Support Vector Machine Learning for Interdependent and Structured Output Spaces

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, ICML, 2004.

And also

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

Journal of Machine Learning Research (JMLR), 6(Sep):1453-1484, 2005.

Presented by

Thorsten Joachims

Cornell University

Department of Computer Science

Examples of Complex Output Spaces

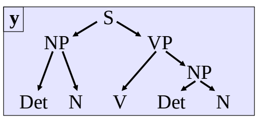
- **Part-of-Speech Tagging**
  - Given a sequence of words *x*, predict sequence of tags *y*.
  - Dependencies from tag-tag transitions in Markov model.

x The bear chased the cat → **y** Det N V Det N

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.

x The dog chased the cat →



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.

x

```
s=(ABJLHBNJYAUGAI)

t=(BHJKBNYGU)
```

→ **y**

```
AB-JLHBNJYAUGAI
 | || || |||
BHJK-BN-YGU
```

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: $S=((x_1,y_1), \ldots, (x_n,y_n)) \sim_{iid} P(X,Y)$
- **Goal: Find *f: X → Y* with low expected loss**
  - Loss function: *Δ(y,y')* (penalty for predicting *y'* if *y* correct)
  - Expected loss (i.e. risk, prediction error):

$$Err_P(f) = \sum_{x,y} \Delta(y, f(x))\ P(X{=}x, Y{=}y)$$

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: $f_w(x) = \text{argmax}_{y \in Y} [F(x, y; w)]$
  - 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

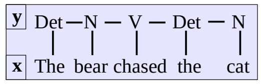
    $$w^* = \underset{w}{\text{argmin}} \sum_{i=1}^{n} \Delta(y_i, f_w(x_i))$$
  - Challenge: How to efficiently find "best" w?

What Form does the Discr. 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

$$f_w(\vec{x}) = argmax_{y \in Y} \left[\vec{w}^T \Psi(x,y)\right]$$

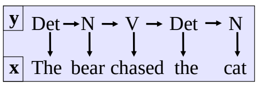
Viterbi



$$\Psi(\mathbf{x}, \mathbf{y}) = \begin{array}{cc} \begin{pmatrix} 1 \\ 0 \\ 2 \\ 1 \\ \vdots \\ 0 \\ 2 \\ 1 \\ 1 \\ 1 \end{pmatrix} & \begin{array}{c} N \to V \\ Det \to V \\ Det \to N \\ V \to Det \\ \\ Det \to bear \\ Det \to the \\ N \to bear \\ V \to chased \\ N \to cat \end{array} \end{array}$$

Connection to Graphical Models

Hidden Markov Model:



- Assumptions

$$P(Y = (y^{(1)}, \ldots, y^{(l)})) = \prod_{i=1}^{l} P(Y_c = y^{(i)} | Y_p = y^{(i-1)})$$

$$P(X = (x^{(1)}, \ldots, x^{(l)}) | Y = (y^{(1)}, \ldots, y^{(l)})) = \prod_{i=1}^{l} P(X_c = x^{(i)} | Y_c = y^{(i)})$$

→ Rule: $f(x) = \underset{y \in Y}{\text{argmax}} [P(X = x | Y = y) P(Y = y)]$

$$= \underset{(y^{(1)}, \ldots, y^{(l)}) \in Y}{\text{argmax}} \left[\prod_{i=1}^{l} P(Y_c = y^{(i)} | Y_p = y^{(i-1)}) P(X_c = x^{(i)} | Y_c = y^{(i)})\right]$$

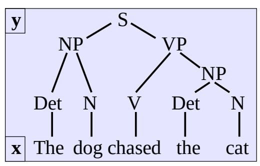
$$= \underset{(y^{(1)}, \ldots, y^{(l)}) \in Y}{\text{argmax}} \left[\vec{w}^T \Phi(x, y)\right]$$

with w_{ab}= *-log[P(Y_c=a|Y_p=b)]* and w_{cd}= *-log[P(X_c=c|Y_c=d)]* and Φ(x,y) histogram

What Form does the Discr. Function Take?

CKY Parser

- **Weighted Context Free Grammar**
  - Each rule r_i (e.g. $S \to NP\ VP$) has a weight w_i
  - Score of a tree is the sum of its weights
  - Find highest scoring tree $f_w(\vec{x}) = argmax_{y \in Y} \left[\vec{w}^T \Psi(x, y)\right]$



$$\Phi(\mathbf{x}, \mathbf{y}) = \begin{array}{cc} \begin{pmatrix} 1 \\ 0 \\ 2 \\ 1 \\ \vdots \\ 0 \\ 2 \\ 1 \\ 1 \\ 1 \end{pmatrix} & \begin{array}{c} S \to NP\ VP \\ S \to NP \\ NP \to Det\ N \\ VP \to V\ NP \\ \\ Det \to dog \\ Det \to the \\ N \to dog \\ V \to chased \\ N \to cat \end{array} \end{array}$$

Outline and Approach

- **What form does the prediction rule take?**
  - Discriminant rule: $f_w(x) = \text{argmax}_{y \in Y} [F(x, y; w)]$
  - 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

    $$w^* = \underset{w}{\text{argmin}} \sum_{i=1}^{n} \Delta(y_i, f_w(x_i))$$
  - Challenge: How to efficiently find "best" w?

How to Compute Prediction Efficiently?

$$f_w(x) = \text{argmax}_{y \in Y} \left[w^T \Psi(x, y)\right]$$

- **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: $f_w(x) = \text{argmax}_{y \in Y} [F(x, y; w)]$
  - 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
    $$w^* = \underset{w}{\text{argmin}} \sum_{i=1}^{n} \Delta(y_i, f_w(x_i))$$
  - Challenge: How to efficiently find "best" w?

Hard-Margin SVM

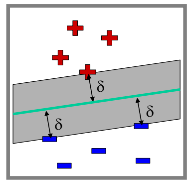
Training Data: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ $\mathbf{x} \in \Re^N$ $y \in \{+1, -1\}$

Classification Rule: $f(\mathbf{x}) = sgn\left[\mathbf{w}^T \mathbf{x} + b\right]$

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

Optimization Problem (Primal):

$$\min_{\vec{w}, b} \quad \frac{1}{2} \vec{w} \cdot \vec{w}$$
$$s.t. \quad y_1(\vec{w} \cdot \vec{x}_1 + b) \geq 1$$
$$\ldots$$
$$y_n(\vec{w} \cdot \vec{x}_n + b) \geq 1$$



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

[Vapnik et al.]

Soft-Margin SVM

Idea: Maximize margin and minimize training error.

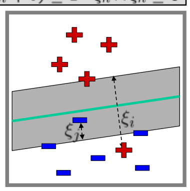
Hard-Margin OP (Primal):

$$\min_{\vec{w}, b} \quad \frac{1}{2} \vec{w} \cdot \vec{w}$$
$$s.t. \quad y_1(\vec{w} \cdot \vec{x}_1 + b) \geq 1$$
$$\ldots$$
$$y_n(\vec{w} \cdot \vec{x}_n + b) \geq 1$$

Soft-Margin OP (Primal):

$$\min_{\vec{w}, \vec{\xi}, b} \quad \frac{1}{2} \vec{w} \cdot \vec{w} + C \sum_{i=1}^{n} \xi_i$$
$$s.t. \quad y_1(\vec{w} \cdot \vec{x}_1 + b) \geq 1 - \xi_1 \wedge \xi_1 \geq 0$$
$$\ldots$$
$$y_n(\vec{w} \cdot \vec{x}_n + b) \geq 1 - \xi_n \wedge \xi_n \geq 0$$

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



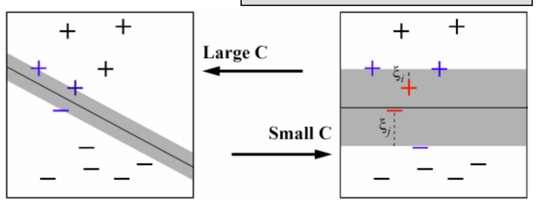
[Cortes et al.]

Controlling Soft-Margin Separation

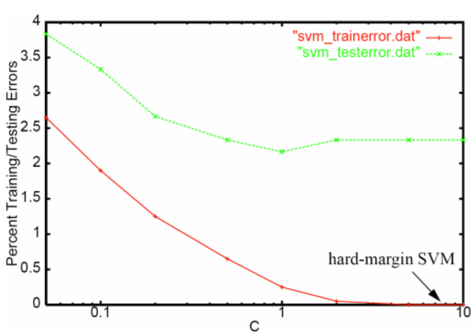
- $\Sigma \xi_i$ is upper bound on number of training errors
- C is a parameter that controls trade-off between margin and training error.

Soft-Margin OP (Primal):

$$\min_{\vec{w}, \vec{\xi}, b} \quad \frac{1}{2} \vec{w} \cdot \vec{w} + C \sum_{i=1}^{n} \xi_i$$
$$s.t. \quad y_1(\vec{w} \cdot \vec{x}_1 + b) \geq 1 - \xi_1 \wedge \xi_1 \geq 0$$
$$\ldots$$
$$y_n(\vec{w} \cdot \vec{x}_n + b) \geq 1 - \xi_n \wedge \xi_n \geq 0$$

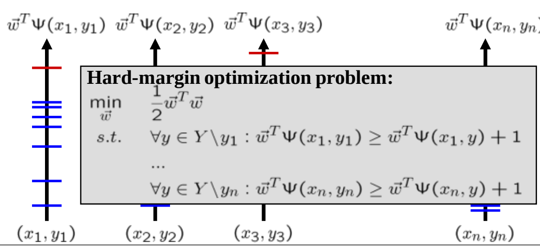


Example Reuters "acq": Varying C



Structural Support Vector Machine

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



Hard-margin optimization problem:

$$\min_{\vec{w}} \quad \frac{1}{2} \vec{w}^T \vec{w}$$
$$s.t. \quad \forall y \in Y \setminus y_1 : \vec{w}^T \Psi(x_1, y_1) \geq \vec{w}^T \Psi(x_1, y) + 1$$
$$\ldots$$
$$\forall y \in Y \setminus y_n : \vec{w}^T \Psi(x_n, y_n) \geq \vec{w}^T \Psi(x_n, y) + 1$$

Soft-Margin Struct SVM (Margin Rescaling)

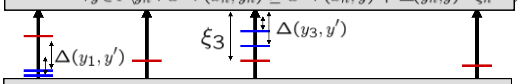
Soft-margin optimization problem:

$$\min_{\vec{w},\vec{\xi}} \quad \frac{1}{2}\vec{w}^T\vec{w} + C\sum_{i=1}^{n}\xi_i$$

$$s.t. \quad \forall y \in Y \backslash y_1 : \vec{w}^T\Psi(x_1,y_1) \geq \vec{w}^T\Psi(x_1,y) + \Delta(y_1,y) - \xi_1$$

$$\ldots$$

$$\forall y \in Y \backslash y_n : \vec{w}^T\Psi(x_n,y_n) \geq \vec{w}^T\Psi(x_n,y) + \Delta(y_n,y) - \xi_n$$



Lemma: The training loss is upper bounded by

$$Err_S(f_w) = \frac{1}{n}\sum_{i=1}^{n}\Delta(y_i, f_w(\vec{x}_i)) \leq \frac{1}{n}\sum_{i=1}^{n}\xi_i$$

(x_1, y_1) (x_2, y_2) (x_3, y_3) (x_n, y_n)

Soft-Margin Struct SVM (Slack Rescaling)

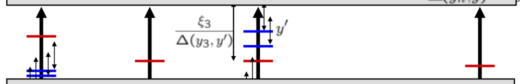
Soft-margin optimization problem:

$$\min_{\vec{w},\vec{\xi}} \quad \frac{1}{2}\vec{w}^T\vec{w} + C\sum_{i=1}^{n}\xi_i$$

$$s.t. \quad \forall y \in Y \backslash y_1 : \vec{w}^T\Psi(x_1,y_1) \geq \vec{w}^T\Psi(x_1,y) + 1 - \frac{\xi_1}{\Delta(y_1,y)}$$

$$\ldots$$

$$\forall y \in Y \backslash y_n : \vec{w}^T\Psi(x_n,y_n) \geq \vec{w}^T\Psi(x_n,y) + 1 - \frac{\xi_n}{\Delta(y_n,y)}$$



Lemma: The training loss is upper bounded by

$$Err_S(f_w) = \frac{1}{n}\sum_{i=1}^{n}\Delta(y_i, f_w(\vec{x}_i)) \leq \frac{1}{n}\sum_{i=1}^{n}\xi_i$$

(x_1, y_1) (x_2, y_2) (x_3, y_3) (x_n, y_n)

Cutting-Plane Algorithm for Structural SVM

- **Input:** $(x_1, y_1), \ldots, (x_n, y_n), C, \epsilon$
- $S \leftarrow \emptyset, \vec{w} \leftarrow 0, \vec{\xi} \leftarrow 0$
- **REPEAT**
  - FOR $i = 1, \ldots, n$
    - compute $\hat{y} = argmax_{y \in Y}\{\Delta(y_i,y) + \vec{w}^T\Psi(x_i,y)\}$
    - IF $(\Delta(y_i,\hat{y}) - \vec{w}^T[\Psi(x_i,y_i) - \Psi(x_i,\hat{y})]) > \xi_i + \epsilon$
      - $S \leftarrow S \cup \{\vec{w}^T[\Psi(x_i,y_i) - \Psi(x_i,\hat{y})] \geq \Delta(y_i,\hat{y}) - \xi_i\}$
      - $[\vec{w}, \vec{\xi}] \leftarrow$ optimize StructSVM over S
    - ENDIF
  - ENDFOR
- **UNTIL** S **has not changed during iteration**

Find most violated constraint

Violated by more than ε ?

Add constraint to working set

[AltHo03] [Jo03] [TsoJoHoAl05]

Polynomial Sparsity Bound

- **Theorem:** The sparse-approximation algorithm finds a solution to the soft-margin optimization problem after adding at most

$$\max\left\{\frac{2nA}{\epsilon}, \frac{8nCAR^2}{\epsilon^2}\right\}$$

constraints to the working set S, so that the Kuhn-Tucker conditions are fulfilled up to a precision ϵ. The loss has to be bounded $0 \leq \Delta(y_i,y) \leq A$, and $||\Phi(x,y)|| \leq R$.

[Jo 03] [Tsochantaridis et al. 04] [Tsochantaridis et al. 05]

Dual QP for Classification SVM

- **Primal Optimization Problem**

$$\text{minimize:} \quad P(\vec{w}, b, \vec{\xi}) = \frac{1}{2}\vec{w}\cdot\vec{w} + C\sum_{i=1}^{n}\xi_i$$

$$\text{subject to:} \quad \forall_{i=1}^{n} : y_i[\vec{w}\cdot\vec{x}_i \cancel{+b}] \geq 1 - \xi_i$$

$$\forall_{i=1}^{n} : \xi_i > 0$$

- **Dual Optimization Problem**

$$\text{maximize:} \quad D(\vec{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\vec{x}_i \cdot \vec{x}_j)$$

$$\text{subject to:} \quad \cancel{\sum_{i=1}^{n} y_i\alpha_i = 0}$$

$$\forall_{i=1}^{n} : 0 \leq \alpha_i \leq C$$

- **Theorem:** If *w** is the solution of the Primal and *α** is the solution of the Dual, then $\vec{w}^* = \sum_{i=1}^{n}\alpha_i^* y_i \vec{x}_i$ and *P(w*,b*,ξ*)=D(α*)*. For all other feasible *w,b,ξ* and *α*, *P(w,b,ξ)>D(α)*.

Dual QP for Structural SVM

- **Primal Optimization Problem**

$$\text{min:} \quad P(\vec{w}, \vec{\xi}) = \frac{1}{2}\vec{w}\cdot\vec{w} + C\sum_{i=1}^{n}\xi_i$$

$$\text{s.t.:} \quad \forall\hat{y} \in Y : \vec{w}\cdot\delta\Psi_1(\hat{y}) \geq \Delta(y_1,\hat{y}) - \xi_1$$

$$\vdots$$

$$\forall\hat{y} \in Y : \vec{w}\cdot\delta\Psi_n(\hat{y}) \geq \Delta(y_n,\hat{y}) - \xi_n$$

- **Dual Optimization Problem**

$$\text{max:} \quad D(\vec{\alpha}) = \sum_{(i,\hat{y})}\Delta(y_i,\hat{y})\alpha_{i\hat{y}} - \frac{1}{2}\sum_{(i,\hat{y}_i)}\sum_{(j,\hat{y}_j)}\alpha_{i\hat{y}_i}\alpha_{j\hat{y}_j}(\delta\Psi_i(\hat{y}_i)\cdot\delta\Psi_j(\hat{y}_j))$$

$$\text{s.t.:} \quad \forall_{i=1}^{n} : \sum_{\hat{y}}\alpha_{i\hat{y}} \leq C$$

$$\forall(i,\hat{y}) : 0 \leq \alpha_{i\hat{y}}$$

- **Theorem:** ditto…

Lemma

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

$$W(\boldsymbol{\alpha}) = -\frac{1}{2}\boldsymbol{\alpha}'\boldsymbol{J}\boldsymbol{\alpha} + \langle \mathbf{h}, \boldsymbol{\alpha}\rangle \quad s.t.\ \boldsymbol{\alpha} \geq 0$$

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

$$\frac{(h_r - \sum_s \alpha_s J_{rs})^2}{2J_{rr}}$$

provided that $h_r \geq \sum_s \alpha_s J_{rs}$.

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

$$\left\lceil \log_2\left(\frac{\Delta}{4R^2C}\right)\right\rceil + \left\lceil \frac{16R^2C}{\varepsilon}\right\rceil$$

constraints to the working set S, so that the primal constraints are feasible up to a precision ϵ and the objective on S is optimal. The loss has to be bounded $0 \leq \Delta(y_i, y) \leq \Delta$, and $2\|\Phi(x,y)\| \leq R$.

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

[Jo06] [TeoLeSmVi07] [JoFinYu08]

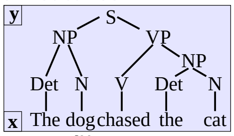
Experiment: Natural Language Parsing

- **Implemention**
  - 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

	Test Accuracy		Training Efficiency		
Method	Acc	F_1	CPU-h	Iter	Const
PCFG with MLE	55.2	86.0	0	N/A	N/A
SVM with (1-F_1)-Loss	**58.9**	**88.5**	3.4	12	8043

[Tsochantaridis et al. 05]

More Expressive Features



- **Linear composition:**

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

- **General form:**

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

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

- **So far:**

$$\phi(x, y_i) = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \ldots \\ 0 \end{pmatrix} \quad \text{if } rule(y_i) = \text{'S} \leftarrow \text{NP VP'}$$

- **Example:**

$$\phi(x, y_i) = \begin{pmatrix} 1 & \text{if } y_i = \text{NP} \wedge x_{end} = \text{'.'} \\ (start - end)^2 & \text{if } y_i = NP \\ 1 & \text{if } y_i = S \wedge \text{span contains } x_i = \text{'and'} \\ \ldots & \end{pmatrix}$$

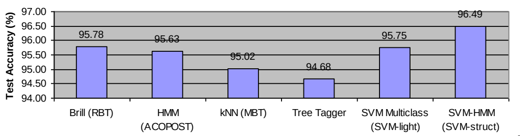
see [Taskar et al. 05]

Experiment: Part-of-Speech Tagging

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

    x The dog chased the cat ➞ y Det➞N➞V➞Det➞N

  - Dependencies from tag-tag transitions in Markov model.
- **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



Applying Structural SVM to New Problem

- **Application specific**
  - Loss function $\Delta(y_i, y)$
  - Representation $\Psi(x, y)$
  - Algorithms to compute

$$\hat{y} = argmax_{y \in Y}\{\vec{w}^T \Psi(x_i, y)\}$$

$$\hat{y} = argmax_{y \in Y}\{\Delta(y_i, y) + \vec{w}^T \Psi(x_i, y)\}$$

- **Implementation SVM-struct: 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

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