble and ϕ, we do not know the best $\overrightarrow{w}$ *a priori* and would like to recover it using the training data. This brings us to the second key idea behind Kernel Plurality. For the case of two-class weighted voting contests, Lin *et al*. [16] show that the reliability of classification increases with the margin of victory. Since a Plurality contest can be defined in terms of multiple two-class contests (Fig. 1), we reason that Plurality would provide more reliable generalization performance on a test set if its weights are set such that the margin of victory with respect to *each* losing class is maximized for the training feature vectors. Note that this is in contrast to maximization of the minimum margin, which some existing techniques [21] try to achieve. The idea of maximizing all-margins as opposed to only the minimum margin is explained with a toy example in Fig. 2. Fig. 2(a) shows four classes in some embedding space with two noisy data points that belong to class 1. Note that due to their proximity to class 2 and 4, respectively, the two data point cannot be reliably classified. If the minimum margin for class 1 is maximized, we get to a situation shown in Fig. 2(b), where class 2, the class closest to class 1 is pushed far away, but other classes have clustered not far from class 2. In this case, ambiguity for the data points which was closer to class 2 has been removed, but the other point is still closer to class 4. If, as proposed, all the margins are maximized with respect to class 1, we get the situation shown in Fig. 2(c) where classes 3 and 4 are pushed farther away than before. Thus it is more likely that ambiguity for the second data point would also be removed.

In terms of the mathematical formulation, the similarity between our objective in the prediction space $\mathbb{P}$ and the objective of Support Vector Machines [7] can be readily noted. Borrowing the formalism from SVM, for a given training set $\mathbb{T}$ of feature vectors x_i with labels l_i, we would like to

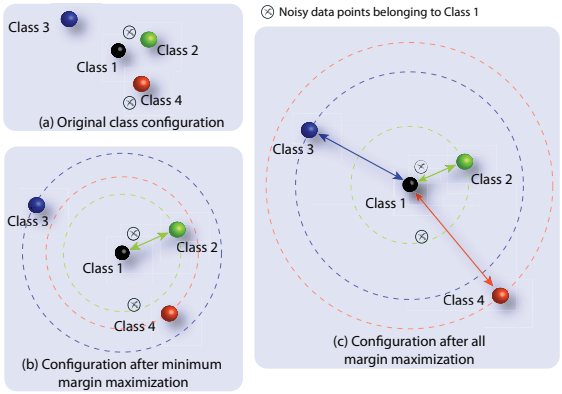


Figure 2. **All margin maximization**: (a) Four classes embedded in some space with two noisy data points that belong to class 1, but seem closer to classes 2 and 4. (b) If for class 1, only the minimum margin is maximized, classes 3 and 4 can possibly cluster just beyond the closest class (1). As a result, ambiguity for the noisy data point closer to class 4, as shown, may remains. (c) If for class 1, all pairwise margins are maximized, classes 3 and 4 can be pushed farther away and ambiguity for both the noisy data points can be reduced.

set the weights $\overrightarrow{w}^\star$ such that

$$\begin{aligned} \overrightarrow{w}^\star &= \arg\min_{w} \|\phi(w)\|^2, \\ s.t.\ &\langle \phi(\overrightarrow{p}_{l_i m}(x_i)), \phi(\overrightarrow{w}) \rangle \geq 1 \\ &\forall x_i \in \mathbb{T}, \forall m \in \mathbb{L}, m \neq l_i. \end{aligned} \tag{7}$$

Note that we have encoded the problem such that the margins should be above a certain threshold and the norm of the weight vector $\overrightarrow{w}$, which is inversely proportional to the margin, should be minimized. To build robustness against outliers we also introduce soft-margins in our formulation and allow for certain $\langle \phi(\overrightarrow{p}_{l_i m}(x_i)), \phi(\overrightarrow{w}) \rangle$ to be less than 1. This transforms Eq. 7 to

$$\begin{aligned} \overrightarrow{w}^\star &= \arg\min_{w,\xi} \|\phi(w)\|^2 + C \sum_{i=1}^{|\mathbb{T}|} \xi_i, \\ s.t.\ &\langle \phi(\overrightarrow{p}_{l_i m}(x_i)), \phi(\overrightarrow{w}) \rangle \geq 1 - \xi_i \\ &\forall x_i \in \mathbb{T}, \forall m \in \mathbb{L}, m \neq l_i, \xi_i \geq 0, \end{aligned} \tag{8}$$

where ξ_i are the slack variables and C is a constant controlling the soft-margin trade-off.

A few salient points should be noted: Firstly, in terms of SVM, we only have one class whose margin has to be maximized with respect to the origin. Consequently, the decision plane runs through the origin and b, the intercept parameter in the standard SVM formulation [7] is set to 0. Secondly, we can generate an equivalent two-class problem by negating all the vectors and labeling them class 2. The symmetry would force the decision plane to pass through the origin. Thirdly, recall from the beginning of this section that unlike most other weighted voting schemes which restrict the weight vector to the positive quadrant, we defined

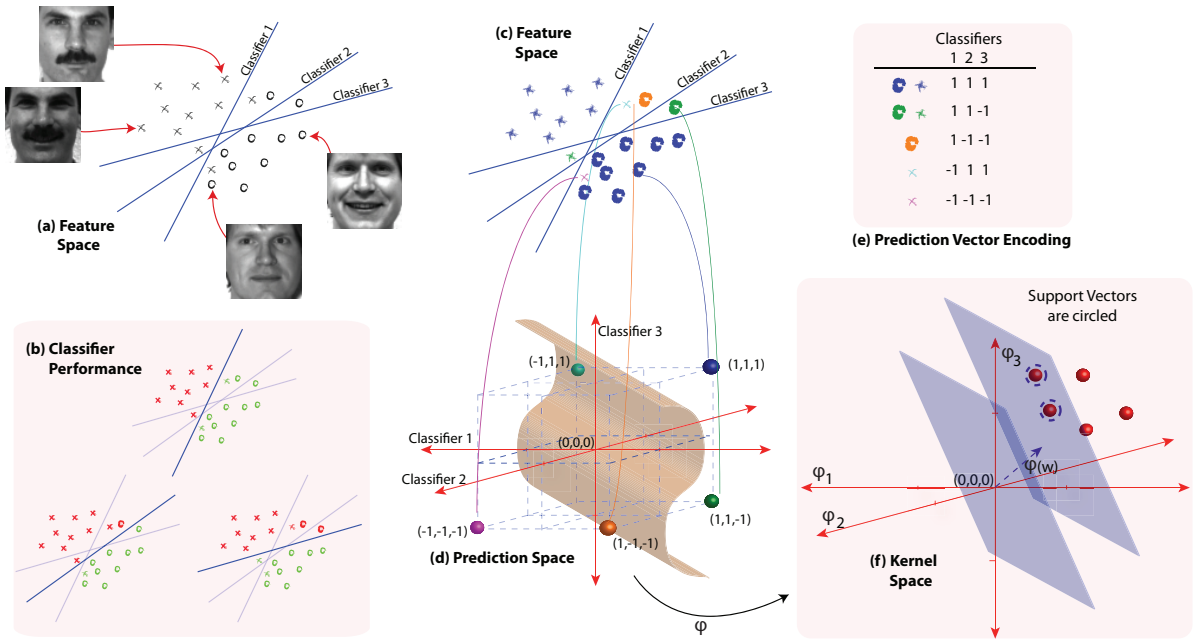


Figure 3. **Kernel Plurality**: Given a set of data points (a) and an ensemble of classifiers that labels them (b), we can encode each data point (c) with a prediction vector p as tabulated in (e). Kernel Plurality tries to find a weight vector in the prediction space $\mathbb{P}$ such that the associated decision boundary separates all the p's from the origin with maximum margin, as shown in (d). A non-linear decision boundary in $\mathbb{P}$ corresponds to a linear hyperplane in the kernel space $\mathbb{K}$ associated with mapping ϕ, as shown in (f). which is were we compute.

$w \in \mathbb{R}^{|\mathbb{C}|}$. This was done since in simple weighted Plurality (ϕ is the identity function), $\forall\ w \in \mathbb{R}^{|\mathbb{C}|}\ \exists\ w' \in \mathbb{R}_+^{|\mathbb{C}|}$ which picks the same winner as w using Eq. 1, but this cannot be guaranteed for any general kernel space $\mathbb{K}$. Finally and most importantly, the procedure outlined above is *not* classifying the feature vectors $x_i \in \mathbb{D}$ using an SVM. We are working in the prediction space $\mathbb{P}$, where we have a two-class problem, while we have an $|\mathbb{L}|$-way classification problem in $\mathbb{D}$. We have simply used the mathematical modeling provided by SVMs to optimize our objective function of maximizing all victory margins in a Plurality contest.

The solution of the mathematical program in Eq. 8 is given by

$$\overrightarrow{w}^\star = \sum_{i=1}^{l} \alpha_i \phi(x_i'), \tag{9}$$

where $\phi(x_i')$ are the support vectors and α_i are the corresponding coefficients. Like in SVMs, the exact form of the mapping ϕ is not required as long as the kernel matrix K, with its i^{th} row j^{th} column given as $K_{ij} = K(p_i, p_j) = \langle \phi(p_i), \phi(p_j) \rangle$, is available for all the prediction vectors $p_i, p_j \in \mathbb{P}$.

We summarize the key ideas behind the Kernel Plurality weight learning algorithm with an example in Fig. 3. In Fig. 3(a) we show feature vectors in the input space $\mathbb{D}$ with 3 linear classifiers. The different labeling imposed by the 3 classifiers is shown in Fig. 3(b). In Fig. 3(c) we have colored feature vectors according to their corresponding prediction vectors listed in Fig. 3(e) (given by Eq. 4). Eq. 8 asks for such a weight vector $\overrightarrow{w}^\star$ that the prediction vectors are separated from the origin with maximum margin in the prediction space $\mathbb{P}$ as shown in Fig. 3(d). We allow for a non-linear boundary in $\mathbb{P}$ by using kernel mapping ϕ. This corresponds to the separation boundary being a hyperplane in the Kernel Space $\mathbb{K}$ as depicted in Fig. 3(f) where we do our computations. We must mention that the complexity of our method is governed by the efficiency of the quadratic program solver used to find the weights.

2.2. Voting with Kernel Plurality

Given the set of prediction labels $\{c_k(y_i)\}$ for a test vector y_i, we now consider the problem of conducting a Kernel Plurality contest among the elements of $\mathbb{L}$ to pick a label for y_i. Combining Eq. 3 and Eq. 1, we pick l as the label for y_i if

$$\prod_{m \in \mathbb{L}, m \neq l} \mathbb{I}_{\mathbb{R}_+}(\langle \phi(\overrightarrow{p}_{lm}(y_i)), \sum_{i=1}^{l} \alpha_i \phi(x_i') \rangle) > 0,$$

$$\Rightarrow \prod_{m \in \mathbb{L}, m \neq l} \mathbb{I}_{\mathbb{R}_+}(\sum_{i=1}^{l} \alpha_i \cdot K(\overrightarrow{p}_{lm}(y_i), x_i')) > 0. \tag{10}$$

In case of a tie for the win, the left hand side of Eq. 10 would be zero for all the tied labels. For the purpose of the results presented in this paper, we randomly choose one of the tied labels as the winner.

In practice, instead of evaluating Eq. 10 explicitly, we found it more efficient to generate the set of pair-wise prediction vectors $\{\overrightarrow{p}_{l_i l_j}(y_i)\}_{l_i, l_j \in \mathbb{L}}$ and classify them using

Algorithms	*Abbr.*	Description
Face Recognition Methods:		
Nearest Neighbor	NN	L_2 distance based classification
Eigenfaces [24]	Eig	PCA + NN
Volterrafaces [14]	Vol	Discriminative filtering + NN
Tensor Subspace Analysis [12]	TSA	Tensor extension of Locality Preserving Projections (LPP) [11]
Local Binary Patterns [2]	LBP	Local features + NN
Label Aggregation Methods:		
Support Vector Machine [7]	SVM	Label vectors classified with linear SVM as in Stacking [26].
Log-Odds Weighted Voting [16]	WMV	Plurality with voters weights set to log of its correct classification odds.
Simple Plurality [16]	Vot	Plurality with weights set to unity.
Linear Kernel Plurality	Lin	Kernel Plurality with $K(u,v) = u'v$.
RBF Kernel Plurality	RBF	Kernel Plurality with $K(u,v) = e^{-\gamma\|\|u-v\|\|^2}, \gamma = \frac{1}{\|\mathbb{C}\|}$.
Polynomial Kernel Plurality	Poly	Kernel Plurality with $K(u,v) = (\gamma u'v)^3, \gamma = \frac{1}{\|\mathbb{C}\|}$.
Sigmoid Kernel Plurality	Sig	Kernel Plurality with $K(u,v) = \tanh(\gamma u'v), \gamma = \frac{1}{\|\mathbb{C}\|}$.

Table 2. Details of Face Recognition and Label Aggregation Algorithms used in our experiments.

a SVM with weight vector $\overrightarrow{w}^\star$ and associated kernel. The classification results are used to build the edges in the voting digraph (Fig. 1) and a winner is picked using a SCC algorithm.

3. Experiments & Results

In order to validate our framework, we conducted extensive experiments using five different benchmark FR datasets - Yale A, CMU PIE, Extended Yale B, Multi-PIE and MERL Dome. Details of these datasets are summarized in Table. 3. We used the preprocessing protocol proposed in [12] that is also used by other methods like [14] and references therein. For the Yale A, CMU PIE, and the Extended Yale B datasets, we obtained the preprocessed images from the authors of [12][2]. For the Multi-PIE and the MERL Dome[3] datasets, we used a subset of 50 labels (subjects), which were then manually cropped and aligned in line with the other three datasets. Note that all the reported results were generated by running various algorithms on the same set of images.

Since the our framework is independent of any one particular FR algorithm, we selected five different publicly available FR methods for our experiments. These are *Eigenfaces* (Eig) [24] - a PCA based method, *Volterrafaces*[4] (Vol) [14] - a recently proposed state-of-the-art method, *Tensor Subspace Analysis* (TSA)[5][12] - a method representative of the class of embedding based techniques, *Local Binary Pattern*[6] (LBP) [2] - a recently proposed features based state-of-the-art method and *Nearest Neighbor* (NN) Classifier - a baseline method. More details for these methods can be found in Table 2. For each algorithm, we also created an associated ensemble of classifiers where each constituent classifier worked with only a 8×8 pixels patch of the face image. The different methods for label aggregation we tested included SVM [7] (an instance of Stacking [26]), log-Odds Weighted Voting (WMV) [16], Simple Plurality (Vot), Plurality with Linear Kernel (Lin), Radial Basis Function Kernel (RBF), Polynomial Kernel (Pol) and Sigmoid Kernel (Sig). We used the LIBSVM [5] software package as our SVM implementation. These methods are summarized in Table 2.

All the conclusions drawn in this section are based on tabulated classification error rates for Extended Yale B, Yale A, MERL Dome, CMU PIE, and Multi-PIE datasets presented in Fig. 4. The reported error rates are the averages over ten different random splits of the data. Each row of these tables is labeled by the name of the algorithm used to generate the results listed in it. The name is given in the format 'ALG + AGG', where ALG is the abbreviated FR method name and AGG is the abbreviated label aggregation method name (see Table 2). Parameters for the FR algorithms were set using cross validation as recommended in [14]. The heading of each column indicates the number (n) of images per label used for training. In each case, $\sim n/2$ images were used as gallery images while the rest were used as probe images while generating the prediction vectors to learn Kernel Plurality weights. The algorithm with the lowest error rate for each FR algorithm is indicated in bold black while the best performer for the whole database is indicated in bold red. We conducted experiments with seven different training set size for each dataset-algorithm combination. Due to lack of space, we have only included results for three representative training set size in Table 4. Our complete results can be found in the Supplementary Material (http://www.seas.harvard.edu/∼rkkumar).

First, we test the broader proposal made in this paper that almost all FR algorithms benefit by patch based clas-

[2]Obtained from http://people.cs.uchicago.edu/∼xiaofei/

[3]Obtained from the authors of [25]

[4]Obtained from http://www.seas.harvard.edu/∼rkkumar

[5]Obtained from http://www.zjucadcg.cn/dengcai/

[6]Obtained from http://ljk.imag.fr/membres/Bill.Triggs/

sification and subsequent label fusion. For this we compare the performance of each selected classifier (ALG) on the whole image to the performance of the corresponding ensemble with traditional label aggregation methods like ALG+WMV and ALG+Vot. It can be noted that across databases, FR methods, and training set sizes, the ensembles results are significantly better than that of corresponding FR methods applied to the whole image (ALG) (only one exception was observed). At the same time, the importance of a good label aggregation method is highlighted by ALG+SVM results. Here we used the labels generated by the ensemble directly as input to a multi-class SVM. Since the number of classes ($|\mathbb{L}|$) is large in all the databases used, it can be noted that ALG+SVM almost always fails in improving the performance over ALG.

Next we examine our second hypothesis that the Kernel Plurality method, which picks voting weights so as to maximize the victory margin with respect to each losing class, is indeed effective. From the tabulated error rates, we can note that across most databases, FR methods, and training set sizes, the ensembles results with Kernel Plurality (Lin, RBF, Pol and Sig) are better than those from the existing methods like (WMV and Vot). We have color coded those cases of Lin, RBF, Pol and Sig that outperform corresponding Vot method in black for easy reading.

The gains provided by Kernel Plurality are quantitatively captured in the plot presented in Fig. 5. For each database-training set size combination presented in Fig. 4, we have plotted the percentage improvement in error rate achieved by the Kernel Plurality variants of the five selected FR algorithms over simple Plurality. Each bar show the range of improvement achieved by the five FR algorithms on a particular database-training set combination and the marker shows the average improvement. The average improvement ranges from $3-21\%$ for different cases. But the maximum improvement, typically achieved by Volterrafaces, spans a more significant $5-66\%$ range.

The effectiveness of the Kernel in Kernel Plurality is demonstrated by the fact that the RBF, Pol, and Sig variants of Kernel Plurality outperform the Lin variant in most cases (Fig. 4). This is highlighted by the fact that in most cases, the best performer for a given database-algorithm-training set size (encoded in bold black font) is one of the Kernel methods. We must point out that the use of patch-wise classification and Kernel Plurality not only improves performance of individual classifiers, but in conjunction with the recent algorithms like Volterrafaces [14] and LBP [2], our framework can achieve state-of-the-art performance. Instances of this are highlighted with red bold font for all of the selected databases. These rates also compare favorably with respect to the performance of many other existing FR methods listed in [14].

Finally, it is instructive to consider a failure case for Kernel Plurality. An easy to understand failure case would be a face image whose prediction vectors falls within the SVM margin due to the slack ξ (Eq. 8). Even though it is possible to assign voters weights such that this face is classified correctly, it is sacrificed in hope for better generalization performance. This face image would likely be correctly classified by other weighting schemes like log-Odds Weighted Voting.

Database	Labels	Images\Label	Total
Yale A [1]	15	11	165
CMU PIE [23]	68	170	11560
Extended Yale B [15]	64	38	2432
MERL Dome [25]	50	16	800
Multi-PIE [10]	50	19	950

Table 3. Databases used in our experiments.

4. Discussion

Here we note the similarities and dissimilarities among Kernel Plurality, Boosting, and SVMs, especially in the context of all margin maximization and Kernel Space voting.

Boosting can be looked as a weighted voting method with the constraint that all the votes sum to unity and be positive. In a two-class scenario, Boosting has been linked to victory margin maximization [22]. Though there is lack of a proof for some of its variants like Adaboost[9] that they indeed maximizes the victory margin, there are other two-class classification algorithms like LPBoost [8] that do so. Thus, barring the important concepts of Kernel voting and possible negative weights, it would seem that Kernel Plurality is similar in spirit to Boosting for the two-class scenario.

As we move to the case of multi-class classification, the notion of margin of victory in a voting scheme must be semantically expanded. For the winner now, there is a margin of victory with respect to each of the losers. But Boosting has traditionally defined the margin in the multi-class scenario as the minimum of all the margins [22]. This has also been noted in the very recently published multi-class generalization of LPBoost [21], which ends up maximizing the minimum margin. At this point Kernel Plurality departs significantly (in addition to having negative weights and kernels) from Boosting since it explicitly tries to maximize all the margins. As in the case of two-class voting [16], the expected improvement in the generalization performance due to all-margin maximization was confirmed by our results.

Investigations into margins have also revealed connections between Boosting and SVMs [22, 20]. They are not exactly the same, but for the binary classification problem, Boosting with a given set of hypotheses is 'similar' to running an SVM with a kernel mapping related to the label vector generated by the hypothesis set [20]. Such a relation is not clear for the multi-class scenario, hence our use of kernels with SVM in the prediction space $\mathbb{P}$ warrants further

		Yale A			CMU PIE			MERL Dome			Multi-PIE			Extended Yale B		
		Training Set Size			Training Set Size			Training Set Size			Training Set Size			Training Set Size		
	Algorithms	3	6	9	3	6	10	3	6	9	3	6	9	3	10	40
	NN	27.50	22.07	20.00	74.89	65.19	55.77	56.99	42.80	32.38	43.36	26.96	18.20	65.56	44.00	23.60
	NN + SVM	77.91	74.27	72.00	97.51	96.88	96.65	95.40	93.72	93.00	95.35	93.91	92.74	94.83	93.31	87.31
	NN + WMV	23.15	19.11	18.15	70.76	66.44	43.07	70.21	48.58	47.84	49.26	40.29	36.64	60.95	32.53	34.27
	NN + Vot	22.64	19.11	19.44	69.08	55.05	37.88	52.91	31.02	24.55	39.65	22.74	12.02	54.18	21.09	2.60
Our Methods	**NN + Lin**	22.87	**18.52**	18.15	67.37	55.00	38.60	53.27	30.31	27.43	39.71	24.93	11.13	53.46	20.96	**2.13**
Our Methods	**NN + RBF**	22.50	18.81	18.15	67.42	53.01	36.89	53.35	**29.31**	24.89	39.60	23.04	11.09	53.58	21.53	2.14
Our Methods	**NN + Pol**	**22.31**	19.41	**17.78**	67.42	53.22	36.64	53.18	29.36	24.98	**39.17**	23.09	**10.93**	53.89	**19.92**	2.17
Our Methods	**NN + Sig**	22.50	18.81	18.50	**67.27**	**52.97**	**36.54**	**52.29**	29.33	**24.16**	39.31	**22.65**	11.24	**53.36**	20.95	2.18
	Eig	30.46	18.52	10.37	75.09	68.22	58.06	62.38	50.40	37.52	44.14	29.25	20.98	61.92	49.42	31.53
	Eig + SVM	79.58	78.00	78.00	97.50	97.19	96.57	94.62	94.16	92.29	95.34	93.10	91.80	94.99	92.33	87.53
	Eig + WMV	23.67	20.93	27.33	75.10	60.31	48.18	71.38	61.67	35.54	47.31	42.85	42.32	52.70	40.03	20.20
	Eig + Vot	**23.98**	14.15	8.70	70.25	54.88	40.70	55.22	36.28	20.68	33.99	14.18	7.87	48.22	22.91	3.16
Our Methods	**Eig + Lin**	24.07	**13.63**	7.78	69.68	56.03	42.18	55.31	36.38	22.63	33.91	14.19	7.73	46.70	22.83	2.89
Our Methods	**Eig + RBF**	24.26	14.07	**8.15**	69.64	53.88	**39.06**	55.91	35.71	20.00	34.00	13.47	7.27	**46.64**	21.76	2.88
Our Methods	**Eig + Pol**	**23.98**	14.07	8.52	69.72	53.93	39.43	55.56	35.20	20.06	34.13	13.62	7.04	46.77	21.72	**2.81**
Our Methods	**Eig + Sig**	**23.98**	14.37	**8.15**	**69.63**	**53.70**	39.08	**54.80**	**34.98**	**19.40**	**33.42**	**13.32**	**7.00**	46.74	**21.63**	2.84
	Vol	26.08	23.46	19.33	46.53	26.70	18.35	11.43	5.12	4.66	3.75	1.35	0.15	30.78	12.87	4.25
	Vol + SVM	92.66	91.46	92.33	97.80	97.16	97.34	95.35	95.40	95.49	96.01	95.95	95.96	96.10	95.46	95.25
	Vol + WMV	11.48	8.30	33.70	38.81	39.11	24.18	78.40	44.84	34.26	2.78	0.74	0.18	15.93	11.62	0.40
	Vol + Vot	11.50	7.41	4.81	38.85	22.93	12.36	3.03	0.34	0.30	1.56	0.26	0.16	14.04	2.83	0.32
Our Methods	**Vol + Lin**	11.02	7.11	**4.44**	**37.35**	22.29	12.71	2.65	**0.20**	0.16	1.51	0.27	0.16	12.87	2.38	0.30
Our Methods	**Vol + RBF**	**11.02**	7.11	4.81	36.75	21.41	12.54	**2.50**	0.24	0.13	1.44	**0.26**	0.16	12.90	**2.36**	0.28
Our Methods	**Vol + Pol**	11.48	**6.81**	**4.44**	36.90	21.53	**12.02**	2.62	0.22	0.13	**1.40**	0.27	0.16	13.17	**2.36**	**0.27**
Our Methods	**Vol + Sig**	11.20	6.96	4.81	36.71	**21.40**	12.46	2.53	0.27	**0.10**	1.46	**0.26**	**0.10**	**12.82**	2.40	0.28
	TSA	43.15	23.85	16.67	57.27	18.74	25.04	52.60	43.69	35.90	**20.24**	**0.74**	**0.20**	52.48	25.06	8.26
	TSA + SVM	77.87	70.52	63.70	97.04	96.26	95.80	94.26	92.36	90.89	93.94	91.52	90.85	93.80	91.57	88.75
	TSA + WMV	25.73	18.22	17.14	60.26	34.29	27.96	35.42	29.80	32.00	38.88	8.44	5.31	50.43	33.41	1.38
	TSA + Vot	25.97	14.44	7.22	56.54	19.06	22.56	46.80	33.03	25.45	32.48	6.15	1.76	46.77	10.62	0.94
Our Methods	**TSA + Lin**	**25.19**	13.93	7.41	56.86	20.47	23.58	45.45	33.22	25.94	30.97	5.06	0.64	**46.32**	10.03	0.85
Our Methods	**TSA + RBF**	25.28	13.93	**7.04**	56.08	19.25	22.18	**45.25**	**31.82**	24.22	31.25	5.15	0.87	46.26	**9.57**	**0.84**
Our Methods	**TSA + Pol**	25.46	**13.78**	7.41	56.23	15.07	22.36	45.50	31.87	24.67	31.13	4.97	0.84	46.44	9.67	0.85
Our Methods	**TSA + Sig**	25.56	14.07	**7.04**	**55.87**	**14.95**	**22.01**	45.52	31.47	**24.19**	31.28	5.30	0.98	**46.32**	9.65	0.85
	LBP	7.41	4.56	3.70	72.16	65.46	54.21	46.39	29.44	20.10	45.31	20.54	12.10	48.83	19.23	3.84
	LBP + SVM	78.15	73.04	72.22	97.26	96.78	96.08	92.68	91.98	91.27	94.54	92.62	91.45	94.38	92.47	87.39
	LBP + WMV	7.04	**4.44**	2.59	57.50	43.81	30.38	49.10	46.32	37.97	28.49	17.68	23.56	48.24	20.21	30.52
	LBP + Vot	6.60	4.67	3.15	47.89	38.19	22.90	31.98	15.87	9.44	29.76	10.67	3.41	46.32	16.68	2.56
Our Methods	**LBP + Lin**	6.30	4.52	2.96	48.48	37.98	23.43	31.04	16.77	10.79	**27.50**	10.08	4.20	**45.07**	16.09	2.52
Our Methods	**LBP + RBF**	6.20	**4.44**	2.96	47.33	**36.99**	22.18	31.28	15.47	9.71	**27.50**	9.77	3.30	45.12	16.23	2.42
Our Methods	**LBP + Pol**	**6.11**	**4.44**	**2.59**	47.28	37.05	**22.11**	31.33	15.93	9.65	28.00	9.77	3.20	45.13	**15.94**	**2.34**
Our Methods	**LBP + Sig**	6.20	4.59	2.96	**47.31**	37.07	21.78	**31.04**	**15.40**	**9.40**	27.69	**9.62**	**3.10**	45.12	16.08	2.35

Algorithms
NN: Nearest Meighbor Classifier
Eig: Eigenfaces
Vol: Volterrafaces
TSA: Tensor Subspace Analysis
LBP: Local Binary Patterns

Label Aggregation Methods
SVM: Support Vector Machine
WMV: Plurality with Log-Odds weights
Vot: Plurality with unit weights
Lin: Kernel Plurality with linear kernel
RBF: Kernel Plurality with Radial Basis Function kernel
Pol: Kernel Plurality with Polynomial kernel

Black Bold Font: best result for dataset-algorithm combination
Red Bold Font: Best result for the dataset
Black Font: better results than corresponding unit weight Pulurality

Figure 4. **Classification Error Rates**: Key for algorithm names and color encoding is provided below the table. Lower the error, better the method. In most cases - across databases, FR algorithms and training set sizes - Kernel Plurality methods (Lin, RBF, Pol and Sig) outperforms the competing methods.

theoretical investigation.

The difference between Kernel Plurality, which maximizes all victory margins, and a collection of SVMs maximizing all pair-wise margins in the feature space $\mathbb{D}$ must also be appreciated. First, the former works in the prediction space while the latter in feature space. Second, in the former case we have one classifier which in required to make $O(2^{|\mathbb{L}|})$ prediction vectors classifications to classify each test feature vector, while the latter case requires training of $O(2^{|\mathbb{L}|})$ classifiers, instead of one.

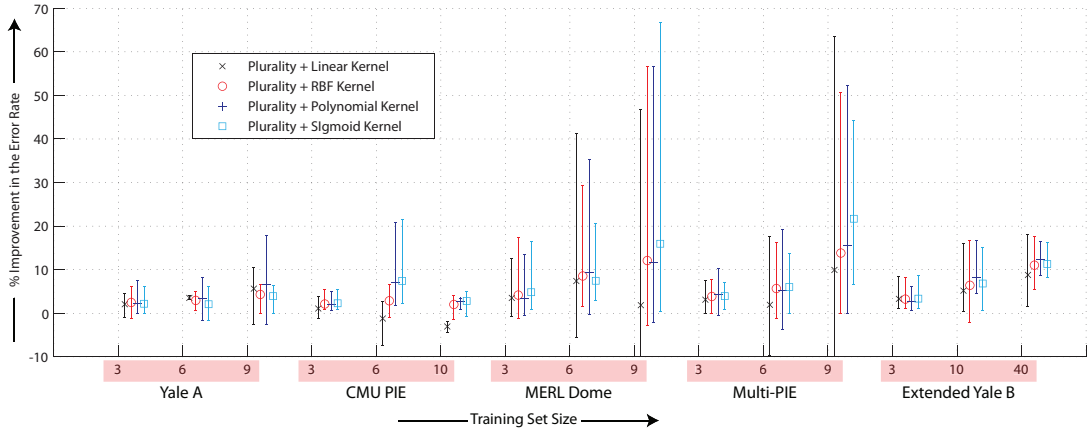


Figure 5. **Percentage Improvement in Error Rates**: For each database-training set size combination in Fig. 4, we have plotted the percentage improvement in error rates achieved by Kernel Plurality methods over Plurality (Vot). Each bar shows that range of improvement achieved by the five selected FR algorithms and the marker shows their average.

5. Conclusions

In a literature landscape teeming with face recognition algorithms, instead of introducing yet another method, here we have made proposals that can potentially improve performance for most of them. We note that face recognition as a classification problem is especially susceptible to over-fitting and for various popular algorithms, this seems to be holding their performance back. We propose and demonstrate that applying face recognition algorithms to patches and then appropriate aggregating the labels tends to do better than the algorithms applied to the whole image. Aggregating labels without taking higher order interactions among patch labels into account amounts to neglect of correlated discriminatory information present in image patches. To remedy this we propose a new voting algorithm called Kernel Plurality, which takes these high order interactions into account while maximizing the margin of victory for the correct label with respect to each of the losers. This results in better generalization performance of Kernel Plurality as compared to Log-Odds weighted Plurality, Simple Plurality and Stacking with SVMs.

6. Acknowledgements

This work was supported in part by NSF Grant No. PHY-0835713 to Hanspeter Pfister.

References

[1] http://cvc.yale.edu/projects/yalefaces/yalefaces.html.
[2] T. Ahonen, A. Hadid, and M. Pietikainen. Face Description with Local Binary Patterns: Application to Face Recognition. *IEEE PAMI*, 28(12):2037–2041, 2006.
[3] P. N. Belhumeur, J. Hespanha, and D. J. Kriegman. Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. *IEEE PAMI*, 19(7):711–720, 1997.
[4] L. Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–140, 1996.
[5] C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines. http://www.csie.ntu.edu.tw/ cjlin/libsvm.
[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to Algorithms, Second Edition. 2001.
[7] C. Cortes and V. Vapnik. Support-Vector Networks. *Machine Learning*, 20, 1995.
[8] A. Demiriz, K. P. Bennett, and J. S. Taylor. Linear Programming Boosting via Column Generation. *Machine Learning*, 2002.
[9] Y. Freund and R. E. Schapire. A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 1997.
[10] R. Gross, I. Matthews, J. Cohn, S. Baker, and T. Kanade. The CMU Multi-Pose, Illumination, and Expression (Multi-PIE) face database. Technical Report TR-07-08, CMU, 2007.
[11] X. He, D. Cai, and P. Niyogi. Locality preserving projections. In *NIPS*, 2003.
[12] X. He, D. Cai, and P. Niyogi. Tensor subspace analysis. In *NIPS*, 2005.
[13] S. Kodipaka, A. Banerjee, and B. C. Vemuri. Large Margin Pursuit for a Conic Section Classifier. *CVPR*, 2008.
[14] R. Kumar, A. Banerjee, and B. C. Vemuri. Volterrafaces: Discriminant Analysis using Volterra Kernels.
[15] K. Lee, J. Ho, and D. J. Kriegman. Acquiring Linear Subspaces for Face Recognition under Variable Lighting. *PAMI*, 2005.
[16] X. Lin, S. Yacoub, J. Burns, and S. Simske. Performance Analysis of Pattern Classifier Combination by Plurality Voting. *Pattern Recognition Letters*, 24, 2002.
[17] N. Littlestone and M. Warmuth. Weighted Majority Algorithm. *IEEE Symposium on Foundations of CS*, 1989.
[18] N. R. Miller. Graph-Theoretical Approaches to the Theory of Voting. *American Journal of Political Science*, 21(4):768–803, 1977.
[19] B. Parhami. Voting Algorithms. *IEEE Tran. on Reliability*, 43(4):617–629, 1994.
[20] G. Ratsch, B. S. S. Mika, and K.-R. Muller. SVM and Boosting: One Class. *Tech. Report*, 2000.
[21] A. Saffari, M. Godec, T. Pock, C. Leistner, and H. Bischof. Online Multi-Class LPBoost. *CVPR*, 2010.
[22] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods. *The Annals of Statistics*, 1998.
[23] T. Sim, S. Baker, and M. Bsat. The CMU Pose, Illumination, and Expression (PIE) Database. *AFGR*, 2002.
[24] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neurosciences*, 3:72–86, 1991.
[25] T. Weyrich, W. Matusik, H. Pfister, B. Bickel, C. Donner, C. Tu, J. McAndless, J. Lee, A. Ngan, H. W. Jensen, and M. Gross. Analysis of human faces using a measurement-based skin reflectance model. *ACM SIGGRAPH*, 2006.
[26] D. Wolpert. Stacked Generalization. *Neural Networks*, 1992.