Support Vector Machine Learning for Interdependent and Structured Output Spaces


And also
I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun

Presented by
Thorsten Joachims
Cornell University
Department of Computer Science

Examples of Complex Output Spaces

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

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.

Examples of Complex Output Spaces

- Protein Sequence Alignment
  - Given two sequences $x=(s,t)$, predict an alignment $y$.
  - Structural dependencies, since prediction has to be a valid global/local alignment.

Learning Task

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

- Goal: Find $f: X \rightarrow Y$ with low expected loss
  - Loss function: $\Delta(y,y')$ (penalty for predicting $y'$ if $y$ correct)
  - Expected loss (i.e. risk, prediction error):
    $$\text{Err}_P(f) = \sum_{x,y} \Delta(y,f(x)) P(X=x,Y=y)$$

Goals of Paper

- Paper proposes Support Vector Machine (SVM) method
  - that does not build generative model, but directly finds rule with low training loss (i.e. ERM).
  - that applies to a large class of structured outputs $Y$
    - sequences (i.e. hidden Markov models)
    - trees (i.e. context-free grammars)
    - hierarchical classification
    - sequence alignment (i.e. string edit distance)
  - allows the use of fairly general loss functions
  - is a generalization of multi-class SVMs
  - has polynomial time training algorithm.
Outline and Approach

- What form does the prediction rule take?
  - Discriminant rule: $f_w(x) = \arg\max_{y \in Y} \psi(x, y) \\ w^T \Psi(x, y)$
  - Challenge: How to compute prediction efficiently?

- What form does the discriminant function take?
  - Linear: $F(x, y; w) = w^T \Psi(x, y)$
  - Challenge: How to represent the model compactly?

- How to train?
  - Discriminative, empirical risk minimization.
  - Minimize upper bound on training loss
  - $w^* = \arg\min \sum \Delta(y, f_w(x))$
  - Challenge: How to efficiently find “best” $w$?

What Form does the Discriminant Function Take?

- Linear Chain Model (HMM)
  - Joint feature map for local dependencies
  - Score for each adjacent label/label and word/label pair
  - Find highest scoring sequence

  $f_w(\vec{x}) = \arg\max_{y \in Y} \psi(x, y) \\ w^T \Psi(x, y)$

  $\Psi(x, y) = \begin{cases} 1 & N \rightarrow V \\ 0 & Det \rightarrow V \\ 2 & Det \rightarrow N \\ 1 & V \rightarrow V \\ \vdots & \vdots \\ 1 & N \rightarrow V \\ 2 & Det \rightarrow \text{bear} \\ 1 & \text{V} \rightarrow \text{Chased} \end{cases}$

Connection to Graphical Models

Hidden Markov Model:

- Assumptions
  - $P(Y = (y_1, \ldots, y_n)) = \prod \{p(y_i | y_{i-1}) | y_1 = y_1^{(0)} \}
  - $P(X = (x_1, \ldots, x_n)) = \prod \{p(x_i | x_{i-1}) | x_1 = x_1^{(0)} \}

- Rule: $f(x) = \arg\max_{y \in Y} \psi(x, y) \\ w^T \Psi(x, y)$

  $= \arg\max_{(y_1, \ldots, y_n) \in Y} \left[ \prod_{i=1}^{n} p(y_{i+1} | y_i) \prod_{i=1}^{n} p(x_{i+1} | x_i) \right]$

  $= \arg\max_{(y_1, \ldots, y_n) \in Y} \left[ w^T \Phi(x, y) \right]$

  with $w_{\alpha, \beta} = \log(p(Y_{\alpha}, \beta))$ and $w_{\alpha, \beta} = \log(p(X_{\alpha}, \beta | Y_{\alpha}, \beta))$

  and $\Phi(x, y)$ histogram

Outline and Approach

- What form does the prediction rule take?
  - Discriminant rule: $f_w(x) = \arg\max_{y \in Y} \psi(x, y) \\ w^T \Psi(x, y)$
  - Challenge: How to compute prediction efficiently?

- What form does the discriminant function take?
  - Linear: $F(x, y; w) = w^T \Psi(x, y)$
  - Challenge: How to represent the model compactly?

- How to train?
  - Discriminative, empirical risk minimization.
  - Minimize upper bound on training loss
  - $w^* = \arg\min \sum \Delta(y, f_w(x))$
  - Challenge: How to efficiently find “best” $w$?

How to Compute Prediction Efficiently?

  $f_w(x) = \arg\max_{y \in Y} \psi(x, y) \\ w^T \Psi(x, y)$

  - Linear Chain (HMM): Viterbi
  - Tree (Weighted Context-Free Grammar): CKY
  - Sequence Alignment: Smith/Waterman algorithm

What Form does the Discriminant Function Take?

- Weighted Context Free Grammar
  - Each rule $r_i$ (e.g. $S \rightarrow NP VP$) has a weight $w_{r_i}$
  - Score of a tree is the sum of its weights
  - Find highest scoring tree $f_w(\vec{x}) = \arg\max_{y \in Y} \psi(x, y) \\ w^T \Psi(x, y)$

  $\Phi(x, y) = \begin{cases} 1 & S \rightarrow NP VP \\ 0 & S \rightarrow NP \\ 2 & NP \rightarrow Det N \\ 1 & NP \rightarrow VP NP \\ \vdots & \vdots \\ 1 & N \rightarrow Det \\ 2 & Det \rightarrow dog \\ 1 & Det \rightarrow the \\ 1 & N \rightarrow dog \\ 1 & V \rightarrow Chased \\ 1 & N \rightarrow cat \end{cases}$

Connection to Graphical Models

Hidden Markov Model:

- Assumptions
  - $P(Y = (y_1, \ldots, y_n)) = \prod \{p(y_i | y_{i-1}) | y_1 = y_1^{(0)} \}
  - $P(X = (x_1, \ldots, x_n)) = \prod \{p(x_i | x_{i-1}) | x_1 = x_1^{(0)} \}

- Rule: $f(x) = \arg\max_{y \in Y} \psi(x, y) \\ w^T \Psi(x, y)$

  $= \arg\max_{(y_1, \ldots, y_n) \in Y} \left[ \prod_{i=1}^{n} p(y_{i+1} | y_i) \prod_{i=1}^{n} p(x_{i+1} | x_i) \right]$

  $= \arg\max_{(y_1, \ldots, y_n) \in Y} \left[ w^T \Phi(x, y) \right]$

  with $w_{\alpha, \beta} = \log(p(Y_{\alpha}, \beta))$ and $w_{\alpha, \beta} = \log(p(X_{\alpha}, \beta | Y_{\alpha}, \beta))$

  and $\Phi(x, y)$ histogram

Outline and Approach

- What form does the prediction rule take?
  - Discriminant rule: $f_w(x) = \arg\max_{y \in Y} \psi(x, y) \\ w^T \Psi(x, y)$
  - Challenge: How to compute prediction efficiently?

- What form does the discriminant function take?
  - Linear: $F(x, y; w) = w^T \Psi(x, y)$
  - Challenge: How to represent the model compactly?

- How to train?
  - Discriminative, empirical risk minimization.
  - Minimize upper bound on training loss
  - $w^* = \arg\min \sum \Delta(y, f_w(x))$
  - Challenge: How to efficiently find “best” $w$?

How to Compute Prediction Efficiently?

  $f_w(x) = \arg\max_{y \in Y} \psi(x, y) \\ w^T \Psi(x, y)$

  - Linear Chain (HMM): Viterbi
  - Tree (Weighted Context-Free Grammar): CKY
  - Sequence Alignment: Smith/Waterman algorithm
Outline and Approach

- What form does the prediction rule take?
  - Discriminant rule: $f_x(x) = \arg\max_{y \in Y} [F(x, y; w)]$
  - Challenge: How to compute prediction efficiently?

- What form does the discriminant function take?
  - Linear: $F(x, y; w) = w^T \psi(x, y)$
  - Challenge: How to represent the model compactly?

- How to train?
  - Discriminative, empirical risk minimization.
  - Minimize upper bound on training loss
    $w^* = \arg\min_w \sum_i \Delta(y_i, f_x(x_i))$
  - Challenge: How to efficiently find “best” $w$?

Hard-Margin SVM

Training Data: $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^N \times \{+1, -1\}$
Classification Rule: $f(x) = sgn[w^T x + b]$

Training: Find hyperplane with the largest distance to the closest training examples.

Support Vectors: Examples with minimal distance (i.e. margin).

Soft-Margin SVM

Idea: Maximize margin and minimize training error.

Hard-Margin OP (Primal): $\min_{w, b} \frac{1}{2} w^T w$ s.t. $y_1(w^T x_1 + b) \geq 1$ and $\ldots$, $y_n(w^T x_n + b) \geq 1$

Soft-Margin OP (Primal): $\min_{w, b} \frac{1}{2} w^T w + C \sum_i \xi_i$ s.t. $y_1(w^T x_1 + b) \geq 1 - \xi_1 \land \xi_1 \geq 0$ and $\ldots$, $y_n(w^T x_n + b) \geq 1 - \xi_n \land \xi_n \geq 0$

- Slack variable $\xi_i$ measures by how much $(x_i, y_i)$ fails to achieve margin $\delta$
- $\sum \xi_i$ is upper bound on number of training errors
- $C$ is a parameter that controls trade-off between margin and training error.

Controlling Soft-Margin Separation

- $\sum \xi_i$ is upper bound on number of training errors
- $C$ is a parameter that controls trade-off between margin and training error.

Structural Support Vector Machine

- Joint features $\psi(x, y)$ describe match between $x$ and $y$
- Learn weights $w$ so that $w^T \psi(x, y)$ is max for correct $y$

Hard-margin optimization problem:

$\min_{w, b} \frac{1}{2} w^T w$

s.t. $\forall y \in Y \setminus \{y_c\}: w^T \psi(x_1, y_1) \geq w^T \psi(x_1, y) + 1$ and $\ldots$, $\forall y \in Y \setminus \{y_c\}: w^T \psi(x_n, y_n) \geq w^T \psi(x_n, y) + 1$

$(x_1, y_1)$, $(x_2, y_2)$, $(x_3, y_3)$, $(x_n, y_n)$
Soft-Margin Struct SVM (Margin Rescaling)

Soft-margin optimization problem:
\[
\min_{\omega, \xi} \frac{1}{2} \omega^T \omega + C \sum_{i=1}^{n} \xi_i \\
\text{s.t.} \quad \forall y \in y_1: \omega^T \Phi(x_1, y) \geq \omega^T \Phi(x_1, y_1) + \Delta(y_1, y) - \xi_i \\
\quad \forall y \in y_0: \omega^T \Phi(x_0, y) \geq \omega^T \Phi(x_0, y_0) + \Delta(y_0, y) - \xi_i 
\]

Lemma: The training loss is upper bounded by
\[
Err_S(f_w) = \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, f_w(x_i)) \leq \frac{1}{n} \sum_{i=1}^{n} \xi_i 
\]

Soft-Margin Struct SVM (Slack Rescaling)

Soft-margin optimization problem:
\[
\min_{\omega, \xi} \frac{1}{2} \omega^T \omega + C \sum_{i=1}^{n} \xi_i \\
\text{s.t.} \quad \forall y \in y_1: \omega^T \Phi(x_1, y) \geq \omega^T \Phi(x_1, y_1) + 1 - \frac{\xi_i}{\Delta(y_1, y)} \\
\quad \forall y \in y_0: \omega^T \Phi(x_0, y) \geq \omega^T \Phi(x_0, y_0) + 1 - \frac{\xi_i}{\Delta(y_0, y_0)} 
\]

Lemma: The training loss is upper bounded by
\[
Err_S(f_w) = \frac{1}{n} \sum_{i=1}^{n} \Delta(y_i, f_w(x_i)) \leq \frac{1}{n} \sum_{i=1}^{n} \xi_i 
\]

Cutting-Plane Algorithm for Structural SVM

- Input: \((x_1, y_1), \ldots, (x_n, y_n), C, \epsilon\)
- \(S = \emptyset, \omega = 0, \xi = 0\)
- REPEAT
  - FOR \(i = 1, \ldots, n\)
    - compute \(\bar{y} = \arg\max_{y \in \{0,1\}} \{\Delta(y_i, y_0) + \omega^T \Phi(x_i, y)\}\)
    - IF \(\Delta(y_i, \bar{y}) - \omega^T \Phi(x_i, \bar{y}) \geq \Delta(y_i, \bar{y}_i) - \xi_i\)
      - \(S = S \cup \{\omega^T \Phi(x_i, \bar{y}) - \omega^T \Phi(x_i, \bar{y}_i) \geq \Delta(y_i, \bar{y}) - \xi_i\}\)
    - \(\omega, \xi\) - optimize StructSVM over \(S\)
    - ENDIF
  - ENDFOR
- UNTIL \(S\) has not changed during iteration

Polynomial Sparsity Bound

- Theorem: The sparse-approximation algorithm finds a solution to the soft-margin optimization problem after adding at most
  \[
  \max \left\{ \frac{2mA}{\epsilon}, \frac{8mCA^2}{\epsilon^2} \right\}
  \]
  constraints to the working set \(S\), so that the Kuhn-Tucker conditions are fulfilled up to a precision \(\epsilon\). The loss has to be bounded \(0 \leq \Delta(y_i, y) \leq A\), and \(||\Phi(x_i, y)|| \leq R\).

Dual QP for Classification SVM

- Primal Optimization Problem
  \[
  \begin{align*}
  \text{minimize:} & \quad P(\omega, b, \xi) = \frac{1}{2} \omega^T \omega + C \sum_{i=1}^{n} \xi_i \\
  \text{subject to:} & \quad \forall y \in Y: \omega^T \Phi(x, y) \geq \omega^T \Phi(x, y_0) + \Delta(y_0, y) - \xi_i 
  \end{align*}
  \]

- Dual Optimization Problem
  \[
  \begin{align*}
  \text{maximize:} & \quad D(\phi) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j \phi(x_i, x_j) \\
  \text{subject to:} & \quad \forall y \in Y: \alpha_y \geq 0, \sum_{y \in Y} \alpha_y = C
  \end{align*}
  \]

- Theorem: If \(\omega^*\) is the solution of the Primal and \(\alpha^*\) is the solution of the Dual, then \(\omega^* = \sum_{y \in Y} \alpha_y^* \phi(x, y)\) and \(P(\omega^*, b^*, \xi^*) = D(\alpha^*)\). For all other feasible \(\omega, b, \xi\) and \(\alpha\), \(P(\omega, b, \xi) \geq D(\alpha)\).

Dual QP for Structural SVM

- Primal Optimization Problem
  \[
  \begin{align*}
  \text{minimize:} & \quad P(\omega, b, \xi) = \frac{1}{2} \omega^T \omega + C \sum_{i=1}^{n} \xi_i \\
  \text{subject to:} & \quad \forall y \in Y: \omega^T \Phi(x, y) \geq \omega^T \Phi(x, y_0) + 1 - \frac{\xi_i}{\Delta(y_0, y)} \\
  \end{align*}
  \]

- Dual Optimization Problem
  \[
  \begin{align*}
  \text{maximize:} & \quad D(\phi) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \sum_{y \in Y} \alpha_y \alpha_{y_j} \phi(x_i, y_j) - \Delta(y_j, y) - \xi_i \\
  \text{subject to:} & \quad \forall y \in Y: \alpha_y \geq 0, \sum_{y \in Y} \alpha_y = C
  \end{align*}
  \]

- Theorem: ditto…
Lemma

**Lemma 1.** Let $J$ be a positive definite matrix and let us define a concave quadratic program

$$W(\alpha) = \frac{1}{2} \alpha' J \alpha + (h, \alpha) \quad \text{s.t. } \alpha \geq 0$$

and assume $\alpha > 0$ is given with $\alpha_r = 0$. Then maximizing $W$ with respect to $\alpha_r$ while keeping all other components fixed will increase the objective by

$$(h_r - \sum_c \alpha_c J_{rc})^2 / 2J_{rr}$$

provided that $h_r \geq \sum_c \alpha_c J_{rc}$.

Improved Training Algorithm and Bound

**Theorem:** The cutting-plane algorithm finds a solution to the Structural SVM soft-margin optimization problem in the 1-slack formulation after adding at most

$$\log \left( \frac{\Delta}{4R^2C} \right) + \frac{16RC^2}{\epsilon}$$

constraints to the working set $S$, so that the primal constraints are feasible up to a precision $\epsilon$ and the objective on $S$ is optimal. The loss has to be bounded by

$$|\Phi(x, y)| \leq \Delta$$

and

$$2|\Phi(x, y)| \leq R.$$

For non-kernelized models, training time scales linearly with number of training examples.

Experiment: Natural Language Parsing

**Implementation**

- Implemented Sparse-Approximation Algorithm in SVM^light
- Incorporated modified version of Mark Johnson’s CKY parser
- Learned weighted CFG with $\epsilon = 0.01$, $C = 1$

**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>$F_1$</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-$F_1$)-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:**

$$\Phi(x, y) = \sum_{i=1}^{m} \phi(x, y_i)$$

**General form:**

$$\phi(x, y_i) = \theta_{\text{kernel}}(\phi(x, [\text{rule}, \text{start}, \text{end}]))$$

$$K(a, b) = \theta_{\text{kernel}}(a)^T \theta_{\text{kernel}}(b)$$

**So far:**

$$\phi(x, y_i) = \begin{cases} 1 & \text{if } y_i = \text{S} \rightarrow \text{NP VP} \\ 0 & \text{otherwise} \end{cases}$$

**Example:**

$$\phi(x, y_i) = \begin{cases} (\text{start} - \text{end})^2 & \text{if } y_i = \text{NP} \land x_{\text{start}} = \text{NP} \\ 0 & \text{otherwise} \end{cases}$$

see [Taskar et al. 05]

**Application specific**

- Loss function $\Delta(y_i, y)$
- Representation $\Psi(x, y)$
- Algorithms to compute $\hat{y} = \arg\max_{y \in Y} \{ w^T \Psi(x, y) \}$

Applying Structural SVM to New Problem

**Implementation SVM-struct:** [http://svmlight.joachims.org](http://svmlight.joachims.org)

- Context-free grammars, sequence alignment, linear chain HMM, diverse rankings, classification with multivariate loss (e.g. F1, ROC Area), etc.
- General API for other problems

Experiment: Part-of-Speech Tagging

**Task**

- Given a sequence of words $x$, predict sequence of tags $y$.

**Model**

- Markov model with one state per tag and words as emissions
- Each word described by ~250,000 dimensional feature vector (all word suffixes/prefixes, word length, capitalization…)

**Experiment (by Dan Fleisher)**

- Train/test on 7966/1700 sentences from Penn Treebank

Experiment: Natural Language Parsing

**Implementation**

- Implemented Sparse-Approximation Algorithm in SVM^light
- Incorporated modified version of Mark Johnson’s CKY parser
- Learned weighted CFG with $\epsilon = 0.01$, $C = 1$

**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>$F_1$</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-$F_1$)-Loss</td>
<td>58.9</td>
<td>88.5</td>
<td>3.4</td>
<td>12</td>
<td>8043</td>
</tr>
</tbody>
</table>
Summary

• Support Vector Machine approach to training
  – Hidden Markov Models
  – Weighted Context-Free Grammars
  – Sequence Alignment cost functions
  – Etc.

• Incorporate loss functions via
  – Margin rescaling
  – Slack rescaling

• General training algorithm based on cutting-plane method
  – Efficient for all linear discriminant models where argmax efficient