Reducing JointBoost-Based Multiclass Classification to Proximity Search

Alexandra Stefan, Vassilis Athitsos
Computer Science and Engineering Department
University of Texas at Arlington

Quan Yuan, Stan Sclaroff
Computer Science Department
Boston University

Abstract

Boosted one-versus-all (OVA) classifiers are commonly used in multiclass problems, such as generic object recognition, biometrics-based identification, or gesture recognition. JointBoost is a recently proposed method where OVA classifiers are trained jointly and are forced to share features. JointBoost has been demonstrated to lead both to higher accuracy and smaller classification time, compared to using OVA classifiers that were trained independently and without sharing features. However, even with the improved efficiency of JointBoost, the time complexity of OVA-based multiclass recognition is still linear to the number of classes, and can lead to prohibitively large running times in domains with a very large number of classes. In this paper, it is shown that JointBoost-based recognition can be reduced, at classification time, to nearest neighbor search in a vector space. Using this reduction, we propose a simple and easy-to-implement vector indexing scheme based on principal component analysis (PCA). In our experiments, the proposed method achieves a speedup of two orders of magnitude over standard JointBoost classification, in a hand pose recognition system where the number of classes is close to 50,000, with negligible loss in classification accuracy. Our method also yields promising results in experiments on the widely used FRGC-2 face recognition dataset, where the number of classes is 535.

1. Introduction

Many real-world applications involve recognizing a very large number of classes, a number that can range from thousands to millions. Examples of such applications include biometrics-based identification (based on faces and/or fingerprints), hand and human body pose classification, speech and sign language recognition, and generic object recognition using computer vision. An important problem in such domains is designing recognition methods that are scalable and that achieve efficient runtime in the presence of such a large number of classes.

Large margin methods, such as boosting [9, 20] and support vector machines (SVMs) [23], have been very successful in recent years in various pattern recognition domains. A common way to apply such methods to multiclass problems is to train a one-versus-all (OVA) classifier for each class [2, 22]. However, a major bottleneck of such approaches is that, given a new pattern to classify, all OVA classifiers must be applied to that pattern, so as to identify the OVA classifier that yields the strongest response. This leads to time complexity that is linear to the number of classes, which can lead to prohibitively large classification times in large multiclass domains with thousands or millions of classes.

JointBoost [22] is a method that has recently attracted significant attention in the vision community. In JointBoost, the OVA classifiers are trained jointly, and are forced to share features. In practice, this typically leads to both higher accuracy and faster classification time. Higher accuracy is obtained because the impact of each feature is evaluated simultaneously on multiple OVA problems, thus making the estimate of that impact more reliable than if measured only on a single OVA problem. Faster classification time is obtained because the total number of unique features that need to be extracted from an input image is drastically reduced, as features are shared among multiple classifiers.

Although JointBoost drastically improves feature extraction time, the time complexity of classifying an input image with JointBoost is still linear to the number of classes, as is the case with other OVA methods based on boosting or SVMs. As the number of classes becomes large, feature extraction time becomes a negligible part of total classification time, and most of the time is spent on computing the response of each OVA classifier. The main contribution of this paper is showing that, given a pattern to classify using JointBoost, identifying the strongest-responding OVA classifier for that pattern can be treated as a proximity search problem, and more specifically as a nearest neighbor search problem in a vector space. This result allows us to use a vast array of vector indexing methods, e.g., [4, 11, 13, 19, 24], so as to improve classification time.

To demonstrate the computational advantage that can be obtained by reducing JointBoost classification to nearest neighbor search, we have implemented and evaluated a sim-

978-1-4244-3991-1/09/$25.00 ©2009 IEEE

ple, easy-to-use vector indexing method based on principal component analysis (PCA). In our experiments, the proposed method achieves a speedup of two orders of magnitude over standard JointBoost classification, in a hand pose recognition system where the number of classes is close to 50,000, with negligible loss in classification accuracy. Our method also yields promising results in experiments on the widely used FRGC-2 face recognition dataset, where the number of classes is 535. Furthermore, the results of the proposed method compare favorably to those of ClassMap [3], a competing method for speeding up multiclass recognition in domains with large numbers of classes.

2. Related Work

Large margin methods, such as boosting methods [9, 20] and support vector machines (SVMs) [23], have been widely used in recent years. Large margin methods are appealing because of their good generalization properties and their state-of-the-art performance in many applications (e.g., [22]). The standard strategy for applying large margin methods to a multiclass problem is to decompose the multiclass problem into a set of binary problems [2, 22].

Different types of multiclass-to-binary decompositions can be defined using error-correcting output codes [2, 8]. The most commonly used decompositions are into all-pairs problems, where a classifier is trained to discriminate between each pair of classes, or into one-vs.-all (OVA) problems, where, for each class, an OVA classifier is trained to discriminate between that class and all other classes. To classify a query, typically all binary classifiers are applied on the query pattern. An exception is the the DAGSVM method [17], that uses the all-pairs scheme but requires a number of classifier evaluations that is linear, not quadratic, to the number of classes.

One way to achieve classification time sublinear to the number of classes is to decompose the multiclass problem into a sublinear number of binary problems. In theory, recognizing n classes can be decomposed to $\log_2 n$ binary problems. However, such sublinear decompositions are rarely used because they define binary problems with unnatural and hard-to-learn class boundaries, leading to low classification accuracy. OVA and all-pairs decompositions, on the other hand, lead to more natural binary classification boundaries, and this explains the popularity of those decompositions in practice.

While OVA and all-pairs methods are frequently used in practice [3, 18, 22], the time complexity of those methods is at least linear to the number of classes. Linear complexity means that these methods are hard to scale to problems with a very large number of classes. Torralba, et al. [22] propose the JointBoost method for speeding up classification time. In JointBoost, the OVA models share weak classifiers among them. While sharing weak classifiers has improved both accuracy and efficiency in the experiments of [22], in JointBoost it is still the case that all OVA classifiers are applied to each pattern at runtime. Our method can be applied on top of JointBoost and significantly reduce classification time, as shown in the experiments.

In ClassMap [3], OVA classifiers and patterns are embedded into a common vector space, where the strongest responding OVA classifier for each pattern can be found more efficiently. ClassMap can be applied on top of more general large-margin methods, whereas the method proposed in this paper is designed for JointBoost-based OVA classifiers. On the other hand, the mapping proposed in this paper is lossess, and preserves information as to which OVA classifier gives the strongest response for a pattern; ClassMap does not guarantee preserving such information.

Some additional methods have been proposed for speeding up specific large multiclass problems. Efficient articulated pose estimation is achieved in [15] by combining hierarchical classifiers into a tree structure. Hierarchical template matching has been used for pedestrian detection [10] and articulated pose estimation [21]. Articulated pose can also be treated as a multidimensional regression problem, and estimators can be trained that directly map observations into vectors from a continuous pose space [1, 7]. However, many domains (e.g., face recognition) do not lend themselves readily either to hierarchical decomposition or to regression-based estimation. In contrast, our method can readily be applied in any domain where JointBoost is applicable, and thus is significantly more general than the above-mentioned domain-specific approaches.

3. Review: Multiclass Recognition Using JointBoost

Let $\mathbb{X}$ be a space of patterns, and $\mathbb{Y}$ be a finite set of class labels. Every pattern $X \in \mathbb{X}$ has a class label $L(X) \in \mathbb{Y}$. In JointBoost [22], for each class $y \in \mathbb{Y}$ a boosted classifier $H_y : \mathbb{X} \rightarrow \mathbb{R}$ is trained to discriminate between patterns of class y and all other patterns. Classifier H_y is of the following form:

$$H_y = \sum_{m=1}^{d} \alpha_{y,m} h_m + k_y \ , \qquad (1)$$

where each h_m is a weak classifier with weight $\alpha_{y,m}$, and k_y is a class-specific constant that gives a way to encode a prior bias for each class y [22]. We should also note that, in JointBoost, weights $\alpha_{y,m}$ are constrained to be either 1 or 0 (depending on whether H_y is using weak classifier h_m or not), but the method proposed in this paper does not use that constraint, and can be applied regardless of the possible values for $\alpha_{y,m}$.

Higher (more positive) responses $H_y(Q)$ indicate higher confidence that the true class label $L(Q)$ of pattern Q is

y. To classify a query $Q \in \mathbb{X}$, we evaluate $H_y(Q)$ for all $y \in \mathbb{Y}$, and classify Q as belonging to the class y for which $H_y(Q)$ is maximized. More specifically, if we denote as $H(Q)$ the output of the multiclass classifier H for pattern Q, $H(Q)$ is defined as:

$$H(Q) = \text{argmax}_{y \in \mathbb{Y}} H_y(Q) \ . \quad (2)$$

At runtime, given a pattern Q to classify, the standard approach is to apply all OVA classifiers H_y, and identify the y such that H_y gives the strongest response. Clearly, this approach has complexity linear to the number of classes. Our goal in this paper is to show that the strongest-responding classifier H_y can be found efficiently, using vector search methods, without needing to evaluate $H_y(Q)$ for all y. This topic is addressed in the next sections.

4. Reduction to Nearest Neighbor Search

The core observation underlying our method is that, for JointBoost-based multiclass recognition, both test patterns and OVA classifiers can be represented as vectors, specifying points on the surface of a hypersphere. Finding for a test pattern Q the strongest-responding OVA classifier H_y can be done by doing nearest neighbor search on those points.

In particular, we will map both OVA classifiers and test patterns into a $(d+2)$-dimensional vector space, where d is the number of weak classifiers that are used to define the OVA classifiers. We denote by $V(Q)$ and $V(H_y)$ respectively the vectors corresponding to test pattern Q and OVA classifier H_y. In defining this mapping, we will explicitly ensure that all resulting vectors have the same norm. Ensuring that all $V(H_y)$ and $V(Q)$ have the same norm will be used in reducing the problem of finding the winning OVA classifier for each Q to the problem of finding the nearest neighbor of $V(Q)$ among all $V(H_y)$.

We begin by defining the vector $V(H_y)$ corresponding to each OVA classifier H_y:

$$V(H_y) = (\alpha_{y,1}, \ldots, \alpha_{y,d}, k_y, c_y) \ . \quad (3)$$

In the above equation, $\alpha_{y,m}$ and k_y are the weights and class-bias terms used in Equation 1, and c_y is a class-specific quantity that ensures that all $V(H_y)$ have the same Euclidean norm.

Quantity c_y can be determined as follows: first, we need to identify what the maximum norm of any $V(H_y)$ would be if we set all c_y to zero:

$$N_{\max} = \sqrt{\max_{y \in \mathbb{Y}} [(\sum_{m=1}^{d} \alpha_{y,m}^2) + k_y^2]} \ . \quad (4)$$

Then, we define c_y as:

$$c_y = \sqrt{N_{\max}^2 - [(\sum_{m=1}^{d} \alpha_{y,m}^2) + k_y^2]} \ . \quad (5)$$

By defining c_y this way, it can easily be verified that the Euclidean norm of every $V(H_y)$ is equal to $N_{\max}$.

Now we can define the vectors corresponding to test patterns. In particular, given a pattern $Q \in \mathbb{X}$, we define an auxiliary vector $V_{\text{orig}}(Q)$, and the vector of interest $V(Q)$, as follows:

$$V_{\text{orig}}(Q) = (h_1(Q), \ldots, h_d(Q), 1, 0) \ , \quad (6)$$

$$V(Q) = \frac{N_{\max} V_{orig}(Q)}{\|V_{orig}(Q)\|} \ , \quad (7)$$

$$(8)$$

where $\|V\|$ denotes the Euclidean norm of V, and h_m are the weak classifiers used in Equation 1.

Using these definitions, Equation 2 can be rewritten as follows:

$$H(Q) = \text{argmax}_{y \in \mathbb{Y}} H_y(Q) \quad (9)$$

$$= \text{argmax}_{y \in \mathbb{Y}} (V_{orig}(Q) \cdot V(H_y)) \quad (10)$$

$$= \text{argmax}_{y \in \mathbb{Y}} (V(Q) \cdot V(H_y)) \ , \quad (11)$$

where $V_1 \cdot V_2$ denotes the dot product between vectors V_1 and V_2. To justify the above lines, we first observe that the $(d+2)$-th coordinate of $V(H_y)$, which is set to c_y, does not influence $V_{\text{orig}}(Q) \cdot V(H_y)$, since the $(d+2)$-th coordinate of each $V_{\text{orig}}(Q)$ is set to zero. Therefore, it can be easily verified that, for all H_y, $H_y(Q) = V_{\text{orig}}(Q) \cdot V(H_y)$. Also, since $V(Q)$ is just a scaled version of $V_{\text{orig}}(Q)$, the same H_y that maximizes $V_{orig}(Q) \cdot V(H_y)$ also maximizes $V(Q) \cdot V(H_y)$.

We will now take one additional step, to show that maximizing the dot product between $V(H_y)$ and $V(Q)$ is the same as minimizing the Euclidean distance between $V(H_y)$ and $V(Q)$. That can be easily shown, by using the fact that both $V(Q)$ and $V(H_y)$ are vectors of norm $N_{\max}$, because the dot product and the Euclidean distance for vectors of norm $N_{\max}$ are related as follows:

$$\|V(Q) - V(H_y)\|^2 = 2N_{\max}^2 - 2(V(Q) \cdot V(H_y)) \ . \quad (12)$$

The above equation can be easily derived as follows:

$$\|V(Q) - V(H_y)\|^2 = \quad (13)$$

$$= (V(Q) - V(H_y)) \cdot (V(Q) - V(H_y)) \quad (14)$$

$$= (V(Q) \cdot V(Q)) + (V(H_y) \cdot V(H_y)) - \quad (15)$$

$$2(V(Q) \cdot V(H_y)) \quad (16)$$

$$= 2N_{\max}^2 - 2(V(Q) \cdot V(H_y)) \ , \quad (17)$$

using the fact that $(V(Q) \cdot V(Q)) = (V(H_y) \cdot V(H_y)) = N_{\max}^2$.

By combining this result with that of Equation 11, it follows readily that:

$$H(Q) = \text{argmin}_{y \in \mathbb{Y}} (\|V(Q) - V(H_y)\|) \ . \quad (18)$$

This result means that, given a test pattern Q, finding the strongest-responding OVA classifier H_y is reduced to finding the nearest neighbor of $V(Q)$ among all vectors $V(H_y)$. The next section describes how to use that fact for speeding up multiclass recognition.

5. A Simple Vector Indexing Scheme

So far we have established that, in order to classify via JointBoost a test pattern Q, it suffices to find the nearest neighbor of $V(Q)$ among all vectors $V(H_y)$. Clearly, vectors $V(H_y)$ can be computed off-line and stored in a database. The importance of reducing JointBoost-based classification to nearest neighbor search is that a vast array of vector indexing methods can be used to speed up this search, such as, e.g., the methods in [4, 11, 13, 19, 24].

In order to illustrate the computational savings that can be obtained by treating JointBoost-based classification as a nearest neighbor search problem, we have implemented a simple and easy-to-use vector indexing method that is based on principal component analysis (PCA) [14]. Since the set of vectors $V(H_y)$ is computed off-line, we can use those vectors for an additional off-line step, where PCA is used to identify the principal components of those vectors and the corresponding projection matrix Φ. Given a test pattern Q, its vector $V(Q)$ can be projected to $\Phi(V(Q))$ online, and then $\Phi(V(Q))$ can be compared to the projections $\Phi(V(H_y))$ of the vectors corresponding to classifiers H_y.

PCA can easily be used within a filter-and-refine retrieval framework [12]: given a user-defined integer parameter p, filter-and-refine works as follows:

- **Input:** A test pattern Q, and its vector representation $V(Q)$.
- **Filter step:** Compute the projection $\Phi(V(Q))$ to the lower-dimensional space, and find the nearest neighbors of $\Phi(V(Q))$ among the set of all $\Phi(V(H_y))$. Keep the top p nearest neighbors, where p is a user-defined parameter, as mentioned above.
- **Refine step:** For each of the top p nearest neighbors, compute $H_y(Q)$.
- **Output:** Return the H_y yielding the strongest response $H_y(Q)$, among the H_y's evaluated during the refine step.

As long as $d' \ll d$ (where d' is the number of dimensions of $\Phi(V(Q))$, and d is the number of weak classifiers), the filter step is significantly faster than simply applying all H_y to Q. At the refine step we do evaluate some classifiers H_y, but, if $p \ll d$, these classifiers are only a small subset of the entire set of OVA classifiers.

5.1. Guarantees of Accuracy

We should note that the simple filter-and-refine method outlined above does not guarantee achieving the same accuracy as brute-force search. In other words, it does not guarantee that the H_y retrieved at the refine step will be truly the one that we would have identified if we had simply evaluated $H_y(Q)$ for all y. However, our filter-and-refine method can be easily modified to guarantee achieving the same accuracy as brute-force search.

More specifically, we can utilize the fact that PCA is a *contractive* mapping, meaning that the Euclidean distance between $\Phi(V(Q))$ and $\Phi(V(H_y))$ is guaranteed to be not greater than the Euclidean distance between $V(Q)$ and $V(H_y)$. When the filter step estimates distances based on a contractive mapping, it is well-known that the refine step can be defined in a way that guarantees finding the true nearest neighbor. Details on that topic can be found at [12]. In our experiments we found that, although we use a filter-and-refine version that does not guarantee finding the nearest neighbor 100% of the time, the accuracy that we obtained in practice was so high that it was not worth implementing a more complicated version.

We should note that a large variety of vector and metric indexing methods also guarantee finding the correct nearest neighbor, e.g., the methods in [19, 24, 25]. Such methods can easily be integrated into the filter step of our method.

6. Classification Time Complexity

Given a test pattern Q, the time that it takes to classify Q using the proposed method can be decomposed to the following costs:

- **Weak classifier cost:** The cost of computing $h_m(Q)$ for each weak classifier h_m. This takes time $O(d)$, where d is the number of weak classifiers. For JointBoost, it is empirically observed in [22] that d tends to increase logarithmically with the number of classes, so this time cost should become a negligible fraction of total time as the number of classes increases.
- **Projection cost:** The cost of computing the PCA projection $\Phi(V(Q))$. If Φ projects from $d+2$ dimensions to d' dimensions, this takes time $O(d^2)$, and becomes a negligible fraction of total time as the number of classes increases, assuming that, as mentioned earlier, d scales logarithmically with the number of classes.
- **Filter cost:** The cost of measuring Euclidean distances between $\Phi(V(Q))$ and $\Phi(V(H_y))$ for each H_y. This takes time $O(d'|\mathbb{Y}|)$, where $|\mathbb{Y}|$ is the number of classes. This is still linear to the number of classes, but we can obtain a big constant factor of savings if $d' \ll d$. We should also note that several methods

exist for sublinear nearest neighbor search in vector spaces, including the popular LSH method [11], and such methods can be easily integrated into our method to achieve time sublinear to the number of classes.

- **Refine cost:** The cost of evaluating $H_y(Q)$ for each H_y selected at the filter step. This takes time $O(dp)$, where p is the number of classifiers H_y selected at the filter step. As shown in our experiments, typically $p \ll |\mathbb{Y}|$, so the refine cost is much smaller than simply evaluating $H_y(Q)$ for each H_y.

7. Experiments

The datasets used in our experiments were generated from two original datasets: a dataset of hand images, where the task is to estimate the handshape and the 3D orientation, and the Face Recognition Grand Challenge (FRGC) Version 2 dataset [16] of 2D face images. Using these datasets, we compare the proposed method to brute-force search, which is the standard way of classifying patterns using OVA classifiers, not only for JointBoost, but in general for methods based on boosting and support vector machines [2, 18]. We also compare our method with ClassMap [3], a method that can be used to speed up OVA-based classification. We only used ClassMap embeddings trained using AdaBoost, as specified in [3], because these embeddings were shown in [3] to outperform other versions of ClassMap embeddings.

7.1. Datasets

7.1.1 The Hand Dataset

This dataset contains hand images of 81 basic hand shapes defined in American Sign Language (ASL). There are 30 different out-of-plane view angles for each shape, and 20 in-plane rotations for each out-of-plane view, for a total of $81 \times 30 \times 20 = 48{,}600$ hand pose classes. The training examples used for each class were 150 synthetic images, generated using Poser 5 [5].

For each synthetic hand image, cluttered background from random real images was added to the regions outside the hand silhouette. From each hand image, a histogram-of-oriented-gradient (HOG) feature vector [6] of dimension 2,025 was extracted. The image was normalized to 48 by 48 pixels, which was divided into cells of size 6 by 6, with neighboring cells overlapping by half. For each cell, nine edge orientation bins were evenly spaced between 0 to 180 degrees. Bins in each cell were normalized with the surrounding 3 by 3 cells. All the bins from all the cells were vectorized into a feature vector of 2025 feature components for a hand sample. Each weak classifier h_m is a feature stump, completely specified by parameters f_m and t_m, that checks whether the f_m-th HOG feature is greater than t_m or not. JointBoost selected 3,000 weak classifiers after training on this dataset.

For evaluation, we used a synthetic test set, disjoint from the training set, and consisting of 281 synthetic hand images (chosen randomly among images from all 48,600 classes). In addition to the synthetic hand images, we also used a second test set of 992 real hand images, collected from 7 subjects and with cluttered background. Because of the difficulties in visually estimating the 3D hand orientation on an image, we assigned to each hand image three different class labels (out of the 48,600 possible class labels). Each of those three class labels corresponded to the same handshape and a 3D orientation within 30 degrees of the manually labeled orientation. The classification result is considered correct iff it is equal to one of those three labels.

7.1.2 The Face Dataset

This dataset contains all 2D face images in the FRGC-2 dataset [16], amounting to 36817 face images from 535 subjects (i.e., 535 classes). The original resolution of the face images was either 1704×2272, or 1200×1600. All images were converted to gray images and normalized to 100 by 100 pixels. A PCA space was learned from 4,000 uniformly sampled training faces of all the subjects. The features of face images were their projections on the top 2,509 PCA components, which accounts for 99.9% of the variance. JointBoost selected 10,000 weak classifiers after training on this dataset. Each weak classifier h_m is a feature stump, completely specified by parameters f_m and t_m, that checks whether the f_m-th PCA dimension is greater than t_m or not. For evaluation, we used 300 face images, that were chosen randomly, and excluded from the training set.

7.2. Results

Performance is measured in terms of speed-up factor with respect to brute force, and classification accuracy. The speed-up factor is the ratio between classification time using our method (or ClassMap) and classification time using bruote-force search. By definition, brute force achieves a speed-up factor of 1. For the proposed method, the parameters that need to be chosen are d', i.e., the dimensionality of the lower-dimensional PCA space, and p, i.e., the number of OVA classifiers to be evaluated at the refine step. To reduce the number of free parameters to one, we decided to set for our method, in all experiments, $p = \frac{nd'}{d}$, where n is the number of classes. This constraint is a simple choice that forces the filter and the refine step to have the same running time. The PCA projection matrix, for each dataset, was computed based solely on the vectors $V(H_y)$ of the OVA classifiers obtained for each dataset.

In presenting the results we refer to the proposed method as OVA-VS, an acronym for OVA-based classification using

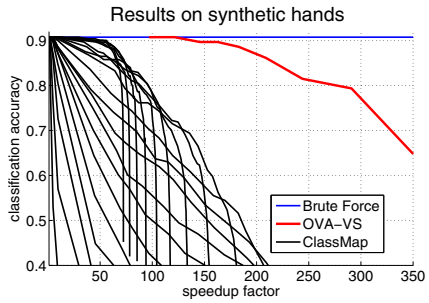

Figure 1. Accuracy vs. speed-up factor obtained by the proposed OVA-VS method, ClassMap, and brute force, on the test set of synthetic hand images. The brute force accuracy, which is a single value equal to 90.75%, is shown as a horizontal line.

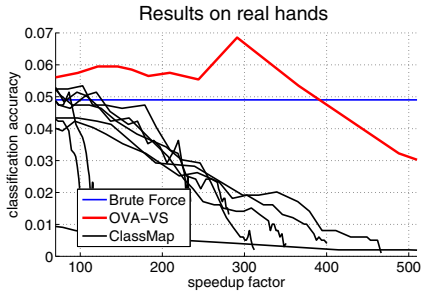

Figure 2. Accuracy vs. speed-up factor obtained by the proposed OVA-VS method, ClassMap, and brute force, on the test set of real hand images. The brute force accuracy, which is a single value equal to 4.9%, is shown as a horizontal line.

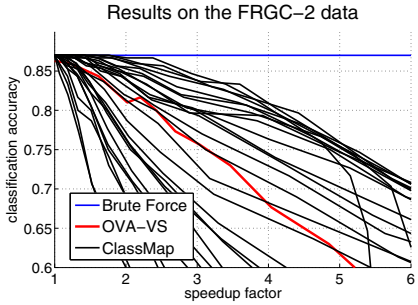

Figure 3. Accuracy vs. speed-up factor obtained by the proposed OVA-VS method, ClassMap, and brute force, on the test images of the FRGC-2 face dataset. The brute force accuracy, which is a single value equal to 87.0%, is shown as a horizontal line.

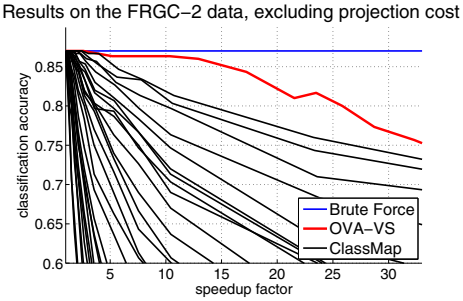

Figure 4. Accuracy vs. speed-up factor (ignoring the projection cost of OVA-VS and ClassMap) obtained by the proposed OVA-VS method, ClassMap, and brute force, on the test images of the FRGC-2 face dataset. The brute force accuracy, which is a single value equal to 87.0%, is shown as a horizontal line.

vector search.

7.2.1 Results on the Synthetic Hands Dataset

Figure 1 compares the performance of the proposed OVA-VS method, ClassMap, and brute-force search on the test set of 281 synthetic hand images. In Figure 1, we plotted a single performance curve for OVA-VS, obtained by constraining d' and p as specified above, and varying d'. In contrast, for ClassMap, we plotted a family of curves, each curve corresponding to a different embedding dimensionality, ranging from 1 to 90 dimensions, and to varying p. The single OVA-VS curve corresponds to much better accuracy vs. efficiency trade-offs than any of the results obtained for ClassMap. As a highlight, OVA-VS gave a speed-up factor of 120 over brute-force search for a classification accuracy of 90.75% (equal to that of brute-force search), whereas ClassMap gave a speed-up factor of only 29 for that accuracy. OVA-VS yielded this result for $d' = 12$ and $p = 194$.

Figure 2 shows results for the test set of 992 real hand images. Once again, the single OVA-VS curve corresponds to much better accuracy vs. efficiency trade-offs than any of the results obtained using different dimensionality and p parameters for ClassMap. OVA-VS gave a speed-up factor of 290 over brute-force search for a classification accuracy of 6.85%. The highest accuracy obtained for ClassMap was 5.3%, for a speedup factor of 105. We note that both ClassMap and OVA-VS attained accuracies higher than brute-force search for the real images. This is a somewhat curious result, that was also reported in the original ClassMap paper [3]. Intuitively, the lower-dimensional projection of OVA-VS and embedding of ClassMap can be seen as new features extracted from the output of the weak classifiers, and according to our results these new features lead to higher accuracy than the original weak classifiers.

Additionally, on the real hand images, OVA-VS yields an accuracy of 5.35%, still slightly better than that of brute force search, for a speedup factor of 365. ClassMap produces accuracies better than or equal to that of brute force search only for speedup factors less than or equal to 127.

It is worth noting that, even using brute force, classification accuracy drops significantly from 90.75% for the synthetic test images to 4.9% for the real hand images. The relatively low accuracy for the real hand images simply re-

flects the difficulty of estimating hand pose from a single image, due to the very large number of possible classes. Since there are 48,600 classes, a classification accuracy of 4.9% is still 2381 times higher than the accuracy of a random classifier. We should also note that even with this accuracy, the proposed hand pose estimation system can be useful for initializing a hand tracker based on particle filtering, where temporal integration can be used to significantly improve overall tracking accuracy.

7.2.2 Results on the FRGC-2 Dataset

Figure 3 plots the results attained with the proposed OVA-VS method, ClassMap, and brute force on the FRGC-2 dataset. We note that for an accuracy of 84.2%, which is 2.8% lower than that of brute-force search, OVA-VS achieved a speedup factor of only 1.6. As seen on the same figure, ClassMap achieved a speedup of 3.0 for the same accuracy of 84.2%, and a speedup of 1.8 for an accuracy of 87%, which is equal to the accuracy of brute-force search.

The main reason for the relatively small improvement in classification time attained by both OVA-VS and ClassMap is the relatively small number of classes in this dataset: only 535, compared to the 48,600 classes of the synthetic hands dataset. As a result, generating a single dimension of a PCA projection, or a single dimension of a ClassMap embedding, are operations that incur 1/535 of the cost of brute-force search, compared to 1/48600 for the hands dataset. In other words, the relatively small number of classes makes the projection cost defined in Section 6 more pronounced. As discussed in Section 6, the projection cost for our method becomes negligible as the number of classes becomes large, and this is also true for ClassMap.

Figure 4 plots classification accuracy vs. speedup factor, but ignores, in computing running times, the projection cost of generating PCA projections and ClassMap embeddings for the queries. This curve is representative of the performance we could expect if we had a much larger number of classes, that would make the projection costs negligible. In that case we see that the proposed OVA-VS method performs better than ClassMap, obtaining, for example, a speedup factor of 12.9 for a classification accuracy of 86.0%. For that same accuracy, ClassMap gives a speedup factor of 4.6.

The results that exclude the projection cost are promising, and indicate that we can expect significant classification time improvements from using our method in face recognition domains with tens of thousands of classes or even more. Unfortunately, we are not aware of any publically available face dataset with such a large number of classes. To the best of our knowledge, no public dataset of 2D face images contains more classes than the FRGC-2 dataset, while still providing a sufficient number of training examples per class to learn OVA classifiers. While several important applications require face recognition in the presence of tens of thousands of classes or more (especially in homeland security and surveillance domains), security and privacy concerns make it difficult for such datasets to be made publically available.

7.2.3 Summary of results

On both synthetic and real hand images, the proposed OVA-VS method significantly outperformed ClassMap and led to speedups of two orders of magnitude compared to brute force with no losses in classification accuracy. On the face dataset, the performance of OVA-VS was hampered by the projection cost, which was relatively high due to the relatively small number (535) of classes. When we excluded the projection cost from the overall running time, to get a picture of the expected performance when the number of classes reaches 10000 or more, OVA-VS again significantly outperformed ClassMap, and gave speedups of over one order of magnitude compared to brute force, with very little reduction (from 87% to 86%) in classification accuracy.

8. Discussion

We have shown that multiclass recognition using JointBoost can be reduced, at runtime, to a nearest neighbor search problem in a vector space. This reduction allows the use of a wide array of vector indexing methods for speeding up multiclass recognition. In our experiments, we have shown that a very simple indexing method, that uses PCA to select a few candidate nearest neighbors, works very well in practice and achieves, for the hands dataset, speedups of two orders of magnitude with no loss in classification accuracy, compared to brute-force search. Our method outperforms ClassMap in the hands dataset, and if we ignore the projection cost (which would be negligible if we had a significantly larger number of classes), our method also outperforms ClassMap on the faces dataset.

In comparing our method with ClassMap, it is worth noting that ClassMap defines a *lossy* vector representation of OVA classifiers H_y and patterns Q. Therefore, if V_{CM} is the vector mapping defined by ClassMap, the nearest neighbor of $V_{\mathrm{CM}}(Q)$ among all $V_{\mathrm{CM}}(H_y)$ is not guaranteed to correspond to the H_y maximizing $H_y(Q)$. In contrast, the method proposed in this paper defines a *lossless* vector representation, where the nearest neighbor always corresponds to the strongest-responding classifier.

We should emphasize that, instead of PCA, any other vector indexing method can also be integrated in the filter step of the proposed method. Several vector indexing methods guarantee finding the correct nearest neighbor for each query. Using such methods for the filter step of our algorithm guarantees that classification accuracy using our method will always equal that of brute-force search.

Naturally, an interesting topic for future exploration is to try a larger number of vector indexing methods, in order to identify methods that tend to work well in practice within the proposed framework. Also, as the proposed method is only applicable to JointBoost-based classification, it will be interesting to investigate whether similar methods can also be designed for other types of large margin classifiers, such as support vector machines. Progress in this area can lead to a broader theory of how to integrate database indexing methods with general large margin methods, so as to achieve scalable classification time complexity in domains with a very large number of classes.

Acknowledgements

This work has been supported by NSF grants CNS-0202067, IIS-0705749, and IIS-0812601, as well as by a UTA startup grant to Professor Athitsos, and UTA STARS awards to Professors Chris Ding and Fillia Makedon.

References

[1] A. Agarwal and B. Triggs. Recovering 3D human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):44–58, 2006.

[2] E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000.

[3] V. Athitsos, A. Stefan, Q. Yuan, and S. Sclaroff. ClassMap: Efficient multiclass recognition via embeddings. In *IEEE International Conference on Computer Vision (ICCV)*, 2007.

[4] C. Böhm, S. Berchtold, and D. A. Keim. Searching in high-dimensional spaces: Index structures for improving the performance of multimedia databases. *ACM Computing Surveys*, 33(3):322–373, 2001.

[5] Curious Labs, Santa Cruz, CA. *Poser 5 Reference Manual*, August 2002.

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 886–893, 2005.

[7] T. E. de Campos and D. W. Murray. Regression-based hand pose estimation from multiple cameras. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 782–789, 2006.

[8] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[9] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2):337–374, 2000.

[10] D. Gavrila and V. Philomin. Real-time object detection for "smart" vehicles. In *IEEE International Conference on Computer Vision*, pages 87–93, 2001.

[11] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *International Conference on Very Large Databases*, pages 518–529, 1999.

[12] G. Hjaltason and H. Samet. Properties of embedding methods for similarity searching in metric spaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):530–549, 2003.

[13] G. R. Hjaltason and H. Samet. Index-driven similarity search in metric spaces. *ACM Transactions on Database Systems*, 28(4):517–580, 2003.

[14] I. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986.

[15] E. J. Ong and R. Bowden. A boosted classifier tree for hand shape detection. In *Face and Gesture Recognition*, pages 889–894, 2004.

[16] P. J. Phillips, P. J. Flynn, T. Scruggs, K. W. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min, and W. Worek. Overview of the face recognition grand challenge. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 947–954, 2005.

[17] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGS for multiclass classification. In *NIPS*, pages 547–553, 2000.

[18] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

[19] Y. Sakurai, M. Yoshikawa, S. Uemura, and H. Kojima. The A-tree: An index structure for high-dimensional spaces using relative approximation. In *International Conference on Very Large Data Bases*, pages 516–526, 2000.

[20] R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[21] B. Stenger, A. Thayananthan, P. H. S. Torr, and R. Cipolla. Hand pose estimation using hierarchical detection. In *ECCV Workshop on Human Computer Interaction*, pages 105–116, 2004.

[22] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(5):854–869, 2007.

[23] V. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995.

[24] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *International Conference on Very Large Data Bases*, pages 194–205, 1998.

[25] P. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 311–321, 1993.