Learning Graph Matching
Joint work with Tiberio Caetano, Li Cheng, and Quoc Le

Alexander J. Smola
sml.nicta.com.au
Statistical Machine Learning Program
Canberra, ACT 0200 Australia
Alex.Smola@nicta.com.au

Firenze, MLG, August 2007
Outline

1. Applications
2. Algorithms
3. Changing the Question
4. Learning Graph Matching
5. Experiments
6. Extensions
Graph Matching

Chemistry and Biology
- Molecules stored in database
- Regulatory networks
- Function estimation for proteins

Computer Vision
- Object matching (e.g. wide baseline match)
- Preprocessing for camera calibration
- 3D reconstruction
- Match maps to aerial photographs (automatic map updates)
Two identical graphs
Large baseline (bad match)
Large baseline (better match)
Problems

**Hardness**
No currently known polynomial time algorithm for matching. Checking is linear in the number of edges.

**Completeness**
- The graphs may not be identical
- We just may want to find a “best match”
- Problem often ill-defined (e.g. largest common subgraph, best matches overall, etc.)

**Attributes**
- SIFT features — unlikely to be identical at all
- Different image resolutions (e.g. different cameras)
- Different image content (e.g. black and white vs. color)
- Different representation (e.g. pixels vs. symbolic)

**Size**
For very large graphs heuristics are popular.
Key observation
Graph matching often needed only for a **restricted domain**.

Idea
- Graph matching on restricted subset of graphs is often **much** easier.
- Attributes in graphs can help a lot (e.g. Bunke’s work for uniquely attributed vertices — matching becomes trivial)
- Local neighborhood may be sufficient for matching.

Strategy
- Use examples of matched graphs. Trivial if both graphs are of the same type: only need collection of graphs, no labeling needed.
- For corresponding objects of different representations training data is needed. Also if we want system to have a robust attribute matching function.
Linear Assignment

Notation

- Graphs $G$ and $G'$ with vertices $V$, $V'$ and edges $E$, $E'$.
- We use $G_{ij} = 1$ to denote presence of an edge between $i$ and $j$ (and $G_{ij} = 0$ to denote its absence).
- $V_i$ denotes vertex $i$ (and its attributes).
- Permutation matrix $\Pi$ describing match between $G$ and $G'$ with $\Pi_{ij} \in \{0; 1\}$ and $\Pi 1 = \Pi^\top 1 = 1$.

Objective Function

- Score $C_{ij}$ for match between vertex $V_i$ and $V'_j$.
- Best assignment by solving

$$\text{minimize} \sum_{i,j} \Pi_{ij} C_{ij}$$

- For uniquely attributed graphs (trivial) we set $C_{ij} = \delta_{V_i, V'_j}$. 
Hungarian Marriage
Hungarian Marriage

Integer Program

$$\text{minimize } \sum_{i,j} \Pi_{ij} C_{ij} \text{ subject to } \Pi_{ij} \in \{0, 1\} \text{ and } \Pi 1 = \Pi^T 1 = 1$$

Linear Programming Relaxation

$$\text{minimize } \sum_{i,j} \Pi_{ij} C_{ij} \text{ subject to } \Pi_{ij} \in [0, 1] \text{ and } \Pi 1 = \Pi^T 1 = 1$$

Properties

- Can be solved in polynomial time (e.g. interior point)
- All vertices are integral, hence the two problems are equivalent.
- Fast shortest path solvers available.
- Adding prior knowledge is easy — clamp $\Pi_{ij}$ to 0 or 1.
Hungarian Marriage

Integer Program

$$\minimize \sum_{i,j} \Pi_{ij} C_{ij} \ \text{subject to} \ \Pi_{ij} \in \{0; 1\} \text{ and } \Pi 1 = \Pi^T 1 = 1$$

Linear Programming Relaxation

$$\minimize \sum_{i,j} \Pi_{ij} C_{ij} \ \text{subject to} \ \Pi_{ij} \in [0, 1] \text{ and } \Pi 1 = \Pi^T 1 = 1$$

Properties

- Can be solved in polynomial time (e.g. interior point)
- All vertices are integral, hence the two problems are equivalent.
- Fast shortest path solvers available.
- Adding prior knowledge is easy — clamp $\Pi_{ij}$ to 0 or 1.
Hungarian Marriage

**Integer Program**

\[
\text{minimize } \sum_{i,j} \Pi_{ij} C_{ij} \text{ subject to } \Pi_{ij} \in \{0; 1\} \text{ and } \Pi 1 = \Pi^T 1 = 1
\]

**Linear Programming Relaxation**

\[
\text{minimize } \sum_{i,j} \Pi_{ij} C_{ij} \text{ subject to } \Pi_{ij} \in [0, 1] \text{ and } \Pi 1 = \Pi^T 1 = 1
\]

**Properties**

- Can be solved in polynomial time (e.g. interior point)
- All vertices are **integral**, hence the two problems are equivalent.
- Fast shortest path solvers available.
- Adding prior knowledge is easy — clamp \( \Pi_{ij} \) to 0 or 1.
When Linear Assignment Fails

Alexander J. Smola: Learning Graph Matching 14 / 35
Failure Diagnosis

Why?
Graph matching is hard, so the Hungarian method (polynomial time algorithm) must fail.

What went wrong?
- Local features insufficient for matching.
- Symmetries create long range dependencies.
- Maybe we used the wrong matching score $C_{ij}$?

How bad is it really?
- Fails on degenerate problems with lots of symmetry.
- Works fine on graphs with enough characteristic features.
- **We should engineer $C_{ij}$ for specific problems.**
Quadratic Assignment

Key Idea

Use edge features for match.

Optimization Problem

\[
\text{minimize} \sum_{i,j} C_{ij} \Pi_{ij} + \sum_{i,j,u,v} Q_{ij,uv} \Pi_{ij} \Pi_{uv}
\]

Properties

- \(C_{ij}\) describes vertex feature match (as before)
- \(Q_{ij,uv}\) describes agreement between (potential) edges \((i, u)\) and \((j, v)\).
- For \(Q_{ij,uv} = 1 - \delta_{G_{iu}, G'_{jv}}\) we have exact matching.
- Problem is NP hard to solve.
Some Algorithms

Genetic algorithms
Tabu search
Ant colony systems
Any other really really desperate heuristic . . .

Graduated Assignment

- First order Taylor approximation of Quadratic Assignment problem is Linear Assignment problem.
- Take small steps.
- Iterative procedure (Sinkhorn, 1964) for small steps.

Semidefinite Relaxations

Not very scalable, $O(m^4)$ storage and $O(m^6)$ computation.

In practice . . .

Can only solve problems of size $< 100$. 
Key Idea

- Exact graph matching is too expensive.
- Linear assignment works if matching scores are good.
- **Use data to learn matching scores** $C_{ij}$.

Bottom line

Work hard to ask the right question not to find the answer for the wrong question. Use structured estimation.

We get **problem dependent scores**.
Outline

1. Applications
2. Algorithms
3. Changing the Question
4. Learning Graph Matching
5. Experiments
6. Extensions
Learning Problem

Optimization Problem

\[
\minimize_{C(\cdot, \cdot)} \sum_{i=1}^{m} \Delta(\Pi^i, 1) \text{ where } \Pi^i = \arg\min \sum_{\Pi} \Pi_{uv} C(V_u^i, V_v^i)
\]

The goal is to find a compatibility function \(C(\cdot, \cdot)\) such that graphs are perfectly matched. Obvious extensions for inexact matches — replace \(1\) by optimal match.

Loss Function

\[
\Delta(\Pi, \Pi') = \|\Pi - \Pi'\|^2 = 2(n - \text{tr} \Pi^\top \Pi')
\]

Obviously other loss functions are possible.

Problem

The optimization is \textit{nonconvex}. Even worse, it is \textit{piecewise constant}. Risk of \textit{overfitting}.
Learning Problem

Optimization Problem

$$\text{minimize} \sum_{i=1}^{m} \Delta(\Pi^i, \mathbf{1}) \text{ where } \Pi^i = \arg\min_{\Pi} \sum_{uv} \Pi_{uv} C(V_u^i, V_v^i)$$

The goal is to find a compatibility function $C(\cdot, \cdot)$ such that graphs are perfectly matched. Obvious extensions for inexact matches — replace $\mathbf{1}$ by optimal match.

Loss Function

$$\Delta(\Pi, \Pi') = \|\Pi - \Pi'\|^2 = 2(n - \text{tr} \, \Pi^\top \Pi')$$

Obviously other loss functions are possible.

Problem

The optimization is nonconvex. Even worse, it is piecewise constant. Risk of overfitting.
Learning Problem

Optimization Problem

\[
\minimize_{C(\cdot, \cdot)} \sum_{i=1}^{m} \Delta(\Pi^i, 1) \quad \text{where} \quad \Pi^i = \arg\min_{\Pi} \sum_{uv} \Pi_{uv} C(V^i_u, V^i_v)
\]

The goal is to find a compatibility function \( C(\cdot, \cdot) \) such that graphs are perfectly matched. Obvious extensions for inexact matches — replace 1 by optimal match.

Loss Function

\[
\Delta(\Pi, \Pi') = \|\Pi - \Pi'\|^2 = 2(n - \text{tr} \, \Pi^\top \Pi')
\]

Obviously other loss functions are possible.

Problem

The optimization is \text{nonconvex}. Even worse, it is \text{piecewise constant}. Risk of \text{overfitting}.
Regularization

Parametric Model for $C$

$$C(V_u, V_j) = \langle \phi(V_u, V_j), w \rangle$$

Regularizer

Assume that small $\|w\|$ corresponds to smooth functions $C$. Hence minimize regularized risk functional

$$\minimize_w \sum_{i=1}^{m} \Delta(\Pi^i, 1) + \lambda \|w\|^2$$
Structured Estimation

Original Objective Function

\[ \Delta(\Pi, 1) \text{ subject to } \Pi = \arg\min_{\Pi} \Pi^\top C \]

Convex Upper Bound

\[ \xi \text{ where } \xi \geq \text{tr}(1 - \Pi')^\top C + \Delta(\Pi', 1) \text{ for all } \Pi'. \]

To see that this is an upper bound, plug in \( \Pi' = \Pi \). The problem is convex in \( \xi \) and \( C \).

Optimization Problem

\[ \min_{w} \sum_{i=1}^{m} \xi_i + \lambda \| w \|^2 \]

subject to \( \xi_i \geq \text{tr}(1 - \Pi')^\top C(G^i, G^i) + 2(n - \text{tr} \Pi'_j) \text{ for all } \Pi'. \)
Issues

- Convex problem but . . .
- Exponential number of constraints
- Need to find most violated constraints efficiently

Column Generation

- Maximizing the constraint is linear assignment problem

\[
\maximize_{\Pi'} - \text{tr} \Pi'^\top [C(G^i, G^i) + 2 \cdot 1]
\]

- Recall that \( C(G^i, G^i) \) is a \textbf{compatibility} score.
- Problem made harder by adding \( 2 \cdot 1 \) to enforce margin.

Algorithm

- Minimize \( w \) for given set of constraints
- Find next set of worst constraints
Large baseline (no learning)
Large baseline (with learning)
Matching Performance

non-learning vs. learning with Linear Assignment and Graduated Assignment

- LA
- GA
- GA normalization 100
- GA normalization 0.1
- GA normalization 0.0001
- LA learning
- GA learning

Error (%) vs. training size/test size (5/106, 8/103, 12/99, 23/88, 37/74, 56/55, 55/56, 74/37, 88/23, 99/12, 103/8, 106/5)
Runtime

![Graph showing time and accuracy of methods](image)
Outline

1 Applications
2 Algorithms
3 Changing the Question
4 Learning Graph Matching
5 Experiments
6 Extensions
Immoral Hungarian Marriage aka $d$-matching
**Fact**

Our algorithm extends to $d$-matching, i.e. where we match every vertex with $d$ vertices of the corresponding graph.

**Million Dollar Question**

What would be a good application for this?
Collaborative Ranking

Setting
- Internet retailer (e.g. Netflix) sells movies $M$ to users $U$.
- Users rate movies if they liked them.
- Retailer wants to suggest some more movies which might be interesting for users.

Goal
Suggest movies that user will like. Pointless to recommend movies that users do not like since they are unlikely to rent.

Problems with Netflix contest
- Error criterion is uniform over all movies.
- Can only recommend a small number of movies at a time (probably no more than 10).
- Need to do well only on top scoring movies.

Insight
We can use linear assignment / sorting for ranking.
More Applications

Retail

eTailer (e.g. Amazon) wants to suggest new books and other products based on past purchase decisions and reviews.

Collaborative photo viewing site

Want to suggest some more photos user might want to see given past viewing behavior (e.g. Flickr.com).

Collaborative bookmark site

Suggest new bookmarks based on which ones users clicked at before. Do this in a personalized fashion. This immediately avoids click spam: only spammer is affected by spam clicks: each user gets his personalized view (e.g. Digg.com).

News site

Suggest new stories personalized to click behavior, such as news.{google, yahoo}.com.
1. Applications
2. Algorithms
3. Changing the Question
4. Learning Graph Matching
5. Experiments
6. Extensions