Advanced Machine Learning & Perception

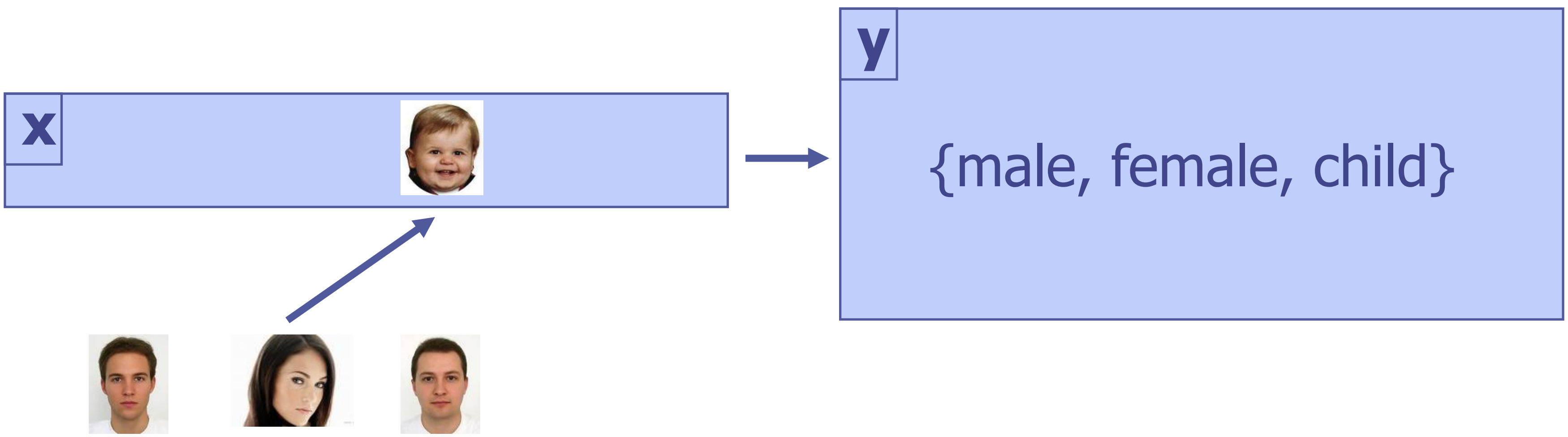
Instructor: Tony Jebara

Graphical Models

- Conditional Multi-Class and Structured Prediction
- Review: Graphical Models
- Review: Junction Tree Algorithm
- MAP Estimation
- Discriminative Multi-Class SVM and Structured SVM
- Cutting Plane Algorithms
- Large Margin versus Large Relative Margin

Multi-Class & Structured Output

Logistic regression initially only handled binary outputs
It can easily also handle multi-class labels

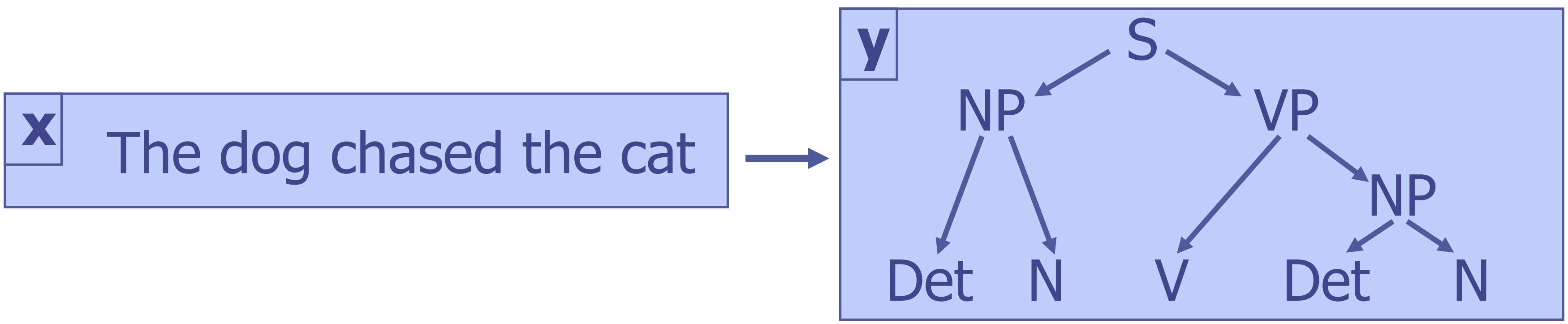


Multi-Class & Structured Output

Can logistic regression or CRF handle structured output?

For example: 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.



Multi-Class & Structured Output

For example: 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
```

Multi-Class & Structured Output

For example: Information Retrieval

- Given a query x, predict a ranking *y*.
- Dependencies between results (e.g. avoid redundant hits)
- Loss function over rankings (e.g. AvgPrec)

x Boosting

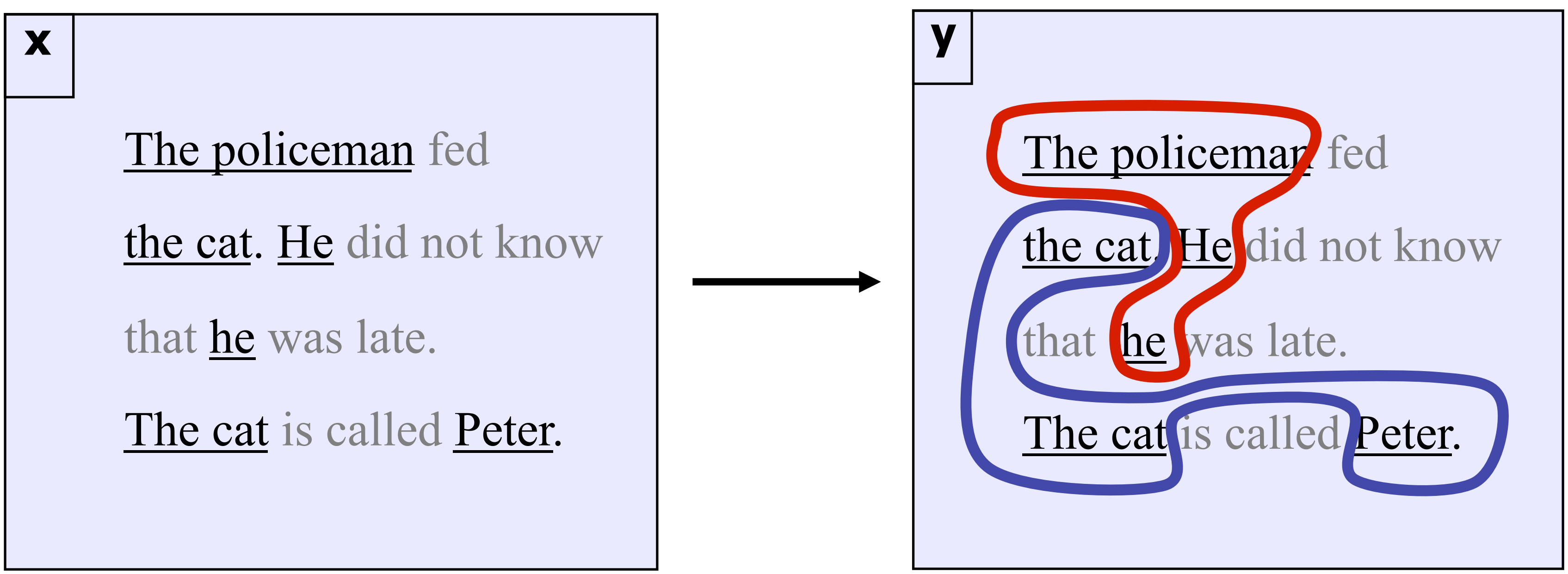
y

1. AdaBoost
2. Freund
3. Schapire
4. Kernel-Machines
5. Support Vector Machines
6. MadaBoost
7. …

Multi-Class & Structured Output

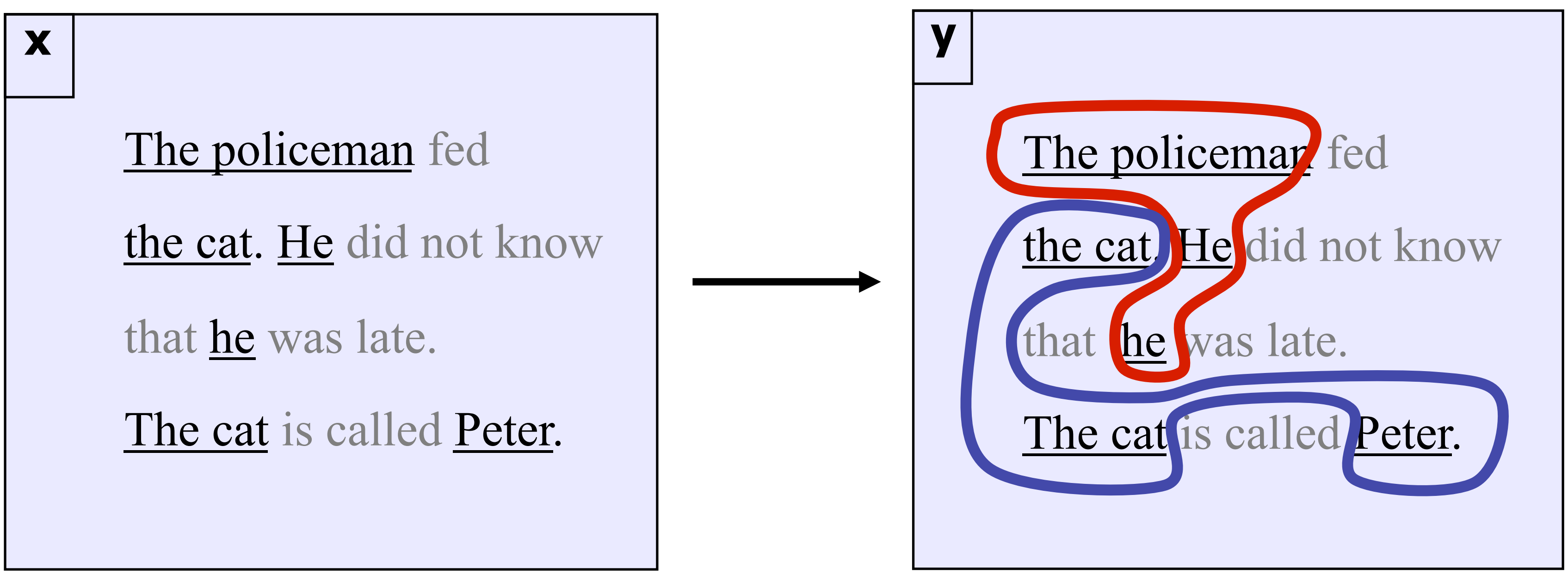
For Example, Noun-Phrase Co-reference

- 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.



Multi-Class & Structured Output

- These problems are usually solved via maximum likelihood
- Or via Bayesian Networks and Graphical Models
- Problem: these methods are not discriminative!
- They learn p(x,y), we want a p(y|x) like a CRF…
- We will adapt the CRF approach to these domains…



CRFs for Structured Prediction

- Recall CRF or log-linear model: $p\left(y\middle|\mathbf{x}\right)=\frac{1}{Z_{\mathbf{x}}\left(\theta\right)}\exp\left(\theta^T\mathbf{f}\left(\mathbf{x},y\right)\right)$

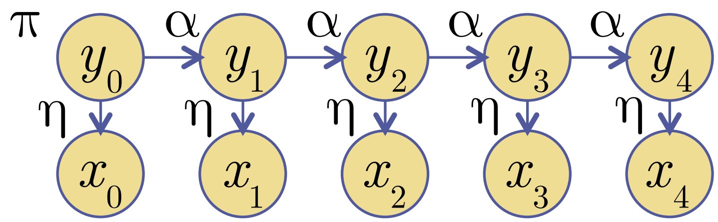
- The key of structured prediction is fast computation of: $\arg\max_y \theta^T\mathbf{f}\left(\mathbf{x},y\right)$

  and fast calculation of: $\sum_y p\left(y\mid\mathbf{x}\right)\mathbf{f}\left(\mathbf{x},y\right)$

- Usually, the space y is too huge to enumerate
- If y splits into many conditionally independent terms
  - → finding the max (Decoding) may be efficient
  - → computing sums (Inference) may be efficient
  - → computing the gradient may be efficient
- Graphical models have three canonical problems to solve: 1) Marginal inference, 2) Decoding and 3) Learning

Structured Prediction & HMMs

- Recall Hidden Markov Model (now x is observed, y hidden):



space of y's is $O(M^T)$

- Here, space of y's is *huge* just like in structured prediction
- Would like to do 3 basic things with graphical models:
  1) Evaluate: given $x_1,\ldots,x_T$ compute likelihood $p(x_1,\ldots,x_T)$
  2) Decode: given $x_1,\ldots x_T$ compute best $y_1,\ldots,y_T$ or $p(y_t)$
  3) Learn: given $x_1,\ldots,x_T$ learn parameters θ

- Typically, HMMs use Baum-Welch, α-β or Viterbi algorithm
- More general graphical models use Junction Tree Algorithm
- The JTA is a way of performing efficient inference

Inference

- Inference: goal is to predict some variables given others

  y1: flu
  y2: fever
  y3: sinus infection
  y4: temperature
  y5: sinus swelling
  y6: headache

  Patient claims headache and high temperature. Does he have a flu?

  Given findings variables Y_f and unknown variables Y_u predict queried variables Y_q

- Classical approach: truth tables (slow) or logic networks

- Modern approach: probability tables (slow) or Bayesian networks (fast belief propagation, junction tree algorithm)

Aka Bayesian Networks

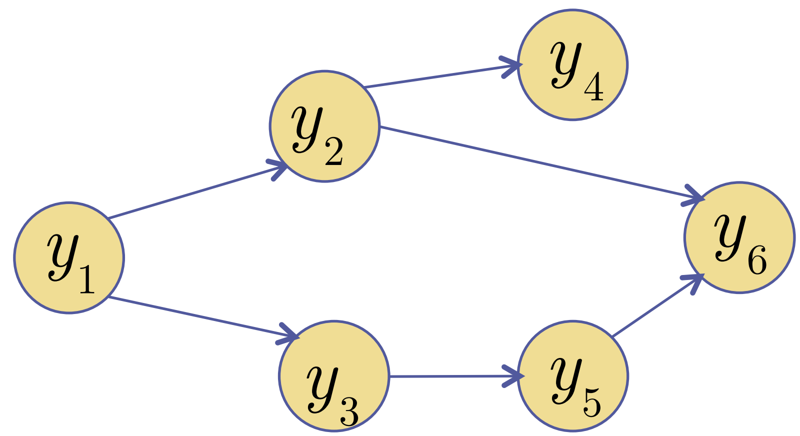
Directed Graphical Models

- Factorize a large (how big?) probability over several vars

$$p\left(y_1,\ldots,y_n\right) = \prod_{i=1}^{n} p\left(y_i \mid pa_i\right) = \prod_{i=1}^{n} p\left(y_i \mid \pi_i\right)$$

- Interpretation

  1: flu
  2: fever
  3: sinus infection
  4: temperature
  5: sinus swelling
  6: headache



$$p\left(y_1,\ldots,y_6\right) = p\left(y_1\right) p\left(y_2 \mid y_1\right) p\left(y_3 \mid y_1\right) p\left(y_4 \mid y_2\right) p\left(y_5 \mid y_3\right) p\left(y_6 \mid y_2, y_5\right)$$

$$2^6 \qquad 2^1 \qquad 2^2 \qquad 2^2 \qquad 2^2 \qquad 2^2 \qquad 2^3$$

Undirected Graphical Models

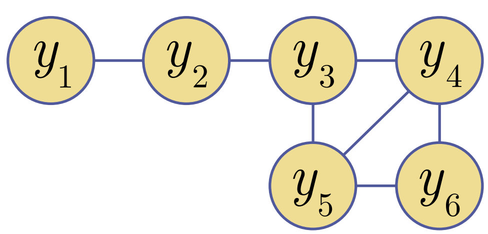
- Probability for undirected is defined via Potential Functions which are more flexible than conditionals or marginals

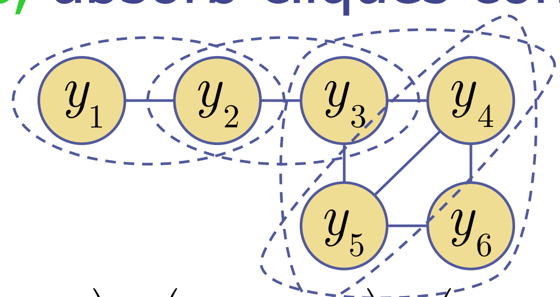
$$p(Y) = p(y_1,\ldots,y_M) = \tfrac{1}{Z}\prod_C \psi(Y_C) \qquad Z = \sum_Y \prod_C \psi(Y_C)$$

- Just a factorization of p(Y), Z just normalizes the pdf
- Potential functions are positive functions of (not mutually exclusive) sub-groups of variables

0.1	**0.2**
0.05	**0.3**

- Potential functions are over complete sub-graphs or cliques C in the graph, clique is a set of fully-interconnected nodes
- Use maximal cliques, absorb cliques contained in larger ψ





$$\psi(y_2,y_3)\,\psi(y_2)\,\psi(y_3) \rightarrow \psi(y_2,y_3)$$

$$p(Y) = \tfrac{1}{Z}\psi(y_1,y_2)\,\psi(y_2,y_3)\,\psi(y_3,y_4,y_5)\,\psi(y_4,y_5,y_6)$$

Junction Tree Algorithm

- Involves 5 steps, the first 4 build the Junction Tree:

1) Moralization

    Polynomial in # of nodes

2) Introduce Evidence (fixed or constant)

    Polynomial in # of nodes (convert pdf to slices)

3) Triangulate (Tarjan & Yannakakis 1984)

    Suboptimal=Polynomial, Optimal=NP

4) Construct Junction Tree (Kruskal)

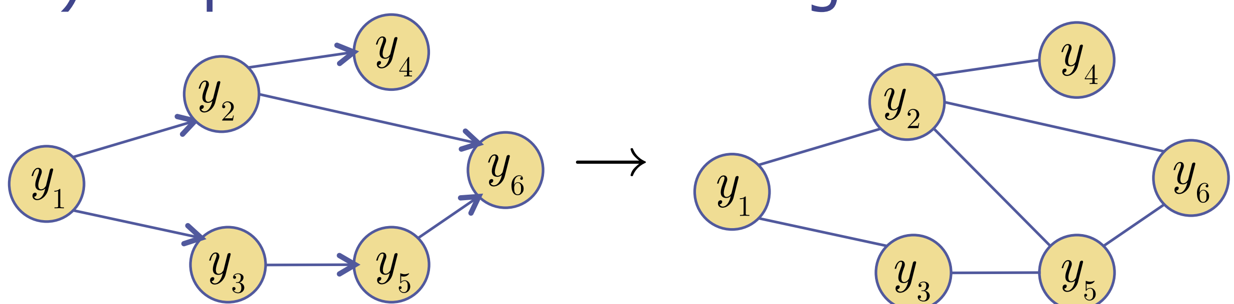
    Polynomial in # of cliques

5) Junction Tree Algorithm (Init,Collect,Distribute,Normalize)

    Polynomial (linear) in # of cliques, *Exponential* in Clique Cardinality

Moralization

- Converts directed graph into undirected graph
- By moralization, marrying the parents:
  1) Connect nodes that have common children
  2) Drop the arrow heads to get undirected



$$p(y_1)\,p(y_2 \mid y_1)\,p(y_3 \mid y_1)\,p(y_4 \mid y_2)\,p(y_5 \mid y_3)\,p(y_6 \mid y_2, y_5)$$
$$\longrightarrow \quad \tfrac{1}{Z}\psi(y_1,y_2)\,\psi(y_1,y_3)\,\psi(y_2,y_4)\,\psi(y_3,y_5)\,\psi(y_2,y_5,y_6)$$

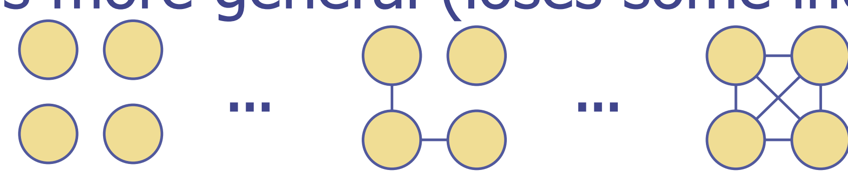
$$p(y_1)\,p(y_2 \mid y_1) \;\longrightarrow\; \psi(y_1,y_2)$$

$$p(y_4 \mid y_2) \;\longrightarrow\; \psi(y_2,y_4)$$

$$Z \longrightarrow 1$$

- Note: moralization resolves *coupling* due to marginalizing
- moral graph is more general (loses some independencies)

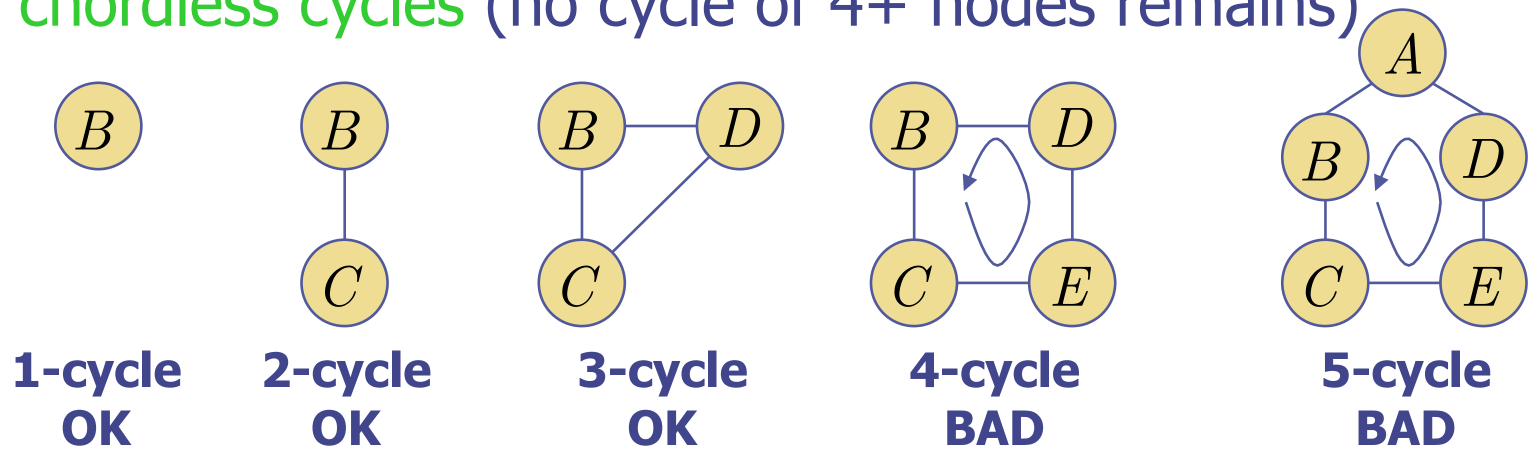
most specific



most general

Triangulation

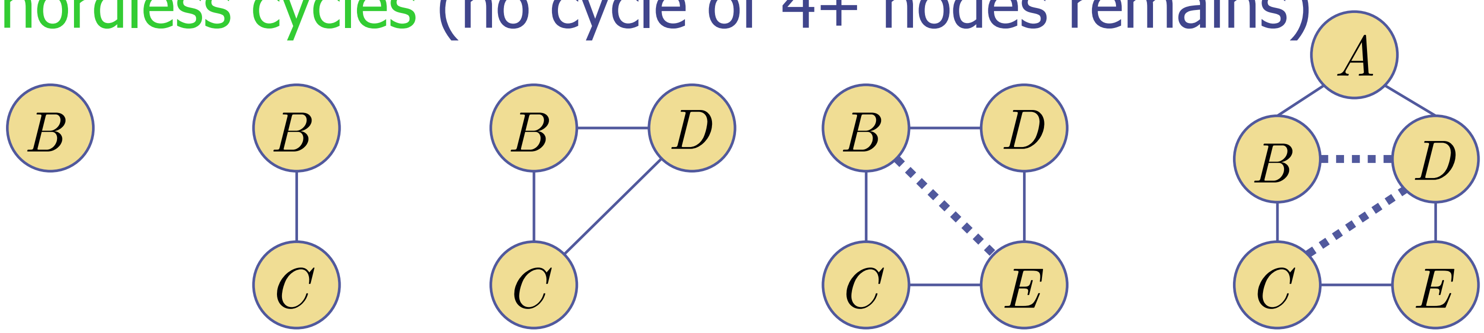
- **Triangulation:** Connect nodes in moral graph such that no chordless cycles (no cycle of 4+ nodes remains)



- So, *add links*, but many possible choices…
- HINT: keep largest clique size small (for efficient JTA)
- **Chordless:** no edges between successor nodes in cycle
- Sub-optimal triangulations of moral graph are Polynomial
- Triangulation that minimizes largest clique size is NP
- But, OK to use a suboptimal triangulation (slower JTA…)

Triangulation

- Triangulation: Connect nodes in moral graph such that no chordless cycles (no cycle of 4+ nodes remains)

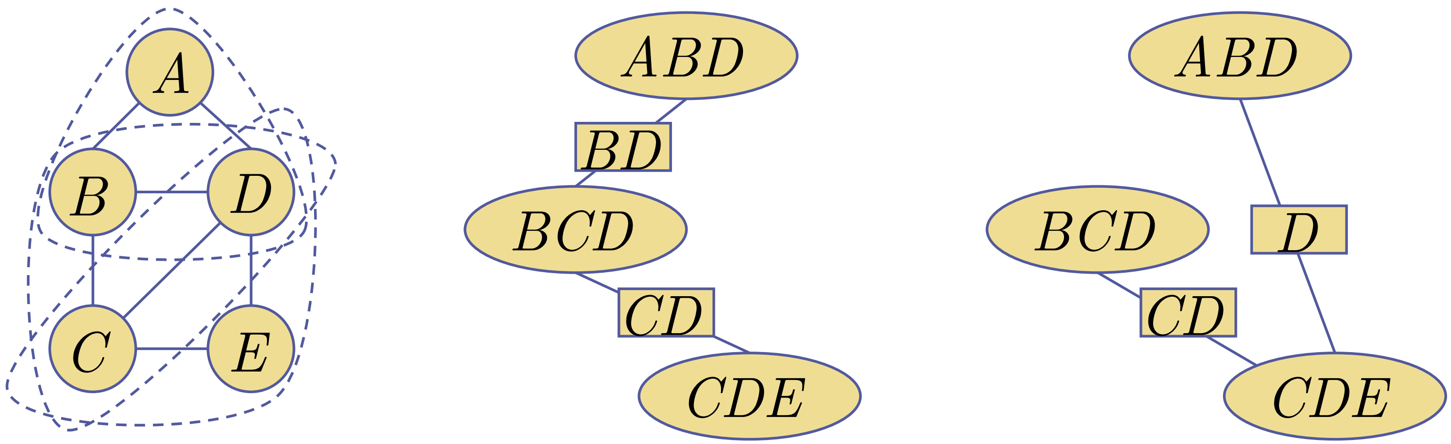


1-cycle OK **2-cycle OK** **3-cycle OK** **3-cycle OK** **3-cycle OK**

- So, *add links*, but many possible choices…
- HINT: keep largest clique size small (for efficient JTA)
- Chordless: no edges between successor nodes in cycle
- Sub-optimal triangulations of moral graph are Polynomial
- Triangulation that minimizes largest clique size is NP
- But, OK to use a suboptimal triangulation (slower JTA…)

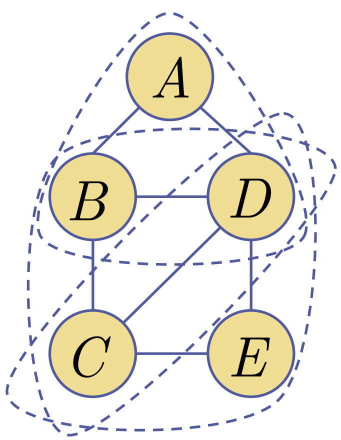
Running Intersection Property

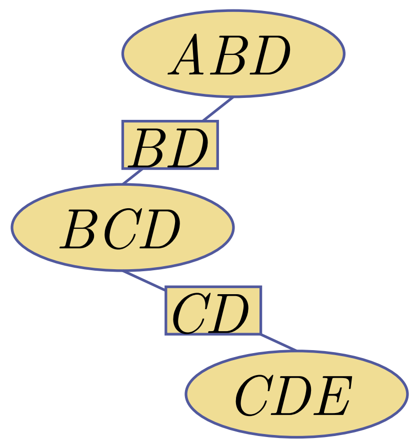
- Junction Tree must satisfy Running Intersection Property
- RIP: On unique path connecting clique V to clique W, all other cliques share nodes in $V \cap W$

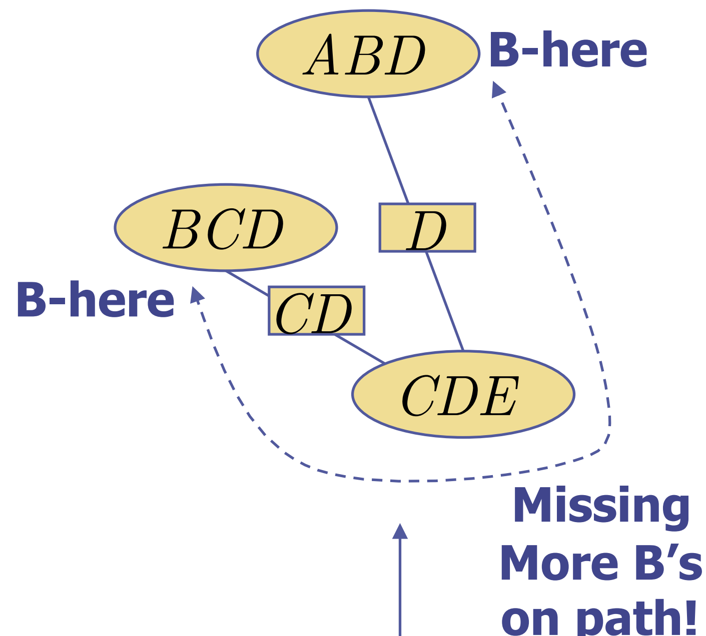


Running Intersection Property

- Junction Tree must satisfy Running Intersection Property
- RIP: On unique path connecting clique V to clique W, all other cliques share nodes in $V \cap W$







HINT: Junction Tree has largest total separator cardinality

$$\left|\Phi\right| = \left|\phi\left(B,C\right)\right| + \left|\phi\left(C,D\right)\right| = 2+2$$

$$\left|\Phi\right| = \left|\phi\left(C,D\right)\right| + \left|\phi\left(D\right)\right| = 2+1$$

Forming the Junction Tree

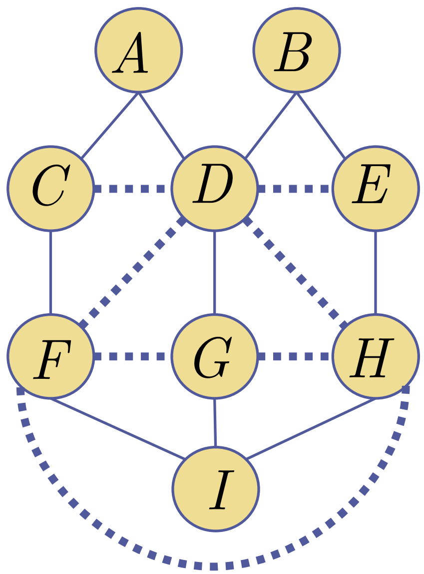
- Now need to connect the cliques into a Junction Tree
- But, must ensure Running Intersection Property
- Theorem: a valid (RIP) Junction Tree connection is one that maximizes the cardinality of the separators

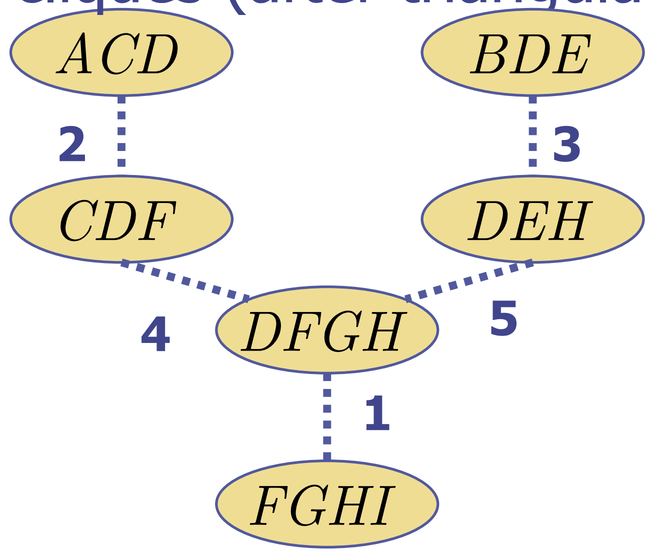
$$JT^* = \max_{TREE\,STRUCTURES} \left|\Phi\right|$$
$$= \max_{TREE\,STRUCTURES} \sum_S \left|\phi\left(Y_S\right)\right|$$

- Use Kruskal's algorithm:
  1) Init Tree with all cliques unconnected (no edges)
  2) Compute size of separators between all pairs
  3) Connect the two cliques with the biggest separator cardinality which doesn't create a loop in current Tree (maintains Tree structure)
  4) Stop when all nodes are connected, else goto 3

Kruskal Example

- Start with unconnected cliques (after triangulation)





	ACD	BDE	CDF	DEH	DFGH	FGHI
ACD	-	1	2	1	1	0
BDE		-	1	2	1	0
CDF			-	1	2	1
DEH				-	2	1
DFGH					-	3
FGHI						-

Junction Tree Probabilities

- We now have a valid Junction Tree!
- What does that mean?
- Recall probability for undirected graphs:

$$p(Y) = p(y_1,\ldots,y_M) = \tfrac{1}{Z}\prod_C \psi(Y_C)$$

- Can write junction tree as potentials of its cliques:

$$p(Y) = \tfrac{1}{Z}\prod_C \tilde{\psi}(Y_C)$$

- Alternatively: clique potentials over separator potentials:

$$p(Y) = \frac{1}{Z}\frac{\prod_C \psi(Y_C)}{\prod_S \phi(Y_S)}$$

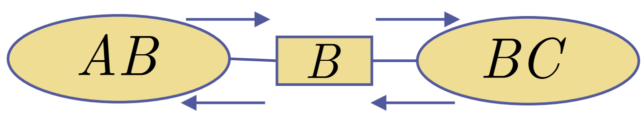
- This doesn't change/do anything! Just less compact…
- Like *de-absorbing* smaller cliques from maximal cliques:

$$\tilde{\psi}(A,B,D) = \frac{\psi(A,B,D)}{\phi(B,D)} \longleftarrow$$

…gives back original formula if $\phi(B,D) \triangleq 1$

Junction Tree Algorithm

- Send message from each clique *to* its separators of what it thinks the submarginal on the separator is.
- Normalize each clique by incoming message *from* its separators so it agrees with them



$$V = \{A,B\} \quad S = \{B\} \quad W = \{B,C\}$$

If agree: $\sum_{V \backslash S} \psi_V = \phi_S = p(S) = \phi_S = \sum_{W \backslash S} \psi_W$ **...Done!**

Else:

Send message From V to W...

$$\phi_S^* = \sum_{V \backslash S} \psi_V$$
$$\psi_W^* = \frac{\phi_S^*}{\phi_S} \psi_W$$
$$\psi_V^* = \psi_V$$

Send message From W to V...

$$\phi_S^{**} = \sum_{W \backslash S} \psi_W^*$$
$$\psi_V^{**} = \frac{\phi_S^{**}}{\phi_S^*} \psi_V^*$$
$$\psi_W^{**} = \psi_W^*$$

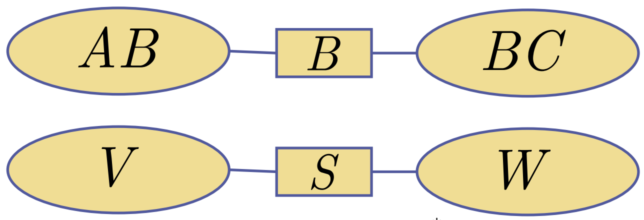
Now they Agree...Done!

$$\sum_{V \backslash S} \psi_V^{**} = \sum_{V \backslash S} \frac{\phi_S^{**}}{\phi_S^*} \psi_V^* = \frac{\phi_S^{**}}{\phi_S^*} \sum_{V \backslash S} \psi_V^* = \phi_S^{**} = \sum_{W \backslash S} \psi_W^{**}$$

Junction Tree Algorithm

- When "Done", all clique potentials are marginals and all separator potentials are submarginals!
- Note that p(X) is unchanged by message passing step:

$$\boxed{\begin{aligned} \phi_S^* &= \sum\nolimits_{V \backslash S} \psi_V \\ \psi_W^* &= \frac{\phi_S^*}{\phi_S} \psi_W \\ \psi_V^* &= \psi_V \end{aligned}}$$



$$p(Y) = \frac{1}{Z} \frac{\psi_V^* \psi_W^*}{\phi_S^*} = \frac{1}{Z} \frac{\psi_V \frac{\phi_S^*}{\phi_S} \psi_W}{\phi_S^*} = \frac{1}{Z} \frac{\psi_V \psi_W}{\phi_S}$$

- Example: if potentials are poorly initialized… get corrected!

$$\begin{aligned} \psi_{AB} &= p(B \mid A)\, p(A) \\ &= p(A,B) \end{aligned} \longrightarrow \phi_B^* = \sum\nolimits_A \psi_{AB} = \sum\nolimits_A p(A,B) = \boxed{p(B)}$$

$$\psi_{BC} = \boxed{p(C \mid B)} \longrightarrow \psi_{BC}^* = \frac{\phi_S^*}{\phi_S} \psi_{BC} = \frac{p(B)}{1} p(C \mid B) = \boxed{p(B,C)}$$

$$\phi_B = \boxed{1}$$

Junction Tree Algorithm

- Use tree recursion rather than iterate messages mindlessly!

initialize(DAG){ Pick root
Set all variables as: $\psi_{C_i} = p\left(y_i \mid \pi_i\right), \phi_S = 1$ **}**

collectEvidence(node) {
for each child of node {
update1(node,collectEvidence(child)); }
return(node); }

distributeEvidence(node) {
for each child of node {
update2(child,node);
distributeEvidence(child); } }

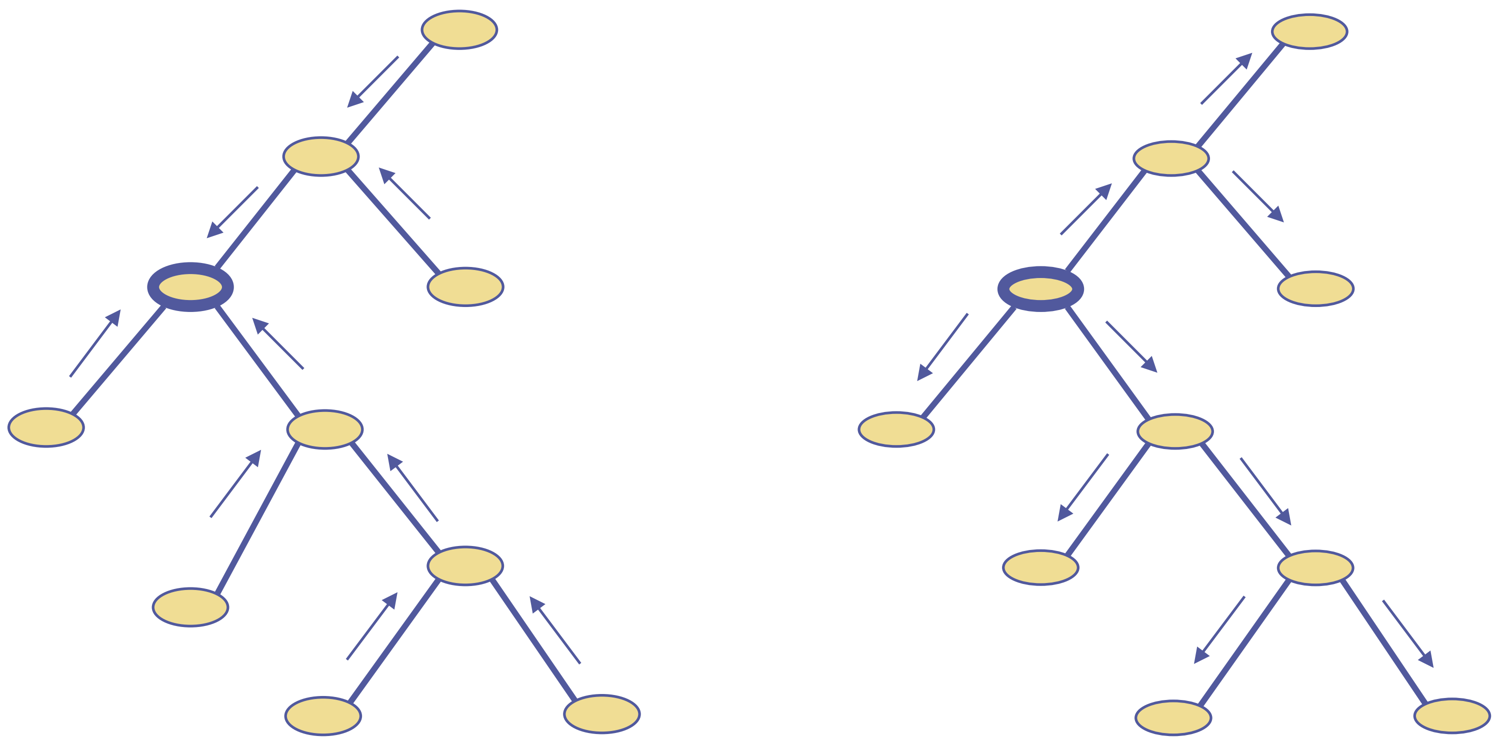
update1(node w,node v) { $\phi^{*}_{V\cap W} = \sum_{V\setminus(V\cap W)} \psi_V, \;\; \psi_W = \frac{\phi^{*}_{V\cap W}}{\phi_{V\cap W}} \psi_W$ **}**

update2(node w,node v) { $\phi^{**}_{V\cap W} = \sum_{V\setminus(V\cap W)} \psi_V, \;\; \psi_W = \frac{\phi^{**}_{V\cap W}}{\phi^{*}_{V\cap W}} \psi_W$ **}**

normalize() { $p\left(Y_C\right) = \frac{1}{\sum_C \psi^{**}_C} \psi^{**}_C \;\; \forall C, \;\; p\left(Y_S\right) = \frac{1}{\sum_S \phi^{**}_S} \phi^{**}_S \;\; \forall S$ **}**

Junction Tree Algorithm

•JTA: 1)*Initialize* 2)*Collect* 3)*Distribute* 4)*Normalize*



•Note: leaves do not change their ψ during *collect*
•Note: the first cliques *collect* changes are parents of leaves
•Note: root does not change its ψ during *distribute*

ArgMax Junction Tree Algorithm

- We can also use JTA for finding the max (not the sum) over the joint to get argmax of marginals & conditionals
- Say have some evidence: $p\left(Y_F, \bar{Y}_E\right) = p\left(y_1, \ldots, y_n, \bar{y}_{n+1}, \ldots, \bar{y}_N\right)$
- Most likely (highest p) Y_F? $Y_F^* = \arg\max_{Y_F} p\left(Y_F, \bar{Y}_E\right)$
- What is most likely state of patient with fever & headache?

$$\begin{aligned} p_F^* &= \max_{y_2, y_3, y_4, y_5} p\left(y_1 = 1, y_2, y_3, y_4, y_5, y_6 = 1\right) \\ &= \max_{y_2} p\left(y_2 \mid y_1 = 1\right) p\left(y_1 = 1\right) \max_{y_3} p\left(y_3 \mid y_1 = 1\right) \\ &\quad \max_{y_4} p\left(y_4 \mid y_2\right) \max_{y_5} p\left(y_5 \mid y_3\right) p\left(y_6 = 1 \mid y_2, y_5\right) \end{aligned}$$

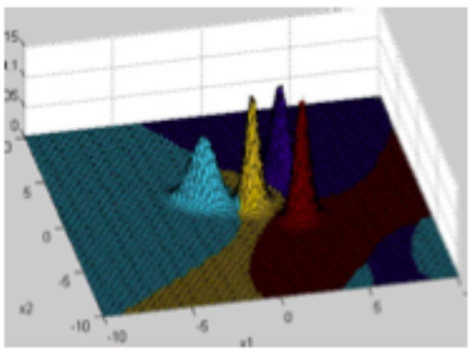
- Solution: update in JTA uses max instead of sum:

$$\phi_S^* = \max_{V \setminus S} \psi_V \qquad \psi_W^* = \frac{\phi_S^*}{\phi_S} \psi_W \qquad \psi_V^* = \psi_V$$

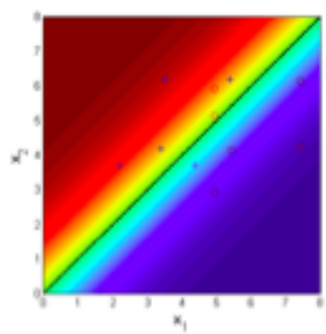
- Final potentials aren't marginals: $\psi\left(X_C\right) = \max_{U \setminus C} p\left(Y\right)$
- Highest value in potential is most likely: $Y_C^* = \arg\max_C \psi\left(Y_C\right)$

Generative, Conditional and Discriminative Prediction

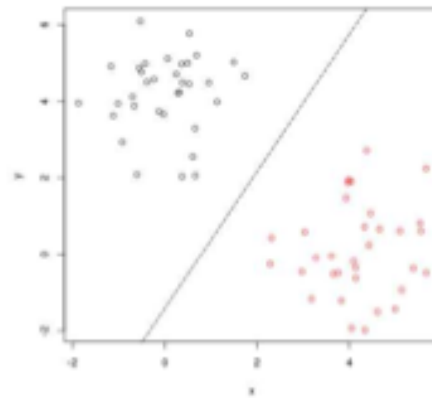
- Generative: hidden Markov model learns p(x,y)
- Conditional: conditional random field learns p(y|x)
- Discriminative: structured SVM learns y=f(x) where y is big
  - Generate & Conditional Need JTA & ArgMax JTA
  - Discriminative only needs ArgMax JTA



Generative



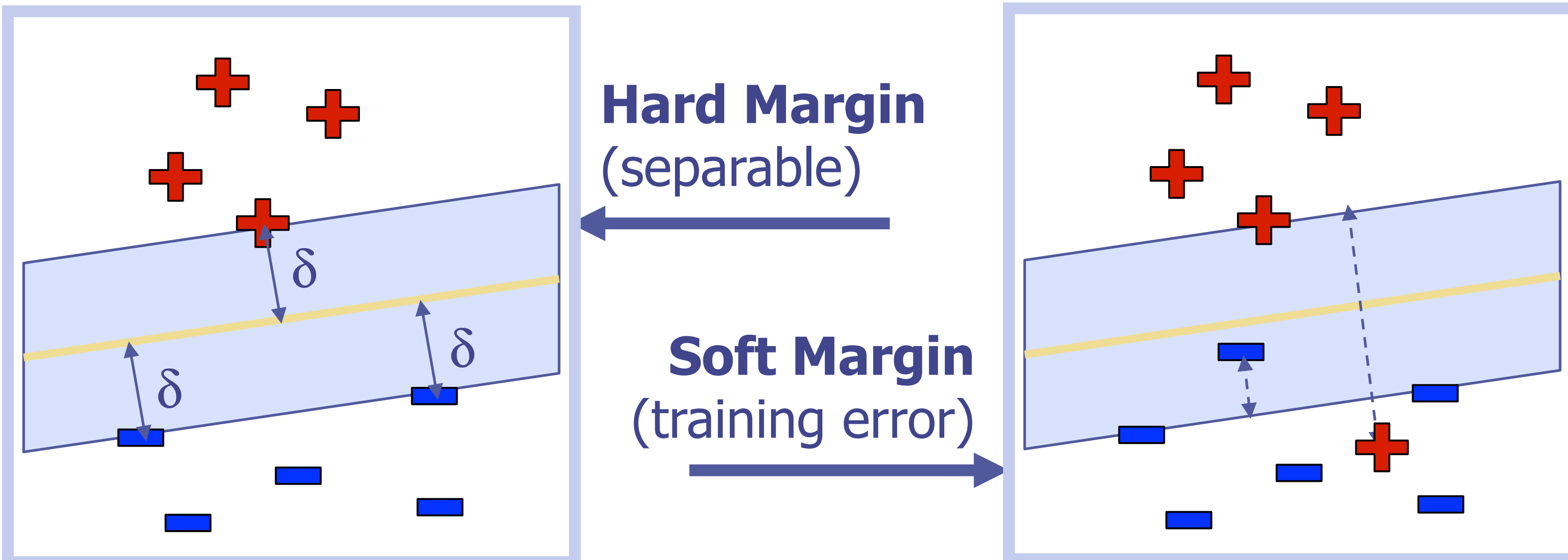
Conditional



Discriminative

Large-Margin SVM

- Binary classification: $\left\{\left(\mathbf{x}_1, y_1\right), \ldots, \left(\mathbf{x}_n, y_n\right)\right\} \rightarrow f\left(\mathbf{x}\right) = \mathbf{w}^T \mathbf{x} + b$



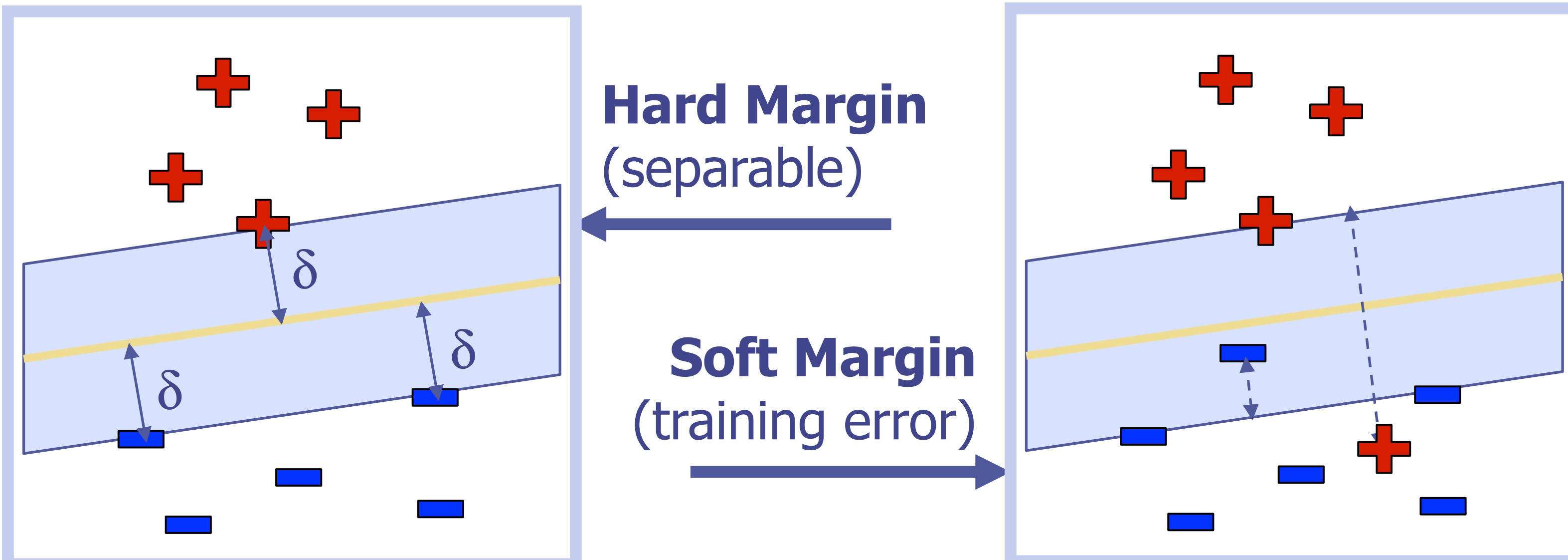
- P: $\min_{w,b,\xi \geq 0} \frac{1}{2}\left\|\mathbf{w}\right\|^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i \quad s.t. \quad y_i\left(\mathbf{w}^T\mathbf{x}_i + b\right) \geq 1 - \xi_i$

- D: $\max_{\lambda} \sum_{i=1}^{n} \lambda_i - \frac{1}{2}\sum_{i,j=1}^{n} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \;\; s.t. 0 \leq \lambda_i \leq \frac{C}{n}, \sum_{i=1}^{n} \lambda_i y_i = 0$

- Primal (P) and dual (D) give same solution $\mathbf{w}^* = \sum_{i=1}^{n} \lambda_i^* y_i \mathbf{x}_i$

Large-Margin SVM with b=0

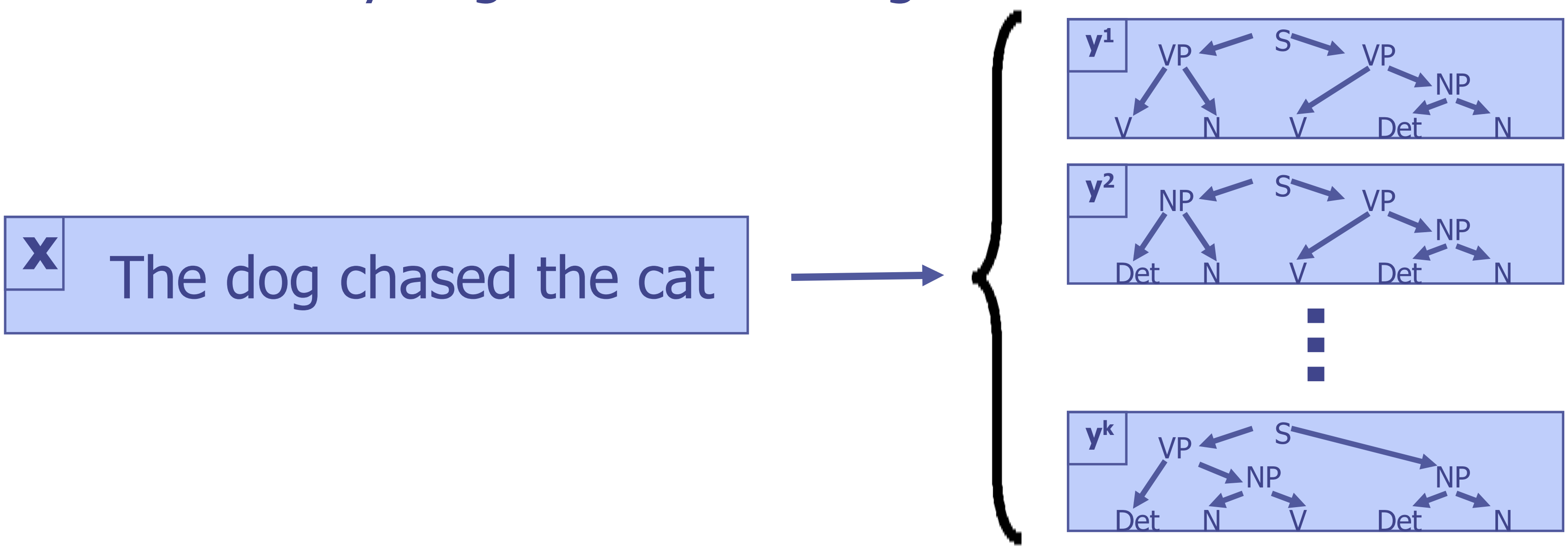
- Binary classification: $\left\{\left(\mathbf{x}_1, y_1\right), \ldots, \left(\mathbf{x}_n, y_n\right)\right\} \rightarrow f\left(\mathbf{x}\right) = \mathbf{w}^T \mathbf{x}$



- P: $\min_{w,b,\xi \geq 0} \frac{1}{2}\left\|\mathbf{w}\right\|^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i \quad s.t. \quad y_i\left(\mathbf{w}^T\mathbf{x}_i\right) \geq 1 - \xi_i$
- D: $\max_{\lambda} \sum_{i=1}^{n} \lambda_i - \frac{1}{2}\sum_{i,j=1}^{n} \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j \;\; s.t. 0 \leq \lambda_i \leq \frac{C}{n}$
- Solution through origin $\mathbf{w}^* = \sum_{i=1}^{n} \lambda_i^* y_i \mathbf{x}_i$ (or just pad x with 1)

Multi-Class & Structured Output

- View the problem as a list of all possible answers
- Approach: view as multi-class classification task
- Every complex output $y_i \in Y$ is one class
- Problems: Exponentially many classes!
    - How to predict efficiently? How to learn efficiently?
    - Potentially huge model! Manageable number of features?



Multi-Class Output

- View the problem as a list of all possible answers
- Approach: view as multi-class classification task
- Every complex output $y_i \in \{1,\ldots,k\}$ is one of K classes
- Enumerate many constraints (slow)…

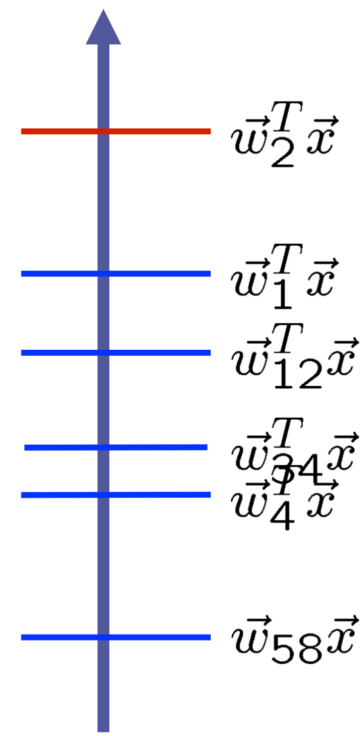
$$\left\{\left(\mathbf{x}_1, y_1\right),\ldots,\left(\mathbf{x}_n, y_n\right)\right\} \rightarrow f\left(\mathbf{x}\right) = \arg\max_{i\in\{1,\ldots,k\}} \mathbf{w}_i^T \mathbf{x}$$

$$\min_{\mathbf{w}_1,\ldots,\mathbf{w}_k,\xi\geq 0} \sum_{i=1}^{k} \left\|\mathbf{w}_i\right\|^2 + \tfrac{C}{n}\sum_{i=1}^{n}\xi_i$$

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

$$s.t.\quad \ldots$$

$$s.t.\quad \forall j \neq y_n : \left(\mathbf{w}_{y_n}^T \mathbf{x}_n\right) \geq \left(\mathbf{w}_j^T \mathbf{x}_n\right) + 1 - \xi_n$$

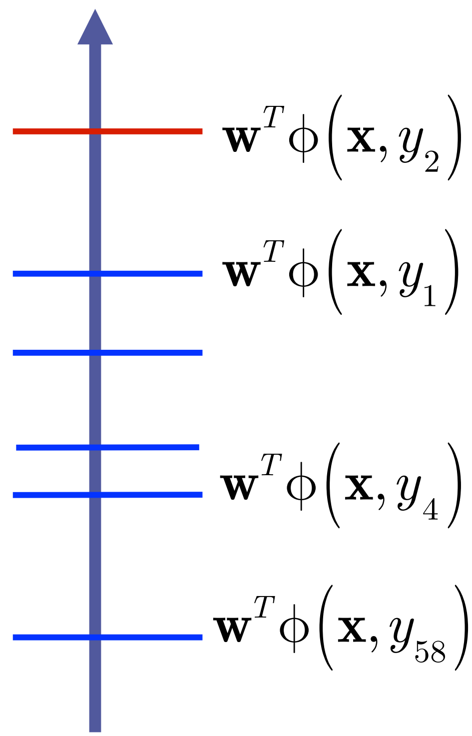


Joint Feature Map

- Instead of solving for K different w's, make 1 long w
- Replace each x with $\phi(\mathbf{x}, y = i) = \left[0^T\, 0^T \ldots 0^T\, \mathbf{x}^T\, 0^T \ldots 0^T\ \right]^T$
- Put the x vector in the i'th position
- The feature vector is DK dimensional

$$y_i \in \{1, \ldots, k\}$$

$$\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \rightarrow f(\mathbf{x}) = \arg\max_{y \in Y} \mathbf{w}^T \phi(\mathbf{x}, y)$$



$$\min_{\mathbf{w}, \xi \geq 0} \|\mathbf{w}\|^2$$

$$s.t. \quad \forall y \in Y \setminus y_1 : \mathbf{w}^T \phi(\mathbf{x}_1, y_1) \geq \mathbf{w}^T \phi(\mathbf{x}_1, y) + 1$$

$$s.t. \quad \ldots$$

$$s.t. \quad \forall y \in Y \setminus y_n : \mathbf{w}^T \phi(\mathbf{x}_n, y_n) \geq \mathbf{w}^T \phi(\mathbf{x}_n, y) + 1$$

Joint Feature Map

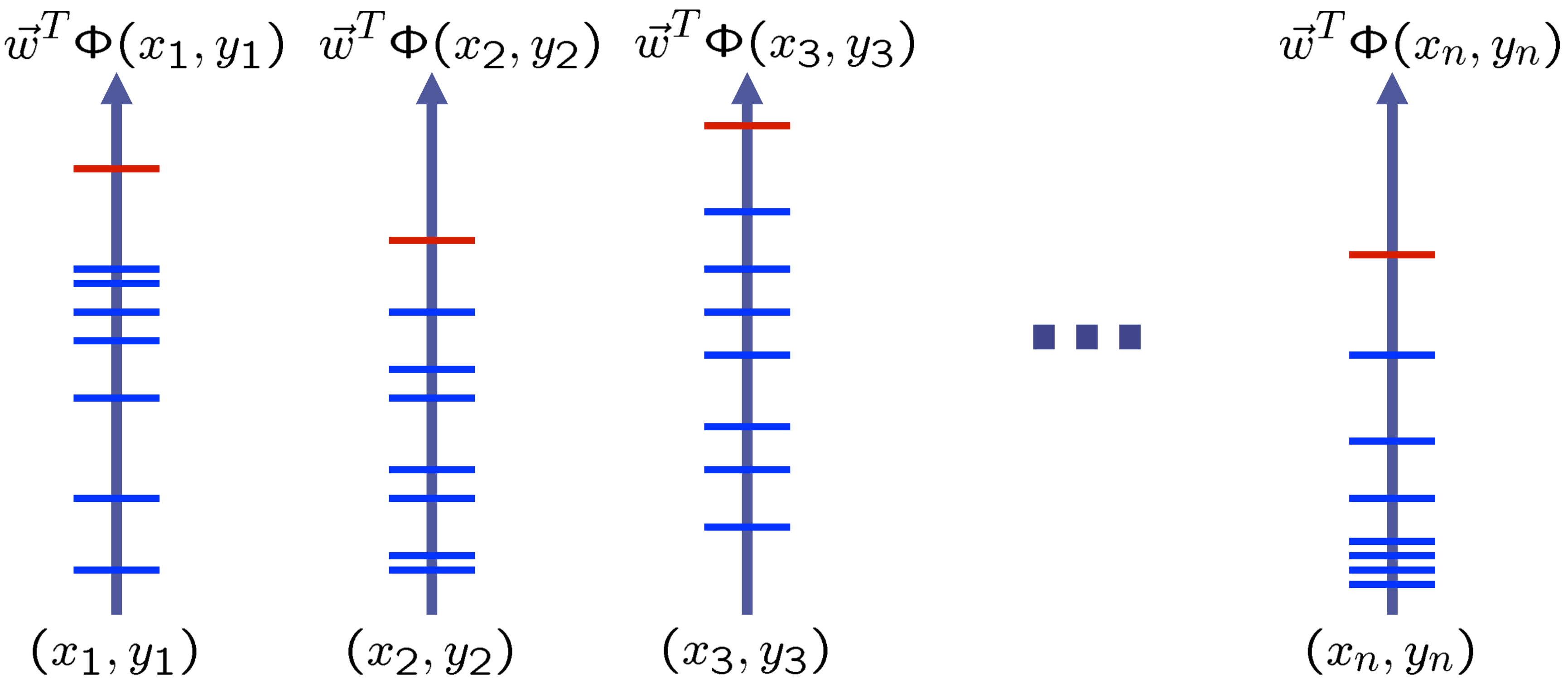
- Learn weight vector so that $\mathbf{w}^T\phi\left(\mathbf{x}_i, y\right)$ is max for correct y

$$\min_{\mathbf{w},\xi\geq 0} \left\|\mathbf{w}\right\|^2$$

$$s.t. \quad \forall y \in Y \setminus y_1 : \mathbf{w}^T\phi\left(\mathbf{x}_1, y_1\right) \geq \mathbf{w}^T\phi\left(\mathbf{x}_1, y\right) + 1$$

$$s.t. \quad \ldots$$

$$s.t. \quad \forall y \in Y \setminus y_n : \mathbf{w}^T\phi\left(\mathbf{x}_n, y_n\right) \geq \mathbf{w}^T\phi\left(\mathbf{x}_n, y\right) + 1$$



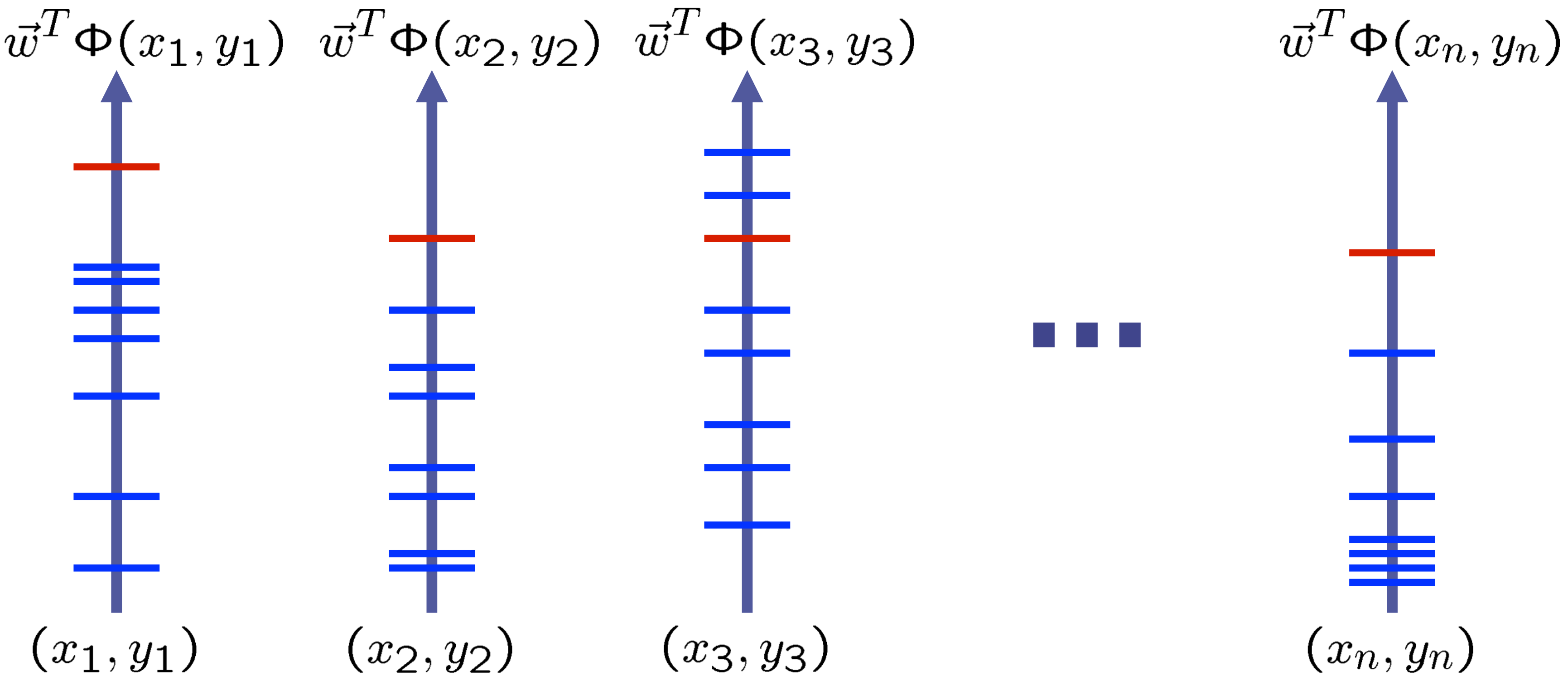
Joint Feature Map with Slack

$$\min_{\mathbf{w},\xi\geq 0} \tfrac{1}{2}\left\|\mathbf{w}\right\|^2 + \tfrac{C}{n}\sum_{i=1}^{n}\xi_i$$

$$s.t.\quad \forall y \in Y \setminus y_1 : \mathbf{w}^T\phi\left(\mathbf{x}_1, y_1\right) \geq \mathbf{w}^T\phi\left(\mathbf{x}_1, y\right) + 1 - \xi_1$$

$$s.t.\quad \ldots$$

$$s.t.\quad \forall y \in Y \setminus y_n : \mathbf{w}^T\phi\left(\mathbf{x}_n, y_n\right) \geq \mathbf{w}^T\phi\left(\mathbf{x}_n, y\right) + 1 - \xi_n$$



The label loss function

- Not all classes are created equal, why clear each by 1?

$$\min_{\mathbf{w},\xi\geq 0} \tfrac{1}{2}\left\|\mathbf{w}\right\|^2 + \tfrac{C}{n}\sum\nolimits_{i=1}^{n}\xi_i$$

$$s.t.\quad \forall y \in Y \setminus y_1 : \mathbf{w}^T\phi\left(\mathbf{x}_1, y_1\right) \geq \mathbf{w}^T\phi\left(\mathbf{x}_1, y\right) + 1 - \xi_1 \qquad \Delta\left(y, y_1\right)$$

$$s.t.\quad \ldots$$

$$s.t.\quad \forall y \in Y \setminus y_n : \mathbf{w}^T\phi\left(\mathbf{x}_n, y_n\right) \geq \mathbf{w}^T\phi\left(\mathbf{x}_n, y\right) + 1 - \xi_n$$

- Instead of a constant 1 value, clear some classes more

$$\Delta\left(y, y_1\right) = Loss\ for\ predicting\ y\ instead\ of\ y_1$$

- For example, if y can be {lion, tiger, cat}

$$\Delta\left(tiger, lion\right) = \Delta\left(lion, tiger\right) = 1$$

$$\Delta\left(cat, lion\right) = \Delta\left(lion, cat\right) = 999$$

$$\Delta\left(tiger, tiger\right) = \Delta\left(cat, cat\right) = \Delta\left(lion, lion\right) = 0$$

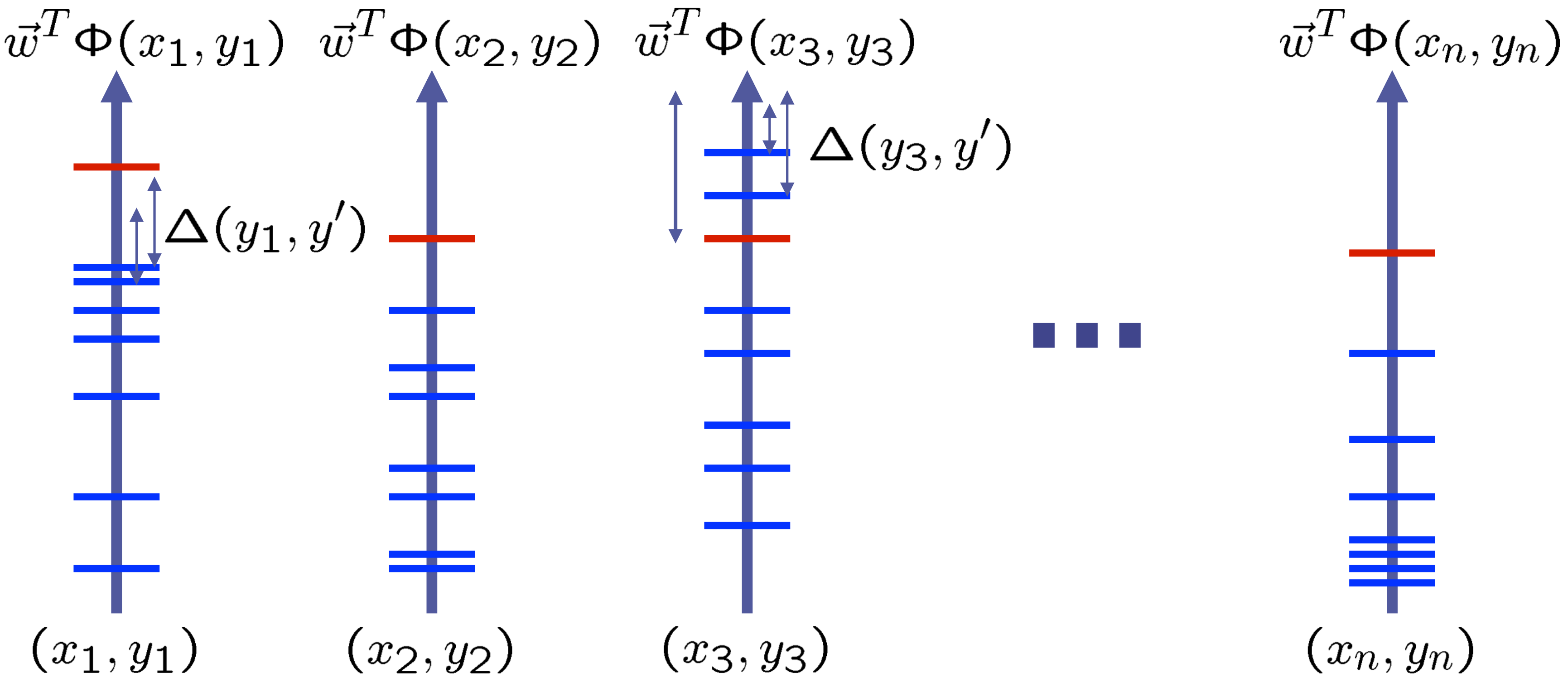
Joint Feature Map with Any Loss

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

$$s.t.\quad \forall y \in Y \setminus y_1 : \mathbf{w}^T\phi\left(\mathbf{x}_1, y_1\right) \geq \mathbf{w}^T\phi\left(\mathbf{x}_1, y\right) + \Delta\left(y, y_1\right) - \xi_1$$

$$s.t.\quad \ldots$$

$$s.t.\quad \forall y \in Y \setminus y_n : \mathbf{w}^T\phi\left(\mathbf{x}_n, y_n\right) \geq \mathbf{w}^T\phi\left(\mathbf{x}_n, y\right) + \Delta\left(y, y_n\right) - \xi_n$$



Joint Feature Map with Slack

- Loss function Δ measures match between target & prediction

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

$$s.t.\quad \forall y \in Y \setminus y_1 : \mathbf{w}^T\phi\left(\mathbf{x}_1, y_1\right) \geq \mathbf{w}^T\phi\left(\mathbf{x}_1, y\right) + \Delta\left(y, y_1\right) - \xi_1$$

$$s.t.\quad \ldots$$

$$s.t.\quad \forall y \in Y \setminus y_n : \mathbf{w}^T\phi\left(\mathbf{x}_n, y_n\right) \geq \mathbf{w}^T\phi\left(\mathbf{x}_n, y\right) + \Delta\left(y, y_n\right) - \xi_n$$

Lemma: The training loss is upper bounded by

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

Generic Structural SVM (slow!)

- Application Specific Design of Model
  - Loss function $\Delta(y_i, y)$
  - Representation $\Phi(x, y)$
    - ➔ Markov Random Fields [Lafferty et al. 01, Taskar et al. 04]
- Prediction:

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

- Training:

$$\begin{aligned}
\min_{\vec{w},\vec{\xi}\geq 0} \quad & \frac{1}{2}\vec{w}^T\vec{w} + \frac{C}{n}\sum_{i=1}^{n}\xi_i \\
s.t. \quad & \forall y \in Y \backslash y_1 : \vec{w}^T\Phi(x_1, y_1) \geq \vec{w}^T\Phi(x_1, y) + \Delta(y_1,y) - \xi_1 \\
& \ldots \\
& \forall y \in Y \backslash y_n : \vec{w}^T\Phi(x_n, y_n) \geq \vec{w}^T\Phi(x_n, y) + \Delta(y_n,y) - \xi_n
\end{aligned}$$

- Applications: Parsing, Sequence Alignment, Clustering, etc.

Reformulating the QP

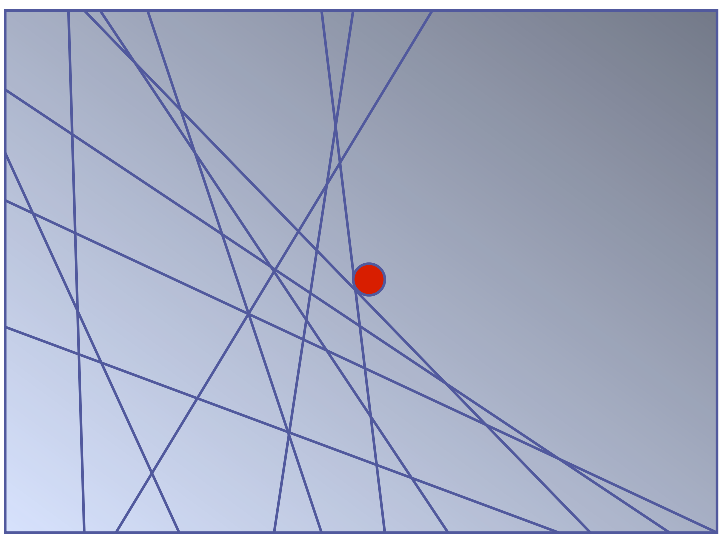
n-Slack Formulation: [TsoJoHoAl04]

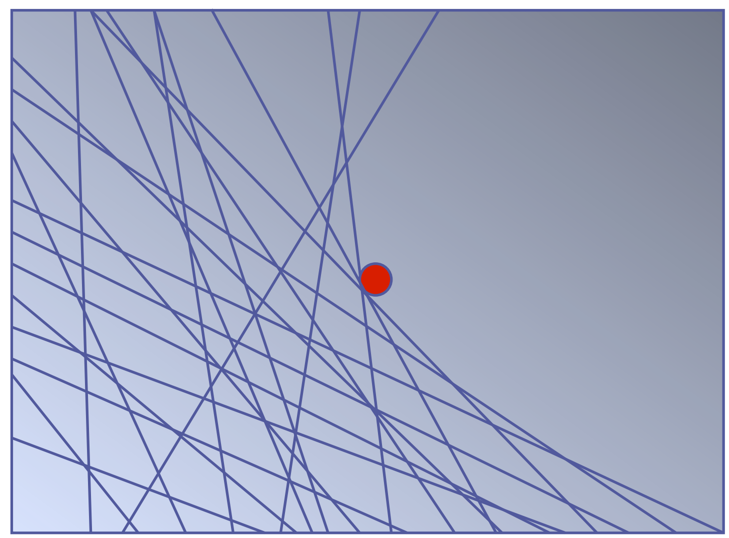
$$\min_{\vec{w},\vec{\xi}} \quad \frac{1}{2}\vec{w}^T\vec{w} + \frac{C}{n}\sum_{i=1}^{n}\xi_i$$

$$s.t. \quad \forall y' \in Y : \vec{w}^T\Phi(x_1,y_1) - \vec{w}^T\Phi(x_1,y') \geq \Delta(y_1,y) - \xi_1$$

$$\ldots$$

$$\forall y' \in Y : \vec{w}^T\Phi(x_n,y_n) - \vec{w}^T\Phi(x_n,y') \geq \Delta(y_n,y) - \xi_n$$





Reformulating the QP

n-Slack Formulation: [TsoJoHoAl04]

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

$$s.t. \quad \forall y' \in Y : \vec{w}^T\Phi(x_1,y_1) - \vec{w}^T\Phi(x_1,y') \geq \Delta(y_1,y) - \xi_1$$

$$\ldots$$

$$\forall y' \in Y : \vec{w}^T\Phi(x_n,y_n) - \vec{w}^T\Phi(x_n,y') \geq \Delta(y_n,y) - \xi_n$$

$$\Leftrightarrow$$

1-Slack Formulation: [JoFinYu08]

$$\min_{\vec{w},\xi} \quad \frac{1}{2}\vec{w}^T\vec{w} + C\xi$$

$$s.t. \quad \forall y_1'...y_n' \in Y : \frac{1}{n}\sum_{i=1}^{n}\left[\vec{w}^T\Phi(x_i,y_i) - \vec{w}^T\Phi(x_i,y_i')\right] \geq \frac{1}{n}\sum_{i=1}^{n}\left[\Delta(y_i,y_i')\right] - \xi$$

Comparing n-Slack & 1-Slack

- Example: $Y = \left\{A, B, C\right\} \; and \;\; y_1 = A, y_2 = A, y_3 = B, y_4 = C$

n-Slack→n(k-1) constraints

$$y_1 \geq B, y_1 \geq C$$
$$y_2 \geq B, y_2 \geq C$$
$$y_3 \geq A, y_3 \geq C$$
$$y_4 \geq A, y_4 \geq B$$

1-Slack→k^n constraints

$$\begin{aligned} y_1 y_2 y_3 y_4 \geq\ & AAAA, AAAB, AAAC, AABA, \\ & AABB, \cdot, AACA, AACB, AACC, \\ & ABAA, ABAB, ABAC, ABBA, \\ & ABBB, ABBC, ABCA, ABCB, \\ & ABCC, ACAA, ACAB, ACAC, \ldots \end{aligned}$$

- Idea: we expect only a few constraints to be active
- Cutting-Plane: a greedy approach to QP
- Solve with only a few constraints at a time
- If solution violates come constraints, add them back in
- If we are smart about which ones to add, may not need k^n

1-Slack Cutting-Plane Algorithm

- Input: $(x_1, y_1), \ldots, (x_n, y_n), C, \epsilon$
- $S \leftarrow \emptyset, \vec{w} \leftarrow 0, \xi \leftarrow 0$
- REPEAT
  - FOR $i = 1, \ldots, n$
    - Compute $y_i' = argmax_{y \in Y} \{\Delta(y_i, y) + \vec{w}^T \Phi(x_i, y)\}$
  - ENDFOR
  - IF $\frac{1}{n}\sum_{i=1}^{n}\left[\Delta(y_i, y_i') - \vec{w}^T[\Phi(x_i, y_i) - \Phi(x_i, y_i')]\right] > \xi + \epsilon$

    $- S \leftarrow S \cup \{\vec{w}^T \frac{1}{n}\sum_{i=1}^{n}[\Phi(x_i, y_i) - \Phi(x_i, y_i')] \geq \frac{1}{n}\sum_{i=1}^{n}\Delta(y_i, y_i') - \xi\}$
    - optimize StructSVM over S to get w and ξ
  - ENDIF
- UNTIL solution has not changed during iteration [Jo06] [JoFinYu08]

Polynomial Sparsity 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

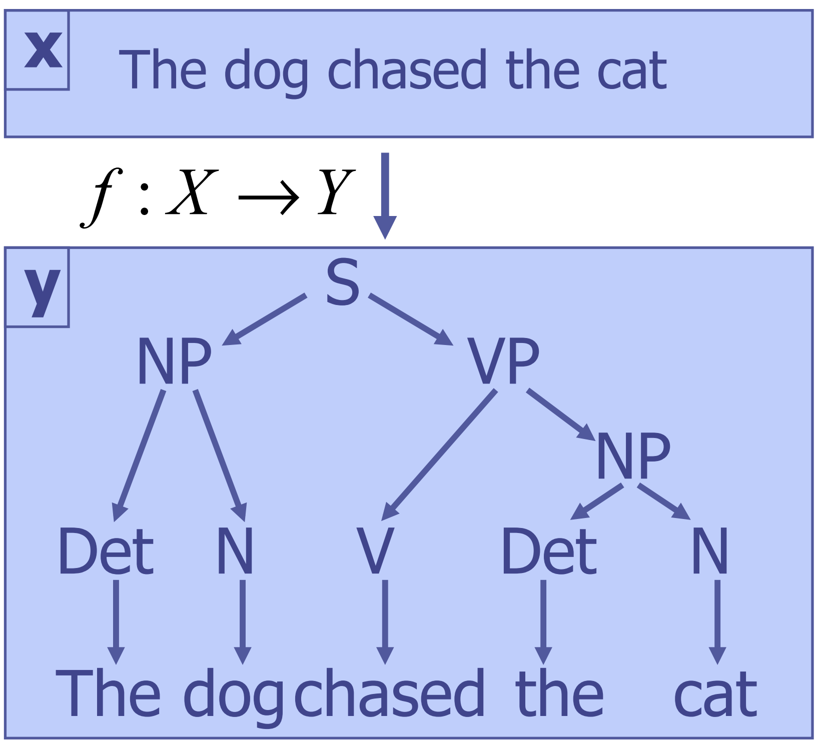
$$\left\lceil \log_2\left(\frac{\Delta}{4R^2C}\right)\right\rceil + \left\lceil \frac{16R^2C}{\varepsilon} \right\rceil$$

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

[Jo03] [Jo06] [TeoLeSmVi07] [JoFinYu08]

Joint Feature Map for Trees

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



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

Experiments: NLP

Implementation

- 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

Method	Test Accuracy Acc	Test Accuracy F_1
PCFG with MLE	55.2	86.0
SVM with (1-F_1)-Loss	**58.9**	**88.5**

[TsoJoHoAl04]

- more complex features [TaKlCoKoMa04]

Experiments: 1-slack vs. n-slack

Part-of-speech tagging on Penn Treebank

~36,000 examples, ~250,000 features in linear HMM model

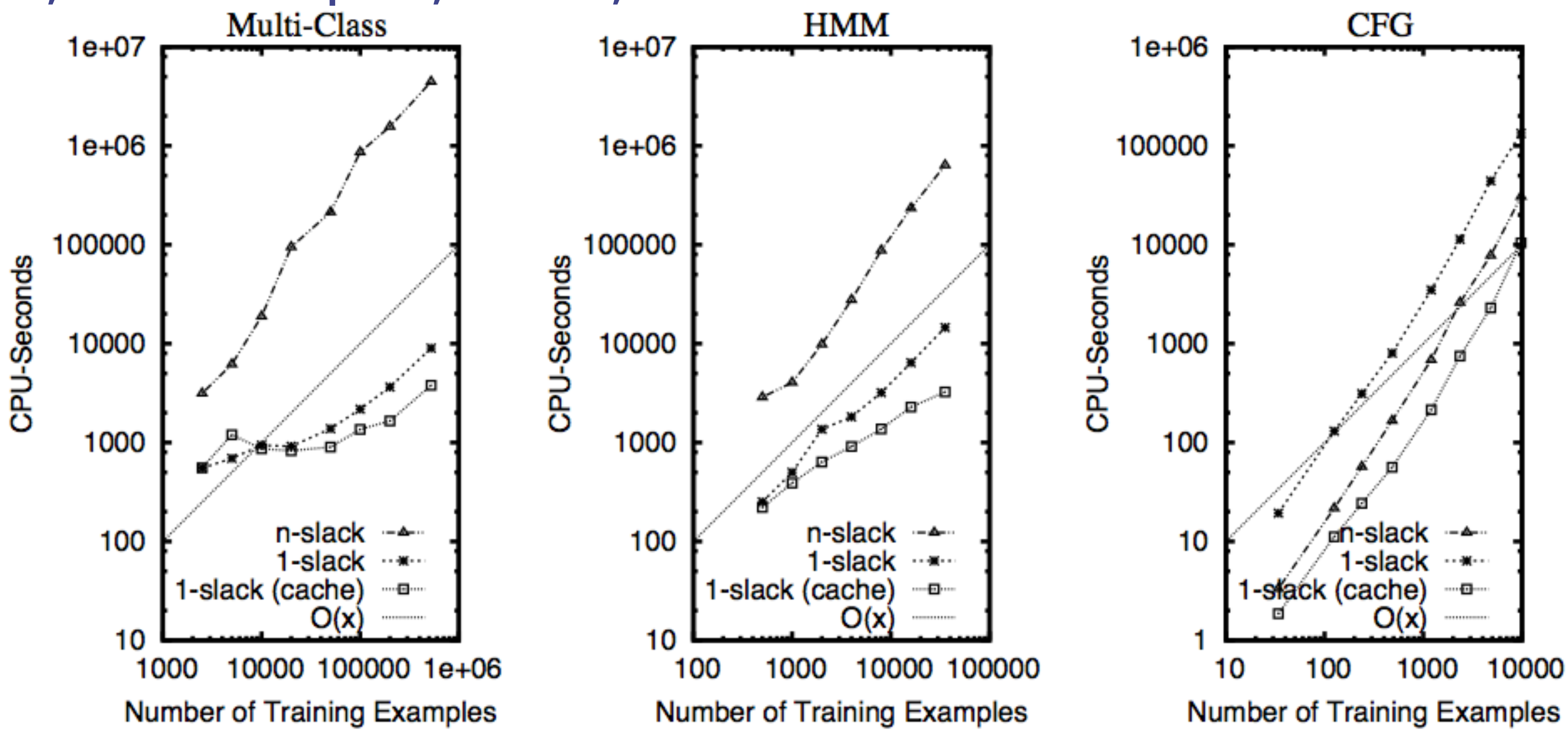


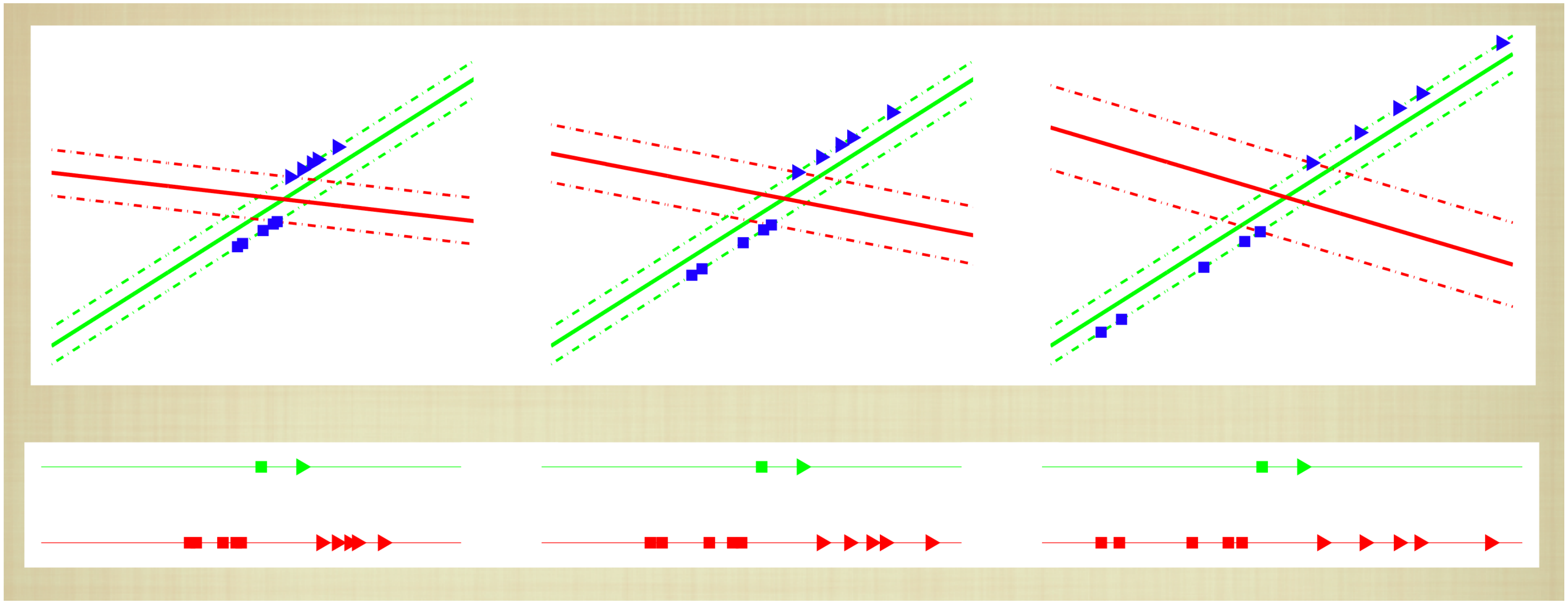
Fig. 1 Training times for multi-class classification (left) HMM part-of-speech tagging (middle) and CFG parsing (right) as a function of n for the n-slack algorithm, the 1-slack algorithm, and the 1-slack algorithm with caching.

StructSVM for Any Problem

- General
  - SVM-struct algorithm and implementation
    http://svmlight.joachims.org
  - Theory (e.g. training-time linear in n)
- Application specific
  - Loss function $\Delta(y_i, y)$
  - Representation $\Phi(x, y)$
  - Algorithms to compute
    $$\hat{y} = argmax_{y \in Y} \{\vec{w}^T \Phi(x_i, y)\}$$
    $$\hat{y} = argmax_{y \in Y} \{\Delta(y_i, y) + \vec{w}^T \Phi(x_i, y)\}$$
- Properties
  - General framework for discriminative learning
  - Direct modeling, not reduction to classification/regression
  - "Plug-and-play"

Maximum Relative Margin

- Details in Shivaswamy and Jebara in NIPS 2008



- Red is maximum margin, Green is max relative margin
- Top is a two d classification problem
- Bottom is projection of data on solution $w^T x+b$
- SVM solution changes as axes get scaled, has large spread

Maximum Relative Margin

- Fast trick to solve the *same* problem as on previous slides:
  Bound the spread of the SVM!
- Recall original SVM primal problem (with slack):

$$\min_{w,b,\xi} \tfrac{1}{2}\left\|w\right\|^2 + C\sum_i \xi_i \quad \textit{subject to} \quad y_i\left(w^T x_i + b\right) \geq 1 - \xi_i$$

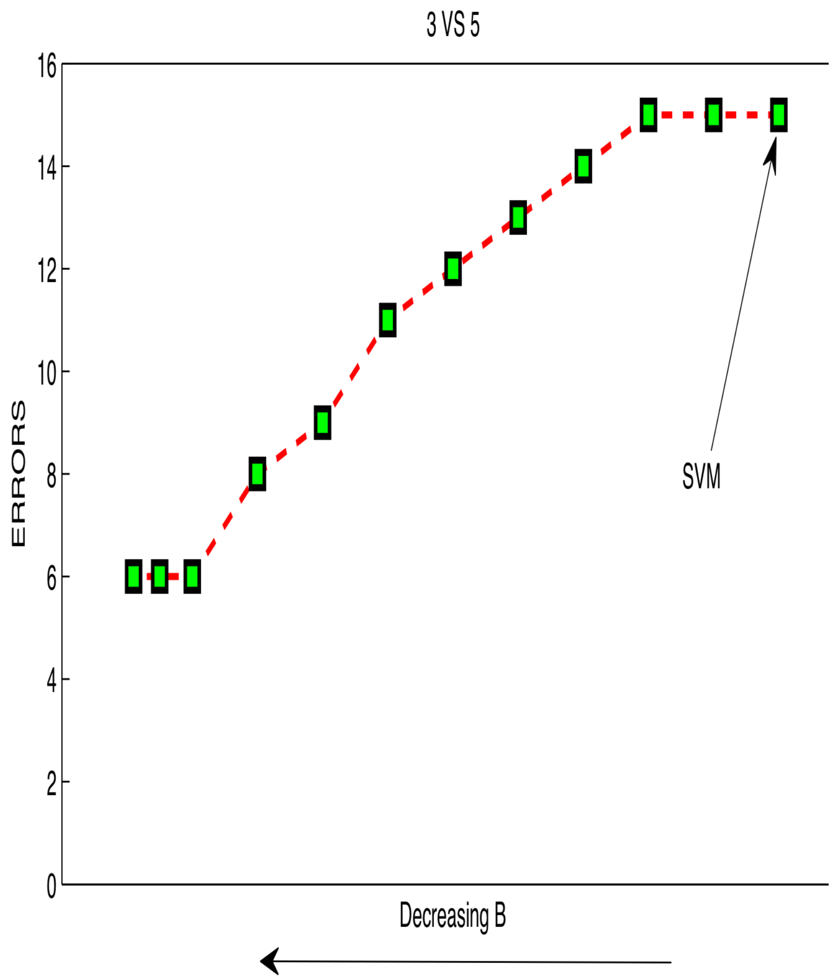
- Add the following constraints: $-B \leq w^T x_i + b \leq B$

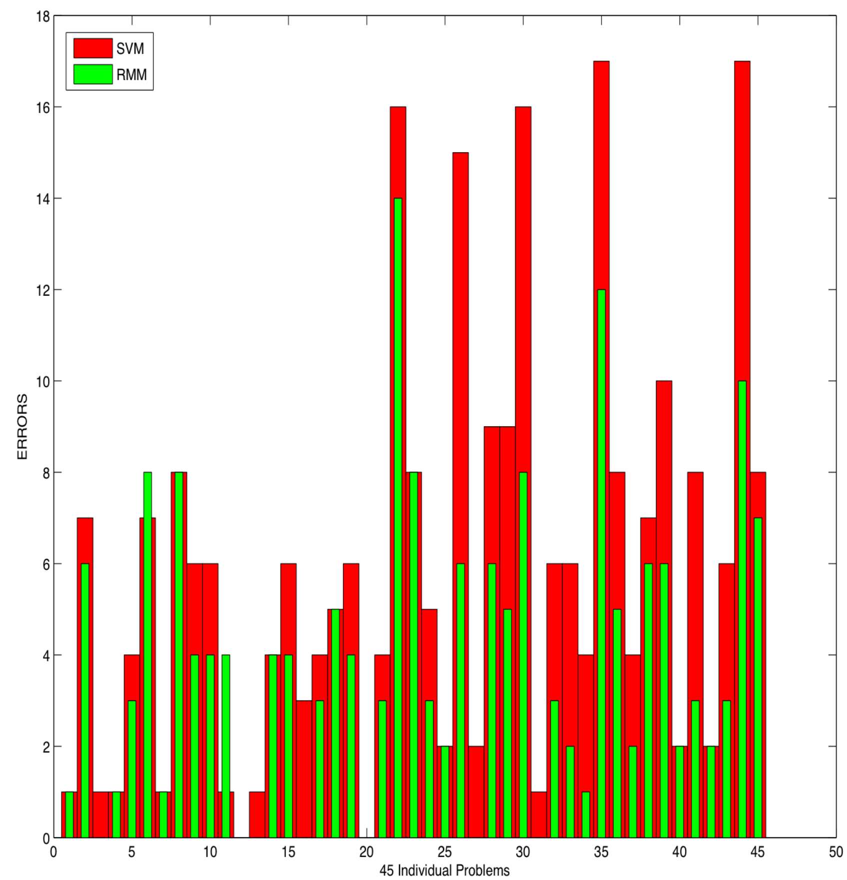
- This bounds the spread. Call it Relative Margin Machine.
- Above is still a QP, scales to 100k examples
- Can also be kernelized, solved in the dual, etc.
- Unlike previous SDP which only runs on ~1k examples

- RMM as fast as SVM but much higher accuracy…

Maximum Relative Margin

- RMM vs. SVM on digit classification (two-class 0,…,9)





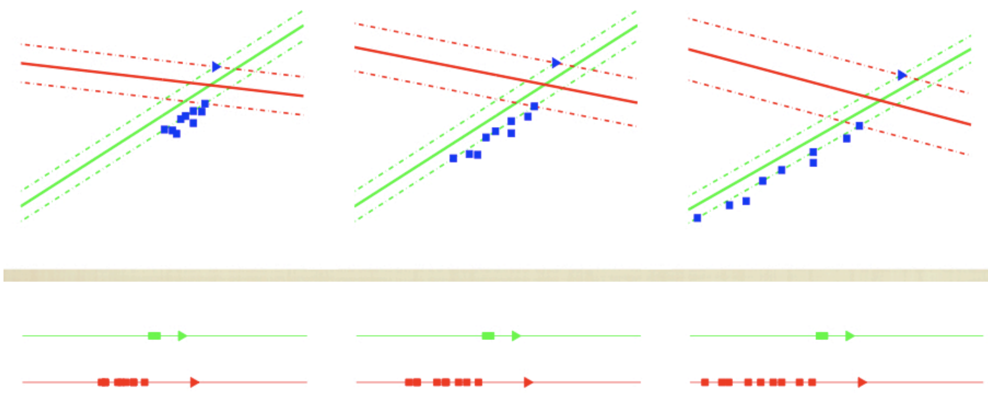
Maximum Relative Margin

- RMM vs. SVM on digit classification (two-class 0,…,9)
- Cross-validate to obtain best B and C fro SVM and RMM
- Compare also to Kernel Linear Discriminant Analysis
- Try different polynomial kernels and RBF
- RMM has consistently lower error for kernel classification

		1	2	3	4	5	6	7	RBF
OPT	SVM	71	57	54	47	40	46	46	51
	Σ-SVM	**61**	48	41	36	35	31	**29**	47
	KLDA	71	57	54	47	40	46	46	**45**
	RMM	71	**36**	**32**	**31**	**33**	**30**	**29**	51
USPS	SVM	145	109	109	103	100	95	93	104
	Σ-SVM	**132**	**108**	99	94	**89**	**87**	**90**	**97**
	KLDA	132	119	121	117	114	118	117	101
	RMM	153	109	**94**	**91**	91	90	**90**	98
Full MNIST	SVM	536	198	170	156	157	141	136	146
	RMM	**521**	**146**	**140**	**130**	**119**	**116**	**115**	**129**

Struct SVM with Relative Margin

- Add relative margin constraints to struct SVM (ShiJeb09)
- Correct beats wrong labels but not by too much (relatively)



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

$$s.t. \quad \forall y \in Y \setminus y_1 : B \geq \mathbf{w}^T\phi\left(\mathbf{x}_1, y_1\right) - \mathbf{w}^T\phi\left(\mathbf{x}_1, y\right) \geq \Delta\left(y, y_1\right) - \xi_1$$

$$s.t. \quad \ldots$$

$$s.t. \quad \forall y \in Y \setminus y_n : B \geq \mathbf{w}^T\phi\left(\mathbf{x}_n, y_n\right) - \mathbf{w}^T\phi\left(\mathbf{x}_n, y\right) \geq \Delta\left(y, y_n\right) - \xi_n$$

- Needs both $\arg\max_{y\in Y} \mathbf{w}^T\phi\left(\mathbf{x}, y\right)$ and $\arg\min_{y\in Y} \mathbf{w}^T\phi\left(\mathbf{x}, y\right)$

Struct SVM with Relative Margin

- Similar bound holds for relative margin
- Maximum # of cuts is

$$\max\left\{\frac{2CR^2}{\varepsilon_B^2}, \frac{2n}{\varepsilon}, \frac{8CR^2}{\varepsilon^2}\right\}$$

- Try sequence learning problems for Hidden Markov Modeling
- Consider named entity recognition (NER) task
- Consider part-of-speech (POS) task

	NER	POS
CRF	5.13 ± 0.28	11.34 ± 0.64
StructSVM	5.09 ± 0.32	11.14 ± 0.60
StructRMM	$\mathbf{5.05 \pm 0.28}$	$\mathbf{10.42 \pm 0.47}$
p-value	0.07	0.00