Supervised Learning

- Find function from input space $X$ to output space $Y$

\[ h : X \rightarrow Y \]

such that the prediction error is low.

Examples of Complex Output Spaces

- **Natural Language Parsing**
  - Given a sequence of words $x$, predict the parse tree $y$.
  - Dependencies from structural constraints, since $y$ has to be a tree.

```
3 The dog chased the cat
```

```
3 2
NP S VP NP
Det N V Det N
```

- **Part-of-Speech Tagging**
  - Given a sequence of words $x$, predict sequence of tags $y$.
  - Dependencies from tag-tag transitions in Markov model.

```
3 The bear chased the cat
```

```
3 2
Det N V Det N
```

Examples of Complex Output Spaces

- **Multi-Label Classification**
  - Given a (bag-of-words) document $x$, predict a set of labels $y$.
  - Dependencies between labels from correlations between labels (“iraq” and “oil” in newswire corpus).

```
3 Due to the continued violence in Baghdad, the oil price is expected to further increase.
OPEC officials met with …
```

```
3 2
antarctica +1 benelux -1 germany +1 iraq +1 oil -1 coal -1 trade -1 acquisitions
```

Examples of Complex Output Spaces

- **Non-Standard Performance Measures (e.g. $F_\beta$-score, Lift)**
  - $F_\beta$-score: harmonic average of precision and recall

\[ F_\beta = \frac{2 \times \text{Precision} \times \text{Recall}}{\beta^2 \times \text{Precision} + \text{Recall}} \]

- New example vector $\mathbf{y}$.
  - Predict $y_3 = 1$, if $P(y_3 = 1 | \mathbf{x}) = 0.4$?
  - Depends on other examples!
Examples of Complex Output Spaces

- **Noun-Phrase Co-reference**
  - Given a set of noun phrases $x$, predict a clustering $y$.
  - Structural dependencies, since prediction has to be an equivalence relation.
  - Correlation dependencies from interactions.

![Noun-Phrase Co-reference Example](image)

Why do we Need Research on Complex Outputs?

- Important applications for which conventional methods don’t fit!
  - Noun-phrase co-reference: two-step approaches of pair-wise classification and clustering as postprocessing, e.g. [Ng & Candie, 2002]
  - Directly optimize complex loss functions (e.g. $F_1$, AvgPrec)

- Improve upon existing methods!
  - Natural language parsing: generative models like probabilistic context-free grammars
  - SVM outperforms Naïve Bayes for text classification [Joachims, 1998]

<table>
<thead>
<tr>
<th>Model</th>
<th>Precision/Recall</th>
<th>Break-Even Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naïve Bayes</td>
<td>72.1</td>
<td>87.5</td>
</tr>
<tr>
<td>Linear SVM</td>
<td>82.0</td>
<td>90.3</td>
</tr>
<tr>
<td>Support Vector Machines</td>
<td>62.4</td>
<td>71.6</td>
</tr>
</tbody>
</table>

Overview

- Task: Learning to predict complex outputs
  - Formalizing the problem
    - Multi-class classification: Generative vs. Discriminative
  - Training models with structured outputs
    - Generative training
    - SVMs and large-margin training
    - Conditional likelihood training

- Example 1: Learning to parse natural language
- Learning weighted context free grammar
- Example 2: Optimizing $F_1$-score in text classification
- Predict vector of class labels
- Example 3: Learning to cluster
- Learning a clustering function that produces desired clusterings

Structured Output Prediction as Multi-Class Classification

- **Learning Task**: $P(X,Y) = P(X) P(Y|X)$
  - Input Space: $X$ (i.e. feature vectors, word sequence, etc.)
  - Output Space: $Y$ (i.e. class, tag sequence, parse tree, etc.)
  - Training Data: $S = (x_1, y_1), ..., (x_n, y_n) \sim_{iid} P(X,Y)$

- **Approach**: view as multi-class classification task
  - Every complex output $y \in Y$ is one class

- **Goal**: Find $h: X \rightarrow Y$ with low expected loss
  - Loss function: $\Delta(x, y')$ (penalty for predicting $y'$ if $y$ correct)
  - Expected loss (i.e. Risk):
    $$\mathbb{E}_{(x,y) \sim S\Delta(x,y)} = \sum_{x \in X, y \in Y} \frac{\Delta(x, y)}{S} P(X=x) = 0$$

Generative Model: Model $P(X,Y)$

- Bayes’ Decision Rule: Optimal Decision is
  $$\hat{y} = \arg\max_{y} \sum_{x \in X} \frac{P(X=x) P(Y=y|X)}{P(Y=y)}$$

- Equivalent Reformulations: For $0(1 - \text{Loss} \Delta(x,y')) = 1$, if $y \neq y'$, 0 else
  $$\Delta(x) = \arg\max_{y, y'} \frac{P(Y=y | X=x) P(X=x)}{\sum_{y' \in Y} P(Y=y' | X=x) P(X=x)}$$

- Learning: maximum likelihood (or MAP, or Bayesian)
  - Assume model class $P(X,Y|\omega)$ with parameters $\omega \in \Omega$
  - Find
    $$\omega = \arg\max_{\omega} \prod_{(x,y) \sim S} \left[ P(Y=y | X=x, \omega) \right]$$

Natural Language Parsing as Multi-Class Classifications

![Natural Language Parsing Example](image)
Naïve Bayes’ Classifier (Multivariate)
- Input Space $X$: Feature Vector
- Output Space $Y$: $\{1,-1\}$
- Model:
  - Prior class probabilities $P(Y=1), P(Y=-1)$
  - Class conditional model (one for each class)
    $$P(X|Y=1) = \prod \frac{P(X = x^i|Y=1)}{P(X = x^i|Y=0)}$$
  - Classification rule:
    $$h(x) = \arg \max_{y \in \Omega} \{P(Y=y) \prod_{x \in X} P(X = x|Y=y)\}$$

Bayes’ Decision Rule:
- Input Space $X$: Feature Vector
- Output Space $Y$: $\{1,-1\}$
- Model discriminant functions $P(Y|X)$
- Discriminative Model: Model $P(Y|X)$
  - Optimal Decision:
    - Assume 0/1 Loss
    - Class conditional model (one for each class)
      $$P(Y|X) = \frac{P(Y=1|X)}{P(Y=-1|X)}$$
    - Prior class probabilities
  - Classification rule:
    $$h(x) = \arg \max_{y \in \Omega} P(Y=y|X=x)$$

Discriminative Model: Model $P(Y|X)$
- Bayes’ Decision Rule:
  - Assume 0/1 Loss $\mathbb{I}(y) = 1$, if $y \neq y'$, 0 else
  - Optimal Decision:
    $$h(x) = \arg \max_{y \in \Omega} P(Y=y|X=x)$$
- Learning: maximum likelihood (or MAP, or Bayesian)
  - Assume model class $P(Y,X,o)$ with parameters $o \in \Omega$
  - Find
    $$\omega^* = \arg \max_{\omega \in \Omega} \sum_{x,y} P(Y=y,X=x,o)$$
- Example: Logistic regression classifier
  - Assume
    $$P(Y = y|X = x, o) = \frac{1}{1 + e^{-x \cdot o^T}}$$

Generative vs. Discriminative Models
- Learning Task:
  - Generator: Generate descriptions according to distribution $P(X)$.
  - Teacher: Assigns a value to each description based on $P(Y|X)$.
- Training Examples $(x_1, y_1), \ldots, (x_n, y_n) \sim P(X, Y)$
- Discriminative Model
  - Model $P(Y|X)$ with $P(Y|X,o)$
    - Find $o \in \Omega$ via MLE
    - Examples: Log. Reg., CRF
  - Model discriminant functions $h(x)$ with low train loss (e.g. Emp. Risk Min.)
    - Examples: naive Bayes, HMM
  - Prediction:
    $$h(x) = \arg \max_{y \in \Omega} P(Y=y|X=x,o)$$

Challenges in Learning with Complex Outputs
- Approach: view as multi-class classification task
  - Every complex output $y \in Y$ is one class
  - Example: The bear chased the cat 
  - Prediction: $h(x) = \arg \max_{y \in \Omega} P(Y=y|X=x)$
- Problem: Exponentially many classes!
  - Generative Model: $P(X,Y)$
  - Discriminative Model: $P(Y|X)$
  - Discriminant Functions $h: X \times Y \rightarrow \Omega$
- Challenges:
  - How to compactly represent model?
  - How to do efficient inference with model (i.e. compute $P(Y|x,o)$)?
  - How to effectively estimate model from data? (e.g. compute $P(Y=x,o)$)
Natural Language Parsing

- Input Space X: Sequences of words
- Output Space Y: Trees over sequences
- Problem:
  How to compute predictions

\[ \lambda(x) = \arg \max_{y \in Y} P(x,y) \]

efficiently?

Multi-Class Linear Discriminant

- Linear discriminant function of the form:
  \[ h(x) = \arg \max_{y \in Y} [\sum \phi(x,y)] \]

Joint Feature Map

- Feature vector \( \phi_n(x,y) \) that describes match between \( x \) and \( y \)
- Linear discriminant function of the form:
  \[ h(x) = \arg \max_{y \in Y} [\sum \phi(x,y)] \]

Joint Feature Map for Trees

- Weighted Context Free Grammar
  - Each rule \( \text{r}_i \) (e.g. \( S \rightarrow NP \)) has a weight
  - Score of a tree is the sum of its weights
  - Find highest scoring tree

Joint Feature Map for Sequences

- Linear Chain Model
  - Only local dependencies
  - Score for each adjacent label/label and word/label pair
  - Find highest scoring sequence

\[ h(x) = \max_{y \in Y} P(x,y) \]

Connection to Graphical Models

Hidden Markov Model:

- Assumptions
  - \[ \prod_{t=1}^{L} P(Y_t=x_t|Y_{t-1}=y_{t-1}) \]
  - Rule: \[ h(x) = \arg \max_{y \in Y} [\sum \phi(x,y)] \]

with \( w_{j,y} = -\log P(Y_j=x_j|Y_{j-1}=y_{j-1}) \) and \( w_{i,x} = -\log P(Y_i=x_i|Y_{i-1}=y_{i-1}) \)
and \( \phi(x,y) \) histogram
Overview

- Task: Learning to predict complex outputs
- Formulating the problem
  - Multi-class classification: Generative vs. Discriminative
  - Compact models for problems with structured outputs
- Training models with structured outputs
  - Generative training
  - SVMs and large-margin training
  - Conditional likelihood training
- Example 1: Learning to parse natural language
  - Learning weighted context free grammar
- Example 2: Optimizing F1-score in text classification
  - Predict vector of class labels
- Example 3: Learning to cluster
  - Learning a clustering function that produces desired clusterings
- Summary & Reading

Training Generative Model: HMM

- Assume:
  - model class P(X,Y|α) with parameters α ∈ Ω
- Maximum Likelihood (or alternative estimator)
  - Find \( \omega^* = \arg \max_{\omega} \prod_{n=1}^{N} P(Y_n|X_n = \omega) \)
- Example: Hidden Markov Model
  - Closed-form solutions
    - \( P(Y = y_i | X = x_j) = \frac{2 \text{times} \text{State}\_\text{B} \text{follows} \text{State}\_\text{B}}{2 \text{times} \text{State}\_\text{B} \text{occurs}} \)
    - \( P(X = x_i | Y = y_j) = \frac{2 \text{times} \text{Output}\_\text{A} \text{occurs} \text{State}\_\text{B}}{2 \text{times} \text{State}\_\text{B} \text{occurs}} \)
  - Need for smoothing the estimates (e.g. max a posteriori)

Support Vector Machine

- Training Examples: \( (x_1, y_1), \ldots, (x_n, y_n) \) \( \in \mathbb{R}^d \times \{1, \ldots, K\} \)
- Hypothesis Space: \( \mathcal{H} = \{ w \in \mathbb{R}^d \times \mathbb{R} \} \)
- Training: Find hyperplane \( \mathbf{w}, b \) with minimal
  \[ \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^{n} \xi_i \]

\[ \begin{align*}
\mathbf{w}^T \mathbf{x}_i + b & \geq 1 - \xi_i \\
\text{if } y_i = 1 \\
\mathbf{w}^T \mathbf{x}_i + b & \leq -1 + \xi_i \\
\text{if } y_i = -1
\end{align*} \]

Multi-Class SVM

- Training Examples: \( (x_1, y_1), \ldots, (x_n, y_n) \) \( \in \mathbb{R}^d \times \{1, \ldots, K\} \)
- Hypothesis Space: \( \mathcal{H} = \{ \phi(x,y) \in \mathbb{R}^d \times \mathbb{R} \} \)

\[ \begin{align*}
\phi(x,y) & = \mathbf{w}_y^T \phi(x) \\
\mathbf{w}_y & = \sum_{i=1}^{n} \alpha_i y_i \mathbf{1}_{y_i = y} \mathbf{x}_i
\end{align*} \]

Structural Support Vector Machine

- Joint features \( \phi(x,y) \) describe match between \( x \) and \( y \)
- Learn weights \( \mathbf{w} \) so that \( \mathbf{w} \phi(x,y) \) is max for correct \( y \)
Loss Functions: Soft-Margin Struct SVM

- Loss function $\Delta(x, y)$ measures match between target and prediction.

Training Approach 1: Factored QP

- Assume:
  - Linearly decomposable loss function $\Delta(x, y) = \sum_i \Delta_i(x_i, y_i)
  - Linear program solution is integral:
    $$\sum_i \Delta_i(x_i, y_i) = \min_{x, y} \left( \sum_i \Delta_i(x_i, y_i) \right)$$

- Algorithm:
  - Min-Max Formulation:
    $$\min_{x, y} \left( \sum_i \Delta_i(x_i, y_i) \right) = \frac{1}{\varepsilon}$$
  - Linear program for “max”
  - Quadratic program can be rewritten so that it has
    - Polynomially many variables
    - Polynomially many constraints

Training Approach 2: Polynomial Sparsity Bound

- Theorem: The sparse-approximation algorithm finds a solution to the soft-margin optimization problem after adding at most $\alpha \log(d + 1) \log(d + \rho)$ constraints to the working set $S$, so that the Kuhn-Tucker conditions are fulfilled up to a precision $\varepsilon$. The loss has to be bounded $0 \leq x_i \leq \rho$ and $\rho \leq n_{y_i}$ for each $y_i$.
Experiment: Natural Language Parsing

- Implementation
  - Implemented Sparse-Approximation Algorithm in SVMstruct
  - Incorporated modified version of Mark Johnson’s CKY parser
  - Learned weighted CFG with $C = 1$
- Data
  - Penn Treebank sentences of length at most 10 (start with POS)
  - Train on Sections 2.22–4098 sentences
- Test on Section 23: 163 sentences

<table>
<thead>
<tr>
<th>Method</th>
<th>Test Accuracy</th>
<th>Training Efficiency</th>
<th>CPU-h</th>
<th>Iter</th>
<th>Const</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCFG with MLE</td>
<td>55.2</td>
<td>86.0</td>
<td>0</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>SVM with (1-F1)-Loss</td>
<td>58.9</td>
<td>88.5</td>
<td>3.4</td>
<td>12</td>
<td>8043</td>
</tr>
</tbody>
</table>

More Expressive Features

- Linear composition:
  - $\phi(x,y) = \sum \phi(x,y)_{i}$
- General form:
  - $\phi(x,y) = \sum \phi(a,y|x_0,y_0,...,y_m)$
- So far:
  - $\phi(x,y) = \begin{cases} 1 & \text{if } \text{rule}(y) = ' N \rightarrow M F $ V P' \\ 0 & \text{otherwise} \end{cases}$
- Example:
  - $\phi(x,y) = \begin{cases} 1 & \text{if } \text{rule}(y) = ' N \rightarrow M F $ V P' \\ 0 & \text{otherwise} \end{cases}$

Applying Structural SVM to New Problem

- Application specific
  - Loss function $\mathcal{L}(y|x,a)$
  - Representation $\Phi(x,y)$
  - Algorithms to compute
    - $\hat{y} = \arg\max_{y} \mathcal{L}(y|x,a)$
    - $z = \arg\max_{y} \mathcal{L}(y|x,a) + \gamma \Phi(x,y)$
- Implementation SVM-struct: [http://svmlight.joachims.org](http://svmlight.joachims.org)
  - Context-free grammars
  - Sequence alignment
  - Classification with multivariate loss (e.g. F1, ROC Area)
  - General API for other problems

Conditional Random Field (CRF)

- Assume:
  - model class $P(Y|X,a)$ with parameters $\omega \in \Omega$
  - In particular,
    - $r(x) = \log \left( \frac{P(Y=y|x)}{P(Y=\bar{y}|x)} \right)$
- Training: Maximum A Posteriori
  - Objective
    - $\omega = \arg\max \left[ \frac{1}{N} \sum_{i=1}^{N} \log \left( \frac{P(Y=y|x,a)}{P(Y=\bar{y}|x,a)} \right) \right]$
  - Gradient
    - $\frac{\partial \omega}{\partial \omega_{xy}} = \frac{1}{N} \sum_{i=1}^{N} \left[ \frac{P(Y=0|x,a)}{P(Y=1|x,a)} \right]_{xy}$
- Methods: iterative scaling, quasi Newton, conjugate gradient
  - See [Taskar et al. 05]

Applying CRF to New Problem

- Application specific
  - Representation $\Phi(x,y)$
  - Algorithms to compute
    - $\hat{y} = \arg\max_{y} \left[ \log \left( \frac{P(Y=y|x,a)}{P(Y=\bar{y}|x,a)} \right) \right]$

Relationship CRF and Structural SVM

- Objective Functions:
  - CRF:
    - $\omega = \arg\min \left[ \frac{1}{N} \sum_{i=1}^{N} \left( \log \left( \frac{P(Y=y|x,a)}{P(Y=\bar{y}|x,a)} \right) \right) \right]$
  - Structural SVM:
    - $\omega = \arg\max \left( \frac{1}{N} \sum_{i=1}^{N} \left( \log \left( \frac{P(Y=y|x,a)}{P(Y=\bar{y}|x,a)} \right) \right) \right) - \frac{1}{C} \sum_{i=1}^{N} \left( \log \left( \frac{P(Y=y|x,a)}{P(Y=\bar{y}|x,a)} \right) \right)$
- Basic Cases for two classes
  - CRF: Regularized logistic regression
  - Structural SVM: binary classification SVM
Overview

- Task: Learning to predict complex outputs
  - Multi-class classification: Generative vs. Discriminative
  - Compact models for problems with structured outputs
- Training models with structured outputs
  - Generative training
  - SVMs and large-margin training
  - Conditional likelihood training
- Example 1: Learning to parse natural language
  - Learning weighted context-free grammar
- Example 2: Optimizing F1-score in text classification
  - Predict vector of class labels
- Example 3: Learning to cluster
  - Learning a clustering function that produces desired clusterings
- Summary & Reading

Examples of Complex Output Spaces

- Non-Standard Performance Measures (e.g. F1-score, Lift)
  - F1-score: harmonic average of precision and recall
  - New example vector $\mathbf{y}_8$: Predict $y_8 = 1$, if $P(y_8=1 | \mathbf{y}_8)=0.4$?
  - Depends on other examples!

Experiment: Text Classification

- Dataset: Reuters-21578 (ModApte)
  - 9663 training / 3299 test examples, 90 categories
  - TFIDF unit vectors (no stemming, no stopword removal)
- Experiment Setup
  - Classification SVM with optimal C in hindsight
  - Linear Cost SVM [Morik et al., 1999] with C/C via 2-CV
  - $F_1$-loss SVM with C via 2-CV
- Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Test $F_1$</th>
<th>Training Efficiency</th>
<th>[CPU-min] Const</th>
<th>SV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification SVM</td>
<td>32.0</td>
<td>0.1</td>
<td>N/A</td>
<td>20+</td>
</tr>
<tr>
<td>Linear Cost SVM</td>
<td>56.1</td>
<td>0.1</td>
<td>N/A</td>
<td>371</td>
</tr>
<tr>
<td>SVM with (1-$F_1$)-Loss</td>
<td>62.0</td>
<td>0.2</td>
<td>173</td>
<td>95</td>
</tr>
</tbody>
</table>

Struct SVM for Optimizing $F_1$-Score

- Loss Function
  - $\Delta(x,y) = (1 - \frac{\text{Precision}}{\text{Recall}})$
- Representation
  - $x = (x_1, x_2, \ldots, x_n)$
  - $y = (y_1, y_2, \ldots, y_n)$
  - Joint feature map $\psi(x,y) = \sum_{i=1}^{n} x_i y_i$
- Prediction
  - $\text{score} = \text{sign}(\psi(x,y) + \theta)$
- Find most violated constraint
  - Only $n^2$ different contingency tables $\Rightarrow$ search brute force

Struct SVM for Optimizing $F_1$-Score

- Loss if
  - $\Delta(x,y)$
- Representation
  - $x = (x_1, x_2, \ldots, x_n)$
  - $y = (y_1, y_2, \ldots, y_n)$
  - Joint
- Prediction
  - $\text{score} = \text{sign}(\psi(x,y) + \theta)$
- Find $\theta^*$
  - Only $n^2$ different contingency tables $\Rightarrow$ search brute force

Multivariate SVM Generalizes Classification SVM

Theorem: The solutions of the multivariate SVM with number of errors as the loss function and an (unbiased) classification SVM are equal.

Multivariate SVM optimizing Error Rate:

Classification SVM (unbiased):

$$\min_{\theta, \mathbf{w}} \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum_{i=1}^{M} \mathbf{1} \left( y_i (\mathbf{w}^T \mathbf{x}_i) \geq 1 - \epsilon_i \right)$$
Overview

- Task: Learning to predict complex outputs
  - Multi-class classification: Generative vs. Discriminative
  - Compact models for problems with structured outputs
- Training models with structured outputs
  - Generative training
  - SVMs and large-margin training
  - Conditional likelihood training
- Example 1: Learning to parse natural language
  - Learning weighted context free grammar
- Example 2: Optimizing F-score in text classification
  - Predict vector of class labels
- Example 3: Learning to cluster
  - Learning a clustering function that produces desired clusterings
- Summary & Reading

Learning to Cluster

- Noun-Phrase Co-reference
  - Given a set of noun phrases \( x \), predict a clustering \( y \).
  - Structural dependencies, since prediction has to be an equivalence relation.
- Correlation dependencies from interactions.

Struct SVM for Supervised Clustering

- Representation
  - Supervised clustering is a structured prediction problem.
    - \( y \) is reflexive \( \forall i y_{ii} = 1 \), symmetric \( y_{ij} = y_{ji} \), and transitive (if \( y_{ij} = 1 \) and \( y_{jk} = 1 \) then \( y_{ik} = 1 \)).
  - Joint feature map: \( f(x, y) = \sum a_i f_i(x) g(y_i, y) \).
- Loss Function
  - \( L(y, f(x, y)) = \| y - f(x, y) \|_1 \).
- Prediction
  - \( f(x, y) = \arg \max_y L(y, f(x, y)) \).
- Find most violated constraint
  - \( \tilde{y} = \arg \min_{y \neq \hat{y}} L(y, f(x, \hat{y})) \).
  - NP hard, use linear relaxation instead [Demaine & Immorlica, 2003]

Summary

- Learning to predict structured and interdependent output
  - Discriminant function: \( A(y) = \arg \max_y A(x, y) \).
- Training:
  - Generative
  - Structural SVM
  - Conditional Random Field
- Examples
  - Learning to predict trees (natural language parsing)
  - Optimize to non-standard performance measures (imbalanced classes)
  - Learning to cluster (noun-phrase coreference resolution)
- Software:
  - SVMs: http://www.cs.joensuu.fi/svv/soft/svmlearn.html
  - Mallet: http://mallet.cs.umass.edu/

Reading

- Generative training
  - Hidden-Markov models [Manning & Schutze, 1999]
  - Probabilistic context-free grammars [Manning & Schutze, 1999]
  - Markov random fields [Geman & Geman, 1984]
  - Etc.
- Discriminative training
  - Multivariate output regression [Lazebnik, 1997] [Breiman & Friedman, 1997]
  - Kernel Dependency Estimation [Weston et al. 2003]
  - Conditional HMM [Krogh, 1994]
  - Transformer networks [LeCun et al., 1998]
  - Conditional random fields [Lafferty et al., 2001] [Sutton & McCallum, 2005]
  - Perceptron training of HMM [Collins, 2002]
  - Structural SVMs / Maximum-margin Markov networks [Taskar et al., 2003]
  - Maximum-margin Markov networks [Taskar et al., 2003] [Tsochantaridis et al., 2004, 2005] [Taskar 2004]