Shape Classification Through Structured Learning of Matching Measures

Longbin Chen
University of California
Santa Barbara, CA 93117
lbchen@cs.ucsb.edu

Julian J. McAuley
NICTA/ANU
Canberra, ACT 0200
julian.mcauley@nicta.com.au

Rogerio S. Feris
IBM T. J. Watson Research Center
Hawthorne, NY 10532
rsferis@us.ibm.com

Tibério S. Caetano
NICTA/ANU
Canberra, ACT 0200
tiberio.caetano@nicta.com.au

Matthew Turk
University of California
Santa Barbara, CA, 93117
mturk@cs.ucsb.edu

Abstract

Many traditional methods for shape classification involve establishing point correspondences between shapes to produce matching scores, which are in turn used as similarity measures for classification. Learning techniques have been applied only in the second stage of this process, after the matching scores have been obtained. In this paper, instead of simply taking for granted the scores obtained by matching and then learning a classifier, we learn the matching scores themselves so as to produce shape similarity scores that minimize the classification loss. The solution is based on a max-margin formulation in the structured prediction setting. Experiments in shape databases reveal that such an integrated learning algorithm substantially improves on existing methods.

1. Introduction

Shape classification through feature-matching scores has been an active research area in recent years [3, 2, 12]. Approaches in this category typically solve the shape classification problem by determining scores based on point correspondences between an input shape and a set of stored shape exemplars. The matching scores obtained are then used for classification in a second stage, which may involve learning [17]. However, the matching objective function is typically handcrafted or engineered in a non data-driven manner.

The key contribution of this paper is to show how the matching criterion itself can be optimized so that the final classification error delivered by matching scores is minimized. In other words, instead of performing learning only after matching scores have been obtained, we learn the matching scores themselves so that the classification loss is small. Figure 1 illustrates the comparison between traditional methods (top) and our proposed learning scheme (bottom). We embed both matching and classification in a unified learning framework, so that the alignment of two shapes is parametrized to produce matching scores with good discriminative power.

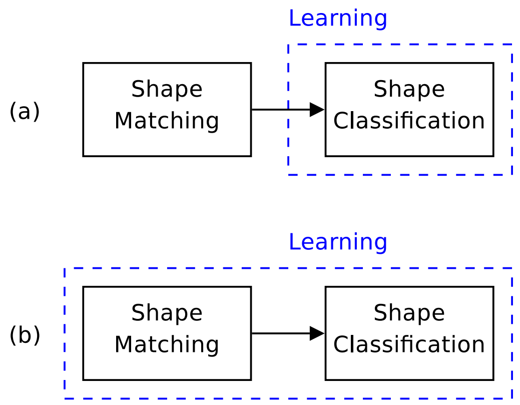


Figure 1. (a) Existing methods; learning happens after the matching scores have been obtained. (b) Our approach; both matching and classification are optimized within a unified learning scheme.

In order to accomplish this task, as part of our training set, we need not only have shapes with labeled classes, but also *labeled matches* between shapes of the same class. Our setting is similar to the one recently introduced in [5], but differs in that here we are concerned with minimizing the classification error, not the matching error *per se*. In our experiments, we will show how the expensive step of manually labeling matches can be avoided.

We will show that the problem of learning matching scores for classification leads to a non-convex optimization problem, which is very hard to solve. Our strategy is to make use of recent methods related to structured prediction, notably [15], enabling us to solve the problem in an elegant way, with good empirical results.

Our approach is independent of the shape alignment method used to compute the matching scores. This is an important property of our formulation, as one could choose *any* matching algorithm and apply our learning scheme to improve classification results.

2. Literature Review

Various methods have addressed the shape classification problem in the framework of deformable shape matching [3, 2, 17], often involving two distinct steps: 1) establishing point correspondences between shapes, and 2) using the scores obtained from this matching process as similarity measures for classification. A well-known technique that follows this methodology is *shape context matching* [2]. The idea consists of describing how each node in the shape "sees" the other nodes, by capturing the distribution of points in the surrounding region. Linear assignment is applied to solve the correspondence problem based on these features, with asymptotic time complexity $O(N^3)$. More efficient matching algorithms are based on representing the shapes as 1D sequences and using dynamic programming to do matching [12].

Berg et al. [3] formulates the correspondence problem as an integer quadratic programming problem, where the cost function has terms based on similarity of corresponding *geometric blur* point descriptors as well as the geometric distortion between pairs of corresponding feature points. Excellent results were obtained for the task of general object categorization. Casting shape matching as a quadratic assignment problem is NP-Hard, but efficient approximations have been proposed, like the spectral matching algorithm of [10]. More recently, Leordeanu et al. [11] showed that good recognition performance can be obtained by using only second-order geometric relationships between features, without relying on the local appearance of shape points.

Machine learning techniques have been applied for shape classification, but only after the matching scores have been obtained. Zhang and Malik [17] proposed a discriminative classifier which is learned based on shape context matching scores. Classification is done based on error-correcting output codes and a significant performance improvement is reported on the MNIST dataset. In more recent work, the authors apply a novel classifier called SVM-KNN [16] over shape and texture measurements to discriminate object categories.

Similar to our approach, Frome et al. [8] use a large-margin formulation to learn local distance functions for shape-based image retrieval and classification. The optimization criterion is based on the property that the distance between images of the same class should be less than the distance between images of different classes. Learning consists of determining weights for patch-based local distance functions, whereas the correspondences between image patches are taken for granted.

Recently, Caetano et al. [5] proposed to learn point-to-point shape correspondences using training pairs of shapes with manually labeled matchings. They reported a very meaningful result showing that linear assignment with such a learning scheme can match (or exceed) the performance of state-of-the-art quadratic assignment relaxations. In terms of methodology, this is the work most closely related to ours. The key novelty of our approach is that we are not concerned with minimizing the matching error between shapes, but the incurred classification loss. In our formulation, the quality of alignment itself is not important as long as the learned matching scores provide good discriminative power over shapes of different categories. Blaschko and Lampert proposed to embed a binary classifier and sliding window detection together in a unified learning framework to locate objects in images [4].

We want to stress that our formulation is not dependent on a specific shape matching or classification algorithm. It is a general formulation which allows *any* shape matching algorithm based on correspondences to be optimized to output matching scores that are meaningful for object class discrimination.

3. Matching-Based Shape Classification

Many shape classification algorithms based on matching scores follow a recipe of the following type:

1. Handcraft a feature-feature similarity measure.
2. For a given input shape, find the optimal correspondence (match) between the shape and every shape in a set of labeled reference shapes. The correspondence between two shapes is obtained by maximizing the *aggregate* feature-feature similarity between the shapes.
3. Use the score of this optimal correspondence as a similarity measure between the shapes.
4. Classify the shape based on this similarity measure.

A well-known algorithm which is an instance of this recipe is [17]. The above can be written in mathematical form as:

$$g(x; R, \theta) = f(s(x, r_1, \phi), \ldots, s(x, r_{|R|}, \phi); \theta) \quad (1)$$

where x is an input shape to be classified, $g(x; R, \theta)$ is the class to which the classifier assigns x and $s : \mathcal{X} \times \mathcal{X} \times \Phi \mapsto \mathbb{R}$ denotes some score function that returns the matching score of input test shape x against a specific shape r in a stored set of shapes R, when the features ϕ are used. The vector θ simply means that after each score $s(x, r_i, \phi)$ has been computed, it may be somehow weighted by θ in order to produce the final class to which x is going to be assigned, i.e., $g(x; R, \theta)$. The function f defines the classification rule, for example the class of the highest scored r_i (weighted by θ). The vector θ is estimated so as to minimize the misclassification rate in the training set (possibly with regularization).

The critical observation to be made here is that learning is only performed *after* the scores (s) have been obtained. If the scores are for any reason not reliable, they will serve as poor features to be later parametrized by θ. In other words, for bad matching scores, even a good learning algorithm may yield poor results. Presenting a solution to this problem is precisely the goal of this paper.

4. Learning Matching Scores for Classification

4.1. Basic Goal

We want to parametrize the matching score itself. The goal then is to perform machine learning such that the matching scores produced will naturally be able to discriminate classes. The classifier can therefore be written as

$$g(x; R, \theta) = f(s(x, r_1, \phi; \theta), \dots, s(x, r_{|R|}, \phi; \theta)). \quad (2)$$

Here s itself is parametrized. It is instructive to compare (eq. 2) against (eq. 1).

For our formulation, we will treat the function f as a K-*nearest-neighbor* (KNN) classifier – i.e., rather than choosing the class of the highest-scored r_i, we will choose the classes of the K shapes in R which result in the highest aggregate score. More precisely,

$$g(x; R, \theta) = \left[\text{class}(k_i^*) \middle| k^* = \underset{k \in \mathcal{K}(R,K)}{\text{argmax}} \sum_{i=1}^{K} s(x, k_i, \phi; \theta) \right], \quad (3)$$

where $\mathcal{K}(R, K)$ is the set of K-subsets of R.

Our training data is organized as follows: for each shape $x_i \in X$, we provide the labeled match $Y^{i,j}$ for each other shape x_j which belongs to the same class, as well as a collection of shapes belonging to *different* classes. Now, our n^{th} training instance is given by the tuple

$$\Bigg(\underbrace{x_n}_{\text{“probe” shape}} ; \underbrace{\{x_i \in X \backslash x_n | \text{class}(x_i) = \text{class}(x_n)\}}_{\text{shapes belonging to the same class}}; \underbrace{\{x_i \in X | \text{class}(x_i) \neq \text{class}(x_n)\}}_{\text{shapes belonging to different classes}}; \underbrace{Y^n}_{\text{matchings}} \Bigg).$$

For simplicity we assume that for each training instance, there are exactly K shapes belonging to the same class.[1]

[1]Were this not the case, we would choose some subset of size K; we may then include multiple training instances with the shape x_n, each using a different subset. For instance, when $K = 1$, we may consider every *pair* of shapes belonging to the same class as a separate training instance.

Our classifier will be trained so as to find a θ that makes objects of the same class more similar than objects of different classes. We will denote this tuple by $(x_n, x_n^{\text{same}}, x_n^{\text{diff}}, Y^n)$.[2]

Our goal now would simply be to find θ so as to minimize the classification loss, i.e.,

$$\theta^* = \underset{\theta}{\text{argmin}} \sum_{n=1}^{N} \Delta\left(x_n, g(x_n; R_n, \theta)\right). \quad (4)$$

where

$$\Delta(x, g) = \frac{1}{K} \sum_{i=1}^{K} 1_{\{\text{class}(x)\}}(g_i), \quad (5)$$

i.e., the proportion of the K neighbors that have the correct class label.[3] Note that our set R is different for each training instance n; here we have $R_n = x_n^{\text{same}} \cup x_n^{\text{diff}}$. For the sake of decreasing training time, we use only a subset of x^{diff} (see section 6).

In order to avoid overfitting, a regularization term may be added to the loss. Typical choices are L_2, L_1 and L_∞ regularizers. In our experiments we use an L_2 regularizer, so that (eq. 4) becomes

$$\theta^* = \underset{\theta}{\text{argmin}} \sum_{n=1}^{N} \Delta\left(x_n, g(x_n; R_n, \theta)\right) + \frac{\lambda}{2} \|\theta\|^2, \quad (6)$$

which is the problem we want to solve (λ is a regularization constant). We will now investigate in detail the parametrization that we use for $s(x, r, \phi; \theta)$, which is the only step left in order to have a final operational description of the optimization problem in (eq. 6).

4.2. The Model

It is left for us to specify the form of the score s of the *best match* between x and r. We assume that this score is given by features linearly parametrized by θ, i.e.,

$$s(x, r, \phi; \theta) = \max_{y} \langle \phi(x, r, y), \theta \rangle, \quad (7)$$

where y represents a putative correspondence between x and r; $y_{ij} = 1$ if $x_i \mapsto r_j$, $y_{ij} = 0$ otherwise (x_i here denotes the ith *point* belonging to the shape x). Moreover we typically enforce one-to-one matches, i.e., $\sum_i y_{ij} \leq 1$

[2]To briefly explain our notation: Y^i represents a *collection* of matches; $Y^{i,j}$ represents the jth entry in this collection. For the (i, j)th entry of the matrix itself, we use $y_{i,j}$.

[3]Alternately, we could use any loss function defined on the K neighbors; for example we could consider only the label of the *most popular* neighbor if we wanted.

and $\sum_j y_{ij} \leq 1$ (the inequalities allow for extra points in either x or r). The best match under θ is now given by

$$y^*(x, r, \phi; \theta) = \underset{y}{\operatorname{argmax}} \langle \phi(x, r, y), \theta \rangle . \tag{8}$$

In particular, we assume that the feature map $\phi(x, r, y)$ is additive on unary feature maps ψ involving individual pairs of points:

$$\phi(x, r, y) = \sum_{ij} \psi(x_i, r_j) y_{ij} \tag{9}$$

(our specific choice for ψ appears in section 5). Substituting (eq. 9) into (eq. 8), we obtain

$$y^*(x, r, \phi; \theta) = \underset{y}{\operatorname{argmax}} \overbrace{\sum_{ij} \underbrace{- \langle \psi(x_i, r_j), \theta \rangle}_{-c_{ij} \text{ (negative "cost")}} y_{ij}}^{\langle \phi(x,r,y), \theta \rangle} . \tag{10}$$

Note that this is a linear assignment problem in the general case. However, we would like to be able to exploit the fact that since we are matching *shapes*, the *order* information of feature points is known. In other words, if $x_i \mapsto r_j$, then $x_{i+1} \mapsto r_{j+1}$ or $x_{i+1} \mapsto r_{j-1}$. In fact this constraint is so convenient that it results in a quadratic time dynamic programming solution to (eq. 10), instead of the general cubic time solution for linear assignment (see [12] for details).[4]

Finally, under our K-nearest-neighbor formulation, we define the "score" (under θ) of a set of matches by

$$sc(x, k, Y^k; \theta) = \sum_{i=1}^{K} \left\langle \phi(x, k_i, Y^{k,i}), \theta \right\rangle \tag{11}$$

(here k is a set of K shapes, Y^k is a set of K matches). We will use k^*, Y^{k^*} to denote the maximizer (over k, Y^k) of this expression.

Learning now consists of solving (eq. 6) under the above model for s.

4.3. A Major Difficulty

We now discuss how difficult it is to solve (eq. 6). Note that there are finitely many K-subsets of R, and for each $r \in R$, there are a *finite* number of possible matches between x and r. Therefore, there are a finite number of matching scores (eq. 11) for a given pair (x, R). On the other hand, $\theta \in \mathbb{R}^d$, where d is the dimensionality of the parametrization. Since there are uncountably many θ's and only a finite number of score values, there are large equivalence classes of θ that produce the same $g(x; R, \theta)$, and therefore the same loss Δ. From the optimization point of view, this is very bad news because it means that Δ is piecewise constant on θ (non-convexity comes as a corollary); the added regularization term does not improve the situation, as it is convex and therefore the sum is still not convex and near-piecewise constant.

Our approach in this paper will follow the strategy proposed in [15] for the solution of analogous problems. The basic trick consists of constructing a convex optimization problem whose optimal solution is an upper bound on the loss.

4.4. The Convex Relaxation

In recent years Machine Learning researchers have found a way to solve optimization problems like those in (eq. 6), most notably in [15].

It is possible to obtain a convex relaxation for this optimization problem by means of the maximum margin classification criterion [15]. First, consider the constraints we aim to satisfy, i.e.,

$$\begin{aligned} &sc(x_n, x_n^{\text{same}}, Y^n; \theta) \geq sc(x_n, k, Y^k; \theta) \\ &\text{for all } n, k \in \mathcal{K}(R_n, K), \text{ and } Y^k \in \mathcal{Y}^k \end{aligned} \tag{12}$$

(here $\mathcal{Y}^n$ denotes the space of all possible matches between x_n and shapes in $\mathcal{K}(R, K)$). In words: for the n^{th} training sample, we want the score for the labeled matches (Y^n) between shapes of the same class (x_n and x_n^{same}) to be no smaller than the score for any possible matches ($Y^k \in \mathcal{Y}^k$) between other possible subsets of the shapes ($k \in \mathcal{K}(R, K)$). If the problem is separable, one needs to regularize the solution since there are infinitely many solutions. One option is to use the intuitive margin-maximization principle, which in addition has theoretical generalization guarantees and in practice consists of minimizing the squared norm of the parameter vector under the constraints in (eq. 12) relaxed by a margin of 1. For non-separable problems slacks must be added to the constraints and penalized in the objective function. The resulting formulation is

$$\underset{\theta, \xi}{\text{minimize}} \ \frac{1}{N} \sum_{n=1}^{N} \xi_n + \frac{\lambda}{2} \|\theta\|^2 \tag{13a}$$

subject to

$$\begin{aligned} &sc(x_n, x_n^{\text{same}}, Y^n; \theta) - sc(x_n, k, Y^k; \theta) \geq \Delta(x_n, k) - \xi_n \\ &\text{for all } n, k \in \mathcal{K}(R_n, K), \text{ and } Y^k \in \mathcal{Y}^k \end{aligned} \tag{13b}$$

Note however that the problem we wanted to solve was that of minimizing the regularized risk, i.e., (eq. 6). The following lemma makes the connection between the two optimization problems (eq. 6) and (eq. 13):

[4]The approach presented in [12] assumes that the first point in each shape is aligned (or that we can "guess" this alignment easily), while their solution becomes cubic if this assumption cannot be made. It is safe to use the quadratic time version if, for instance, our shapes are not subject to rotations (as happens to be the case in our experiments). Of course, we could easily resort to the cubic time version were this not the case, though doing inference would obviously be slower.

Lemma 4.1. *The optimal solution (ξ^*, θ) in (eq. 13) is such that $\xi_n^* \geq \Delta(x_n, g(x_n; R, \theta))$.*

Proof. The constraint in (eq. 13b) holds for every k, Y^k, so in particular it holds for Y^{k^*}. We then obtain the inequality $sc(x_n, x_n^{\text{same}}, Y^n; \theta) - sc(x_n, k^*, Y^{k^*}; \theta) \geq \Delta(x_n, k^*) - \xi_n^*$. But from (eq. 11) we see that Y^{k^*} is the maximizer of the $sc(\cdot)$ function, therefore the LHS of the inequality cannot be positive. This implies $\xi_n^* \geq \Delta(x_n, g(x_n; R, \theta))$. $\square$

What this lemma (essentially due to [15]) implies is that, since (eq. 13) tries to minimize $\sum_n \xi_n$, it will also bring down the aggregated loss $\sum_n \Delta(x_n, g(x_n; R, \theta))$.

We now have a convex relaxation, but the optimization problem (eq. 13) still has an exponential number of constraints. To address this issue, we proceed with a cutting-plane method, which consists of a systematic search for a small set of critical constraints, which will be the only ones to be ultimately enforced [15].

4.5. The BMRM Algorithm

We use the "Bundle Methods for Regularized Risk Minimization" (BMRM) solver of [14], which merely requires that for each candidate θ, we compute the difference in gradient (w.r.t. θ) of the score function of the *true* assignments (x_n^{same}), and the *most violated* constraint (k^{viol}, $Y^{k^{\text{viol}}}$):

$$\phi(x_n, x_n^{\text{same}}, Y^n) - \phi(x_n, k^{\text{viol}}, Y^{k^{\text{viol}}}), \tag{14}$$

and also the loss ($\frac{1}{N} \sum_n \Delta(x_n, k^{\text{viol}})$). See [14] for further details (for more explanation of BMRM, see Algorithm 1 in [14], which we have omitted due to space constraints).

Clearly the critical step consists of finding the most violated constraint for the current solution of the optimization problem. Note that the most violated constraint for the n^{th} observation and current solution θ is the one which maximizes ξ_n. Therefore, from (eq. 13b), we need to solve

$$\underset{k, Y^k}{\operatorname{argmax}} \big[sc(x_n, R_n, Y^k; \theta) + \Delta(x_n, k) \big] \tag{15}$$

as this is the k, Y^k for which the constraint (eq. 13b) is tightest. Note that, algorithmically, this problem is precisely the same as the matching problem and can be solved in quadratic time using dynamic programming [12].

5. Implementation Details

In this section we describe the implementation details of our method, including our features, shape matching algorithm, and the way we labeled matches in the training set.

5.1. Features and Labeling Samples

Each shape is represented as a set of evenly spaced points along the contour of an object, for which shape descriptors are extracted. Consider a specific point O along the shape, with four immediate point predecessors p_1, p_2, p_3, p_4 and four immediate point successors q_1, q_2, q_3, q_4. $p_1 O q_1$ forms a *turning angle* (TA) for that specific feature point. To make the feature set more robust to scale variance, we also use $p_2 O q_2$, $p_3 O q_3$, and $p_4 O q_4$ as turning angles. Therefore, we extract four TAs for each point in the shape as part of our feature set.

We have also used a feature that we call *distance across the shape* (DAS) to describe each point along the shape. Consider the interior bisector of the angle $p_1 O q_1$, which intersects the contour at point O'. The length of OO' is the distance across the shape (DAS) at point O. If the bisector intersects with the shape multiple times, the distance to the closest intersection is used. In case the exterior bisection intersects with the contour, the distance to the intersection is the exterior distance across the shape (EDAS).

Finally, we used the *inner-distance shape context* (IDSC) feature at each contour point, which was proposed by Huang et al. [12]. This feature has proven to be very useful to match shapes under significant non-linear deformations.

Manually labeling point-to-point correspondences between shapes of the same class in the training set is time-demanding. We provide a semi-automatic way to alleviate this problem. By providing manual correspondences for a few salient feature points, we can use the dynamic programming matching algorithm to *complete* the labeling in quadratic time. In fact, we found that this was necessary only for the "difficult" shapes in our dataset, while the remainder may be labeled automatically. Although this kind of noise in our training set would present a major issue if we were learning *matching* scores, we found that this "noisy" data is nevertheless useful when we are learning *classification* scores.

5.2. Shape Matching Parametrization

Since we have a linear assignment criterion to be optimized *in addition* to the order constraint, we can solve the problem using a simple dynamic programming algorithm of quadratic time complexity, as presented in [12]. We implemented the algorithm in [12] in order to obtain the optimal solution of the optimization problem (eq. 8). Note that we apply this algorithm both at test time (to predict the class of a new shape instance) and at training time (to generate the most violated constraint).

Once the feature vectors for every point are obtained, the cost features between pairs of points are computed. For point a_i in one shape, point b_j in the other shape, and their feature vectors $(A_1, \ldots, A_F)$, $(B_1, \ldots, B_F)$ (F is the feature dimension), we define the feature vector

$$\Phi(a_i, b_j) = \left(\frac{(A_1 - B_1)^2}{|A_1| + |B_1|}, \ldots, \frac{(A_F - B_F)^2}{|A_F| + |B_F|} \right). \tag{16}$$

We also define a dummy feature for a point not having a corresponding match in the other shape, in which case a dummy variable d is introduced in each shape and we have $\Phi(a_i, d) := (1, |A_1|, \ldots, |A_F|)$ and $\Phi(d, b_j) := (1, |B_1|, \ldots, |B_F|)$. Linearly parametrizing $\psi(a_i, b_j)$, we then obtain the final cost c_{ij} as an inner product:

$$c_{ij} = \langle \psi(a_i, b_j), (\theta_1, \ldots, \theta_{2F+1}) \rangle , \tag{17}$$

where we define the vector

$$\psi(a_i, b_j) := (\Phi(a_i, b_j); \underbrace{0, \ldots, 0}_{F+1 \text{ times}}) \tag{18}$$

$$\psi(a_i, d) := (\underbrace{0, \ldots, 0}_{F \text{ times}}; \Phi(a_i, d)) \text{ and} \tag{19}$$

$$\psi(d, b_j) := (\underbrace{0, \ldots, 0}_{F \text{ times}}; \Phi(d, b_j)). \tag{20}$$

Note in particular that if the parameter coefficients θ_i are all equal, then c_{ij} reduces to the standard similarity measure as used in [2, 12].

6. Experiments

6.1. MPEG-7 Dataset

The MPEG7 dataset is frequently used to evaluate shape matching and recognition algorithms. This shape database contains 70 shape categories, each of which has 20 samples with in-plane rotations, articulations, and occlusions.

Learning within the same shape category

In this experiment we want to evaluate the capability of our algorithm to discriminate one particular object category from all other object categories.

We divided the MPEG-7 dataset into 70 subsets, each of which contains objects belonging to a particular category. We further divided each subset into three parts: 5 training shapes (i.e., 10 unique correspondences), 5 validation shapes, and 10 test shapes. The purpose of our validation set is to choose the value of our regularization constant, λ. Thus, we train our algorithm using the *training* set, and report the error on our *test* set, for whichever choice of λ results in the lowest error on our *validation* set.

Denoting the j^{th} shape in the i^{th} category by $x_{i,j}$, our training set for the experiment on category i becomes

$$\Big\{ \{ x_{i,j}, \underbrace{\{x_{i,k} | k \neq j\}}_{\text{same category}}, \underbrace{\{x_{m,l} | m \neq i\}}_{\text{different categories}}, Y^i \} \Big| j, k, l \in 1 \ldots 5 \Big\}.$$

For validation and testing, we simply compare each shape to *all* other shapes in the dataset, and compute the K-nearest-neighbors. Since our training sets have five elements, we have $K = 4$ in this case.[5] When searching for a max-violator for a match against $x_{i,k}$ (see Section 4.5)), we only consider shapes with the same index k (this is done simply to decrease training time).

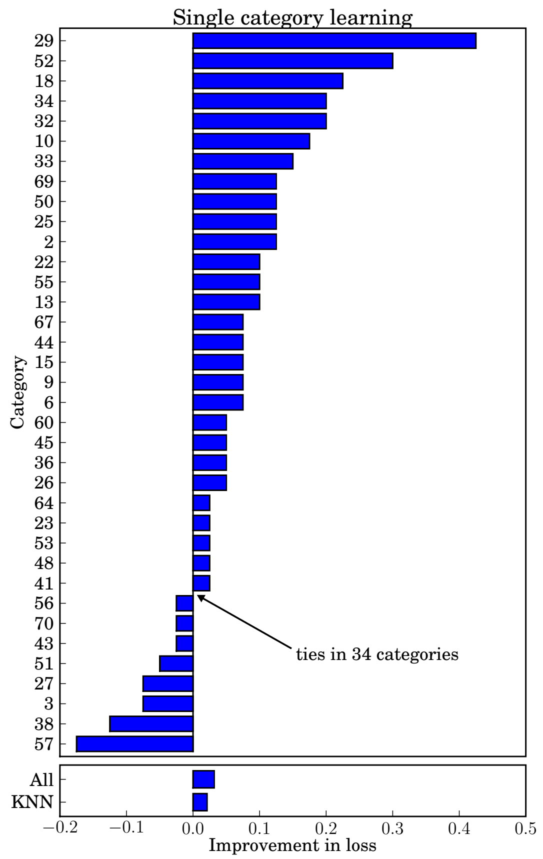


Figure 2. Results - Learning within the same shape category. Our method underperforms in 8 categories, ties in 34 categories, and achieves an improvement in 28 categories. Categories in which we achieve a tie are not shown. The bottom of the figure shows learning across all categories ("All"), and a model which learns classification scores only ("KNN").

We compare our retrieval results on each category with the IDSC+DP algorithm [12, 7] (this is in fact the non-learning version of our approach). Figure 2 shows the improvement of learning versus non-learning using our approach (i.e., the loss incurred before learning minus the loss incurred after learning). Our method underperforms in 8 categories, ties in 34 categories, and achieves an improvement in the remaining 28 categories.

Figure 3 analyzes these success and failure cases: learn-

[5]Note that in this experiment, our learning algorithm only explicitly minimizes false-negatives (i.e., shapes in the current category being identified as belonging to a different category). At the expense of training time, we could of course include negative training data as well, so as to minimize false-positives. However, we found in practice that the number of false-positives was not adversely effected by this learning scheme.

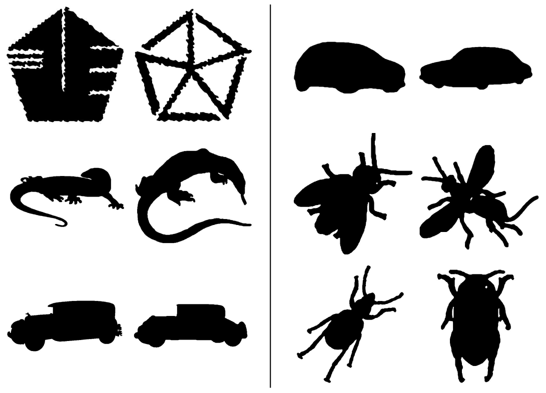


Figure 3. Left: the three categories in which our method achieves the greatest improvement. Right: the three categories in which our method achieves the least improvement (i.e., in which non-learning outperforms learning). The left images seem to be subject to noise and occlusions (which our matching algorithm handles well), whereas the right images seem to be subject to rotation and deformation (which our matching algorithm handles poorly).

ing seems to perform well when the shapes are subject to noise and conclusions, but poorly when subject to rotations and deformations. This is probably in part due to the small size of our training set (meaning that the test set may be very different), and due to the fact that we are not using the rotationally invariant version of the dynamic programming matching algorithm of [12] (though as mentioned, we could handle this case at the expense of increased running time).

Learning across different shape categories

Similarly, we can learn a single model which is able to discriminate all 70 shape categories. This is done simply by combining the 70 training sets from the previous experiment into a single set (similarly for the validation and test sets). Results are shown at the bottom of Figure 2 ("All"). Specifically, the algorithm achieves an error of 15.9% before learning, and 12.7% after learning.

Finally, we compare our results to an algorithm which learns classification scores only – i.e., in the theme of Figure 1 (a). This is done in a framework very similar to that already described, the only difference being that matching scores are fixed, and we parametrize only classification scores. That is, we learn a 70 dimensional weight vector (for our 70 object categories); when performing nearest-neighbor classification, the distance from a particular shape is scaled by the weight for that shape's category. Without learning, this is identical to the previous model, but only achieves an error of 13.8% after learning. Although learning improves over non-learning in both models, the *benefit* of learning is increased by tuning the matching scores.

Figure 4 shows an example match before and after learning; Figure 5 shows the weight vector for these two models.

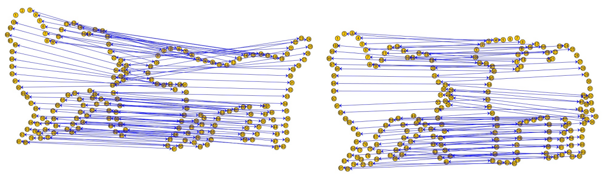
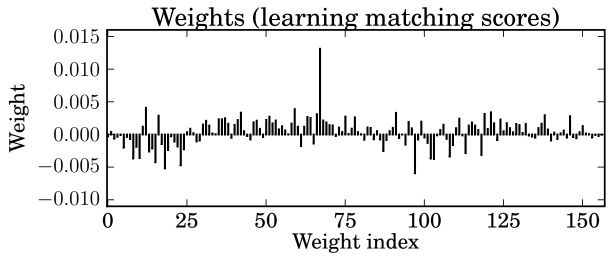


Figure 4. Left: classification without weights; the match with the highest score belongs to the incorrect category. Right: classification with weights; the categories are the same.



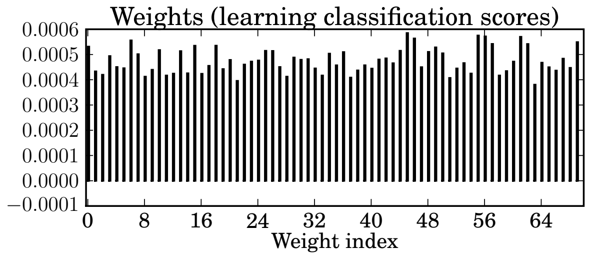


Figure 5. Top: weights when learning matching scores; the first half of the weights correspond to matching costs, the second half to occlusion costs (see Section 5.2). Bottom: weights when learning classification scores.

6.2. MNIST Dataset

For our second experiment, we do matching on the MNIST dataset [9], which consists of $70,000$ handwritten digits ($60,000$ for training and $10,000$ for testing). We represent a digit as a set of contours, and use the Smith-Waterman algorithm (see [13, 6]) to match multiple contours between two shapes, using the same features as in the previous section. It is almost impossible to use all of the images in the training set (due to running time), so we select the 20 most similar shapes from each category (based on the Euclidean distance) for each probe shape.

The first experiment is to build a model to classify all digits, which is similar to our "Learning across different categories" experiment in Section 6.1. We randomly select 10 samples along with their neighboring shapes for training; the shape matching score is combined with the Euclidean distance to classify digits. The second experiment is similar to the "Learning within the same shape category" experiment in Section 6.1 – instead of learning a general model for all shapes, we learn a shape-specific model for every probe shape using its neighboring shapes. This idea is similar to

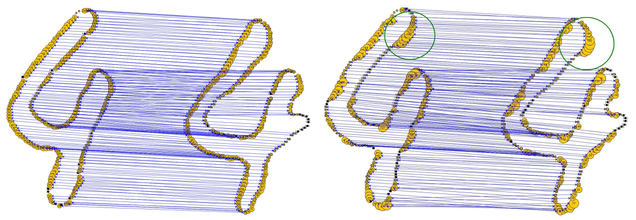


Figure 6. Left: matching using the general model. Right: matching using the shape-specific model; the point sizes indicate the matching scores (see [13]).

the KNN-SVM method [16], which achieves a significant improvement over generic SVMs. The Smith-Waterman algorithm is used to automatically generate matchings between training shapes (so that we don't need to manually label the training samples).

Figure 6 shows an example where the shape-specific model outperforms the general model. The general shape model misclassifies the probe shape as "9", whereas the shape-specific model is able to distinguish them. The circles on the right of Figure 6 show that the matching scores are large for these points, which appear to be critical in distinguishing "4" from "9". This example shows our method's strength in that we aim to optimize the matching score for classification, not for matching itself.

Overall, the shape matching method without learning gives an error of (1.84%), which is smaller than KNN with L_2 distance (3.09%, [1]). The error rate of the learned general model depends on the amount of training data and on model parameters: with regularization constant $C = 0.001$, error bound $e = 0.1$, using 3-NN classifiers, the classification error rate is 0.88%. We also perform a comparison with the shape context matching method of [2]. Among 63 misclassified samples by [2], only 32 are misclassified by our non-learning shape matching method and only 28 are missclassified by our learned general model. This shows that our method is complimentary to the shape matching method (and possibly others), and by combining them, it is likely to build an even stronger classifier.

The shape-specific classifiers are more computationally intensive so we apply this method only for probe shapes where the Euclidean distance and matching score give conflicting predictions. There are a total of 236 samples that need a shape-specific classifier, 210 of which are classified correctly by it. The overall error rate when applying shape-specific learning is 0.58%, which is comparable to state-of-the-art methods (see [1]).

7. Conclusion

We have presented a novel approach that embeds shape matching and classification in a unified learning scheme. Instead of relying on handcrafted or engineered matching functions, we learn point matching measures via structured estimation with the goal of minimizing the classification loss. Our method can be applied to improve any shape classification algorithm based on correspondences.

References

[1] `http://yann.lecun.com/exdb/mnist/`.

[2] J. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. on PAMI*, 24(4):509–522, 2002.

[3] A. Berg, T. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *CVPR*, 2005.

[4] M. B. Blaschko and C. H. Lampert. Learning to localize objects with structured output regression. In *ECCV*, 2008.

[5] T. S. Caetano, L. Cheng, Q. Le, and A. J. Smola. Learning graph matching. In *ICCV*, 2007.

[6] L. Chen, R. Feris, and M. Turk. Efficient partial shape matching using smith-waterman algorithm. In *CVPR workshop on Non-Rigid Shape Analysis and Deformable Image Alignment*, 2008.

[7] P. Felzenszwalb. Hierarchical matching of deformable shapes. In *CVPR*, 2007.

[8] A. Frome, Y. Singer, F. Sha, and J. Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *ICCV*, 2007.

[9] Y. Lecun, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[10] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, 2005.

[11] M. Leordeanu, M. Hebert, and R. Sukthankar. Beyond local appearance: Category recognition from pairwise interactions of simple features. In *CVPR*, 2007.

[12] H. Ling and D. Jacobs. Shape classification using the inner-distance. *IEEE Trans. on PAMI*, 29(2):286–299, 2007.

[13] T. Smith and M. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[14] C. Teo, Q. Le, A. Smola, and S. Vishwanathan. A scalable modular convex solver for regularized risk minimization. In *KDD*, 2007.

[15] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 6:1453–1484, 2005.

[16] H. Zhang, A. Berg, M. Maire, and J. Malik. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *CVPR*, 2006.

[17] H. Zhang and J. Malik. Learning a discriminative classifier using shape context distances. In *CVPR*, 2003.