Learning to Predict Trees, Sequences, and other Structured Outputs

Tutorial at NESCAI 2006

Thorsten Joachims

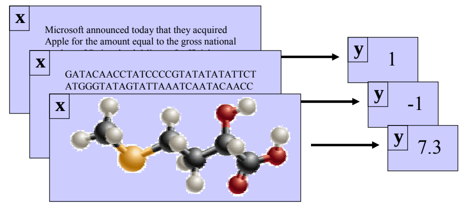
Cornell University
Department of Computer Science

Supervised Learning

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

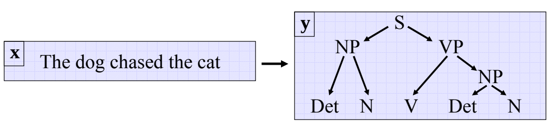
$$h : X \longrightarrow Y$$

such that the prediction error is low.



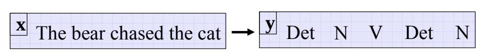
Examples of Complex Output Spaces

- **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.



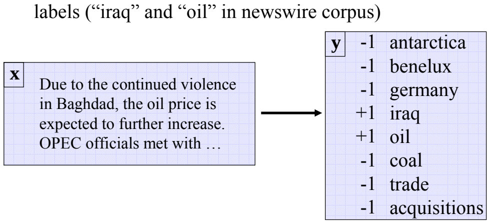
Examples of Complex Output Spaces

- **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

- **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)

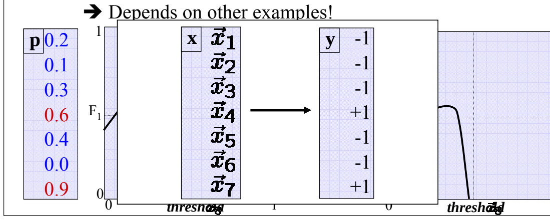


Examples of Complex Output Spaces

- **Non-Standard Performance Measures (e.g. F_1-score, Lift)**
  – F_1-score: harmonic average of precision and recall

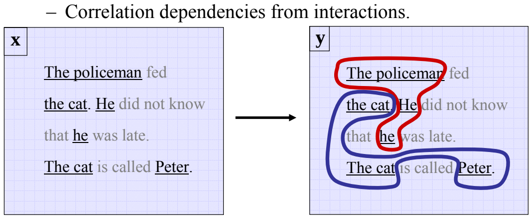
$$F_1 = \frac{2\,Prec\,Rec}{Prec+Rec}$$

  – New example vector $\vec{x}_8$. Predict $y_8=1$, if $P(y_8=1|\vec{x}_8)=0.4$?
  ➔ 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.



Overview

- **Task: Learning to predict complex outputs**
- **Formalizing 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 F_1-score in text classification**
  - Predict vector of class labels
- **Example 3: Learning to cluster**
  - Learning a clustering function that produces desired clusterings
- **Summary & Reading**

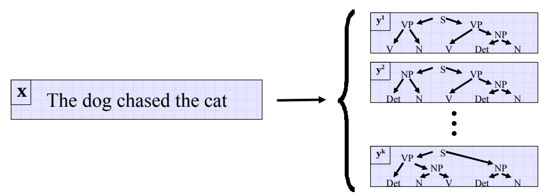
Why do we Need Research on Complex Outputs?

- **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 & Cardie, 2002]
  - Directly optimize complex loss functions (e.g. F1, AvgPrec)
- **Improve upon existing methods!**
  - Natural language parsing: generative models like probabilistic context-free grammars
  - SVM outperforms naïve Bayes for text classification [Joachims, 1998] [Dumais et al., 1998]
- **Mor**
  - –
  - –
- **Tra**
  - –
  - –
  - Support Vector Machines

Precision/Recall Break-Even Point	Naïve Bayes	Linear SVM
Reuters	72.1	87.5
WebKB	82.0	90.3
Ohsumed	62.4	71.6

Natural Language Parsing as Multi-Class Classifications

- **Input Space *X*:** Sequences of words
- **Output Space *Y*:** Each tree is a class



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: $S=((x_1,y_1), \ldots, (x_n,y_n)) \sim_{iid} P(X,Y)$
- **Approach: view as multi-class classification task**
  - Every complex output $y \in Y$ is one class
- **Goal: Find h: X → Y with low expected loss**
  - Loss function: $\Delta(y,y')$ (penalty for predicting y' if y correct)
  - Expected loss (i.e. Risk):

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

Generative Model: Model P(X,Y)

- **Bayes' Decision Rule:** Optimal Decision is

$$h(x) = \operatorname*{argmin}_{y \in Y} \left[\sum_{y' \in Y} \Delta(y',y)\, P(Y=y'|X=x) \right]$$

- **Equivalent Reformulations:** For 0/1-Loss $\Delta(y,y') = 1$, if $y \neq y'$, 0 else

$$\begin{aligned} h(x) &= \operatorname*{argmax}_{y \in Y} [P(Y=y|X=x)] \\ &= \operatorname*{argmax}_{y \in Y} \left[\frac{P(X=x|Y=y)P(Y=y)}{P(X=x)} \right] \\ &= \operatorname*{argmax}_{y \in Y} [P(X=x|Y=y)P(Y=y)] \\ &= \operatorname*{argmax}_{y \in Y} [P(X=x, Y=y)] \\ &= \operatorname*{argmax}_{y \in Y} [P(Y=y|X=x)P(X=x)] \end{aligned}$$

- **Learning: maximum likelihood (or MAP, or Bayesian)**
  - Assume model class $P(X,Y|\omega)$ with parameters $\omega \in \Omega$
  - Find $\omega' = \operatorname*{argmax}_{\omega \in \Omega} \prod_{i=1}^{n} [P(Y=y_i, X=x_i|\omega)]$

Naïve Bayes' Classifier (Multivariate)

fever (3)	cough (2)	pukes (2)	flu?
high	yes	no	1
high	no	yes	1
low	yes	no	-1
low	yes	yes	1
high	no	yes	???

- **Input Space X: Feature Vector**
- **Output Space Y: {1,-1}**
- **Model:**
  - Prior class probabilities
    $P(Y=+1) \quad P(Y=-1)$
  - Class conditional model (one for each class)
    $$P(X=\vec{x}|Y=+1)=\prod_{j=1}^{N} P(X^{(j)}=x^{(j)}|Y=+1)$$
    $$P(X=\vec{x}|Y=-1)=\prod_{j=1}^{N} P(X^{(j)}=x^{(j)}|Y=-1)$$
- **Classification rule:**
  $$h_{naive}(\vec{x}) = argmax_{y\in\{+1,-1\}} \left\{ P(Y=y) \prod_{j=1}^{N} P(X^{(j)}=x^{(j)}|Y=y) \right\}$$

Estimating the Parameters of Naïve Bayes

fever (3)	cough (2)	pukes (2)	flu?
high	yes	no	1
high	no	yes	1
low	yes	no	-1
low	yes	yes	1
high	no	yes	???

- **Count frequencies in training data**
  - n: number of training examples
  - n_+ / n_- : number of pos/neg examples
  - $\#(X^{(j)}=x^{(j)}, y)$: number of times feature $X^{(j)}$ takes value $x^{(j)}$ for examples in class y
  - $|X^{(j)}|$: number of values attribute of $X^{(j)}$
- **Estimating:** $\omega' = \operatorname{argmax}_{\omega\in\Omega} \prod_{i=1}^{n} \left[P(Y=y_i) \prod_{j=1}^{N} \left[P(X_i^{(j)}=x_i^{(j)}|Y=y_i) \right] \right]$
  - P(Y): Maximum Likelihood Estimate
    $$\hat{P}(Y=1)=\frac{n_+}{n} \quad \hat{P}(Y=-1)=\frac{n_-}{n}$$
  - P(X|Y): Maximum Likelihood Estimate
    $$\hat{P}(X^{(j)}=x^{(j)}|Y=y)=\frac{\#(X^{(j)}=x^{(j)},y)}{n_y}$$
  - P(X|Y): Smoothing with Laplace estimate
    $$\hat{P}(X^{(j)}=x^{(j)}|Y=y)=\frac{\#(X^{(j)}=x^{(j)},y)+1}{n_y+|X^{(j)}|}$$

Discriminative Model: Model P(Y|X)

- **Bayes' Decision Rule:**
  - Assume 0/1 Loss $\Delta(y,y') = 1$, if $y \neq y'$, 0 else
  - Optimal Decision: $h(x) = \operatorname{argmax}_{y\in Y} [P(Y=y|X=x)]$
- **Learning: maximum likelihood (or MAP, or Bayesian)**
  - Assume model class $P(Y|X,\omega)$ with parameters $\omega \in \Omega$
  - Find
    $$\omega' = \operatorname{argmax}_{\omega\in\Omega} \prod_{i=1}^{n} [P(Y=y_i|X=x_i,\omega)]$$
- **Example: Logistic regression classifier**
  - Assume
    $$P(Y=y|X=x,w)=\frac{e^{w_y^T x}}{\sum_{y\in Y} e^{w_y^T x}}$$

Discriminative Model: Model Discriminant Function *h* Directly

- **Discriminant Function: h_ω: $X \times Y \rightarrow \Re$**
  $$h(x) = \operatorname{argmax}_{y\in Y} \left[\sum_{y'\in Y} \Delta(y',y) P(Y=y'|X=x) \right] = \operatorname{argmax}_{y\in Y} [h(x,y)]$$
- **Consistency of Empirical Risk:**
  - Empirical Risk (i.e. training error): $Err_S(h)=\sum_{i=1}^{n}\Delta(y_i,h(x_i))$
  - For sufficiently "small" H_Ω and "large" S: Rule $h' \in H$ with best $Err_S(h')$ has $Err_P(h')$ close to $min_{h\in H}\{Err_P(h)\}$
- **Learning: Empirical Risk Minimization (ERM)**
  - Assume class H_Ω of discriminant functions h_ω: $X \rightarrow Y$
  - Find
    $$h'_\omega = \operatorname{argmin}_{h_\omega\in H_\Omega} \sum_{i=1}^{n} \Delta(y_i, h_\omega(x_i))$$

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 $(x_1,y_1),...,(x_n,y_n) \sim P(X,Y)$

Discriminative Model
- Model *P(Y|X)* with *P(Y|X,ω)*
  - Find ω e.g. via MLE
  - Examples: Log. Reg., CRF
- Model discriminant functions
  - Find $h_\omega \in H$ with low train loss (e.g. Emp. Risk Min.)
  - Examples: SVM, Dec. Tree

Generative Model
- Model *P(X,Y)* with distributions *P(Y,X|ω)*
- Find ω that best matches *P(X,Y)* on training data (e.g. MLE)
- Examples: naive Bayes, HMM

Prediction: $h(x) = \operatorname{argmax}_{y\in Y} [h(x,y)]$

Challenges in Learning with Complex Outputs

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

    x The bear chased the cat → **y** Det→N→V→Det→N
- **Problem: Exponentially many classes!**
  - Generative Model: $P(X,Y|\omega)$
  - Discriminative Model: $P(Y|X,\omega)$
  - Discriminant Functions: h_ω: $X \times Y \rightarrow \Re$
- **Challenges**
  - How to compactly represent model?
  - How to do efficient inference with model (i.e. $\operatorname{argmax}_{y\in Y} [h(x,y)]$)?
  - How to effectively estimate model from data? (e.g. compute $h'_\omega = \operatorname{argmin}_{h_\omega\in H_\Omega} \sum_{i=1}^{n} \Delta(y_i, h_\omega(x_i))$)

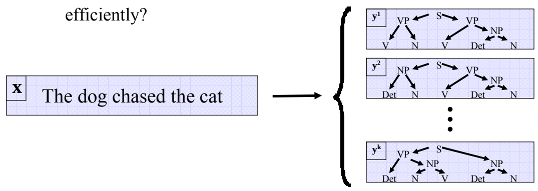
Natural Language Parsing

- **Input Space X:** Sequences of words
- **Output Space Y:** Trees over sequences
- **Problem:**

  How to compute predictions

  $$h(x) = \text{argmax}_{y \in Y}\,[h(x,y)]$$

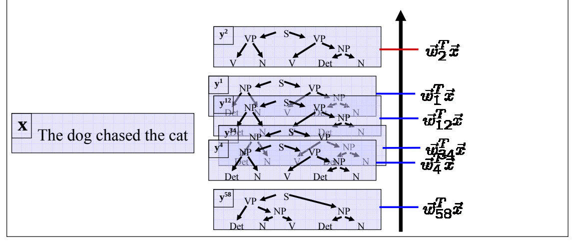
  efficiently?



Multi-Class Linear Discriminant

- **Linear discriminant function of the form:** $h_w(x,y) = w_y^T x$
- **Learn one weight vector w_y for each class $y \in Y$**

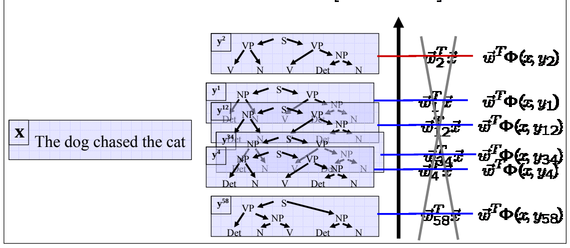
$$h(\vec{x}) = argmax_{y \in Y}\left[w_y^T x\right]$$



Joint Feature Map

- **Feature vector $\Phi(x,y)$ that describes match between x and y**
- **Linear discriminant function of the form:** $h_w(x,y) = \vec{w}^T\Phi(x,y)$

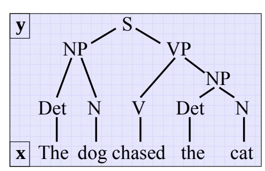
$$h(\vec{x}) = argmax_{y \in Y}\left[\vec{w}^T\Phi(x,y)\right]$$



Joint Feature Map for Trees

- **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 $h(\vec{x}) = argmax_{y \in Y}\left[\vec{w}^T\Phi(x,y)\right]$

CKY Parser



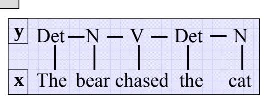
$$\Phi(\mathbf{x},\mathbf{y}) = \begin{pmatrix} 1 \\ 0 \\ 2 \\ 1 \\ \vdots \\ 0 \\ 2 \\ 1 \\ 1 \\ 1 \end{pmatrix} \begin{array}{c} S \to NP\ VP \\ S \to NP \\ NP \to Det\ N \\ VP \to V\ NP \\ \\ Det \to dog \\ Det \to the \\ N \to dog \\ V \to chased \\ N \to cat \end{array}$$

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

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

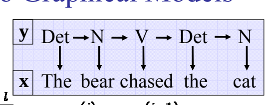
Viterbi



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

Connection to Graphical Models

Hidden Markov Model:



- Assumptions

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

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

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

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

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

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

Overview

- **Task: Learning to predict complex outputs**
- **Formalizing 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 F_1-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 $P(X,Y|\omega)$ with parameters $\omega \in \Omega$
- **Maximum Likelihood (or alternative estimator)**
  - Find $\omega' = \operatorname{argmax}_{\omega\in\Omega} \prod_{i=1}^{n} [P(Y=y_i, X=x_i|\omega)]$

$$= \operatorname{argmax}_{\omega\in\Omega} \left[\prod_{i=1}^{n}\prod_{j=1}^{l} P(Y_c=y_i^{(j)}|Y_p=y_i^{(j-1)})\,P(X_c=x_i^{(j)}|Y_c=y_i^{(j)})\right]$$

- **Example: Hidden Markov Model**
  - Closed-form solutions

$$P(Y_c=y_a|Y_p=y_b)=\frac{\#ofTimesStateAFollowsStateB}{\#ofTimesStateBOccurs}$$

$$P(X_c=x_a|Y_c=y_b)=\frac{\#ofTimesOutputAIsObservedInStateB}{\#ofTimesStateBOccurs}$$

  - Need for smoothing the estimates (e.g. max a posteriori)

Support Vector Machine

- **Training Examples:** $(\vec{x}_1,y_1),\ldots,(\vec{x}_n,y_n)$ $\vec{x}\in\Re^N$ $y\in\{+1,-1\}$
- **Hypothesis Space:** $h(\vec{x})=sgn\left[\vec{w}^T\vec{x}+b\right]$ with $\vec{w}=\sum_{i=1}^{n}\alpha_i y_i \vec{x}_i$
- **Training:** Find hyperplane $<\vec{w},b>$ with minimal $\frac{1}{\delta^2}+C\sum_{i=1}^{n}\xi_i$

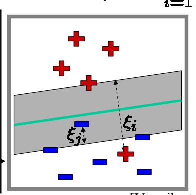
Optimization Problem:

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

$$s.t.\quad y_1(\vec{w}^T\vec{x}_1+b)\geq 1-\xi_1$$

$$\ldots$$

$$y_n(\vec{w}^T\vec{x}_n+b)\geq 1-\xi_n$$

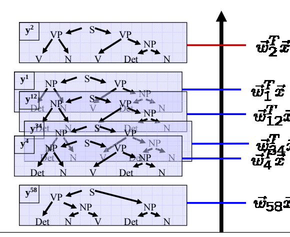


[Vapnik et al.]

Multi-Class SVM

- **Training Examples:** $(\vec{x}_1,y_1),\ldots,(\vec{x}_n,y_n)$ $\vec{x}\in\Re^N$ $y\in\{1,\ldots,k\}$
- **Hypothesis Space:** $h(\vec{x})= argmax_{i\in\{1,\ldots,k\}}\left[\vec{w}_i^T\vec{x}\right]$

x The dog chased the cat



[Crammer & Singer 02]

Training: Find $<\vec{w}_1,\ldots,\vec{w}_k>$ that solve

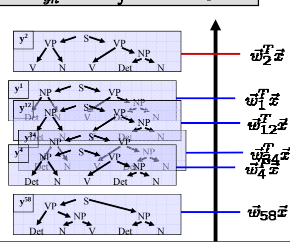
$$\min_{\vec{w}_1,\ldots,\vec{w}_n,\vec{\xi}} \sum_{i=1}^{k}\vec{w}_i^2+C\sum_{i=1}^{n}\xi_i$$

$$s.t.\quad \forall j\neq y_1: \vec{w}_{y_1}^T\vec{x}_1\geq\vec{w}_j^T\vec{x}_1+1-\xi_1$$

$$\ldots$$

$$\forall j\neq y_n: \vec{w}_{y_n}^T\vec{x}_n\geq\vec{w}_j^T\vec{x}_n+1-\xi_n$$

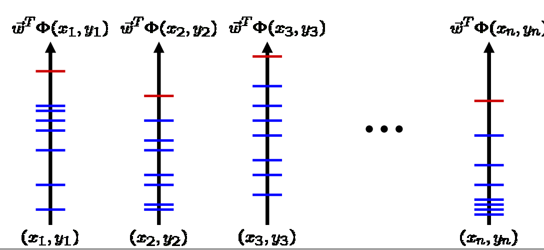
x The dog chased the cat



[Crammer & Singer 02]

Structural Support Vector Machine

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



Structural Support Vector Machine

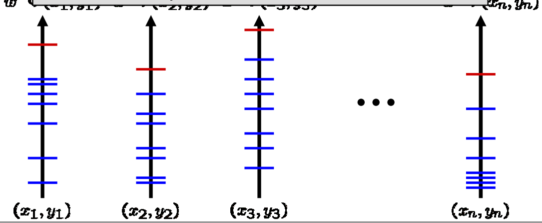
Hard-margin optimization problem:

$$\min_{\vec{w}} \quad \frac{1}{2}\vec{w}^T\vec{w}$$

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

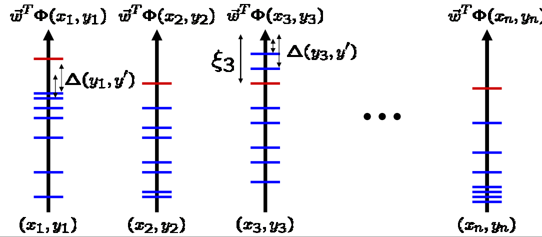
...

$$\forall y \in Y \backslash y_n : \vec{w}^T\Phi(x_n,y_n) \geq \vec{w}^T\Phi(x_n,y) + 1$$



Loss Functions: Soft-Margin Struct SVM

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



Loss Functions: Soft-Margin Struct SVM

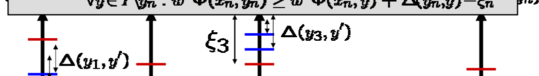
Soft-margin optimization problem:

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

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

...

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



Lemma: The training loss is upper bounded by

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

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

Training Approach 1: Factored QP

- **Assume:**
  - Linearly decomposable loss function: $\Delta(y',y) = \sum_{i=1}^{l}\delta(y'^{(i)},y^{(i)})$
  - Linear program solution is integral:

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

- **Algorithm:**
  - Min-Max Formulation:

$$\min_{\vec{w},\vec{\xi}} \quad \frac{1}{2}\vec{w}^T\vec{w} - C\left(\sum_{i=1}^{n}\underbrace{\vec{w}^T\Phi(x_i,y_i) - \max_{y'\in Y}\left[\Delta(y_i,y) + \vec{w}^T\Phi(x_i,y')\right]}_{= \xi_i}\right)$$

  - Plug in linear program for "max"
  - Quadratic program can be rewritten so that it has
    - Polynomially many variables
    - Polynomially many constraints

[Taskar et al. 03/04]

Training Approach 2: 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 $\hat{y} = argmax_{y\in Y}\{\Delta(y_i,y) + \vec{w}^T\Phi(x_i,y)\}$
    - IF $(\Delta(y_i,\hat{y}) - \vec{w}^T[\Phi(x_i,y_i) - \Phi(x_i,\hat{y})]) > \xi_i + \epsilon$
      - $S \leftarrow S \cup \{\vec{w}^T[\Phi(x_i,y_i) - \Phi(x_i,\hat{y})] \geq \Delta(y_i,\hat{y}) - \xi_i\}$
      - $[\vec{w},\vec{\xi}] \leftarrow$ 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]

Training Approach 2: Polynomial Sparsity Bound

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

$$n\frac{4CA^2R^2}{\epsilon^2}$$

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]

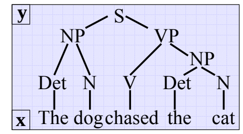
Experiment: Natural Language Parsing

- **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:**

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

- **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]

Applying Structural SVM to New Problem

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

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

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

- **Implementation SVM-struct: http://svmlight.joachims.org**
  - Context-free grammars
  - Sequence alignment
  - Classification with multivariate loss (e.g. F1, ROC Area)
  - General API for other problems

[Lafferty et al. 01]

Conditional Random Field (CRF)

- **Assume:**
  - model class P(Y|X,ω) with parameters ω ∈ Ω
  - In particular, $P(Y = y | X = x, \vec{w}) = \dfrac{e^{\vec{w}^T \Phi(x,y)}}{\sum_{y' \in Y} e^{\vec{w}^T \Phi(x,y')}}$
- **Training:** Maximum A Posteriori
  - Objective

$$\vec{w}' = \underset{\vec{w}}{\text{argmax}}\, e^{-\frac{1}{2\sigma^2}\vec{w}^T\vec{w}} \prod_{i=1}^{n} \left[\frac{e^{\vec{w}^T \Phi(x_i, y_i)}}{\sum_{y' \in Y} e^{\vec{w}^T \Phi(x_i, y')}} \right]$$

$$= \underset{\vec{w}}{\text{argmin}}\, \frac{1}{2\sigma^2}\vec{w}^T\vec{w} - \sum_{i=1}^{n} \left[\vec{w}^T \Phi(x_i, y_i) - \log \left[\sum_{y' \in Y} e^{\vec{w}^T \Phi(x_i, y')} \right] \right]$$

  - Gradient

$$\frac{\delta \mathcal{L}}{\delta w_j} = \frac{\vec{w}_j}{\sigma^2} - \sum_{i=1}^{n} \left[\Phi_j(x_i, y_i) - \frac{\delta}{\delta w_j} \left[\log \left[\sum_{y' \in Y} e^{\vec{w}^T \Phi(x_i, y')} \right] \right] \right]$$

  - Methods: iterative scaling, quasi Newton, conjugate gradient [Sha & Pereira 03] [Wallach 02] [Malouf 02] [Minka 03]

Applying CRF to New Problem

- **Application specific**
  - Representation $\Phi(x, y)$
  - Algorithms to compute

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

$$\frac{\delta Z(x_i)}{\delta w_j} = \frac{\delta}{\delta w_j} \left[\log \left[\sum_{y' \in Y} e^{\vec{w}^T \Phi(x_i, y')} \right] \right]$$

- **Implementation of CRF: http://mallet.cs.umass.edu/**

Relationship CRF and Structural SVM

- **Objective Functions:**
  - CRF:

$$\vec{w}' = \underset{\vec{w}}{\text{argmin}}\, \frac{1}{2\sigma^2}\vec{w}^T\vec{w} - \sum_{i=1}^{n} \left[\vec{w}^T \Phi(x_i, y_i) - \log \left[\sum_{y' \in Y} e^{\vec{w}^T \Phi(x_i, y')} \right] \right]$$

  - Structural SVM:

$$\vec{w}' = \underset{\vec{w}}{\text{argmin}}\, \frac{1}{2}\vec{w}^T\vec{w} - C \left(\sum_{i=1}^{n} \vec{w}^T \Phi(x_i, y_i) - \max_{y' \in Y} \left[\Delta(y_i, y') + \vec{w}^T \Phi(x_i, y') \right] \right)$$

- **Basic Cases for two classes**
  - CRF: Regularized logistic regression
  - Structural SVM: binary classification SVM

Overview

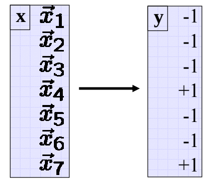
- **Task: Learning to predict complex outputs**
- **Formalizing the problem**
  - Multi-class classification: Generative vs. Discriminative
  - Compact models for problems with structured outputs
- **Training models with structured outputs**
  - Generative training
  - SVMs and large-margin training
  - Conditional likelihood training
- **Example 1: Learning to parse natural language**
  - Learning weighted context free grammar
- **Example 2: Optimizing F_1-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. F_1-score, Lift)**
  - F_1-score: harmonic average of precision and recall

$$F_1 = \frac{2\,Prec\,Rec}{Prec + Rec}$$

  - New example vector $\vec{x}_8$. Predict $y_8=1$, if $P(y_8=1|\vec{x}_8)=0.4$?
    ➔ Depends on other examples!

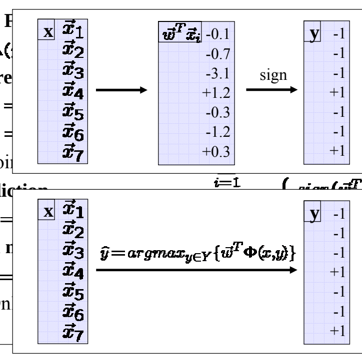


Struct SVM for Optimizing F_1-Score

- **Loss Function**
  - $\Delta(y,y') = (1-F_1) = \left(1 - \frac{2\,Prec\,Rec}{Prec + Rec}\right)$
- **Representation**
  - $x = (\vec{x}_1, \vec{x}_2, \vec{x}_3, \ldots, \vec{x}_n)$
  - $y = (y_1, y_2, y_3, \ldots, y_n)$
  - Joint feature map $\Phi(x,y) = \sum_{i=1}^{n} y_i \vec{x}_i$
- **Prediction**
  - $\hat{y} = argmax_{y \in Y}\{\vec{w}^T \Phi(x_i,y)\} \Longleftrightarrow \hat{y} = \begin{cases} sign(\vec{w}^T\vec{x}_1) \\ sign(\vec{w}^T\vec{x}_2) \\ \vdots \\ sign(\vec{w}^T\vec{x}_n) \end{cases}$
- **Find most violated constraint**
  - $\hat{y} = argmax_{y \in Y}\{\Delta(y_i,y) + \vec{w}^T\Phi(x_i,y)\}$
  - Only n^2 different contingency tables ➔ search brute force

[Jo 05]

Struct SVM for Optimizing F_1-Score



[Jo 05]

Experiment: Text Classification

- **Dataset: Reuters-21578 (ModApte)**
  - 9603 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**

Method	Test F_1	Training Efficiency CPU-min	Const	SV
Classification SVM	52.8	0.1	N/A	264
Linear Cost SVM	56.1	0.1	N/A	371
SVM with $(1\text{-}F_1)$-Loss	**62.0**	32.2	173	86

[Jo 05]

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:

$$\min_{\mathbf{w},\xi\geq 0} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi$$

$$s.t.\ \forall \vec{y}' \in \bar{\mathcal{Y}}: \mathbf{w}^T\Phi(\bar{x},\bar{y}) \geq \mathbf{w}^T\Phi(\bar{x},\vec{y}') + 2Err(\bar{y},\vec{y}') - \xi$$

$\Leftrightarrow$

Classification SVM (unbiased):

$$\min_{\mathbf{w},\xi\geq 0} \frac{1}{2}\mathbf{w}^T\mathbf{w} + 2C\sum_{i=1}^{n}\xi_i$$

$$s.t.\quad y_1(\mathbf{w}^T\mathbf{x}_1) \geq 1-\xi_1,\ \ldots\ ,\ y_n(\mathbf{w}^T\mathbf{x}_n) \geq 1-\xi_n$$

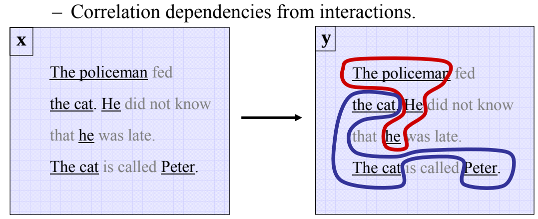
[Jo 05]

Overview

- **Task: Learning to predict complex outputs**
- **Formalizing 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 F_1-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.



[Finley & Jo 05], see also [McCallum & Wellner 04]

Struct SVM for Supervised Clustering

- **Representation**
  - $x = \begin{pmatrix} \vec{x}_{11} & \vec{x}_{12} & \cdots & \vec{x}_{1n} \\ \vec{x}_{21} & \vec{x}_{22} & \cdots & \vec{x}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \vec{x}_{n1} & \vec{x}_{n2} & \cdots & \vec{x}_{nn} \end{pmatrix} \quad y = \begin{pmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{n1} & y_{n2} & \cdots & y_{nn} \end{pmatrix} y_{ij} \in \{0,1\}$
  - *y* is reflexive ($y_{ii}=1$), symmetric ($y_{ij}=y_{ji}$), and transitive (if $y_{ij}=1$ and $y_{jk}=1$, then $y_{ik}=1$)
  - Joint feature map $\Phi(x,y) = \sum_{i=1}^{n}\sum_{j=1}^{n} y_{ij}\vec{x}_{ij}$
- **Loss Function**
  - $\Delta(y,y') = ||y-y'||_1$
- **Prediction**
  - $\hat{y} = argmax_{y\in Y}\{\vec{w}^T\Phi(x_i,y)\} = argmax_{y\in Y}\{\sum\sum y_{ij}(\vec{w}^T x_{ij})\}$
  - NP hard, use linear relaxation instead [Demaine & Immorlica, 2003]
- **Find most violated constraint**
  - $\hat{y} = argmax_{y\in Y}\{\Delta(y_i,y) + \vec{w}^T\Phi(x_i,y)\}$
  - NP hard, use linear relaxation instead [Demaine & Immorlica, 2003]

[Finley & Jo 05], see also [McCallum & Wellner 04]

Struct SVM for Supervised Clustering

- **Rep**
  - … $\{0,1\}$
  - … nd
- **Los**
  - $\Delta(y,y') = ||y-y'||_1$
- **Predic**
  - $\hat{y}=$ … $\sum y_{ij}(\vec{w}^T x_{ij})\}$
  - NP … orlica, 2003]
- **Find n**
  - $\hat{y}=$
  - NP … orlica, 2003]

$\vec{w}^T\vec{x}_i$						
1.0	2.5	0.0	0.0	-2.1	-0.1	0.0
0.0	1.0	3.0	0.1	0.1	0.3	0.0
0.0	0.0	1.0	0.3	2.2	-0.9	0.0
0.9	-0.4	-0.1	1.0	-1.1	-0.1	-2.0
0.0	0.0	0.0	-0.3	1.0	2.3	0.0
0.1	0.1	0.0	-0.1	0.1	1.0	1.8
-1.0	-0.7	0.0	-0.2	-1.1	0.1	1.0

→

y						
1	1	1	1	0	0	0
1	1	1	1	0	0	0
1	1	1	1	0	0	0
1	1	1	1	0	0	0
0	0	0	0	1	1	1
0	0	0	0	1	1	1
0	0	0	0	1	1	1

y						
1	1	1	1	0	0	0
1	1	1	1	0	0	0
1	1	1	1	0	0	0
1	1	1	1	0	0	0
0	0	0	0	1	1	1
0	0	0	0	1	1	1
0	0	0	0	1	1	1

y'						
1	1	1	**0**	0	0	0
1	1	1	**0**	0	0	0
1	1	1	**0**	0	0	0
0	**0**	**0**	1	0	0	0
0	0	0	0	1	1	1
0	0	0	0	1	1	1
0	0	0	0	1	1	1

Summary

- **Learning to predict structured and interdependent output**
  - Discriminant function: $h(x) = \text{argmax}_{y\in Y}\,[h(x,y)]$
- **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:**
  - SVM[struct]: `http://svmlight.joachims.org/`
  - Mallet: `http://mallet.cs.umass.edu/`

Reading

- **Generative training**
  - Hidden-Markov models [Manning & Schuetze, 1999]
  - Probabilistic context-free grammars [Manning & Schuetze, 1999]
  - Markov random fields [Geman & Geman, 1984]
  - Etc.
- **Discriminative training**
  - Multivariate output regression [Izeman, 1975] [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] [Tschochantaridis et al., 2004, 2005] [Taskar 2004]