Learning Graph Matching

Tibério S. Caetano, Li Cheng, Quoc V. Le and Alex J. Smola
Statistical Machine Learning Program, NICTA and ANU
Canberra ACT 0200, Australia

Abstract

As a fundamental problem in pattern recognition, graph matching has found a variety of applications in the field of computer vision. In graph matching, patterns are modeled as graphs and pattern recognition amounts to finding a correspondence between the nodes of different graphs. There are many ways in which the problem has been formulated, but most can be cast in general as a quadratic assignment problem, where a linear term in the objective function encodes node compatibility functions and a quadratic term encodes edge compatibility functions. The main research focus in this theme is about designing efficient algorithms for solving approximately the quadratic assignment problem, since it is NP-hard.

In this paper, we turn our attention to the complementary problem: 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 matchings between pairs of graphs. We present experimental results with real image data which give evidence that learning can improve the performance of standard graph matching algorithms. In particular, it turns out that linear assignment with such a learning scheme may improve over state-of-the-art quadratic assignment relaxations. This finding suggests that for a range of problems where quadratic assignment was thought to be essential for securing good results, linear assignment, which is far more efficient, could be just sufficient if learning is performed. This enables speed-ups of graph matching by up to 4 orders of magnitude while retaining state-of-the-art accuracy.

1. Introduction

Graphs are commonly used as abstract representations for complex scenes, and many computer vision problems can be formulated as an attributed graph matching problem, where 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 in 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 in 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, G and G', should the optimal match be uniquely determined? For example, assume first that G and G' come from two images acquired with 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 same[1] since the graph pair is the same in both cases. This is how 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 is obviously not practical. Instead, we approach the problem from a purely statistical inference perspective. First we extract graphs from a number of images acquired in the same conditions as those for which we want to solve, whatever the word "conditions" mean (e.g. from the surveillance camera or the photographer's database). We then *manually* provide what we understand to be the optimal matches between pairs of 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 a loose language) adjusting the node and edge compatibility functions in a way that the expected optimal match in a test pair of graphs agrees with the expected match they would have had they been in the training set. In this formulation, the learning problem consists of a 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 im-

[1] Assuming a single optimal solution and that the algorithm finds it.

proves their performance. In fact, we show that learning improves on non-learning results so dramatically that *linear* assignment with learning can perform similarly or better than state-of-the-art *quadratic* assignment relaxation algorithms without learning. This suggests that a range of problems for which quadratic assignment (NP-hard) was thought to be essential in order to secure good matching results may be accurately solved with a much simpler (worst case cubic time) linear assignment algorithm under the proposed learning framework.

1.1. Related Literature

A variety of approaches has been proposed to solve the attributed graph matching problem. An incomplete list includes spectral methods [13, 18], semidefinite programming [17], probabilistic methods [10, 5, 7] and the well-known graduated assignment method [9].

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

In [16] 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. In terms of methodology, possibly the paper most closely related to ours is [12], 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 [8]; 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 vertex i and edge 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 G maps to node i' in G' ($i \mapsto i'$) 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* [3]

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;
n - index for training instance; N - no. 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; g^* - optimal predictor function;
f - discriminant function;
Δ - loss function;
Φ, ϕ_1, ϕ_2 - joint, node and edge feature maps respectively;
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 j^{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 $ij \mapsto i'j'$
α_{ny} - dual variable
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; C - regularization parameter;
δ - convergence threshold in bistochastic normalization;

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

typically subject to either the injectivity constraint (one-to-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 [15]. 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 parameterized 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 here 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. In a way, this could be seen as directly "changing the question" in graph matching, instead of trying a "better answer" to the same question.

3. Learning Graph Matching

3.1. General Problem Setting

We approach the problem of learning the compatibility functions as a supervised learning problem [19]. 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. We will see that this peculiar aspect of our problem will give rise to a non-trivial learning problem which will demand the employment of elaborate optimization techniques in order to be solved.

If $\mathcal{X} = \mathcal{G} \times \mathcal{G}$ is the space of pairs of graphs, $\mathcal{Y}$ is the space of matching matrices and $\mathcal{W}$ 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 g^* will then be the one which minimizes an expression of the type

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

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

3.2. The Model

We start by specifying a w-parameterized 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

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

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. $y^* = g(G, G'; w)$. This allows us to interpret the discriminant function in (3) as the objective function to be maximized in (1):

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

which 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 parameterization for the compatibility functions. We assume a simple linear parameterization,

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

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

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. 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 traditional graph matching 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 then arrive at the final form for $\Phi(G, G', y)$ from (4) and (5):

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

Naturally, the final specification of the predictor g depends on the choices of ϕ_1 and ϕ_2. In our experiments we use SIFT features and Shape Context features for constructing ϕ_1 and a simple edge-match criterion for constructing ϕ_2 (details follow in section 4).

Next we define the loss $\Delta(y, y^n)$ incurred by estimating the matching matrix y instead of the correct one, y^n. This is

simply defined as the fraction of mismatches between matrices y and y^n, i.e.

$$\Delta(y, y^n) = 1 - \frac{\langle y, y^n \rangle}{\langle y^n, y^n \rangle}. \tag{7}$$

where here we used $\langle y, y^n \rangle := \sum_{ii'} y_{ii'} y^n_{ii'}$. Finally, we specify a quadratic regularizer $\Omega(g) = \frac{1}{2} \|w\|^2$.

3.3. The Optimization Problem

Here we combine the elements discussed in section 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 section 3.2 still consists in a very difficult (in particular non-convex) optimization problem. Although the regularization term is convex in the parameters w, the empirical risk, i.e. the first term in (2), is not: it depends in a very complicated way on w as can be seen from the nonlinear dependency of g on w (3).

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, as shown in [19]. By minimizing the convex upper bound, we will then also decrease the empirical risk (by the definition of an upper bound). It is easy to show that the convex (in particular, linear) function $\frac{1}{N} \sum_n \xi_n$ is an upper bound for $\frac{1}{N} \sum_n \Delta(g(G^n, G'^n), y^n)$ for the solution of (2) with appropriately chosen constraints:

$$\underset{w,\xi}{\text{minimize}} \quad \frac{C}{N} \sum_{n=1}^{N} \xi_n + \frac{1}{2} \|w\|^2 \tag{8a}$$

subject to

$$\langle w, \Psi^n(y) \rangle \geq \Delta(y, y^n) - \xi_n$$

$$\text{for all } n \text{ and } y \in \mathcal{Y}. \tag{8b}$$

where $\Psi^n(y) := \Phi(G^n, G'^n, y^n) - \Phi(G^n, G'^n, y)$. This is because at the optimal solution we have $\xi_n^* = \max\{0, \max_y\{\Delta(y, y^n) - \langle w, \Psi^n(y) \rangle\}\}$, which always upper bounds $\Delta(y, y^n)$ for y such that $\langle w, \Psi^n(y) \rangle \leq 0$. It then follows that $\xi_n \geq \Delta(g(G^n, G'^n), y^n)$, and immediately we have $\frac{1}{N} \sum_n \xi_n \geq \frac{1}{N} \sum_n \Delta(g(G^n, G'^n, y^n))$.

The constraints (8b) 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 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 n by adding to the objective function the cost $\frac{C}{N} \xi_n$.

Instead of solving the primal optimization problem given in (8), we actually solve its dual for reasons to be soon made clear. Denote by $K_{ny,mz} := \langle \Phi(G^n, G'^n, y), \Phi(G^m, G'^m, z) \rangle$. Then the dual problem of (8) can be derived as

$$\underset{\alpha}{\text{minimize}} \quad \frac{1}{2} \sum_{n,y,m,z} \alpha_{ny} \alpha_{mz} K_{ny,mz} - \sum_{n,y} \Delta(y, y^n) \alpha_{ny}$$

$$\text{s.t.} \sum_y \alpha_{ny} \leq \frac{C}{N} \text{ and } \alpha_{ny} \geq 0 \text{ for all } y \in \mathcal{Y}, n \tag{9}$$

where the primal and dual variables at optimality can be shown to be related by

$$w = \sum_{n,y} \alpha_{ny} (\Phi(G^n, G'^n, y^n) - \Phi(G^n, G'^n, y)), \tag{10}$$

i.e. we can recover the solution in the primal from the solution in the dual.

We will see in what follows that there is an efficient way of approximating the solution in the dual (9) by a column generation procedure, while the direct solution of the primal (8) is not feasible.

3.4. The Algorithm

Note that the number of constraints in the primal (8) and the dual (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 [15]. Instead of solving (9) immediately, one computes the most violated constraints in (8) iteratively for the current solution of (9) and adds those constraints to the optimization problem. In order to do so, we need to solve

$$\underset{y}{\text{argmax}} \left[\langle w, \Phi(G^n, G'^n, y) \rangle + \Delta(y, y^n) \right], \tag{11}$$

as this is the term for which the constraint (8b) is tightest. The resulting algorithm (analogous to the column generation scheme described in [19]) is given in table 2. Column generation has good convergence properties, and approximates the optimal solution to arbitrary precision in a polynomial number of iterations [19].

Let's investigate the complexity of solving (11). Using the joint feature map Φ as in (6) and the loss as in (7), the objective function in (11) becomes

$$\langle \Phi(G, G', y), w \rangle + \Delta(y, y^n) = \tag{12}$$

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

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

Table 2. Column Generation

Define:
$\Psi^n(y) := \Phi(G^n, G'^n, y^n) - \Phi(G^n, G'^n, y)$
$H^n(y) := \langle w, \Phi(G^n, G'^n, y)\rangle + \Delta(y, y^n)$
Input: training graph pairs $\{G^n\}$,$\{G'^n\}$, training matching matrices $\{y^n\}$, sample size N, tolerance ϵ
Initialize $S_n = \emptyset$ for all n, and $w = 0$.
repeat
  for $n = 1$ **to** N **do**
    $w = \sum_n \sum_{y \in S_n} \alpha_{ny} \Psi^n(y)$
    $\hat{y} = \operatorname{argmax}_{y \in \mathcal{Y}} H^n(y)$
    $\xi_n = \max(0, \max_{y \in S_n} H^n(y))$
    if $\langle w, \Phi(G^n, G'^n, \hat{y})\rangle + \Delta(y, y^n) > \xi_n + \epsilon$ **then**
      Increase constraint set $S_n \leftarrow S_n \cup \hat{y}$
      Optimize (9) using only α_{ny} where $y \in S_n$.
    end if
  end for
until no S_n has changed in this iteration

The maximization of (12), which needs to be carried out at *training* time, is a quadratic assignment problem just as the problem to be solved at test time is. 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 [11]. For quadratic assignment, we developed a C implementation of the well-known Graduated Assignment algorithm [9]. However it should be stressed that the learning scheme discussed here is completely independent of which algorithm we use for solving either linear or quadratic assignment.

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 $\phi_1(G_i, G'_{i'})$ and then proceed to $\phi_2(G_{ij}, G'_{i'j'})$. For concreteness, here we only describe options actually used in our experiments.

4.1. Node Features

We construct $\phi_1(G_i, G'_{i'})$ by using a standard coordinate-wise exponential decay given by $\phi_1(G_i, G'_{i'}) = (\dots, \exp(-|G_i(r) - G'_{i'}(r)|^2), \dots)$. 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 $\phi_1(G_i, G'_{i'}) = (\dots, \exp(-\|G_i - G'_{i'}\|^2), \dots)$ and $w_1 = (\dots, 1/R, \dots)$, where R is the dimension of ϕ_1 and w_1 [8]. 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 how relevant is the r^{th} feature of ϕ_1. In our experiments to be described in the next section, we use two types of node features (i.e. $G_i, G'_{i'}$): the well-known SIFT features [14] and Shape Context features [4]. SIFT is a 128-dimensional rotation and scale-invariant descriptor which has also some invariance with respect to viewpoint and illumination changes and has been widely used in computer vision [14]. Our Shape Context descriptor is 60-dimensional, as in [4], and roughly encodes 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. See [4] for details.

4.2. Edge Features

For 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 (i' and j') and 0 otherwise. We then set $\phi_2(G_{ij}, G'_{i'j'}) = G_{ij} G'_{i'j'}$.

5. Experiments

Graph matching has applications in problems where consistent correspondence between sets of features is required, such as object recognition, shape matching, wide baseline stereo, 2D and 3D registration. Examples of features may be points, lines, descriptors of interest points, etc. Here we select a few applications and present some experimental results of learning versus non-learning.

5.1. Matching Frames of the 'house' Sequence

Here we performed experiments with the CMU 'house' dataset [1]. This dataset contains 111 frames of a video sequence of a toy house for which labeling of the same 30 landmark points is available across the whole sequence [6]. We can easily deal with outliers by augmenting the smaller graph with dummy nodes, in the same way as described in [4]. The sequence is such that the first and last frames are separated by a very wide baseline. For each image in the sequence, we compute from its landmark points the Shape Context features and define them as being the unary attributes G_i. In order to generate the underlying graph topology, we create a Delaunay triangulation of the points and set G_{ij} according to it (i.e. 1 or 0 depending if there is an edge or not). We then compare linear assignment and graduated assignment both with and without learning, which gives us four algorithmic settings. In addition, we compare these settings against a state-of-the-art setting, graduated assignment with *bistochastic normalization*, as introduced in [8]. We do so for three different values of the convergence threshold δ (which trades off accuracy versus speed–see [8]). This gives in total seven settings, which correspond to the seven bars per training size/test size split in Figure 1.

The experimental setting for Figure 1 is the following. For each of the training/test splits (x axis), the *train* set contains all images whose order in the sequence is divisible by k (where $k \in \{25, 15, 10, 5, 3, 2\}$), the rest is used for test. With six values of k, we have six splits of the dataset. We also *swap* these train and test sets to check performance for the cases where we have plenty of training instances. That means we have six other splits where the *test* set contains all images whose order in the sequence is divisible by k (where $k \in \{2, 3, 5, 10, 15, 25\}$) and the rest is for training. Thus, we have in total 12 splits of the dataset.

Note that as we move to the right in the figure the training set size is increasing. This partly explains why the gaps between learning and non-learning versions of the algorithms increase. The other explanation is due to the increasing difficulty in the matching task: in the right of the figure the baseline between frames is wider. Note in particular that as we get to the right of the figure the non-learning algorithms get worse (due to the matching task becoming harder) while the learning algorithms get better (i.e. if sufficient training data is provided the accuracy improves regardless of the increase in the baseline). Note that the sizes of the error bars increase as well, but this is due to the shrinking of the test set, which produces higher variance in testing accuracy. Note in particular that from the middle to the right in the figure the accuracy of linear assignment with learning becomes at least as good as that for the state-of-the-art quadratic assignment relaxation given by graduated assignment with bistochastic normalization [8].[2] This is a significant result, given the large processing time differences between running a linear assignment algorithm and graduated assignment with bistochastic normalization (see Table 3 and Figure 2, which indicate a gap of up to 4 orders of magnitude in processing times). Even if faster quadratic assignment relaxations are used (such as SMAC from [8]), the gap is roughly unchanged since most of the time is taken by the bistochastic normalization procedure, not by the matching algorithm per se (see table 3). It is possible to speed-up the bistochastic normalization procedure by using a larger convergence monitoring threshold δ [8], but as Figure 1 shows the price to be paid is a significant increase in error rate (see bars for $\delta = 0.0001$, $\delta = 0.1$ and $\delta = 100$: δ is the L_1 norm between consecutive compatibility matrices d in the bistochastic normalization procedure, where c is encoded in d as $d_{ii'ii'} \leftarrow c_{ii'}$). Figure 3 shows the weight vector learned for a particular split of Figure 1. Figure 4 shows matches between the first and last frames in the sequence for linear assignment without and with learning.

[2] Although this is partly due to the use of context features, which create unary compatibilities that are correlated with the structure of the graph, this is not sufficient, as can be seen from the poor accuracy of linear assignment without learning.

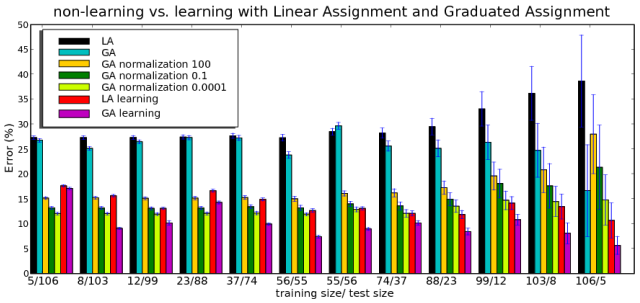

Figure 1. Mismatch percentages of all algorithms in the 'house' dataset. The baseline between frames increases as we move to the right in the plot. The increasing sizes of error bars simply reflect the decreasing test set sizes. For wide baseline matching (right in the figure), learning is essential for good results.

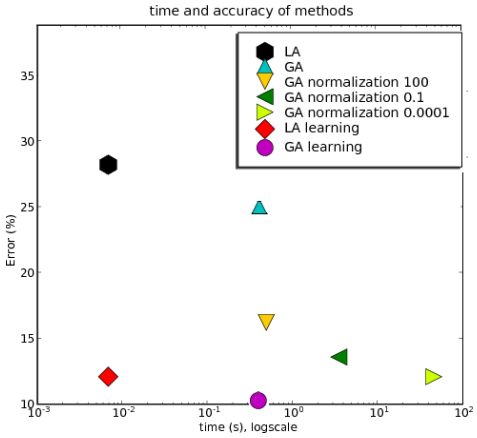

Figure 2. Trade-off between processing time and accuracy for a particular split from Figure 1 (74/37). Note that although the best version of bistochastic GA normalization (light green) has similar accuracy than LA learning (red), there is a gap of 4 orders of magnitude in processing times.

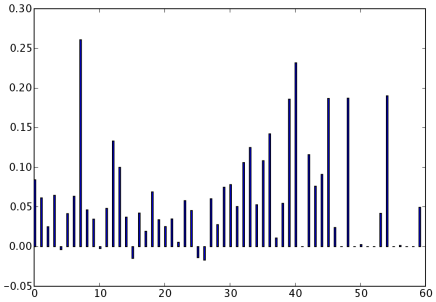

Figure 3. Weight vector w_1 for the 74/37 split from Figure 1 (LA learning). The uneven distribution reflects the fact that the learning algorithm tunes different components of the Shape Context descriptor so as to produce matches that best mimic those from human labeling. Standard graph matching with no learning corresponds to a flat distribution.

Table 3. Average processing times to match two 30-node graphs in a dual core Pentium 4, 3GHz, 1Gb RAM, with C implementations. Legend: la-linear assignment; ga-graduated assignment; bn-bistochastic GA normalization with threshold as shown; lal-linear assignment learning; gal-graduated assignment learning.

la	ga	bn100	bn0.1	bn0.0001	lal	gal
0.0068s	0.4s	0.49s	3.5s	45.6s	0.0068s	0.4s

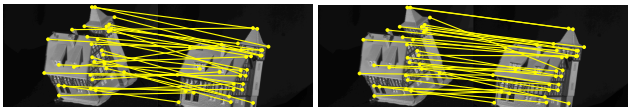

Figure 4. Wide baseline matching using only linear assignment. Left: match without learning (7/30 correct matches). Right: match with learning (19/30 correct matches).

5.2. Matching Frames of the 'hotel' Sequence

Although all image pairs in training are different from those in the test set, the experiment described in section 5.1 still involves only images in a *single* image sequence. In order to evaluate how learning can generalize from one to *another* image sequence, we have manually labeled data for a second image sequence. This way, we trained on pairs of the 'house' image sequence and tested on pairs of the 'hotel' image sequence [2]. In order to illustrate the fact that the size of the error bars only depends on the size of the test set and also the fact that the non-learning versions of the algorithms have the same accuracy if the test set is the same, we used a fixed test set by sampling the 'hotel' sequence at every 7 frames. This resulted in 105 image pairs for testing. The results are shown in Figure 5. We can see that the accuracy of all non-learning methods is constant, since the training set size only affects learning. As the training set size increases, the learning versions of the algorithms get progressively better. Eventually, linear assignment with learning surpasses bistochastic GA normalization. GA learning again outperforms all other algorithmic settings. Note also that the bistochastic GA normalization used in this plot is the one with highest accuracy among the three versions from section 5.1, i.e. the errors for the other 2 cases would be larger. (Recall the gap of 4 orders of magnitude between processing times for the algorithm that produces the red bar and the one that produces the light green bar in Figure 5) The fact that GA no-learning underperforms LA no-learning seems intriguing, but may be due to the fact that if learning is not performed the relative importance of the linear and quadratic terms is not properly calibrated.

5.3. Matching Images of Humans

In this experiment, the dataset contains images of humans performing simple actions: walking or running in different directions. The feature vector is the SIFT, and each graph has 12 nodes, corresponding to 12 points that showed up consistently when running the SIFT algorithm.

We split the data into training set and test set according to their "modes": moving from the left as training set (12 images – 66 training graph pairs) and moving from the right as test set (4 images — 6 test graph pairs). Figure 6 shows some instances of the training set. A random pair of graphs is selected from the training set as the validation set (used to tune the regularization parameter C). Every remaining pair of graphs is used as a training instance. We compare the statistics of the error rates of the test examples on Table 4. In this particular experiment we focus on linear assignment only since the humans are very flexible and stringent structural constraints have high variation, thus making the structural information very noisy. The results indicate that learning, again, significantly outperforms non-learning. One matching example is shown in Figure 7.

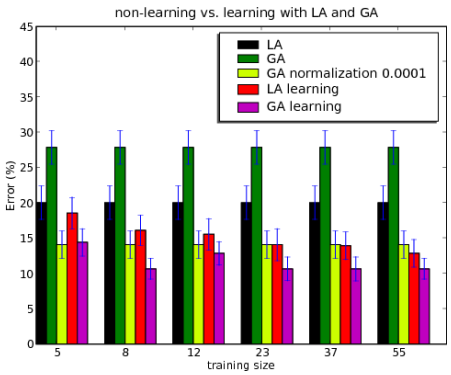

Figure 5. Mismatch proportions for increasing training set size and fixed test set. Training is done in pairs of the 'house' sequence and test in pairs of the 'hotel' sequence.

Table 4. The error rates of LA vs LA-learning on 'Humans' dataset. Paired t-test shows $p = 0.0058$ (very significant).

	mean	std. error
LA	32.41%	2.57%
LA learning	13.88%	2.77%

Figure 6. Examples of images from the 'Humans' dataset.

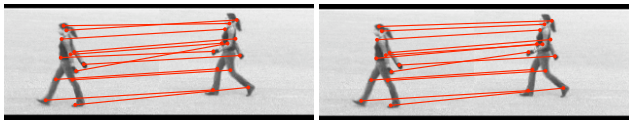

Figure 7. Left: match without learning. Right: match with learning. Note that without learning ear and hair are swapped, hand matches elbow, elbow matches belly and belly matches hand. With learning, there are no mistakes.

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 large-margin 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 present experimental results in three different settings, in all of which the solutions for the graph matching problem have been improved by means of learning.

A major finding in this work has been the realization that *linear* assignment with learning may perform similarly or better than state-of-the-art *quadratic* assignment relaxation algorithms (without learning). This clearly suggests a direct possibility of "transporting" computational burden from the online matching task to the off-line learning procedure. This enables one to solve large matching problems quickly using a fast linear assignment solver without jeopardizing accuracy. It is however important that context is somehow encoded into the unary features. More research into this topic is necessary in order to see how far we can go with linear assignment only.

In brief, the main conclusions from this paper are that, by using learning, (i) the speed of graph matching can be significantly boosted while retaining state-of-the-art accuracy (in case linear assignment is used) or (ii) the accuracy of graph matching can be significantly improved without decreasing the speed (in case quadratic assignment is used).

In our experiments we have only scratched the surface of potential applications for learning graph matching. For example, we can learn compatibility functions for cases where the graphs are very different. Imagine we want to match images of real people with images of cartooned people. The features will likely be very different, so standard graph matching is hopeless. However by doing learning we can find the appropriate feature calibration such that the matching loss is small. Similarly, we may match color against gray-level images, high-resolution against low-resolution images, images from cameras with completely different specification and calibration, etc. All we need are some training examples of matches in similar conditions. Even different *types* of features can be used in each of the graphs, the only thing needed being sensible constructions of ϕ_1 and ϕ_2 (different features may have different dimensionalities for instance, so the straightforward constructions used in section 4 should be adapted accordingly).

To summarize, by learning a matching criterion from previously labeled data (obtained under conditions similar to those in which we want the algorithm to be used), we are able to implicitly account for the peculiarities of the situation, and customize the graph matching algorithm accordingly.

Acknowledgements

We thank M. Barbosa, T. Barra and J. McAuley for critical readings of the manuscript. Part of this work was carried out while Quoc Le was with the Max Planck Institute for Biological Cybernetics. NICTA is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council. This work was supported by the Pascal Network.

References

[1] CMU 'house' dataset: http://vasc.ri.cmu.edu/idb/html/motion/house/index.html. 5

[2] CMU 'hotel' dataset: http://vasc.ri.cmu.edu/idb/html/motion/hotel/index.html. 7

[3] K. Anstreicher. Recent advances in the solution of quadratic assignment problems. *Math. Program., Ser. B*, 2003. 2

[4] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *PAMI*, 2002. 5

[5] T. S. Caetano, T. Caelli, and D. A. C. Barone. Graphical models for graph matching. In *CVPR*, 2004. 2

[6] T. S. Caetano, T. Caelli, D. Schuurmans, and D. A. C. Barone. Graphical models and point pattern matching. *PAMI*, 2006. 5

[7] W. J. Christmas, J. Kittler, and M. Petrou. Structural matching in computer vision using probabilistic relaxation. *PAMI*, 1994. 2

[8] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. In *NIPS*, 2006. 2, 5, 6

[9] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *PAMI*, 1996. 2, 5

[10] E. Hancock and R. C. Wilson. Graph-based methods for vision: A yorkist manifesto. *SSPR & SPR 2002*, 2002. 2

[11] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 1987. 5

[12] S. Lacoste-Julien, B. Taskar, D. Klein, and M. Jordan. Word alignment via quadratic assignment. In *HLT-NAACL06*, 2006. 2

[13] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, 2005. 2

[14] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 5

[15] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, New Jersey, 1982. 2, 4

[16] M. Pelillo and M. Refice. Learning compatibility coefficients for relaxation labeling processes. *PAMI*, 1994. 2

[17] C. Schellewald. *Convex mathematical programs for relational matching of object views*. PhD thesis, University of Mannhein, 2004. 2

[18] L. Shapiro and J. Brady. Feature-based correspondence - an eigenvector approach. *IVC*, 1992. 2

[19] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 2005. 3, 4