Learning Graph Matching

Tibério S. Caetano, Julian J. McAuley, Li Cheng, Quoc V. Le and Alex J. Smola*

February 19, 2013

arXiv:0806.2890v1 [cs.CV] 17 Jun 2008

Abstract

As a fundamental problem in pattern recognition, graph matching has applications in a variety of fields, from computer vision to computational biology. In graph matching, patterns are modeled as graphs and pattern recognition amounts to finding a correspondence between the nodes of different graphs. Many formulations of this problem can be cast in general as a quadratic assignment problem, where a linear term in the objective function encodes node compatibility and a quadratic term encodes edge compatibility. The main research focus in this theme is about designing efficient algorithms for approximately solving the quadratic assignment problem, since it is NP-hard. In this paper we turn our attention to a different question: how to estimate compatibility functions such that the solution of the resulting graph matching problem best matches the expected solution that a human would manually provide. We present a method for *learning graph matching*: the training examples are pairs of graphs and the 'labels' are matches between them. Our experimental results reveal that learning can substantially improve the performance of standard graph matching algorithms. In particular, we find that simple linear assignment with such a learning scheme outperforms Graduated Assignment with bistochastic normalisation, a state-of-the-art quadratic assignment relaxation algorithm.

1 Introduction

Graphs are commonly used as abstract representations for complex structures, including DNA sequences, documents, text, and images. In particular they are extensively used in the field of computer vision, where many problems can be formulated as an attributed graph matching problem. Here the nodes of the graphs correspond to local features of the image and edges correspond to relational aspects between features (both nodes and edges can be attributed, i.e. they can encode feature vectors). Graph matching then consists of finding a correspondence between nodes of the two graphs such that they 'look most similar' when the vertices are labeled according to such a correspondence.

Typically, the problem is mathematically formulated as a quadratic assignment problem, which consists of finding the assignment that maximizes an objective function encoding local compatibilities (a linear term) and structural compatibilities (a quadratic term). The main body of research in graph matching has then been focused on devising more accurate and/or faster algorithms to solve the problem approximately (since it is NP-hard); the compatibility functions used in graph matching are typically handcrafted.

An interesting question arises in this context: If we are given two attributed graphs to match, G and G', should the optimal match be uniquely determined? For example, assume first that G and G' come from two images acquired by a surveillance camera in an airport's lounge; now, assume the same G and G' instead come from two images in a photographer's image database; should the optimal match be the same in both situations? If the algorithm takes into account exclusively the graphs to be matched, the optimal solutions will be the same¹ since the graph pair is the same in both cases. This is the standard way graph matching is approached today.

In this paper we address what we believe to be a limitation of this approach. We argue that if we know the 'conditions' under which a pair of graphs has been extracted, then we should take into account how graphs arising in those conditions are typically matched. However, we do not take the information on the conditions explicitly into account, since this would obviously be impractical. Instead, we approach the problem purely from a statistical inference perspective. First, we extract graphs from a number of images acquired under the same conditions as those for which we want to solve, whatever the word 'conditions' means (e.g. from the surveillance camera or the photographer's database). We then *manually* provide what we understand to be the optimal matches between the resulting graphs. This information is then used in a *learning* algorithm which learns a map from the space of pairs of graphs to the space of matches.

In terms of the quadratic assignment problem, this learning algorithm amounts to (in loose language) adjusting the node and edge compatibility functions such that the expected optimal match in a test pair of graphs agrees with the expected match they would have had, had they been in the training set. In this formulation, the learning problem consists of a convex, quadratic program which is readily solvable by means of a column generation procedure.

We provide experimental evidence that applying learning to standard graph matching algorithms significantly improves their performance. In fact, we show that learning improves upon non-learning results so dramatically that linear assignment *with learning* outperforms Graduated As-

^{*}Tibério Caetano, Julian McAuley, Li Cheng, and Alex Smola are with the Statistical Machine Learning Program at NICTA, and the Research School of Information Sciences and Engineering, Australian National University. Quoc Le is with the Department of Computer Science, Stanford University.

 $^{^1\}mathrm{Assuming}$ there is a single optimal solution and that the algorithm finds it.

signment with bistochastic normalisation, a state-of-the-art quadratic assignment relaxation algorithm. Also, by introducing learning in Graduated Assignment itself, we obtain results that improve both in accuracy and speed over the best existing quadratic assignment relaxations.

A preliminary version of this paper appeared in [1].

1.1 Related Literature

The graph matching literature is extensive, and many different types of approaches have been proposed, which mainly focus on approximations and heuristics for the quadratic assignment problem. An incomplete list includes spectral methods [2–6], relaxation labeling and probabilistic approaches [7–13], semidefinite relaxations [14], replicator equations [15], tree search [16], and graduated assignment [17]. Spectral methods consist of studying the similarities between the spectra of the adjacency or Laplacian matrices of the graphs and using them for matching. Relaxation and probabilistic methods define a probability distribution over mappings, and optimize using discrete relaxation algorithms or variants of belief propagation. Semidefinite relaxations solve a convex relaxation of the original combinatorial problem. Replicator equations draw an analogy with models from biology where an equilibrium state is sought which solves a system of differential equations on the nodes of the graphs. Tree-search techniques in general have worst case exponential complexity and work via sequential tests of compatibility of local parts of the graphs. Graduated Assignment combines the 'softassign' method [18] with Sinkhorn's method [19] and essentially consists of a series of first-order approximations to the quadratic assignment objective function. This method is particularly popular in computer vision since it produces accurate results while scaling reasonably in the size of the graph.

The above literature strictly focuses on trying better *algorithms* for approximating a solution for the graph matching problem, but does not address the issue of how to determine the compatibility functions in a principled way.

In [20] the authors learn compatibility functions for the relaxation labeling process; this is however a different problem than graph matching, and the 'compatibility functions' have a different meaning. Nevertheless it does provide an initial motivation for learning in the context of matching tasks. In terms of methodology, the paper most closely related to ours is possibly [21], which uses structured estimation tools in a quadratic assignment setting for word alignment. A recent paper of interest shows that *very* significant improvements on the performance of graph matching can be obtained by an appropriate *normalization* of the compatibility functions [22]; however, no learning is involved.

2 The Graph Matching Problem

The notation used in this paper is summarized in table 1. In the following we denote a graph by G. We will often refer to a *pair* of graphs, and the second graph in the pair will be denoted by G'. We study the general case of *attributed* graph matching, and attributes of the vertex i and the edge

 Table 1: Definitions and Notation

- G generic graph (similarly, G');
- G_i attribute of node *i* in *G* (similarly, $G'_{i'}$ for G');
- G_{ij} attribute of edge ij in G (similarly, $G'_{i'j'}$ for G');
- \mathcal{G} space of graphs ($\mathcal{G} \times \mathcal{G}$ space of pairs of graphs);
- x generic observation: graph pair (G, G'); $x \in \mathcal{X}$, space of observations;
- y generic label: matching matrix; $y \in \mathcal{Y}$, space of labels;
- \boldsymbol{n} index for training instance; N number of training instances;
- x^n n^{th} training observation: graph pair (G^n, G'^n) ;
- y^n n^{th} training label: matching matrix;
- g predictor function;
- y^w optimal prediction for g under w;
- f discriminant function;
- Δ loss function;
- Φ joint feature map;
- ϕ_1 node feature map;
- ϕ_2 edge feature map;
- S_n constraint set for training instance n;
- y^* solution of the quadratic assignment problem;

 \hat{y} - most violated constraint in column generation;

- $y_{ii'}$ i^{th} row and i'^{th} column element of y;
- $c_{ii'}$ value of compatibility function for map $i \mapsto i'$;
- $d_{ii'jj'}$ value of compatibility function for map $ij \mapsto i'j'$;
- ϵ tolerance for column generation;
- w_1 node parameter vector; w_2 edge parameter vector;
- $w := [w_1 \ w_2]$ joint parameter vector; $w \in \mathcal{W}$;

 ξ_n - slack variable for training instance n;

 Ω - regularization function; λ - regularization parameter;

 δ - convergence monitoring threshold in bistochastic normalization.

ij in G are denoted by G_i and G_{ij} respectively. Standard graphs are obtained if the node attributes are empty and the edge attributes $G_{ij} \in \{0,1\}$ are binary denoting the absence or presence of an edge, in which case we get the so-called *exact* graph matching problem.

Define a matching matrix y by $y_{ii'} \in \{0,1\}$ such that $y_{ii'} = 1$ if node i in the first graph maps to node i' in the second graph $(i \mapsto i')$ and $y_{ii'} = 0$ otherwise. Define by $c_{ii'}$ the value of the compatibility function for the unary assignment $i \mapsto i'$ and by $d_{ii'jj'}$ the value of the compatibility function for the pairwise assignment $ij \mapsto i'j'$. Then, a generic formulation of the graph matching problem consists of finding the optimal matching matrix y^* given by the solution of the following (NP-hard) quadratic assignment problem [23]

$$y^* = \underset{y}{\operatorname{argmax}} \left[\sum_{ii'} c_{ii'} y_{ii'} + \sum_{ii'jj'} d_{ii'jj'} y_{ii'} y_{jj'} \right], \quad (1)$$

typically subject to either the injectivity constraint (oneto-one, that is $\sum_{i} y_{ii'} \leq 1$ for all $i', \sum_{i'} y_{ii'} \leq 1$ for all i) or simply the constraint that the map should be a function (many-to-one, that is $\sum_{i'} y_{ii'} = 1$ for all *i*). If $d_{ii'jj'} = 0$ for all ii'jj' then (1) becomes a linear assignment problem, exactly solvable in worst case cubic time [24]. Although the compatibility functions c and d obviously depend on the attributes $\{G_i, G'_{i'}\}$ and $\{G_{ij}, G'_{i'j'}\}$, the functional form of this dependency is typically assumed to be fixed in graph matching. This is precisely the restriction we are going to relax in this paper: both the functions c and d will be parametrized by vectors whose coefficients will be learned within a convex optimization framework. In a way, instead of proposing yet another algorithm for determining how to approximate the solution for (1), we are aiming at finding a way to determine what should be maximized in (1), since different c and d will produce different criteria to be maximized.

3 Learning Graph Matching

3.1 General Problem Setting

We approach the problem of learning the compatibility functions for graph matching as a supervised learning problem [25]. The training set comprises N observations x from an input set \mathcal{X} , N corresponding labels y from an output set \mathcal{Y} , and can be represented by $\{(x^1; y^1), \ldots, (x^N; y^N)\}$. Critical in our setting is the fact that the observations and labels are structured objects. In typical supervised learning scenarios, observations are vectors and labels are elements from some discrete set of small cardinality, for example $y^n \in \{-1, 1\}$ in the case of binary classification. However, in our case an observation x^n is a pair of graphs, i.e. $x^n = (G^n, G'^n)$, and the label y^n is a match between graphs, represented by a matching matrix as defined in section 2.

If $\mathfrak{X} = \mathfrak{G} \times \mathfrak{G}$ is the space of pairs of graphs, \mathfrak{Y} is the space of matching matrices, and \mathcal{W} is the space of parameters of our model, then learning graph matching amounts to estimating

a function $g: \mathcal{G} \times \mathcal{G} \times \mathcal{W} \mapsto \mathcal{Y}$ which minimizes the prediction loss on the test set. Since the test set here is assumed not to be available at training time, we use the standard approach of minimizing the empirical risk (average loss in the training set) plus a regularization term in order to avoid overfitting. The optimal predictor will then be the one which minimizes an expression of the following type:

$$\underbrace{\frac{1}{N}\sum_{n=1}^{N}\Delta(g(G^n, G'^n; w), y^n)}_{\text{empirical risk}} + \underbrace{\lambda\Omega(w)}_{\text{regularization term}}, \quad (2)$$

where $\Delta(g(G^n, G'^n; w), y^n)$ is the loss incurred by the predictor g when predicting, for training input (G^n, G'^n) , the output $g(G^n, G'^n; w)$ instead of the training output y^n . The function $\Omega(w)$ penalizes 'complex' vectors w, and λ is a parameter that trades off data fitting against generalization ability, which is in practice determined using a validation set. In order to completely specify such an optimization problem, we need to define the parametrized class of predictors g(G, G'; w) whose parameters w we will optimize over, the loss function Δ and the regularization term $\Omega(w)$. In the following we will focus on setting up the optimization problem by addressing each of these points.

3.2 The Model

We start by specifying a *w*-parametrized class of predictors g(G, G'; w). We use the standard approach of discriminant functions, which consists of picking as our optimal estimate the one for which the discriminant function f(G, G', y; w) is maximal, i.e. $g(G, G'; w) = \operatorname{argmax}_y f(G, G', y; w)$. We assume linear discriminant functions $f(G, G', y; w) = \langle w, \Phi(G, G', y) \rangle$, so that our predictor has the form

$$g(G, G', w) = \operatorname*{argmax}_{y \in \mathcal{Y}} \langle w, \Phi(G, G', y) \rangle.$$
(3)

Effectively we are solving an inverse optimization problem, as described by [25, 26], that is, we are trying to find fsuch that g has desirable properties. Further specification of g(G, G'; w) requires determining the joint feature map $\Phi(G, G', y)$, which has to encode the properties of both graphs as well as the properties of a match y between these graphs. The key observation here is that we can relate the quadratic assignment formulation of graph matching, given by (1), with the predictor given by (3), and interpret the solution of the graph matching problem as being the estimate of g, i.e. $y^w = g(G, G'; w)$. This allows us to interpret the discriminant function in (3) as the objective function to be maximized in (1):

$$\langle \Phi(G, G', y), w \rangle = \sum_{ii'} c_{ii'} y_{ii'} + \sum_{ii'jj'} d_{ii'jj'} y_{ii'} y_{jj'}.$$
 (4)

This clearly reveals that the graphs and the parameters must be encoded in the compatibility functions. The last step before obtaining Φ consists of choosing a parametrization for the compatibility functions. We assume a simple linear parametrization

$$c_{ii'} = \langle \phi_1(G_i, G'_{i'}), w_1 \rangle \tag{5a}$$

$$d_{ii'jj'} = \left\langle \phi_2(G_{ij}, G'_{i'j'}), w_2 \right\rangle, \tag{5b}$$

i.e. the compatibility functions are linearly dependent on the parameters, and on new feature maps ϕ_1 and ϕ_2 that only involve the graphs (section 4 specifies the feature maps ϕ_1 and ϕ_2). As already defined, G_i is the attribute of node *i* and G_{ij} is the attribute of edge ij (similarly for G'). However, we stress here that these are not necessarily *local* attributes, but are arbitrary features *simply indexed* by the nodes and edges.² For instance, we will see in section 4 an example where G_i encodes the graph structure of G as 'seen' from node *i*, or from the 'perspective' of node *i*.

Note that the traditional way in which graph matching is approached arises as a particular case of equations (5): if w_1 and w_2 are constants, then $c_{ii'}$ and $d_{ii'jj'}$ depend only on the features of the graphs. By defining $w := [w_1 w_2]$, we arrive at the final form for $\Phi(G, G', y)$ from (4) and (5):

$$\Phi(G, G', y) = \left[\sum_{ii'} y_{ii'} \phi_1(G_i, G'_{i'}), \sum_{ii'jj'} y_{ii'} y_{jj'} \phi_2(G_{ij}, G'_{i'j'})\right].$$
 (6)

Naturally, the final specification of the predictor g depends on the choices of ϕ_1 and ϕ_2 . Since our experiments are concentrated on the computer vision domain, we use typical computer vision features (e.g. Shape Context) for constructing ϕ_1 and a simple edge-match criterion for constructing ϕ_2 (details follow in section 4).

3.3 The Loss

Next we define the loss $\Delta(y, y^n)$ incurred by estimating the matching matrix y instead of the correct one, y^n . When both graphs have large sizes, we define this as the fraction of mismatches between matrices y and y^n , i.e.

$$\Delta(y, y^n) = 1 - \frac{1}{\|y^n\|_F^2} \sum_{ii'} y_{ii'} y_{ii'}^n.$$
(7)

(where $\|\cdot\|_F$ is the Frobenius norm). If one of the graphs has a small size, this measure may be too rough. In our experiments we will encounter such a situation in the context of matching in images. In this case, we instead use the loss

$$\Delta(G, G', \pi) = 1 - \frac{1}{|\pi|} \sum_{i} \left[\frac{d(G_i, G'_{\pi(i)})}{\sigma} \right].$$
 (8)

Here graph nodes correspond to point sets in the images, G corresponds to the smaller, 'query' graph, and G' is the larger, 'target' graph (in this expression, G_i and G'_j are particular points in G and G'; $\pi(i)$ is the index of the point in

G' to which the i^{th} point in G should be correctly mapped; d is simply the Euclidean distance, and is scaled by σ , which is simply the width of the image in question). Hence we are penalising matches based on how distant they are from the correct match; this is commonly referred to as the 'endpoint error'.

Finally, we specify a quadratic regularizer $\Omega(w) = \frac{1}{2} ||w||^2$.

3.4 The Optimization Problem

Here we combine the elements discussed in 3.2 in order to formally set up a mathematical optimization problem that corresponds to the learning procedure. The expression that arises from (2) by incorporating the specifics discussed in 3.2/3.3 still consists of a very difficult (in particular nonconvex) optimization problem. Although the regularization term is convex in the parameters w, the empirical risk, i.e. the first term in (2), is not. Note that there is a finite number of possible matches y, and therefore a finite number of possible values for the loss Δ ; however, the space of parameters \mathcal{W} is continuous. What this means is that there are large equivalence classes of w (an equivalence class in this case is a given set of w's each of which produces the same loss). Therefore, the loss is piecewise constant on w, and as a result certainly not amenable to any type of smooth optimization.

One approach to render the problem of minimizing (2) more tractable is to replace the empirical risk by a convex upper bound on the empirical risk, an idea that has been exploited in Machine Learning in recent years [25,27,28]. By minimizing this convex upper bound, we hope to decrease the empirical risk as well. It is easy to show that the convex (in particular, linear) function $\frac{1}{N} \sum_n \xi_n$ is an upper bound for $\frac{1}{N} \sum_n \Delta(g(G^n, G'^n; w), y^n)$ for the solution of (2) with appropriately chosen constraints:

$$\underset{w,\xi}{\text{minimize}} \quad \frac{1}{N} \sum_{n=1}^{N} \xi_n + \frac{\lambda}{2} \|w\|^2$$
(9a)

subject to
$$\langle w, \Psi^n(y) \rangle \ge \Delta(y, y^n) - \xi_n$$
 (9b)
for all n and $y \in \mathcal{Y}$.

Here we define $\Psi^n(y) := \Phi(G^n, G'^n, y^n) - \Phi(G^n, G'^n, y)$. Formally, we have:

Lemma 3.1 For any feasible (ξ, w) of (9) the inequality $\xi_n \geq \Delta(g(G^n, G'^n; w), y^n)$ holds for all n. In particular, for the optimal solution (ξ^*, w^*) we have $\frac{1}{N} \sum_n \xi_n^* \geq \frac{1}{N} \sum_n \Delta(g(G^n, G'^n; w^*), y^n).$

Proof The constraint (9b) needs to hold for all y, hence in particular for $y^{w^*} = g(G^n, G'^n; w^*)$. By construction y^{w^*} satisfies $\langle w, \Psi^n(y^{w^*}) \rangle \leq 0$. Consequently $\xi_n \geq \Delta(y^{w^*}, y^n)$. The second part of the claim follows immediately.

The constraints (9b) mean that the margin $f(G^n, G'^n, y^n; w) - f(G^n, G'^n, y; w)$, i.e. the gap between the discriminant functions for y^n and y should exceed the loss induced by estimating y instead of the

 $^{^{2}}$ As a result in our general setting 'node' compatibilities and 'edge' compatibilities become somewhat misnomers, being more appropriately described as unary and binary compatibilities. We however stick to the standard terminology for simplicity of exposition.

training matching matrix y^n . This is highly intuitive since it reflects the fact that we want to safeguard ourselves most against mis-predictions y which incur a large loss (i.e. the smaller is the loss, the less we should care about making a mis-prediction, so we can enforce a smaller margin). The presence of ξ_n in the constraints and in the objective function means that we allow the hard inequality (without ξ_n) to be violated, but we penalize violations for a given nby adding to the objective function the cost $\frac{1}{N}\xi_n$.

Despite the fact that (9) has exponentially many constraints (every possible matching y is a constraint), we will see in what follows that there is an efficient way of finding an ϵ -approximation to the optimal solution of (9) by finding the worst violators of the constrained optimization problem.

3.5 The Algorithm

Note that the number of constraints in (9) is given by the number of *possible* matching matrices $|\mathcal{Y}|$ times the number of training instances N. In graph matching the number of possible matches between two graphs grows factorially with their size. In this case it is infeasible to solve (9) exactly.

There is however a way out of this problem by using an optimization technique known as *column generation* [24]. Instead of solving (9) directly, one computes the most violated constraint in (9) iteratively for the current solution and adds this constraint to the optimization problem. In order to do so, we need to solve

$$\underset{y}{\operatorname{argmax}} \left[\langle w, \Phi(G^n, G'^n, y) \rangle + \Delta(y, y^n) \right], \qquad (10)$$

as this is the term for which the constraint (9b) is tightest (i.e. the constraint that maximizes ξ_n).

The resulting algorithm is given in algorithm 1. We use the 'Bundle Methods for Regularized Risk Minimization' (BMRM) solver of [29], which merely requires that for each candidate w, we compute the gradient of $\frac{1}{N} \sum \langle w, \Psi(\hat{y}) \rangle + \frac{\lambda}{2} \|w\|^2$ with respect to w, and the loss $(\frac{1}{N} \sum_n \Delta(\hat{y}, y^n))$ (\hat{y} is the most violated constraint in column generation). See [29] for further details

Let us investigate the complexity of solving (10). Using the joint feature map Φ as in (6) and the loss as in (7), the argument in (10) becomes

$$\langle \Phi(G, G', y), w \rangle + \Delta(y, y^n) =$$

$$= \sum_{ii'} y_{ii'} \bar{c}_{ii'} + \sum_{ii'jj'} y_{ii'} y_{jj'} d_{ii'jj'} + \text{constant},$$
(11)

where $\bar{c}_{ii'} = \langle \phi_1(G_i, G'_{i'}), w_1 \rangle + y^n_{ii'} / \|y^n\|_F^2$ and $d_{ii'jj'}$ is defined as in (5b).

The maximization of (11), which needs to be carried out at *training* time, is a quadratic assignment problem, as is the problem to be solved at test time. In the particular case where $d_{ii'jj'} = 0$ throughout, both the problems at training and at test time are linear assignment problems, which can be solved efficiently in worst case cubic time.

In our experiments, we solve the linear assignment problem with the efficient solver from [30] ('house' sequence), and the Hungarian algorithm (video/bikes dataset). For

Algorithm 1 Column Generation

 $\begin{array}{l} \textbf{Define:}\\ \Psi^n(y) &:= \Phi(G^n,G'^n,y^n) - \Phi(G^n,G'^n,y)\\ H^n(y) &:= \langle w, \Phi(G^n,G'^n,y) \rangle + \Delta(y,y^n)\\ \textbf{Input:} \text{ training graph pairs } \{G^n\}, \{G'^n\}, \text{ training match$ $ing matrices } \{y^n\}, \text{ sample size } N, \text{ tolerance } \epsilon\\ \textbf{Initialize } S_n &= \emptyset \text{ for all } n, \text{ and } w = 0\\ \textbf{repeat}\\ \textbf{Get current } w \text{ from BMRM}\\ \textbf{for } n = 1 \text{ to } N \text{ do}\\ \hat{y} &= \operatorname{argmax}_{y \in \mathfrak{Y}} H^n(y)\\ \textbf{Compute gradient of } \langle w, \Psi(G^n, G'^n, y) \rangle + \frac{\lambda}{2} \|w\|^2\\ \textbf{w.r.t. } w &(= \Psi_n(\hat{y}) + \lambda w)\\ \textbf{Compute loss } \Delta(\hat{y}, y^n)\\ \textbf{end for}\\ \textbf{Report } \frac{1}{N} \sum \xi_n \text{ and } \frac{1}{N} \sum_n \Delta(\hat{y}, y^n) \text{ to BMRM}\\ \textbf{until } \frac{1}{N} \sum \xi_n \text{ is sufficiently small} \end{array}$

quadratic assignment, we developed a C++ implementation of the well-known Graduated Assignment algorithm [17]. However the learning scheme discussed here is independent of which algorithm we use for solving either linear or quadratic assignment. Note that the estimator is but a mere approximation in the case of quadratic assignment: since we are unable to find the most violated constraints of (10), we cannot be sure that the duality gap is properly minimized in the constrained optimization problem.

4 Features for the Compatibility Functions

The joint feature map $\Phi(G, G', y)$ has been derived in its full generality (6), but in order to have a working model we need to choose a specific form for $\phi_1(G_i, G'_{i'})$ and $\phi_2(G_{ij}, G'_{i'j'})$, as mentioned in section 3. We first discuss the linear features ϕ_1 and then proceed to the quadratic terms ϕ_2 . For concreteness, here we only discuss options actually used in our experiments.

4.1 Node Features

We construct $\phi_1(G_i, G'_{i'})$ using the squared difference $\phi_1(G_i, G'_{i'}) = (\ldots, -|G_i(r) - G'_{i'}(r)|^2, \ldots)$. This differs from what is shown in [1], in which an exponential decay is used (i.e. $\exp(-|G_i(r) - G'_{i'}(r)|^2/)$); we found that using the squared difference resulted in much better performance after learning. Here $G_i(r)$ and $G'_{i'}(r)$ denote the r^{th} coordinates of the corresponding attribute vectors. Note that in standard graph matching without learning we typically have $c_{ii'} = \exp(-||G_i - G'_{i'}||^2)$, which can be seen as the particular case of (5a) for both ϕ_1 and w_1 flat, given by $\phi_1(G_i, G'_{i'}) = (\ldots, \exp(-||G_i - G'_{i'}||^2), \ldots)$ and $w_1 = (\ldots, 1, \ldots)$ [22]. Here instead we have $c_{ii'} = \langle \phi_1(G_i, G'_{i'}), w_1 \rangle$, where w_1 is learned from training data. In this way, by tuning the r^{th} coordinate of w_1 accordingly, the learning process finds the relevance of the r^{th} feature of ϕ_1 . In our experiments (to be described in the next section), we use the well-known 60-dimensional Shape Context features [31]. They encode how each node 'sees' the other nodes. It is an instance of what we called in section 3 a feature that captures the node 'perspective' with respect to the graph. We use 12 angular bins (for angles in $[0, \frac{\pi}{6}) \dots [\frac{11\pi}{6}, 2\pi)$), and 5 radial bins (for radii in $(0, 0.125), [0.125, 0.25) \dots [1, 2)$, where the radius is scaled by the average of all distances in the scene) to obtain our 60 features. This is similar to the setting described in [31].

4.2 Edge Features

For the edge features G_{ij} $(G'_{i'j'})$, we use standard graphs, i.e. G_{ij} $(G'_{i'j'})$ is 1 if there is an edge between *i* and *j* and 0 otherwise. In this case, we set $\phi_2(G_{ij}, G'_{i'j'}) = G_{ij}G'_{i'j'}$ (so that w_2 is a scalar).

5 Experiments

5.1 House Sequence

For our first experiment, we consider the CMU 'house' sequence – a dataset consisting of 111 frames of a toy house [32]. Each frame in this sequence has been hand-labelled, with the same 30 landmarks identified in each frame [33]. We explore the performance of our method as the baseline (separation between frames) varies.

For each baseline (from 0 to 90, by 10), we identified all pairs of images separated by exactly this many frames. We then split these pairs into three sets, for training, validation, and testing. In order to determine the adjacency matrix for our edge features, we triangulated the set of landmarks using the Delaunay triangulation (see figure 1).

Figure 1 (top) shows the performance of our method as the baseline increases, for both linear and quadratic assignment (for quadratic assignment we use the Graduated Assignment algorithm, as mentioned previously). The values shown report the normalised Hamming loss (i.e. the proportion of points incorrectly matched); the regularization constant resulting in the best performance on our validation set is used for testing. Graduated assignment using bistochastic normalisation, which to the best of our knowledge is the state-of-the-art relaxation, is shown for comparison [22].³

For both linear and quadratic assignment, figure 1 shows that learning significantly outperforms non-learning in terms of accuracy. Interestingly, quadratic assignment performs *worse* than linear assignment before learning is applied – this is likely because the relative scale of the linear and quadratic features is badly tuned before learning. Indeed, this demonstrates exactly why learning is important. It is also worth noting that linear assignment *with* learning performs similarly to quadratic assignment with bistochastic normalisation (without learning) – this is an important result, since quadratic assignment via Graduated Assignment is significantly more computationally intensive.



Figure 1: Top: Performance on the 'house' sequence as the baseline (separation between frames) varies (the normalised Hamming loss on all testing examples is reported, with error bars indicating the standard error). Centre: The weights learned for the quadratic model (baseline = 90, $\lambda = 1$). Bottom: A frame from the sequence, together with its landmarks and triangulation; the 3^{rd} and the 93^{rd} frames, matched using linear assignment (without learning, loss = 12/30), and the same match after learning ($\lambda = 10$, loss = 6/30). Mismatches are shown in red.

³Exponential decay on the node features was beneficial when using the method of [22], and has hence been maintained in this case (see section 4.1); a normalisation constant of $\delta = 0.00001$ was used.



Figure 2: Top: Performance on the video sequence as the baseline (separation between frames) varies (the endpoint error on all testing examples is reported, with error bars indicating the standard error). Centre: The weights learned for the model (baseline = 90, $\lambda = 100$). Bottom: The 7th and the 97th frames, matched using linear assignment (loss = 0.028), and the same match after learning ($\lambda = 100$, loss = 0.009). The outline of the points to be matched (left), and the correct match (right) are shown in green; the inferred match is outlined in red; the match after learning is much closer to the correct match.



Figure 3: Running time versus accuracy on the 'house' dataset, for a baseline of 90. Standard errors of both running time and performance are shown (the standard error for the running time is almost zero). Note that linear assignment is around three orders of magnitude faster than quadratic assignment.

Figure 1 (centre) shows the weight vector learned using quadratic assignment (for a baseline of 90 frames, with $\lambda = 1$). Note that the first 60 points show the weights of the Shape Context features, whereas the final point corresponds to the edge features. The final point is given a very high score after learning, indicating that the edge features are important in this model.⁴ Here the first 12 features correspond to the first radial bin (as described in section 4) etc. The first radial bin appears to be more important than the last, for example. Figure 1 (bottom) also shows an example match, using the 3^{rd} and the 93^{rd} frames of the sequence for linear assignment, before and after learning.

Finally, Figure 3 shows the running time of our method compared to its accuracy. Firstly, it should be noted that the use of learning has no effect on running time; since learning outperforms non-learning in all cases, this presents a very strong case for learning. Quadratic assignment with bistochastic normalisation gives the best non-learning performance, however, it is still worse than either linear or quadratic assignment *with* learning and it is significantly slower.

5.2 Video Sequence

For our second experiment, we consider matching features of a human in a video sequence. We used a video sequence from the SAMPL dataset [34] – a 108 frame sequence of a human face (see figure 2, bottom). To identify landmarks for these scenes, we used the SUSAN corner detector [35, 36]. This detector essentially identifies points as corners if their neighbours within a small radius are dissimilar. This

⁴This should be interpreted with some caution: the features have different scales, meaning that their importances cannot be compared directly. However, from the point of view of the regularizer, assigning this feature a high weight bares a high cost, implying that it is an important feature.

detector was tuned such that no more than 200 landmarks were identified in each scene.

In this setting, we are no longer interested in matching *all* of the landmarks in both images, but rather those that correspond to important parts of the human figure. We identified the same 11 points in each image (figure 2, bottom). It is assumed that these points are known in advance for the template scene (G), and are to be found in the target scene (G'). Clearly, since the correct match corresponds to only a tiny proportion of the scene, using the normalised Hamming loss is no longer appropriate – we wish to penalise incorrect matches less if they are 'close to' the correct match. Hence we use the loss function (as introduced in section 3.2)

$$\Delta(G, G', \pi) = 1 - \frac{1}{|\pi|} \sum_{i} \left[\frac{d(G_i, G'_{\pi(i)})}{\sigma} \right].$$
 (12)

Here the loss is small if the distance between the chosen match and the correct match is small.

Since we are interested in only a few of our landmarks, triangulating the graph is no longer meaningful. Hence we present results only for linear assignment.

Figure 2 (top) shows the performance of our method as the baseline increases. In this case, the performance is nonmonotonic as the subject moves in and out of view throughout the sequence. This sequence presents additional difficulties over the 'house' dataset, as we are subject to noise in the detected landmarks, and possibly in their labelling also. Nevertheless, learning outperforms non-learning for all baselines. The weight vector (figure 2, centre) is heavily peaked about particular angular bins.

5.3 Bikes

For our final experiment, we used images from the Caltech 256 dataset [37]. We chose to match images in the 'touring bike' class, which contains 110 images of bicycles. Since the Shape Context features we are using are robust to only a small amount of rotation (and not to reflection), we only included images in this dataset that were taken 'side-on'. Some of these were then reflected to ensure that each image had a consistent orientation (in total, 78 images remained). Again, the SUSAN corner detector was used to identify the landmarks in each scene; 6 points corresponding to the frame of the bicycle were identified in each frame (see figure 4, bottom).

Rather than matching all *pairs* of bicycles, we used a fixed template (G), and only varied the target. This is an easier problem than matching all pairs, but is realistic in many scenarios, such as image retrieval.

Table 2 shows the endpoint error of our method, and gives further evidence of the improvement of learning over nonlearning. Figure 4 shows a selection of data from our training set, as well as an example matching, with and without learning.

	Loss	Loss (learning)
Training	$0.094\ (0.005)$	0.057 (0.004)
Validation	$0.040\ (0.007)$	$0.040 \ (0.006)$
Testing	0.101 (0.005)	0.062(0.004)

Table 2: Performance on the 'bikes' dataset. Results for the minimiser of the validation loss ($\lambda = 10000$) are reported. Standard errors are in parentheses.



Figure 4: Top: Some of our training scenes. Bottom: A match from our test set. The top frame shows the points as matched without learning (loss = 0.105), and the bottom frame shows the match with learning (loss = 0.038). The outline of the points to be matched (left), and the correct match (right) are outlined in green; the inferred match is outlined in red.

6 Conclusions and Discussion

We have shown how the compatibility functions for the graph matching problem can be estimated from labeled training examples, where a training input is a pair of graphs and a training output is a matching matrix. We use largemargin structured estimation techniques with column generation in order to solve the learning problem efficiently, despite the huge number of constraints in the optimization problem. We presented experimental results in three different settings, each of which revealed that the graph matching problem can be significantly improved by means of learning.

An interesting finding in this work has been that *linear* assignment with learning performs similarly to Graduated Assignment with bistochastic normalisation, a state-of-theart *quadratic* assignment relaxation algorithm. This suggests that, in situations where speed is a major issue, linear assignment may be resurrected as a means for graph matching. In addition to that, if learning is introduced to Graduated Assignment *itself*, then the performance of graph matching *improves significantly* both on accuracy and speed when compared to the best existing quadratic assignment relaxation [22].

There are many other situations in which learning a matching criterion can be useful. In multi-camera settings for example, when different cameras may be of different types and have different calibrations and viewpoints, it is reasonable to expect that the optimal compatibility functions will be different depending on which camera pair we consider. In surveillance applications we should take advantage of the fact that much of the context does not change: the camera and the viewpoint are typically the same.

To summarize, by learning a matching criterion from previously labeled data, we are able to substantially improve the accuracy of graph matching algorithms.

References

- T. S. Caetano, L. Cheng, Q. V. Le, and A. J. Smola, "Learning graph matching," in *International Confer*ence on Computer Vision, 2007.
- M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems using pairwise constraints," in *ICCV*, 2005.
- [3] H. Wang and E. R. Hancock, "A kernel view of spectral point pattern matching," in *International Workshops* SSPR & SPR, LNCS 3138, 2004, pp. 361–369.
- [4] L. Shapiro and J. Brady, "Feature-based correspondence - an eigenvector approach," *Image and Vision Computing*, vol. 10, pp. 283–288, 1992.
- [5] M. Carcassoni and E. R. Hancock, "Spectral correspondence for point pattern matching," *Pattern Recognition*, vol. 36, pp. 193–204, 2003.
- [6] T. Caelli and S. Kosinov, "An eigenspace projection clustering method for inexact graph matching," *IEEE Trans. PAMI*, vol. 26, no. 4, pp. 515–519, 2004.

- [7] T. S. Caetano, T. Caelli, and D. A. C. Barone, "Graphical models for graph matching," in *IEEE International Conference on Computer Vision and Pattern Recognition*, Washington, DC, 2004, pp. 466–473.
- [8] A. Rosenfeld and A. C. Kak, *Digital Picture Processing*. New York, NY: Academic Press, 1982.
- [9] R. C. Wilson and E. R. Hancock, "Structural matching by discrete relaxation," *IEEE Trans. PAMI*, vol. 19, no. 6, pp. 634–648, 1997.
- [10] E. Hancock and R. C. Wilson, "Graph-based methods for vision: A yorkist manifesto," SSPR & SPR 2002, LNCS, vol. 2396, pp. 31–46, 2002.
- [11] W. J. Christmas, J. Kittler, and M. Petrou, "Structural matching in computer vision using probabilistic relaxation," *IEEE Trans. PAMI*, vol. 17, no. 8, pp. 749–764, 1994.
- [12] J. V. Kittler and E. R. Hancock, "Combining evidence in probabilistic relaxation," Int. Journal of Pattern Recognition and Artificial Intelligence, vol. 3, pp. 29– 51, 1989.
- [13] S. Z. Li, "A markov random field model for object matching under contextual constraints," in *Interna*tional Conference on Computer Vision and Pattern Recognition, 1994, pp. 866–869.
- [14] C. Schellewald, "Convex mathematical programs for relational matching of object views," Ph.D. dissertation, University of Mannhein, 2004.
- [15] M. Pelillo, "Replicator equations, maximal cliques, and graph isomorphism," *Neural Comput.*, vol. 11, pp. 1933–1955, 1999.
- [16] B. T. Messmer and H. Bunke, "A new algorithm for error-tolerant subgraph isomorphism detection," *IEEE Trans. PAMI*, vol. 20, no. 5, pp. 493–503, 1998.
- [17] S. Gold and A. Rangarajan, "A graduated assignment algorithm for graph matching," *IEEE Trans. PAMI*, vol. 18, no. 4, pp. 377–388, 1996.
- [18] A. Rangarajan, A. Yuille, and E. Mjolsness, "Convergence properties of the softassign quadratic assignment algorithm," *Neural Computation*, vol. 11, pp. 1455– 1474, 1999.
- [19] R. Sinkhorn, "A relationship between arbitrary positive matrices and doubly stochastic matrices," Ann. Math. Statis., vol. 35, pp. 876–879, 1964.
- [20] M. Pelillo and M. Refice, "Learning compatibility coefficients for relaxation labeling processes," *IEEE Trans.* on PAMI, vol. 16, no. 9, pp. 933–945, 1994.
- [21] S. Lacoste-Julien, B. Taskar, D. Klein, and M. Jordan, "Word alignment via quadratic assignment," in *HLT-NAACL06*, 2006.

- [22] T. Cour, P. Srinivasan, and J. Shi, "Balanced graph matching," in NIPS, 2006.
- [23] K. Anstreicher, "Recent advances in the solution of quadratic assignment problems," *Math. Program., Ser. B*, vol. 97, pp. 27–42, 2003.
- [24] C. Papadimitriou and K. Steiglitz, Combinatorial Optimization : Algorithms and Complexity. Dover Publications, July 1998.
- [25] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large margin methods for structured and interdependent output variables," *J. Mach. Learn. Res.*, vol. 6, pp. 1453–1484, 2005.
- [26] R. K. Ahuja and J. B. Orlin, "Inverse optimization," Operations Research, vol. 49, no. 5, pp. 771–783, 2001.
- [27] A. Smola, S. V. N. Vishwanathan, and Q. Le, "Bundle methods for machine learning," in Advances in Neural Information Processing Systems 20, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. Cambridge, MA: MIT Press, 2008, pp. 1377–1384.
- [28] B. Taskar, C. Guestrin, and D. Koller, "Max-margin markov networks," in Advances in Neural Information Processing Systems 16, S. Thrun, L. Saul, and B. Schölkopf, Eds. Cambridge, MA: MIT Press, 2004.
- [29] C. Teo, Q. Le, A. Smola, and S. Vishwanathan, "A scalable modular convex solver for regularized risk minimization," in *Knowledge Discovery and Data Mining KDD*'07, 2007.
- [30] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing*, vol. 38, no. 4, pp. 325–340, 1987.
- [31] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. on PAMI*, vol. 24, no. 24, pp. 509–521, 2002.
- [32] CMU 'house' dataset: vasc.ri.cmu.edu/idb/html/motion/house/index.html.
- [33] T. S. Caetano, T. Caelli, D. Schuurmans, and D. A. C. Barone, "Graphical models and point pattern matching," *IEEE Trans. on PAMI*, vol. 28, no. 10, pp. 1646– 1663, 2006.
- [34] SAMPLE motion dataset: http://sampl.ece.ohio-state.edu/database.htm.
- [35] S. Smith, "A new class of corner finder," in *BMVC*, 1992, pp. 139–148.
- [36] —, "Flexible filter neighbourhood designation," in *ICPR*, 1996, pp. 206–212.
- [37] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Institute of Technology, Tech. Rep. 7694, 2007. [Online]. Available: http://authors.library.caltech.edu/7694