The Convergence of Structural and Statistical Pattern Recognition

Tibério Caetano

NICTA/Australian National University

ICPRAM, Vilamoura, Algarve, Portugal, 8 February 2012
Thanks to my collaborators

- L. Cheng
- Q. Le
- J. McAuley
- J. Petterson
- A. Ramisa
- A. Smola
Structural and Statistical Approaches to PR

- Statistical Pattern Recognition
  - Weakness: Simplistic representations (vectors)
  - Strength: Powerful and efficient manipulation of such representations (linear algebra)

- Structural Pattern Recognition
  - Strength: Complex representations (graphs, strings, taxonomies)
  - Weakness: Lack of efficient manipulation tools (discrete world is hard)

Can we have the best of both? Yes. This talk is a tour on some examples.
Structural and Statistical Approaches to PR

- **Statistical Pattern Recognition**
  - Weakness: Simplistic representations (vectors)
  - Strength: Powerful and efficient manipulation of such representations (linear algebra)

- **Structural Pattern Recognition**
  - Strength: Complex representations (graphs, strings, taxonomies)
  - Weakness: Lack of efficient manipulation tools (discrete world is hard)
Structural and Statistical Approaches to PR

- **Statistical Pattern Recognition**
  - Weakness: Simplistic representations (vectors)
  - Strength: Powerful and efficient manipulation of such representations (linear algebra)

- **Structural Pattern Recognition**
  - Strength: Complex representations (graphs, strings, taxonomies)
  - Weakness: Lack of efficient manipulation tools (discrete world is hard)

Can we have the best of both?
Structural and Statistical Approaches to PR

- **Statistical Pattern Recognition**
  - Weakness: Simplistic representations (vectors)
  - Strength: Powerful and efficient manipulation of such representations (linear algebra)

- **Structural Pattern Recognition**
  - Strength: Complex representations (graphs, strings, taxonomies)
  - Weakness: Lack of efficient manipulation tools (discrete world is hard)

**Can we have the best of both?**

Yes. This talk is a tour on some examples.
Outline

- Part I: The Structured Prediction Paradigm
- Part II: Structured Prediction in Action
  - Example 1: Learning to predict matchings
  - Example 2: Learning image taxonomies
  - Example 3: Learning to predict sets
- Future Directions
Part I:
The Structured Prediction Paradigm
Goal of pattern recognition: find predictor $f$

\[ f : \mathcal{X} \rightarrow \mathcal{Y} \]

where $\mathcal{X}$ is arbitrary and $\mathcal{Y}$ is discrete and finite.
Structured Prediction

- \( \mathcal{Y} \) is discrete but grows exponentially with the number of input variables.
The Supervised Learning approach to PR

- Assume you have corresponding instances from $\mathcal{X}$ and $\mathcal{Y}$: 
  $\{x^n, y^n\}$
  
  → Search for $f$ that is consistent with these correspondences
  → Constrain $f$ to be simple

\[
\text{argmin}_f \sum_n \ell(y^n, f(x^n)) + \Omega(f)
\]

- $f(x) \rightarrow$ prediction for input $x$
- $\ell(y, f(x)) \rightarrow$ loss incurred when predicting $f(x)$ instead of $y$
- $\Omega(f) \rightarrow$ penalty for complexity of $f$ (regularizer)
Linear Predictors

- Standard assumption: $f$ is a linear predictor of the form

$$f(x) \in \operatorname*{argmax}_{y \in Y} \langle \phi(x, y), \theta \rangle$$

- $\phi(x, y)$: arbitrary joint feature map (can be $\infty$-dimensional)
- $\theta$: parameterization
- $\langle \cdot, \cdot \rangle$: inner product

- Since $Y$ is exponentially large, computing $f(x)$ is a non-trivial combinatorial optimization problem. Hardness depends on $\phi$
The optimization problem now becomes

\[
\argmin_{\theta} \sum_{n} \ell(y^n, f_{\phi}(x^n; \theta)) + \Omega(\theta)
\]

- Design parameters: \( \ell, \phi \) and \( \Omega \)
- In this talk I focus only on \( \ell \) and \( \phi \)
- I will fix \( \Omega(\theta) = \lambda \|\theta\|^2 \)
The optimization problem now becomes

$$\arg\min_{\theta} \sum_n \ell(y^n, f_\phi(x^n; \theta)) + \Omega(\theta)$$

- Design parameters: $\ell$, $\phi$ and $\Omega$
- In this talk I focus only on $\ell$ and $\phi$
- I will fix $\Omega(\theta) = \lambda \|\theta\|^2$

(Francis will talk about structure in $\Omega$)
The optimization problem now becomes

\[ \argmin_{\theta} \sum_{n} \ell(y^n, f_\phi(x^n; \theta)) + \Omega(\theta) \]

Design parameters: \( \ell, \phi \) and \( \Omega \)

In this talk I focus only on \( \ell \) and \( \phi \)

I will fix \( \Omega(\theta) = \lambda \|\theta\|^2 \)

(Francis will talk about structure in \( \Omega \))

Much of ML is about designing \( \ell, \phi \) and \( \Omega \) and optimization algorithms to solve the resulting problem
Structural + Statistical PR

- We have a data-driven, statistical formulation
- $\phi$ will be designed so as to encode rich structures
- $\ell$ will be designed so as to compare structures in terms of which performance criterion we care about
The Optimization is Hard

$$\arg\min_\theta \sum_n \ell(y^n, f(x^n; \theta)) + \lambda \|\theta\|^2$$

- $\ell(y^n, f(x^n; \theta))$ only takes finitely many values (since $Y$ is finite)
- For $\theta$ continuous, we therefore have a piecewise-constant optimization problem $\rightarrow$ hard!
- For a given pair $(x^n, y^n)$, there are uncountably many $\theta$ with precisely the same loss $\ell(y^n, f(x^n; \theta))$
Relaxation: Convex Upper Bound

\[
[\theta^*, \xi^*] = \arg\min_{\theta, \xi} \left[ \sum_n \xi_n + \lambda \|\theta\|^2 \right]
\]

s.t. \( \langle \phi(x_n, y^n), \theta \rangle - \langle \phi(x_n, y), \theta \rangle \geq \ell(y, y^n) - \xi_n \)
\[ \xi_n \geq 0 \]
\[ \forall n, y \in \mathcal{Y} \]

**Theorem:** \( \ell(y^*_n, y^n) \leq \xi^*_n \)

where \( y^*_n = \arg\max_y \langle \phi(x^n, y), \theta^* \rangle \)

[Tsochantaridis et al. JMLR '05]
Relaxation: Convex Upper Bound

\[
[\theta^*, \xi^*] = \arg\min_{\theta, \xi} \left[ \sum_n \xi_n + \lambda \|\theta\|^2 \right]
\]

s.t. \[ \langle \phi(x^n, y^n), \theta \rangle - \langle \phi(x^n, y), \theta \rangle \geq \ell(y, y^n) - \xi_n \]
\[ \xi_n \geq 0 \]
\[ \forall n, \forall y \in Y \]
Relaxation: Convex Upper Bound

\[ [\theta^*, \xi^*] = \arg \min_{\theta, \xi} \left[ \sum_n \xi_n + \lambda \|\theta\|^2 \right] \]

s.t. \( \langle \phi(x^n, y^n), \theta \rangle - \langle \phi(x^n, y), \theta \rangle \geq \ell(y, y^n) - \xi_n \)

\( \xi_n \geq 0 \)

\( \forall n, \forall y \in Y \)
Relaxation: Convex Upper Bound

\[
[\theta^*, \xi^*] = \underset{\theta, \xi}{\text{argmin}} \left[ \sum_n \xi_n + \lambda \|\theta\|^2 \right]
\]

s.t. \hspace{1cm} \langle \phi(x^n, y^n), \theta \rangle - \langle \phi(x^n, y), \theta \rangle \geq \ell(y, y^n) - \xi_n

\xi_n \geq 0

\forall n, \forall y \in Y

Exponentially many constraints!
Solution: Constraint Generation

- We can proceed by constraint generation.
- Start with no constraints and iteratively add most violated constraint for the current solution.
- $\epsilon$-approximation to optimal solution in $O(\epsilon^{-1})$ iterations.
Solution: Constraint Generation

- Most violated constraint: $y$ that maximizes violation gap $\xi$

$$y_{*n} = \arg\max_y \xi_n(y)$$

$$= \arg\max_y [\ell(y, y^n) + \langle \phi(x^n, y), \theta \rangle]$$
Solution: Constraint Generation

\[ [\theta^1, \xi^1] = \arg\min_{\theta, \xi} \left[ \sum_n \xi_n + \lambda \|\theta\|^2 \right] \]

s.t.

\[ \langle \phi(x^n, y^n), \theta \rangle - \langle \phi(x^n, y^0), \theta \rangle \geq \ell(y^0, y^n) - \xi_n \]
\[
[\theta^2, \xi^2] = \arg\min_{\theta, \xi} \left[ \sum_n \xi_n + \lambda \|\theta\|^2 \right]
\]
s.t.
\[
\langle \phi(x^n, y^n), \theta \rangle - \langle \phi(x^n, y^0_\ast), \theta \rangle \geq \ell(y^0_\ast, y^n) - \xi_n
\]
\[
\langle \phi(x^n, y^n), \theta \rangle - \langle \phi(x^n, y^1_\ast), \theta \rangle \geq \ell(y^1_\ast, y^n) - \xi_n
\]
Solution: Constraint Generation

$$[\theta^3, \xi^3] = \arg\min_{\theta, \xi} \left[ \sum_n \xi_n + \lambda \|\theta\|^2 \right]$$

s.t.

$$\langle \phi(x^n, y^n), \theta \rangle - \langle \phi(x^n, y_*^0), \theta \rangle \geq \ell(y_*^0, y^n) - \xi_n$$

$$\langle \phi(x^n, y^n), \theta \rangle - \langle \phi(x^n, y_*^1), \theta \rangle \geq \ell(y_*^1, y^n) - \xi_n$$

$$\langle \phi(x^n, y^n), \theta \rangle - \langle \phi(x^n, y_*^2), \theta \rangle \geq \ell(y_*^2, y^n) - \xi_n$$
Solution: Constraint Generation

\[
\begin{align*}
[\theta^{t+1}, \xi^{t+1}] &= \arg \min_{\theta, \xi} \left[ \sum_n \xi_n + \lambda \|\theta\|^2 \right] \\
\text{s.t.} \\
&\langle \phi(x^n, y^n), \theta \rangle - \langle \phi(x^n, y^0_*), \theta \rangle \geq \ell(y^0_*, y^n) - \xi_n \\
&\langle \phi(x^n, y^n), \theta \rangle - \langle \phi(x^n, y^1_*), \theta \rangle \geq \ell(y^1_*, y^n) - \xi_n \\
&\langle \phi(x^n, y^n), \theta \rangle - \langle \phi(x^n, y^2_*), \theta \rangle \geq \ell(y^2_*, y^n) - \xi_n \\
&\cdots \\
&\langle \phi(x^n, y^n), \theta \rangle - \langle \phi(x^n, y^t_*), \theta \rangle \geq \ell(y^t_*, y^n) - \xi_n
\end{align*}
\]
In summary, you have to:

- Specify $\ell$ and $\phi$ that make sense for your problem
- Find an efficient algorithm to solve

$$\arg\max_y [\ell(y, y^n) + \langle \phi(x^n, y), \theta \rangle]$$
Part II:
Structured Prediction in Action
Example 1:
Learning Graph Matching

Joint work with J. McAuley, L. Cheng, Q. Le and A. Smola
Example 1:
Learning Graph Matching
Graph Matching

- Classical problem in Structural PR
- Given two graphs $G = (V, E)$ and $G' = (V', E')$
- Given attributes of vertices: $v : V \rightarrow \mathbb{R}^n$
- Given attributes of edges: $e : E \rightarrow \mathbb{R}^n$
- Problem: Find a ‘consistent’ set of correspondences $C$ between the vertices of $G$ and $G'$: $C \subseteq V \times V'$
- Typical assumption: $C$ is an injective function, or a one-to-one mapping (can be relaxed).
Graph Matching

- Most popular formulation: Quadratic Assignment Problem (QAP)
- Assemble ‘Compatibility Function’ or ‘Affinity Function’ between pairs of associations:

$$A_{ijkl} = f(v_i, v_j, v_k, v_l, e_{ij}, e_{kl})$$

- Solution space: set of permutation matrices $y$:
  $$\mathcal{Y} = \{ y : y_{ij} \in \{0, 1\}, y1 = 1, y^T 1 = 1 \}$$
- Goal: solve quadratic assignment problem:

$$\arg\max_{y \in \mathcal{Y}} \sum_{ijkl} A_{ijkl} y_{ik} y_{jl}$$
Graph Matching

- Most popular formulation: Quadratic Assignment Problem (QAP)
- Assemble ‘Compatibility Function’ or ‘Affinity Function’ between pairs of associations:

\[ A_{ijkl} = f(v_i, v_j, v_k, v_l, e_{ij}, e_{kl}) \]

- Solution space: set of permutation matrices \( y \):

\[ y = \{ y : y_{ij} \in \{0, 1\}, y_1 = 1, y^T1 = 1 \} \]

- Goal: solve quadratic assignment problem:

\[ \arg\max_{y \in Y} \sum_{ijkl} A_{ijkl} y_{ik} y_{jl} \]

\( \text{NP-hard!} \)
Lots of algorithms to find an approximate solution to QAP:

[Zaslavskiy et al. '09], [Gold and Rangarajan '06], [Pelillo '99],
[Wilson and Hancock '97], [Messmer and Bunke '98],
[Leordeanu and Hebert '05], [Cour et al. '06], [Caelli and Kosinov '02], [Caetano et al. '04]

... 100's of papers
Graph Matching

- **Lots** of algorithms to find an approximate solution to QAP:
  
  [Zaslavskiy et al. '09], [Gold and Rangarajan '06], [Pelillo '99], [Wilson and Hancock '97], [Messmer and Bunke '98], [Leordeanu and Hebert '05], [Cour et al. '06], [Caelli and Kosinov '02], [Caetano et al. '04]

  ... 100's of papers

- But... **do we really need** QAP?
Graph Matching

- Isomorphism easily modeled as QAP: $A_{ijkl} = e_{ij}e_{kl}$
  ($e_{ij} \in \{0, 1\}$, $e_{ij} = 1$ iff there is an edge between $i$ and $j$)

However, if we use the right features, we can solve isomorphism as a linear assignment problem (LAP): argmax $y \sum ikd_{ik}y_{ik}$, where $d_{ik} = \text{degree}(i) = \text{degree}(k)$. LAP is polynomial!
Graph Matching

- Isomorphism easily modeled as QAP: $A_{ijkl} = e_{ij}e_{kl}$ ($e_{ij} \in \{0, 1\}$, $e_{ij} = 1$ iff there is an edge between $i$ and $j$)

- However, if we use the right features, we can solve isomorphism as a linear assignment problem (LAP): $\arg\max_y \sum_{ik} d_{ik}y_{ik}$, where $d_{ik} = [\text{degree}(i) = \text{degree}(k)]$. LAP is polynomial!
Graph Matching

- **Problem:** If we have symmetries, then LAP will not recover the results of QAP
Graph Matching

- **Problem:** If we have symmetries, then LAP will not recover the results of QAP

- Obvious question: given a real, complex graph matching problem, is it possible to find measurements such that LAP gives the same solution as QAP?

▶ Yes: use data to brake symmetry

▶ Since real graphs contain rich vertex and edge attributes, there probably exist measurements of a graph that can make LAP produce the same results as QAP.

▶ Let's find them through machine learning!
Graph Matching

- **Problem:** If we have symmetries, then LAP will not recover the results of QAP

- Obvious question: given a real, complex graph matching problem, is it possible to find measurements such that LAP gives the same solution as QAP?

- **Yes:** use data to brake symmetry
Graph Matching

- **Problem:** If we have symmetries, then LAP will not recover the results of QAP

- Obvious question: given a real, complex graph matching problem, is it possible to find measurements such that LAP gives the same solution as QAP?

- **Yes:** use data to brake symmetry

- Since real graphs contain rich vertex and edge attributes, there probably exist measurements of a graph that can make LAP produce the same results as QAP.

- Let’s find them through machine learning!
Learning Graph Matching

- Let’s parameterize the graph attributes (and any other measurements of the graph attributes)

\[ A_{ijkl} = f(v_i, v_j, v_k, v_l, e_{ij}, e_{kl}, \theta) \]

- Let’s manually provide the correct matchings for several graph matching problems

- Let’s estimate \( \theta \) so as to minimize some matching loss in the training matches (plus regularization)

[Caetano et al. ICCV’07, PAMI’09]
Loss, Feature Map and Constraint Generation

- $\ell(y, \bar{y}) = 1 - \frac{\langle y, \bar{y} \rangle}{\langle y, y \rangle}$: fraction of incorrect correspondences

- $\phi(x^n, y) = \sum_{ij} y_{ij} h_{ij}(G^n, G'^n)$, where $h$ is any vector of measurements of the graphs $G$ and $G'$ from the ‘perspective’ of vertices $i$ and $j$ (for example one of the entries in $h_{ij}$ can be $[\text{degree}(i) = \text{degree}(j)]$)

- Since both $\ell$ and $\phi$ are linear in $y$, constraint generation becomes a LAP:

$$\text{argmax} \sum_{ij} c^n_{ij} y_{ij}$$

where $c^n_{ij} = \langle h^n_{ij}, \theta \rangle + y^n_{ij} / \langle y^n_{ij}, y^n_{ij} \rangle$
Results on CMU ‘Hotel’ dataset

Before Learning:

After Learning:

RED: Mistakes
Results

House (baseline = 90)

- Linear
- Linear+Learning
- Quadratic
- Quadratic+Learning
- Quadratic normalisation 0.00001
- SMAC (MATLAB)
Example 2:
Learning Image Taxonomies

Joint work with J. McAuley and A. Ramisa
mammal → placental → carnivore → canine → dog → working dog → husky

vehicle → craft → watercraft → sailing vessel → sailboat → trimaran
Data and Annotation

- **ImageNet** dataset

- Each image manually annotated with only **one** category

- Each image often contains **more than one category** though

- Categories are organised in a taxonomy tree
The Loss Function

- The loss should account for the imperfect labelling

- If prediction is ‘cat’ in an image with both dogs and cats, but which is labeled as ‘dog’, the prediction should still be considered correct

\[ \ell(Y, y_n) = \min_{y \in Y} d(y, y_n) \]

- $Y$ is the set of 5 predicted categories; $y$ is one of them
- $y_n$ is the annotated category
- $d(y, y_n)$ is the distance between node $y$ and the nearest common ancestor between $y$ and $y_n$
The Loss Function

- The loss should account for the imperfect labelling

- If prediction is ‘cat’ in an image with both dogs and cats, but which is labeled as ‘dog’, the prediction should still be considered correct

- **Imagenet Competition**: Organisers proposed the following error measure:
  - An algorithm is allowed to predict 5 categories
  - The loss of a prediction is:

  \[
  \ell(Y, y^n) = \min_{y \in Y} d(y, y^n)
  \]
The Loss Function

- The loss should account for the imperfect labelling

- If prediction is ‘cat’ in an image with both dogs and cats, but which is labeled as ‘dog’, the prediction should still be considered correct

- **Imagenet Competition**: Organisers proposed the following error measure:
  - An algorithm is allowed to predict 5 categories
  - The loss of a prediction is:
    \[
    \ell(Y, y^n) = \min_{y \in Y} d(y, y^n)
    \]
  - \(Y\) is the set of 5 predicted categories; \(y\) is one of them
  - \(y^n\) is the annotated category
  - \(d(y, y^n)\) is the distance between node \(y\) and the nearest common ancestor between \(y\) and \(y^n\)
\[ d(y, y^n) = 2 \]
Our Approach

- We directly optimize the competition error measure (loss function)
- We again use the convex upper bound trick
- Constraint generation needs a dedicated algorithm
Constraint Generation

\[
\hat{Y}^n = \arg\max_{Y \in y} \left\{ \min_{y \in Y} d(y, y^n) + \sum_{y \in Y} \langle \phi(x^n, y), \theta \rangle \right\}
\]

- Needs a dedicated algorithm. We developed an algorithm that is exact and efficient.

[McAuley, Ramisa and Caetano EMMCVPR'11]
Results: 1024-dimensional feature vector

- Approach: re-weight parameters obtained with binary SVMs by competition winners [Perronin and Sanchez ’11] using the structured loss

![Graphs showing reweighting of 1024 dimensional features and training error reduction with latent variables.]

<table>
<thead>
<tr>
<th></th>
<th>1nn</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before learning</td>
<td>11.35</td>
<td>9.29</td>
<td>8.08</td>
<td>7.25</td>
<td>6.64</td>
</tr>
<tr>
<td>After learning</td>
<td>10.88</td>
<td>8.85</td>
<td>7.71</td>
<td>6.93</td>
<td>6.36</td>
</tr>
</tbody>
</table>
Results: 4096-dimensional feature vector

Many additional results upcoming

[McAuley, Ramisa and Caetano ’12]
Example 3:
Learning to Predict Sets

Joint work with J. Petterson
Predicting sets

- Example: Image Tagging
- Input: Image
- Output: A set of tags
- Tag vocabulary of size \( V \)
- \( 2^V \) possible outputs
Predicting sets

chess, WTC, NY

Kangaroo, Sun, Sea, Australia

crocodile, water, green
Predicting sets

- It’s important to predict the correct labels (recall)
- It’s important not to predict incorrect labels (precision)
- $F$-score: harmonic mean of precision and recall
Loss Function

- $y \in \{0, 1\}^V$ indicates which labels are predicted

- Recall and Precision:
  
  \[ R = \frac{y^T \bar{y}}{y^T y} \quad \quad P = \frac{\bar{y}^T \bar{y}}{\bar{y}^T \bar{y}} \]

- $F$-score is the harmonic mean of precision and recall, so a corresponding loss is

  \[ \ell_F(y, \bar{y}) = 1 - \frac{2PR}{P + R} = 1 - \left[ \frac{2y^T \bar{y}}{y^T y + \bar{y}^T \bar{y}} \right] \]

  \(F\)-score
It’s important to have parameters associating the input image with individual tags (blue images more likely to have sky tag) [Petterson and Caetano NIPS’10]

In addition we would like to explicitly model tag dependencies (if there is strong evidence for tag ship then tag ocean should become more likely) [Petterson and Caetano NIPS’11]

We encode in $\phi$ both unary tag features and pairwise tag dependencies
Constraint Generation

- If we constraint tag dependencies to be \textit{submodular}, then we can find an approximation algorithm for constraint generation which is provably very accurate.

- This basically means that we can model positive pairwise correlations, i.e., the fact that \texttt{wheel} and \texttt{car} co-occur more often than not.

- For generic pairwise dependencies, constraint generation is completely intractable.
Constraint Generation

- Constraint generation is an integer quadratic program
  \[ y = \arg \max_{y} y^T A(y) y \]

- Off-diagonal elements of \( A \) are forced to be **positive** (submodularity) and are independent of \( y \)

- **Certain elements** of the diagonal of \( A \) depend on the cardinality of \( y \), \( \sum_i y_i \)

- This unique setting requires a dedicated algorithm [Petterson and Caetano NIPS’11]
### Results

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Ours</th>
<th>CCA</th>
<th>CC</th>
<th>BM</th>
<th>SM</th>
<th>MS</th>
<th>ECC</th>
<th>EBM</th>
<th>EPS</th>
<th>RAKEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yeast</td>
<td>0.440</td>
<td>0.346</td>
<td>0.346</td>
<td>0.326</td>
<td>0.327</td>
<td>0.331</td>
<td>0.362</td>
<td>0.364</td>
<td>0.420</td>
<td>0.413</td>
</tr>
<tr>
<td>Scene</td>
<td>0.671</td>
<td>0.374</td>
<td>0.696</td>
<td>0.685</td>
<td>0.666</td>
<td>0.694</td>
<td>0.742</td>
<td>0.729</td>
<td>0.763</td>
<td>0.750</td>
</tr>
<tr>
<td>Medical</td>
<td>0.420</td>
<td>-</td>
<td>0.377</td>
<td>0.364</td>
<td>0.321</td>
<td>0.370</td>
<td>0.386</td>
<td>0.382</td>
<td>0.324</td>
<td>0.377</td>
</tr>
<tr>
<td>Enron</td>
<td>0.243</td>
<td>-</td>
<td>0.198</td>
<td>0.197</td>
<td>0.144</td>
<td>0.198</td>
<td>0.201</td>
<td>0.201</td>
<td>0.155</td>
<td>0.206</td>
</tr>
</tbody>
</table>

[Peterson and Caetano NIPS’10]
Results

% of unary features used for training

F-Score

yeast

Ours
RML
ML-KNN
RaKEL
BR

[Petterson and Caetano NIPS’11]
Summary

- Structural and Statistical Pattern Recognition can coexist

- Structured prediction seems to be a good starting point

- The optimization of structured losses with structured feature maps is possible and improves over non-structured formulations

- Tractable classes of combinatorial optimization problems can deliver basically the same results as non-tractable classes if we use statistics to estimate the objective function from data (such as replacing QAP for LAP formulations in graph matching)
Summary

- Structural and Statistical Pattern Recognition can coexist

- Structured prediction seems to be a good starting point

- The optimization of structured losses with structured feature maps is possible and improves over non-structured formulations

- Tractable classes of combinatorial optimization problems can deliver basically the same results as non-tractable classes if we use statistics to estimate the objective function from data (such as replacing QAP for LAP formulations in graph matching)
Future Directions
Future Directions

- Structured prediction should be seen as a somewhat primitive attempt to **incorporate Statistics into Optimization**

- What we are really trying to do is to **estimate from data which optimization problem we should be solving at prediction time**

- Optimization is **deduction**, Statistical Inference is **induction**. Intelligence relies on both. If we make serious progress in unifying these fields, we may have a conceptually **complete** basis for Artificial Intelligence, which may lead to AI systems with human-like capabilities and beyond.