Introduction to Support Vector Machines

BTR Workshop
Fall 2006

Thorsten Joachims
Cornell University

Outline

- **Statistical Machine Learning Basics**
  - Training error, generalization error, hypothesis space
- **Support Vector Machines for Classification**
  - Optimal hyperplanes and margins
  - Soft-margin Support Vector Machine
  - Primal vs. dual optimization problem
  - Kernels
- **Support Vector Machines for Structured Outputs**
  - Linear discriminant models
  - Solving exponentially-size training problems
  - Example: Predicting the alignment between proteins

Supervised Learning

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

$$h : X \longrightarrow Y$$

such that the prediction error is low.

x

Microsoft announced today that they acquired Apple for the amount equal to the gross national product of Switzerland. Microsoft officials stated that they first wanted to buy Switzerland, but eventually were turned off by the mountains and the snowy winters…

$\longrightarrow$

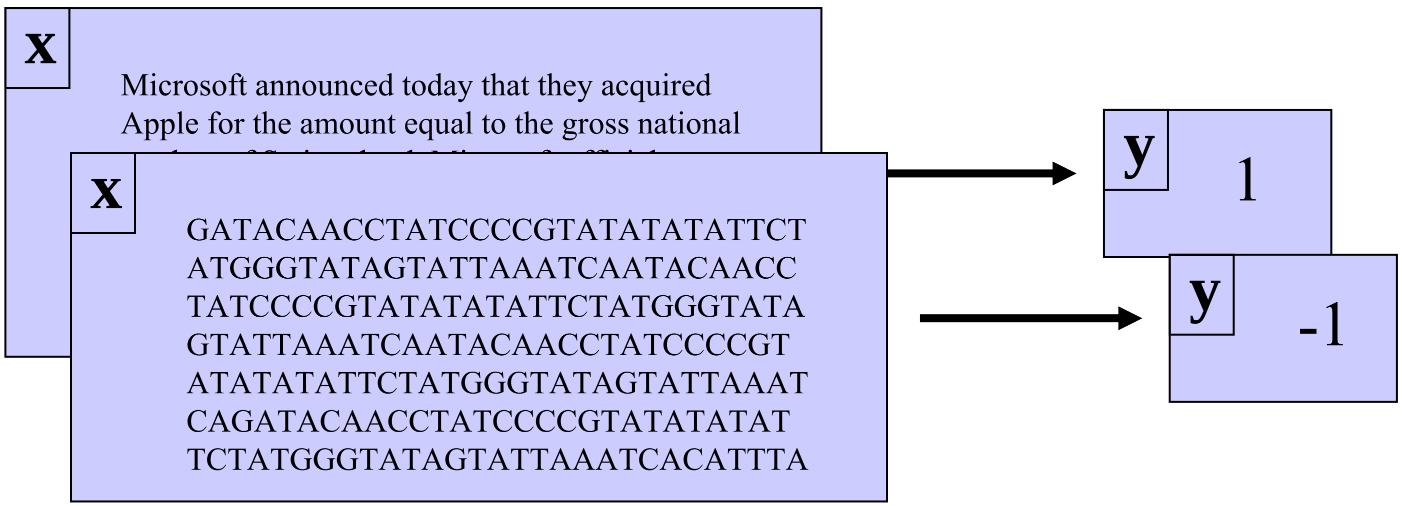
y 1

Supervised Learning

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

$$h : X \longrightarrow Y$$

such that the prediction error is low.

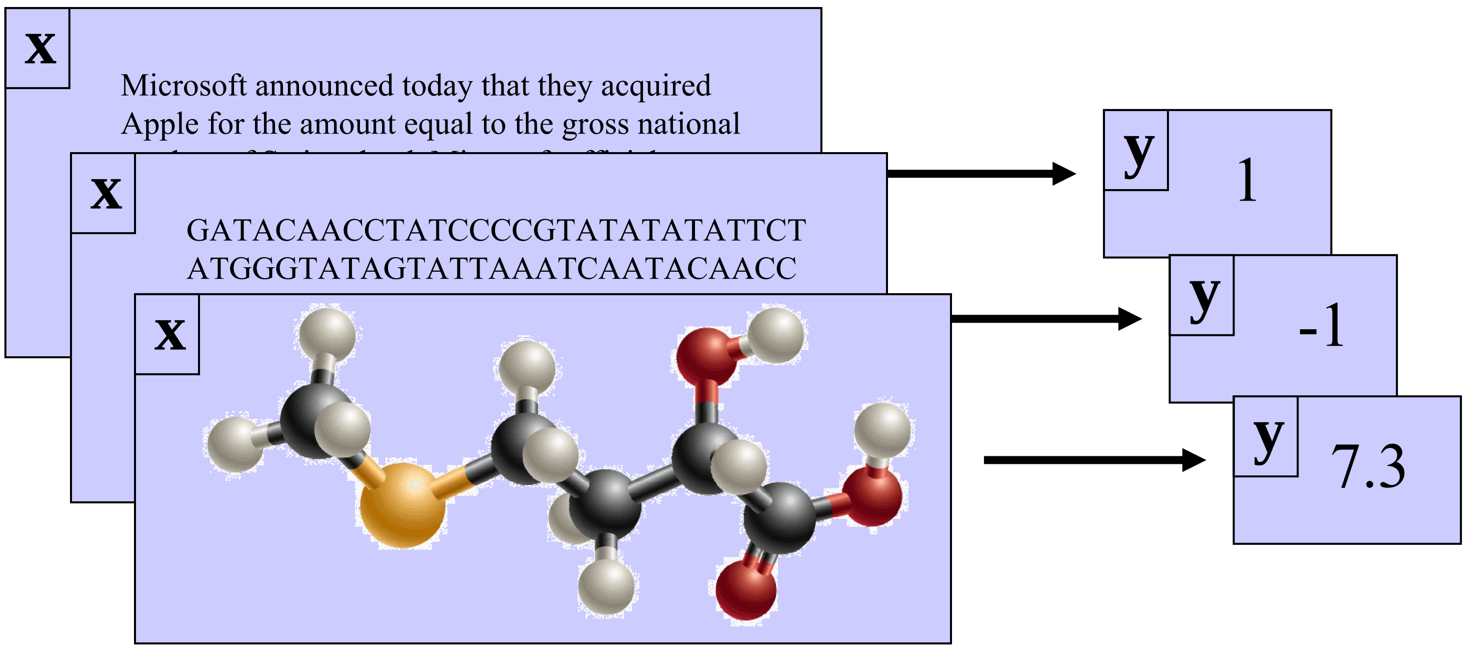


Supervised Learning

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

$$h : X \longrightarrow Y$$

such that the prediction error is low.

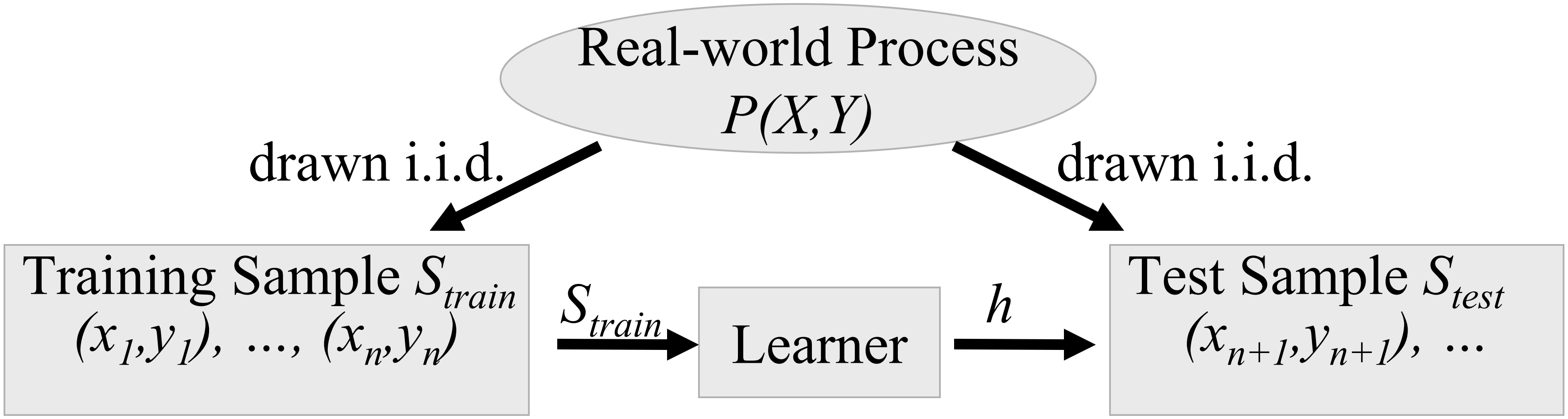


Example: Spam Filtering

	viagra	learning	the	dating	nigeria	$spam?$
$\vec{x}_1 = ($	1	0	1	0	0)	$y_1 = 1$
$\vec{x}_2 = ($	0	1	1	0	0)	$y_2 = -1$
$\vec{x}_3 = ($	0	0	0	0	1)	$y_3 = 1$

- **Instance Space X:**
  - Feature vector of word occurrences => binary features
  - N features (N typically > 50000)
- **Target Concept c:**
  - Spam (+1) / Ham (-1)

Learning as Prediction Task



- **Goal:** Find h with small prediction error $Err_P(h)$ over $P(X,Y)$.
- **Strategy:** Find (any?) h with small error $Err_{S_{train}}(h)$ on training sample S_{train}.

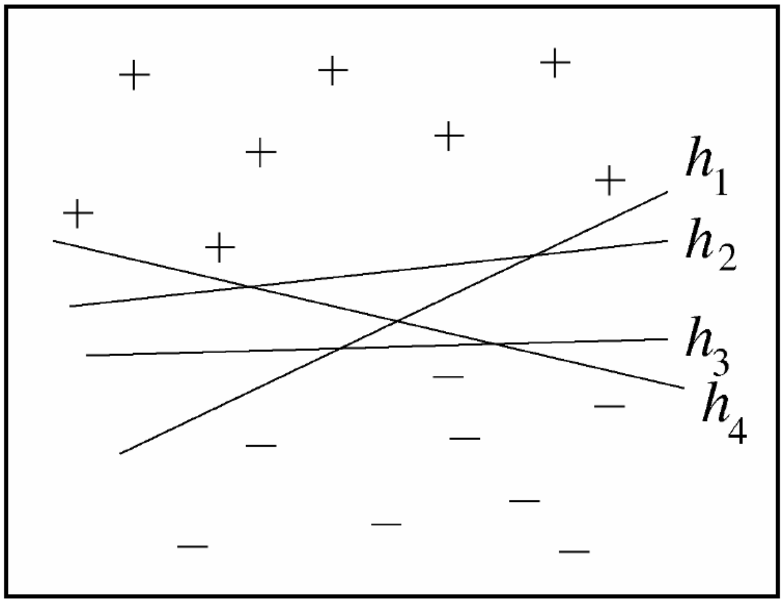
- **Training Error:** Error $Err_{S_{train}}(h)$ on training sample.
- **Test Error:** Error $Err_{S_{test}}(h)$ on test sample is an estimate of $Err_P(h)$.

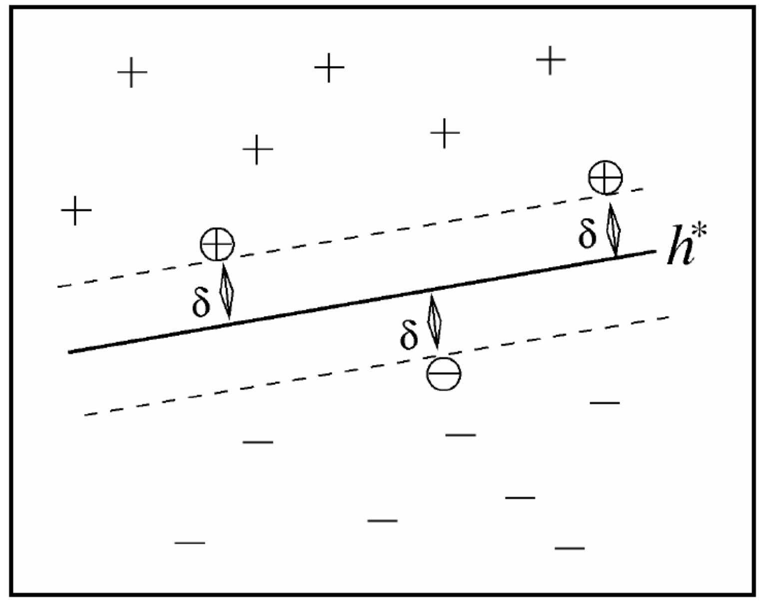
Linear Classification Rules

- **Hypotheses of the form**
  - unbiased: $h_{\vec{w}}(\vec{x}) = \begin{cases} 1 & w_1x_1 + ... + w_Nx_N > 0 \\ -1 & else \end{cases}$
  - biased: $h_{\vec{w},b}(\vec{x}) = \begin{cases} 1 & w_1x_1 + ... + w_Nx_N + b > 0 \\ -1 & else \end{cases}$
  - Parameter vector *w*, scalar *b*
- **Hypothesis space H**
  - $H_{unbiased} = \{h_{\vec{w}} : \vec{w} \in \Re^N\}$
  - $H_{biased} = \{h_{\vec{w},b} : \vec{w} \in \Re^N\ b \in \Re\}$
- **Notation**
  - $w_1x_1 + ... + w_Nx_N = \vec{w} \cdot \vec{x}$ and $sign(a) = \begin{cases} 1 & a > 0 \\ -1 & else \end{cases}$
  - $h_{\vec{w}}(\vec{x}) = sign(\vec{w} \cdot \vec{x})$
  - $h_{\vec{w},b}(\vec{x}) = sign(\vec{w} \cdot \vec{x} + b)$

Optimal Hyperplanes
Linear Hard-Margin Support Vector Machine

Assumption: Training examples are linearly separable.





Margin of a Linear Classifier

Definition: *For a linear classifier h_w, the* **margin** δ *of an example* $(\vec{x}, y)$ *is* $\delta = y(\vec{w} \cdot \vec{x})$.

Definition: *The margin is called* geometric margin, *if* $||\vec{w}|| = 1$. *Otherwise,* functional margin.

Definition: *The (hard) margin of an unbiased linear classifier* $h_{\vec{w}}$ *on a sample* S *is* $\delta = min_{(\vec{x},y)\in S} y(\vec{w} \cdot \vec{x})$.

Definition: *The (hard) margin of an unbiased linear classifier* $h_{\vec{w}}$ *on a task* $P(X, Y)$ *is*

$$\delta = inf_{S\sim P(X,Y)} min_{(\vec{x},y)\in S} y(\vec{w} \cdot \vec{x}).$$

Hard-Margin Separation

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

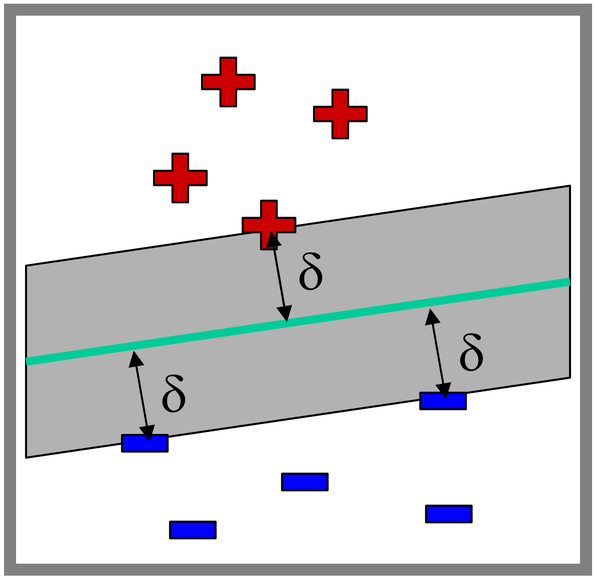
Optimization Problem (Primal):

$$\min_{\vec{w},b} \quad \frac{1}{2}\vec{w}\cdot\vec{w}$$

$$s.t. \quad y_1(\vec{w}\cdot\vec{x}_1 + b) \geq 1$$

$$\ldots$$

$$y_n(\vec{w}\cdot\vec{x}_n + b) \geq 1$$

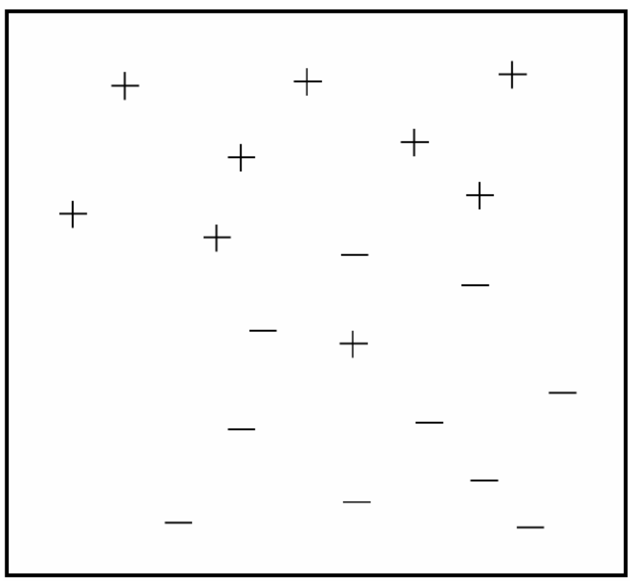


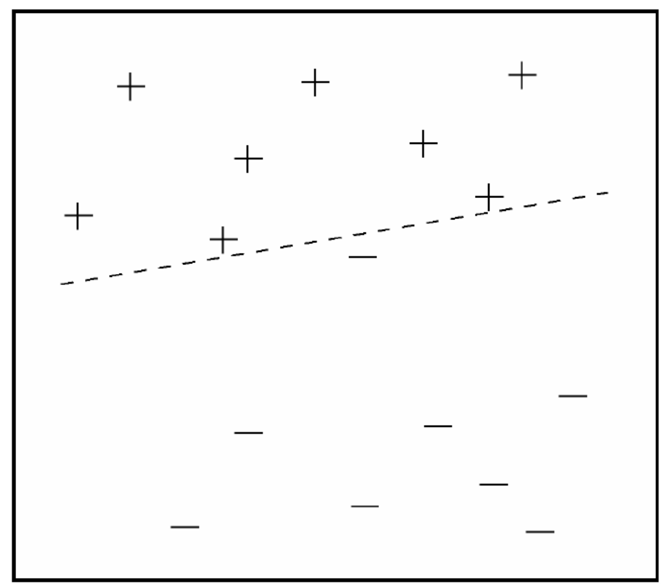
Support Vectors: Examples with minimal distance (i.e. margin).

Non-Separable Training Data

Limitations of hard-margin formulation

- For some training data, there is no separating hyperplane.
- Complete separation (i.e. zero training error) can lead to suboptimal prediction error.





Soft-Margin Separation

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):

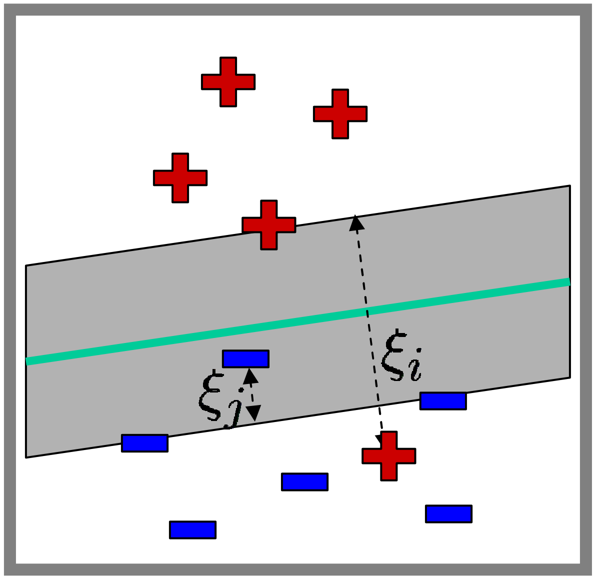
$$\min_{\vec{w},\vec{\xi},b} \frac{1}{2}\vec{w}\cdot\vec{w} + C\sum_{i=1}^{n}\xi_i$$

$$s.t. \; y_1(\vec{w}\cdot\vec{x}_1 + b) \geq 1 - \xi_1 \wedge \xi_1 \geq 0$$

$$\ldots$$

$$y_n(\vec{w}\cdot\vec{x}_n + b) \geq 1 - \xi_n \wedge \xi_n \geq 0$$

- 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.



Soft-Margin Separation

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):

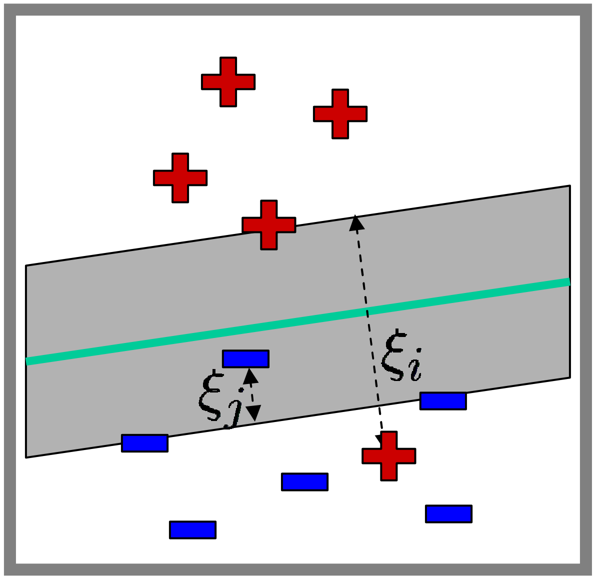
$$\min_{\vec{w},\vec{\xi},b} \quad \frac{1}{2}\vec{w}\cdot\vec{w} + C\sum_{i=1}^{n}\xi_i$$

$$s.t. \quad y_1(\vec{w}\cdot\vec{x}_1+b) \geq 1-\xi_1 \wedge \xi_1 \geq 0$$

$$\ldots$$

$$y_n(\vec{w}\cdot\vec{x}_n+b) \geq 1-\xi_n \wedge \xi_n \geq 0$$

- 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.



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.

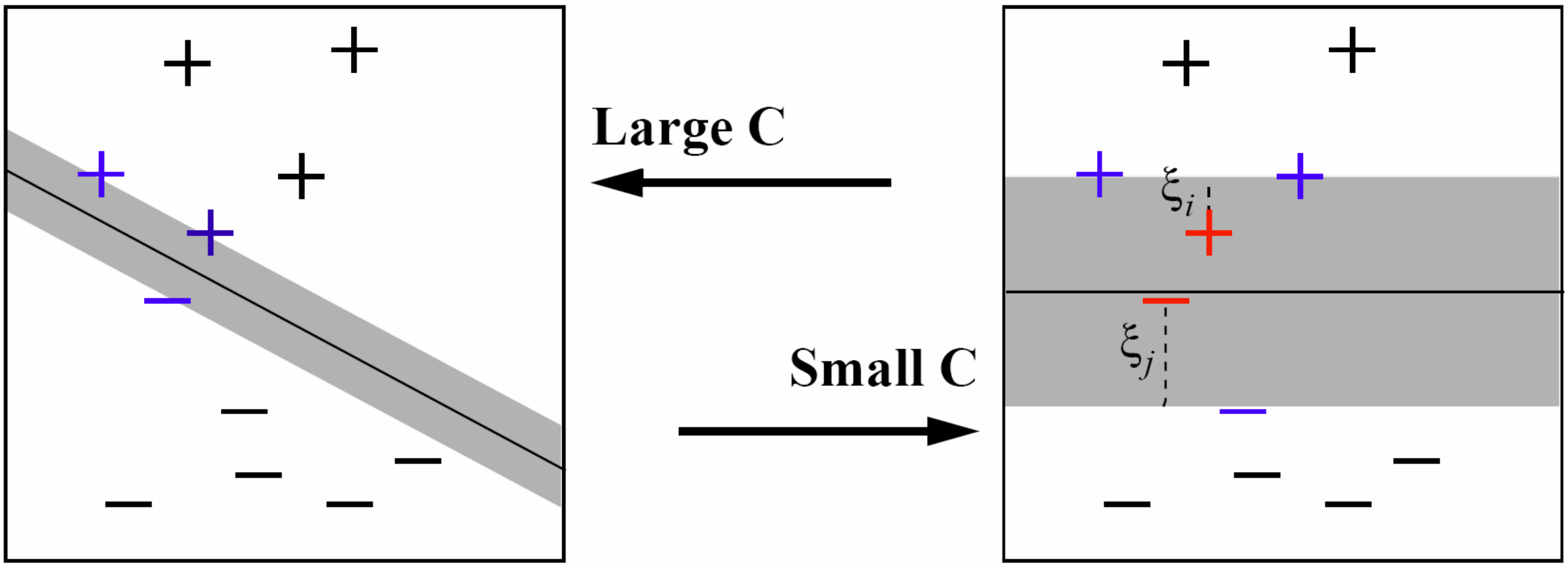
Soft-Margin OP (Primal):

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

$$s.t.\ y_1(\vec{w}\cdot\vec{x}_1 + b) \geq 1 - \xi_1 \wedge \xi_1 \geq 0$$

$$\ldots$$

$$y_n(\vec{w}\cdot\vec{x}_n + b) \geq 1 - \xi_n \wedge \xi_n \geq 0$$



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.

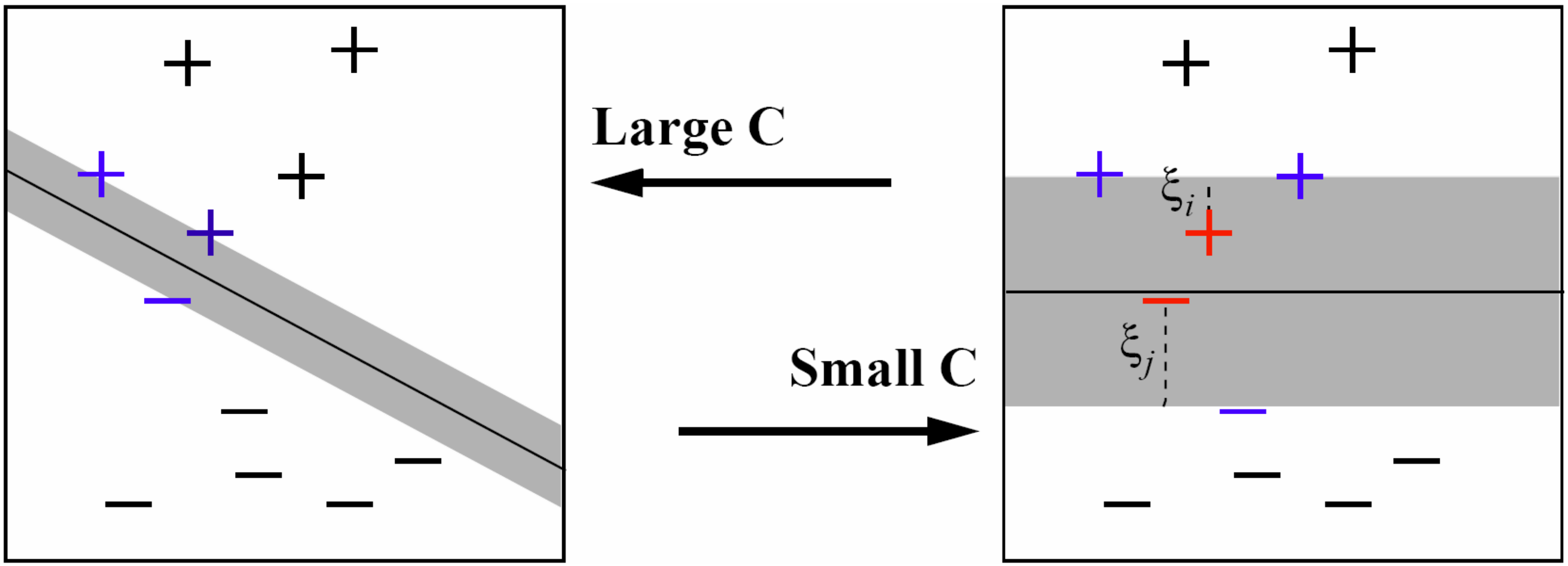
Soft-Margin OP (Primal):

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

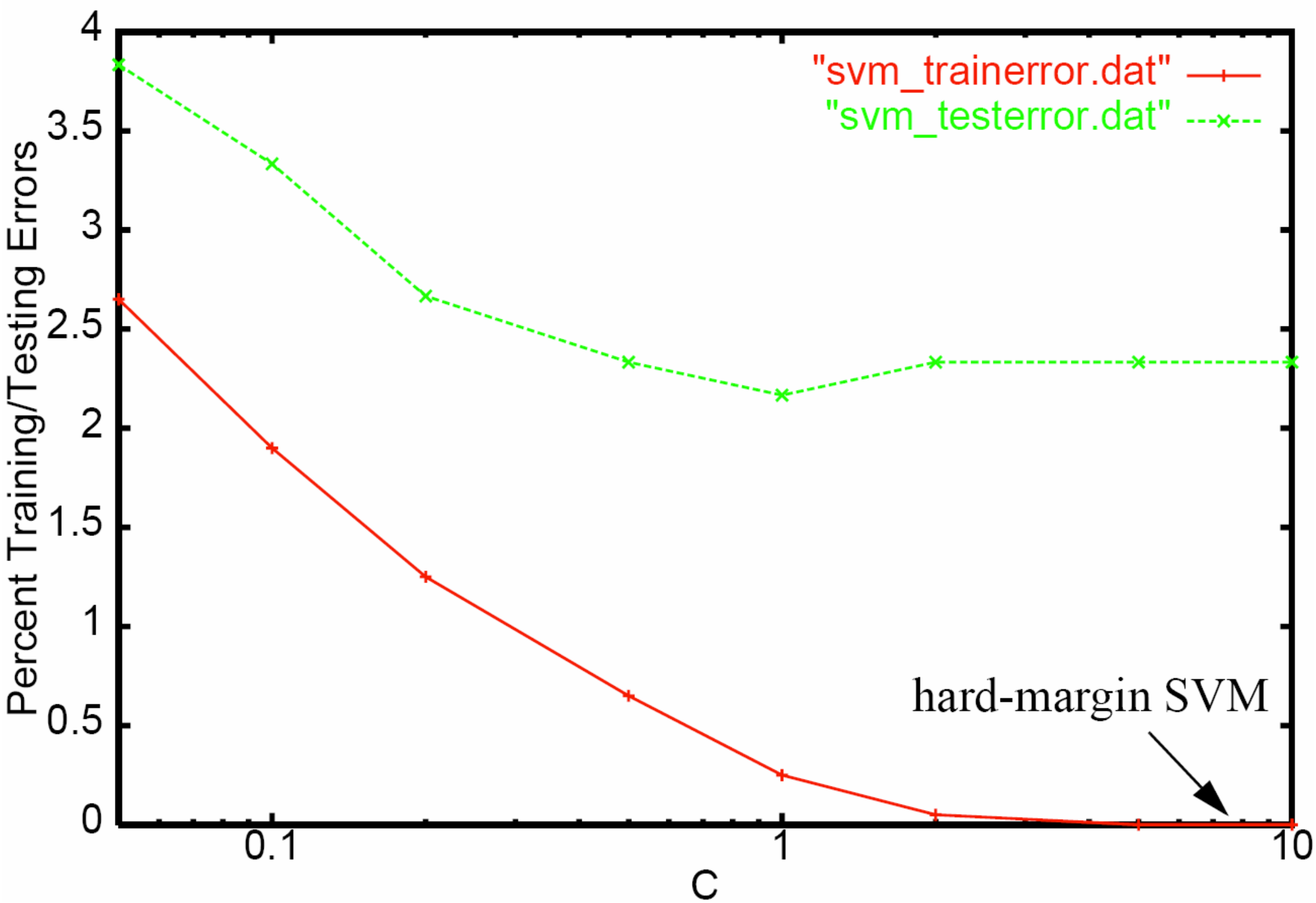
$$s.t.\ y_1(\vec{w}\cdot\vec{x}_1 + b) \geq 1 - \xi_1 \wedge \xi_1 \geq 0$$

...

$$y_n(\vec{w}\cdot\vec{x}_n + b) \geq 1 - \xi_n \wedge \xi_n \geq 0$$



Example Reuters "acq": Varying C



Example: Margin in High-Dimension

Training Sample S_{train}	$\vec{x}$							**y**
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
$(\vec{x}_1, y_1)$	1	0	0	1	0	0	0	1
$(\vec{x}_2, y_2)$	1	0	0	0	1	0	0	1
$(\vec{x}_3, y_3)$	0	1	0	0	0	1	0	-1
$(\vec{x}_4, y_4)$	0	1	0	0	0	0	1	-1
	$\vec{w}$							**b**
	w_1	w_2	w_3	w_4	w_5	w_6	w_7	
Hyperplane 1	1	1	0	0	0	0	0	2
Hyperplane 2	0	0	0	1	1	-1	-1	0
Hyperplane 3	1	-1	1	0	0	0	0	0
Hyperplane 4	0.5	-0.5	0	0	0	0	0	0
Hyperplane 5	1	-1	0	0	0	0	0	0
Hyperplane 6	0.95	-0.95	0	0.05	0.05	-0.05	-0.05	0

SVM Solution as Linear Combination

- **Primal OP:**

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

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

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

- **Theorem:** The solution w^* can always be written as a linear combination
$$\vec{w}^* = \sum_{i=1}^{n} \alpha_i y_i \vec{x}_i \text{ with } 0 \leq \alpha_i \leq C$$
of the training vectors.
- **Properties:**
  - Factor α_i indicates "influence" of training example (x_i, y_i).
  - If $\xi_i > 0$, then $\alpha_i = C$.
  - If $0 \leq \alpha_i < C$, then $\xi_i = 0$.
  - (x_i, y_i) is a Support Vector, if and only if $\alpha_i > 0$.
  - If $0 < \alpha_i < C$, then $y_i(x_i\, w+b)=1$.
  - SVM-light outputs α_i using the "-a" option

Dual SVM Optimization Problem

- **Primal Optimization Problem**

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

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

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

- **Dual Optimization Problem**

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

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

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

- **Theorem:** If $w*$ is the solution of the Primal and $\alpha*$ is the solution of the Dual, then $\vec{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \vec{x}_i$

Leave-One-Out (i.e. n-fold CV)

Training Set: $S = ((\vec{x}_1, y_1), ..., (\vec{x}_n, y_n))$

Approach: Repeatedly leave one example out for testing.

train on	test on
$(\vec{x}_2, y_2), (\vec{x}_3, y_3), (\vec{x}_4, y_4), \ldots, (\vec{x}_n, y_n)$	$(\vec{x}_1, y_1)$
$(\vec{x}_1, y_1), (\vec{x}_3, y_3), (\vec{x}_4, y_4), \ldots, (\vec{x}_n, y_n)$	$(\vec{x}_2, y_2)$
$(\vec{x}_1, y_1), (\vec{x}_2, y_2), (\vec{x}_4, y_4), \ldots, (\vec{x}_n, y_n)$	$(\vec{x}_3, y_3)$
...	...
$(\vec{x}_1, y_1), (\vec{x}_2, y_2), (\vec{x}_3, y_3), \ldots, (\vec{x}_{n-1}, y_{n-1})$	$(\vec{x}_n, y_n)$

Estimate: $Err_{loo}(A) = \frac{1}{n} \sum_{i=1}^{n} \Delta(h_i(\vec{x}_i), y_i)$

Question: Is there a cheaper way to compute this estimate?

Necessary Condition for Leave-One-Out Error

Lemma: For SVM, $[h_i(\vec{x}_i) \neq y_i] \Longrightarrow \left[2\alpha_i R^2 + \xi_i \geq 1\right]$

Input:

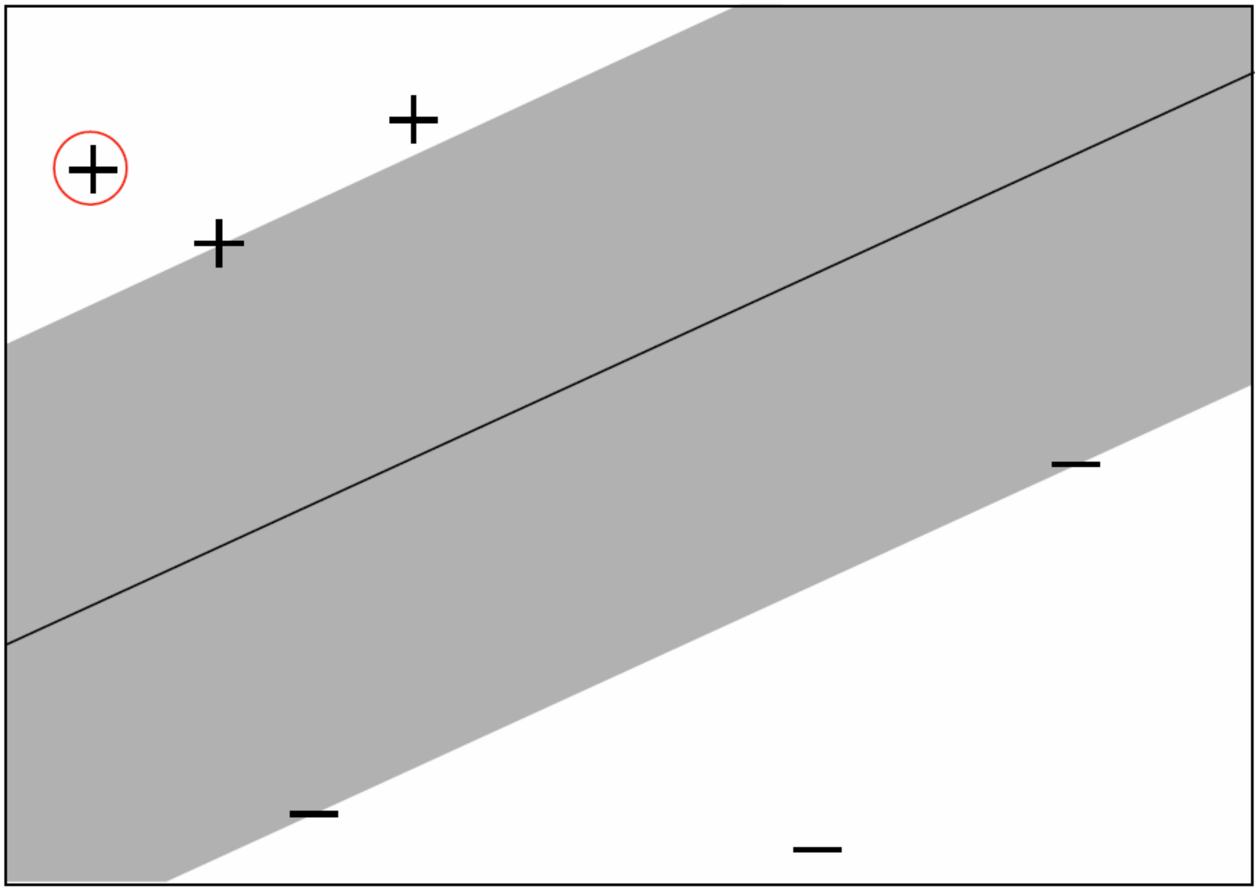
- α_i dual variable of example i
- ξ_i slack variable of example i
- $||x|| \leq R$ bound on length

Example:

Value of $2 \alpha_i R^2 + \xi_i$	Leave-one-out Error?
0.0	Correct
0.7	Correct
3.5	Error
0.1	Correct
1.3	Correct
…	…

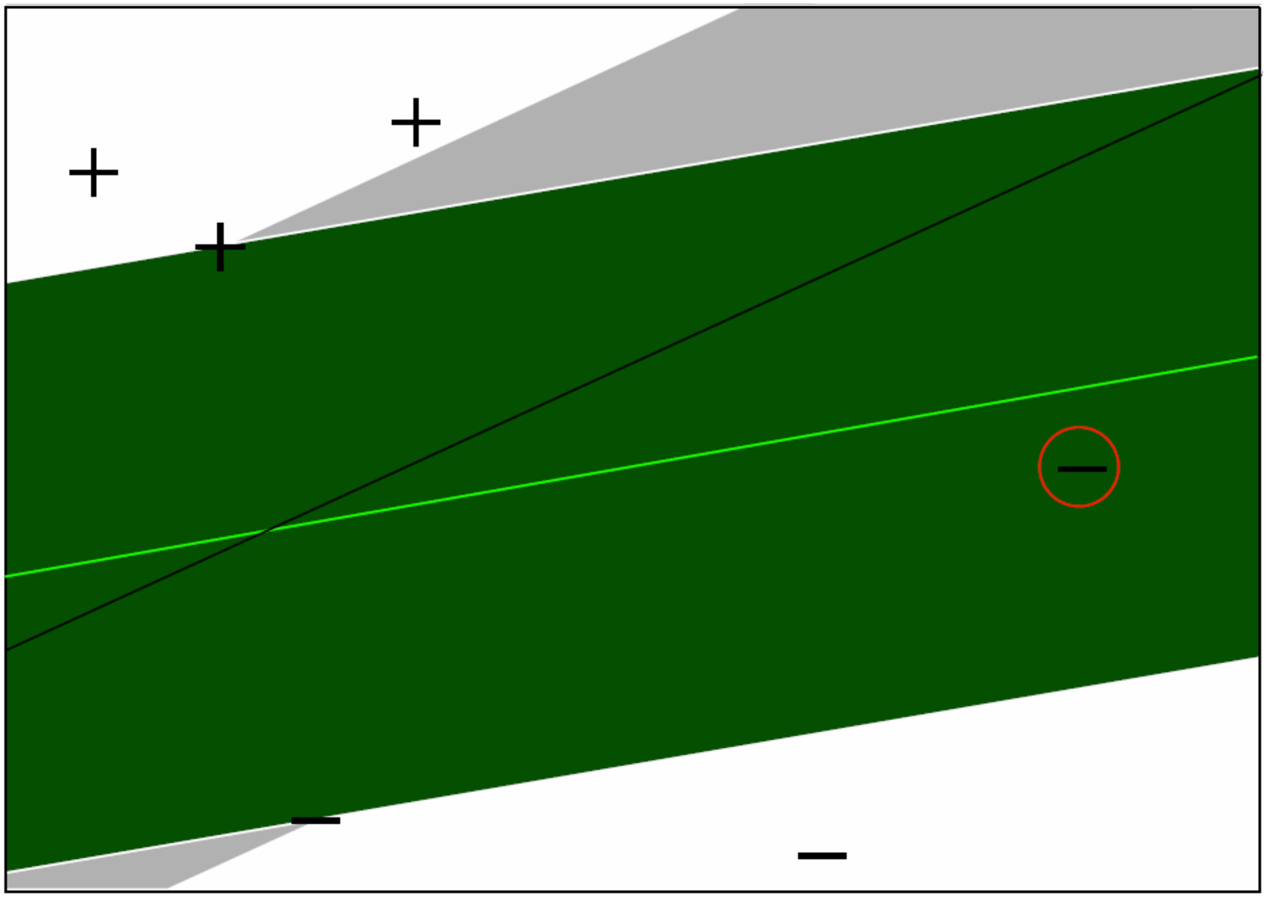
Case 1: Example is not SV

Criterion: $(\alpha_i = 0)$) $(\xi_i=0)$) $(2\,\alpha_i R^2 + \xi_i < 1)$) Correct



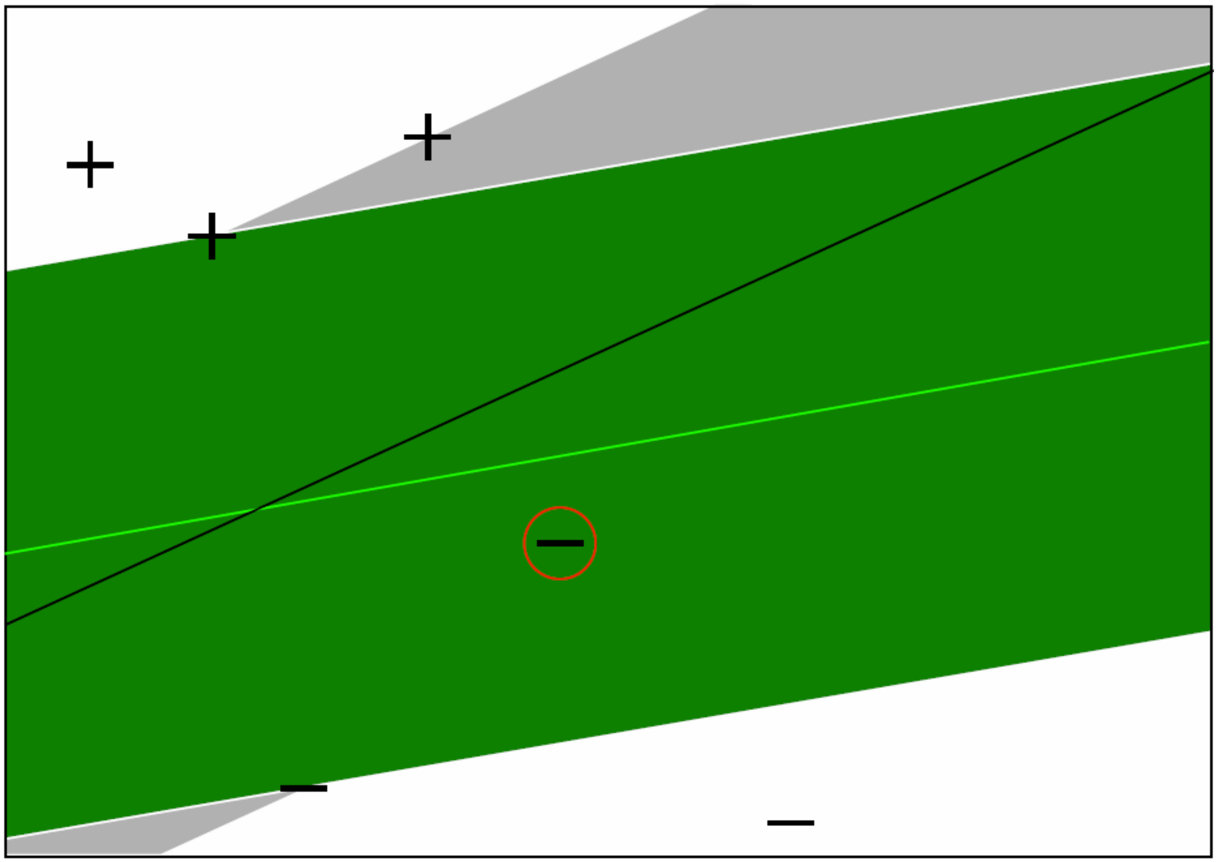
Case 2: Example is SV with Low Influence

Criterion: $(\alpha_i < 0.5/R^2 < C)$) $(\xi_i = 0)$) $(2\alpha_i R^2 + \xi_i < 1)$) Correct



Case 3: Example has Small Training Error

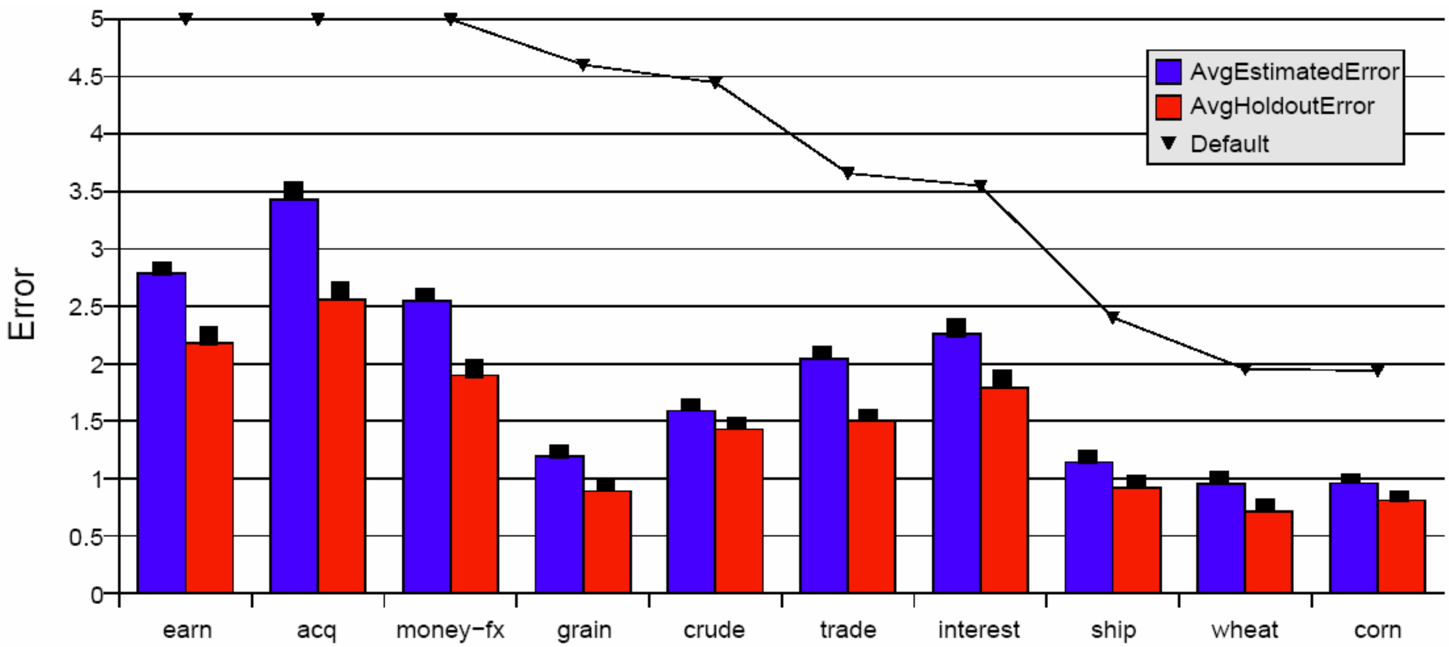
Criterion: $(\alpha_I = C)$) $(\xi_i < 1\text{-}2CR^2)$) $(2\alpha_i R^2+\xi_i < 1)$) Correct



Experiment: Reuters Text Classification

Experiment Setup

- 6451 Training Examples
- 6451 Validation Examples to estimate true Prediction Error
- Comparison between Leave-One-Out upper bound and error on Validation Set (average over 10 test/validation splits)



Fast Leave-One-Out Estimation for SVMs

Lemma: Training errors are always Leave-One-Out Errors.

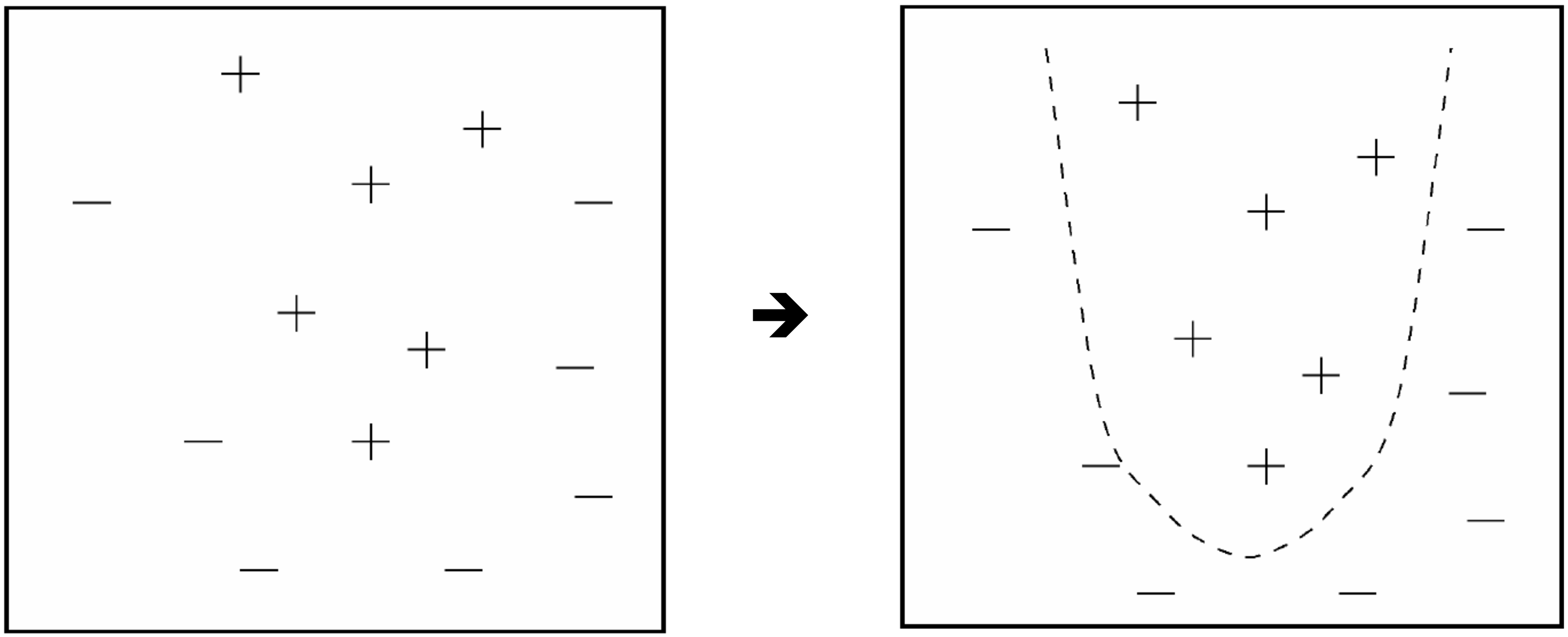
Algorithm:

- $(R,\alpha,\xi) = \text{trainSVM}(S_{train})$
- FOR (x_i,y_i) 2 S_{train}
  - IF $\xi_{\mathbf{i}} > 1$ THEN loo++;
  - ELSE IF $(2\ \alpha_{\mathbf{i}}\ R^2 + \xi_{\mathbf{i}} < 1)$ THEN loo = loo;
  - ELSE $\text{trainSVM}(S_{train} \setminus \{(x_i,y_{\mathbf{i}})\})$ and test explicitly

Experiment:

Training Sample	Retraining Steps (%)	CPU-Time (sec)
Reuters (n=6451)	**0.58%**	**32.3**
WebKB (n=2092)	**20.42%**	**235.4**
Ohsumed (n=10000)	**2.56%**	**1132.3**

Non-Linear Problems



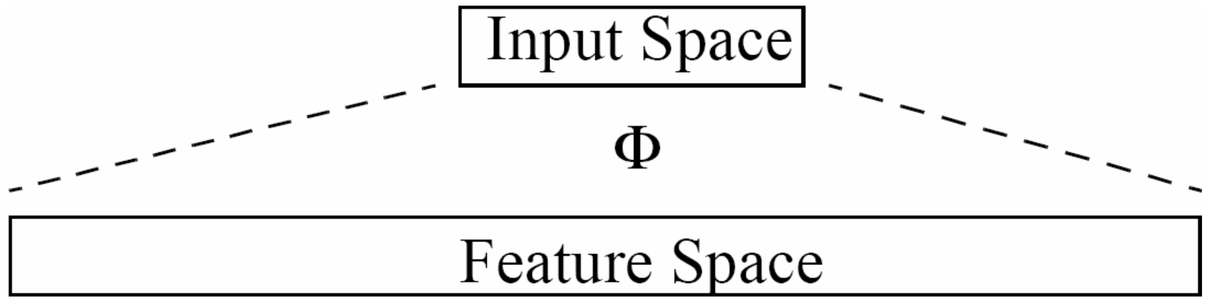
Problem:

- some tasks have non-linear structure
- no hyperplane is sufficiently accurate

How can SVMs learn non-linear classification rules?

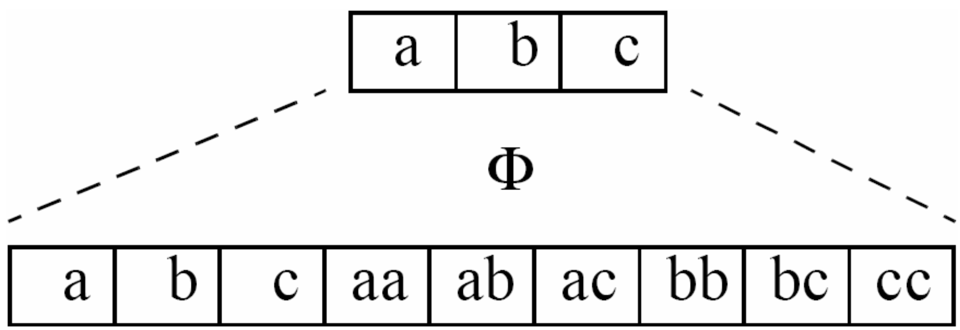
Extending the Hypothesis Space

Idea: add more features



➔ Learn linear rule in feature space.

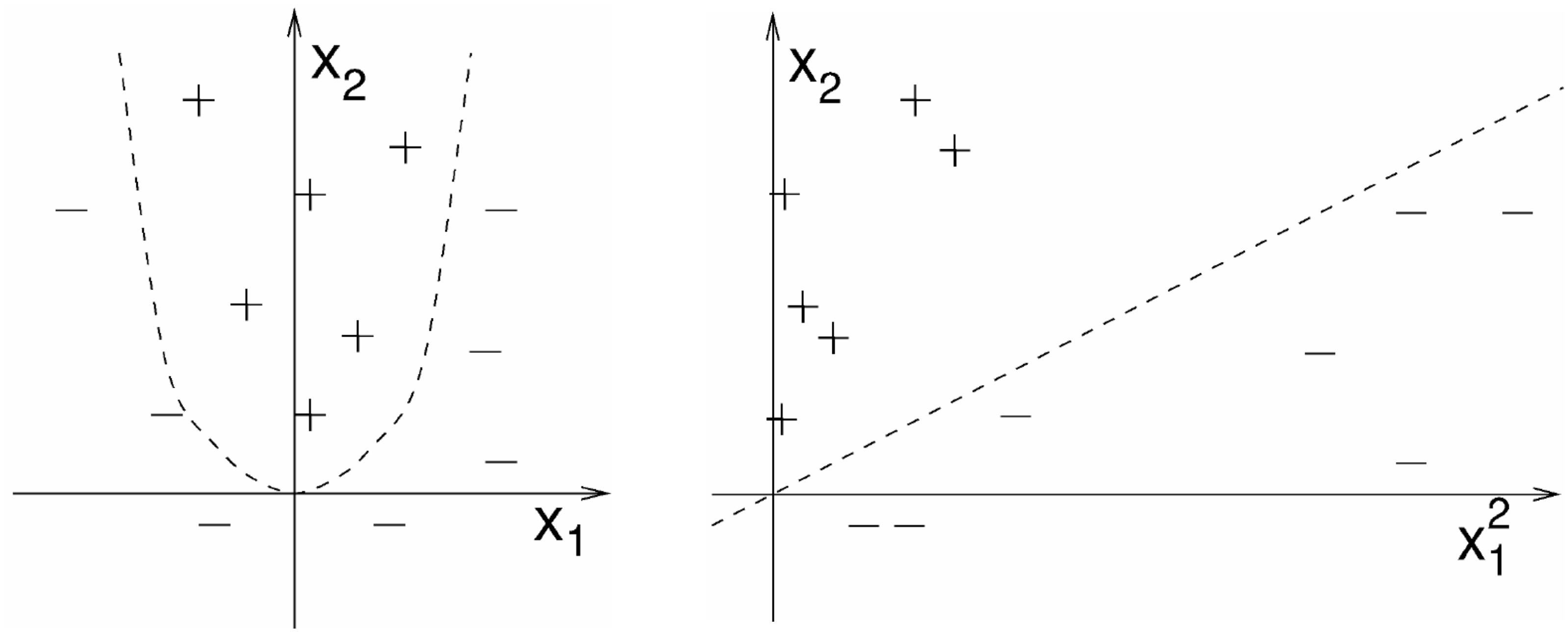
Example:



➔ The separating hyperplane in feature space is degree two polynomial in input space.

Example

- **Input Space:** $\vec{x} = (x_1, x_2)$ **(2 attributes)**
- **Feature Space:** $\Phi(\vec{x}) = (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, 1)$
  (6 attributes)



Dual SVM Optimization Problem

- **Primal Optimization Problem**

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

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

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

- **Dual Optimization Problem**

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

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

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

- **Theorem:** If $w*$ is the solution of the Primal and $\alpha*$ is the solution of the Dual, then $\vec{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \vec{x}_i$

Kernels

Problem: Very many Parameters! Polynomials of degree *p* over *N* attributes in input space lead to attributes in feature space!

Solution: [Boser et al.] The dual OP depends only on inner products => Kernel Functions

$$K(\vec{a},\vec{b}) = \Phi(\vec{a}) \cdot \Phi(\vec{b})$$

Example: For $\Phi(\vec{x}) = (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, 1)$ calculating $K(\vec{a},\vec{b}) = [\vec{a} \cdot \vec{b} + 1]^2$ computes inner product in feature space.

➔ no need to represent feature space explicitly.

SVM with Kernel

Training:

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

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

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

Classification:

$$h(\vec{x}) = sign\left(\left[\sum_{i=1}^{n} \alpha_i y_i \Phi(\vec{x}_i)\right] \cdot \Phi(\vec{x}) + b\right)$$

$$= sign\left(\sum_{i=1}^{n} \alpha_i y_i K(\vec{x}_i, \vec{x}) + b\right)$$

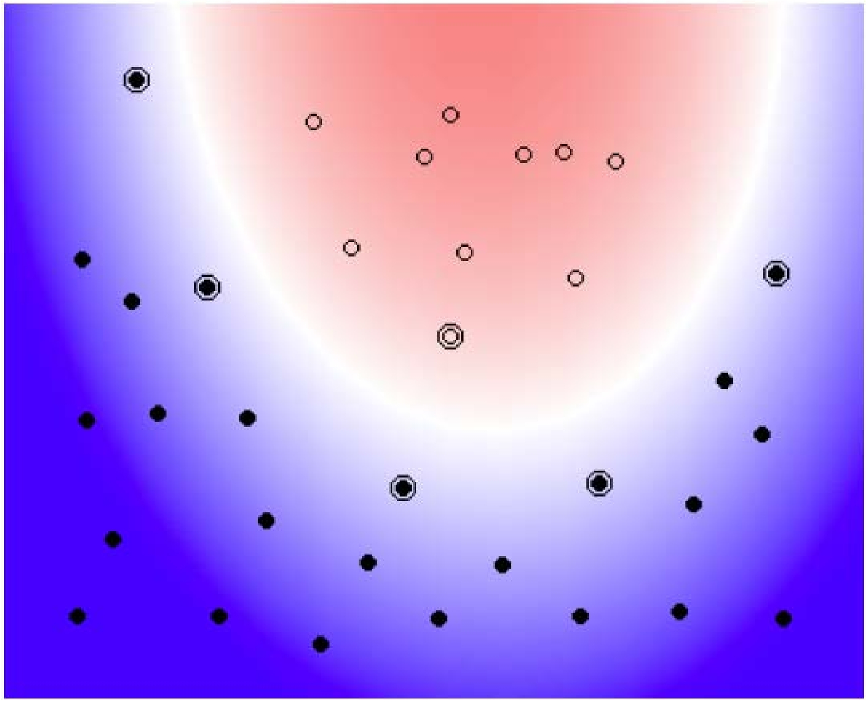
New hypotheses spaces through new Kernels:

- **Linear:** $K(\vec{a}, \vec{b}) = \vec{a} \cdot \vec{b}$
- **Polynomial:** $K(\vec{a}, \vec{b}) = [\vec{a} \cdot \vec{b} + 1]^d$
- **Radial Basis Function:** $K(\vec{a}, \vec{b}) = exp(-\gamma[\vec{a} - \vec{b}]^2)$
- **Sigmoid:** $K(\vec{a}, \vec{b}) = tanh(\gamma[\vec{a} \cdot \vec{b}] + c)$

Examples of Kernels

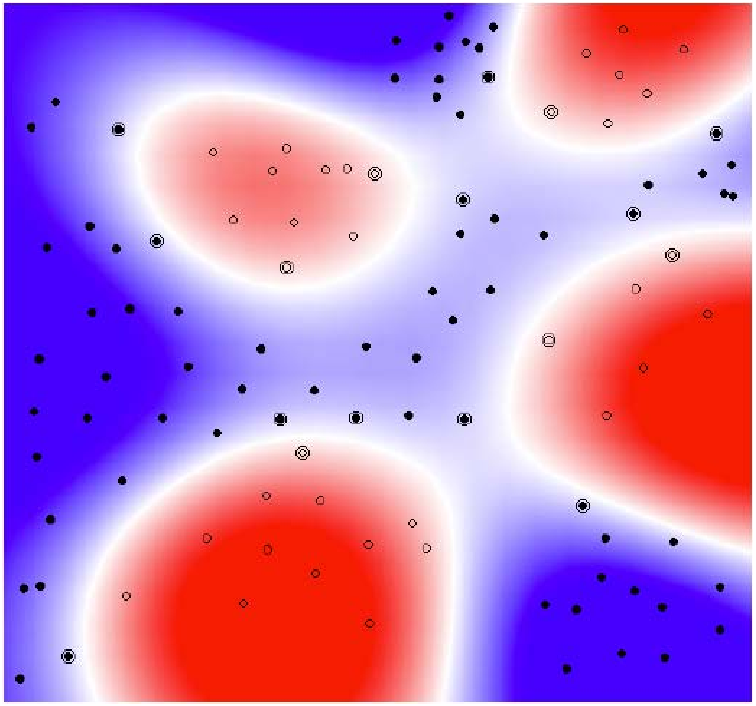
Polynomial

$K(\vec{a}, \vec{b}) = [\vec{a} \cdot \vec{b} + 1]^2$



Radial Basis Function

$K(\vec{a}, \vec{b}) = exp(-\gamma[\vec{a} - \vec{b}]^2)$



What is a Valid Kernel?

Definition: Let X be a nonempty set. A function is a valid kernel in X if for all n and all $x_1, \ldots, x_n$ 2 X it produces a Gram matrix

$$G_{ij} = K(x_i, x_j)$$

that is symmetric

$$G = G^T$$

and positive semi-definite

$$\forall \vec{\alpha} : \vec{\alpha}^T G \vec{\alpha} \geq 0$$

How to Construct Valid Kernels

Theorem: Let K_1 and K_2 be valid Kernels over X £ X, X µ **<**^{N}, $\alpha \geq 0$, $0 \leq \lambda \leq 1$, f a real-valued function on X, ϕ:X! **<**^{m} with a kernel K_3 over **<**$^{\mathbf{m}}$ **£ <**$^{\mathbf{m}}$, and K a symmetric positive semi-definite matrix. Then the following functions are valid Kernels

$$K(x,z) = \lambda\, K_1(x,z) + (1-\lambda)\, K_2(x,z)$$

$$K(x,z) = \alpha\, K_1(x,z)$$

$$K(x,z) = K_1(x,z)\, K_2(x,z)$$

$$K(x,z) = f(x)\, f(z)$$

$$K(x,z) = K_3(\phi(x),\phi(z))$$

$$K(x,z) = x^{T}\, K\, z$$

Kernels for Discrete and Structured Data

Kernels for Sequences: Two sequences are similar, if the have many common and consecutive subsequences.

Example [Lodhi et al., 2000]: For $0 \leq \lambda \leq 1$ consider the following features space

	c-a	c-t	a-r	b-a	b-t	c-r	a-r	b-r
ϕ(cat)	λ^2	λ^3	λ^2	0	0	0	0	0
ϕ(car)	λ^2	0	0	0	0	λ^3	λ^2	0
ϕ(bat)	0	0	λ^2	λ^2	λ^3	0	0	0
ϕ(bar)	0	0	0	λ^2	0	0	λ^2	λ^3

=> $K(car,cat) = \lambda^4$, efficient computation via dynamic programming

Kernels for Non-Vectorial Data

- **Applications with Non-Vectorial Input Data**
  → classify non-vectorial objects
  - Protein classification (x is string of amino acids)
  - Drug activity prediction (x is molecule structure)
  - Information extraction (x is sentence of words)
  - Etc.
- **Applications with Non-Vectorial Output Data**
  → predict non-vectorial objects
  - Natural Language Parsing (y is parse tree)
  - Noun-Phrase Co-reference Resolution (y is clustering)
  - Search engines (y is ranking)

→ Kernels can compute inner products efficiently!

Properties of SVMs with Kernels

- **Expressiveness**
  - SVMs with Kernel can represent any boolean function (for appropriate choice of kernel)
  - SVMs with Kernel can represent any sufficiently “smooth” function to arbitrary accuracy (for appropriate choice of kernel)
- **Computational**
  - Objective function has no local optima (only one global)
  - Independent of dimensionality of feature space
- **Design decisions**
  - Kernel type and parameters
  - Value of C

Reading: Support Vector Machines

- **Books**
  - Schoelkopf, Smola, “Learning with Kernels”, MIT Press, 2002.
  - Cristianini, Shawe-Taylor. “Introduction to Support Vector Machines”, Cambridge University Press, 2000.
  - Cristianini, Shawe-Taylor. ???

SVMs for other Problems

- **Multi-class Classification**
  - [Schoelkopf/Smola Book, Section 7.6]
- **Regression**
  - [Schoelkopf/Smola Book, Section 1.6]
- **Outlier Detection**
  - D.M.J. Tax and R.P.W. Duin, "Support vector domain description", Pattern Recognition Letters, vol. 20, pp. 1191-1199, 1999b. 26
- **Ordinal Regression and Ranking**
  - Herbrich et al., “Large Margin Rank Boundaries for Ordinal Regression”, Advances in Large Margin Classifiers, MIT Press, 1999.
  - Joachims, “Optimizing Search Engines using Clickthrough Data”, ACM SIGKDD Conference (KDD), 2001.

Supervised Learning

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

$$h : X \longrightarrow Y$$

such that the prediction error is low.

x

Microsoft announced today that they acquired Apple for the amount equal to the gross national product of Switzerland. Microsoft officials stated that they first wanted to buy Switzerland, but eventually were turned off by the mountains and the snowy winters…

→

y

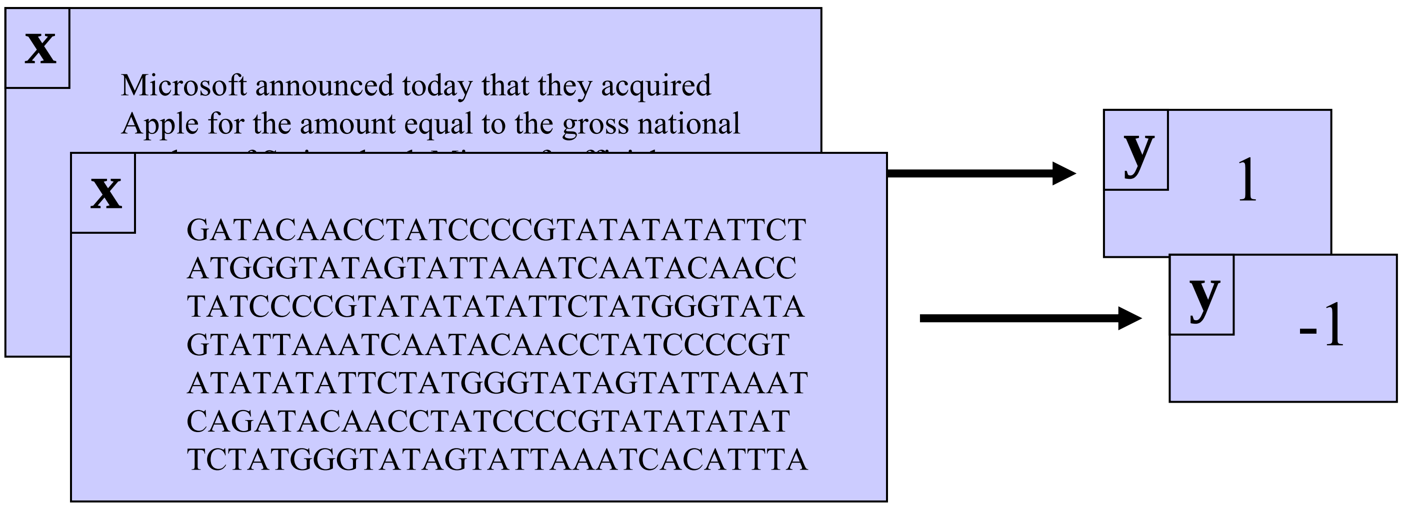
1

Supervised Learning

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

$$h : X \longrightarrow Y$$

such that the prediction error is low.

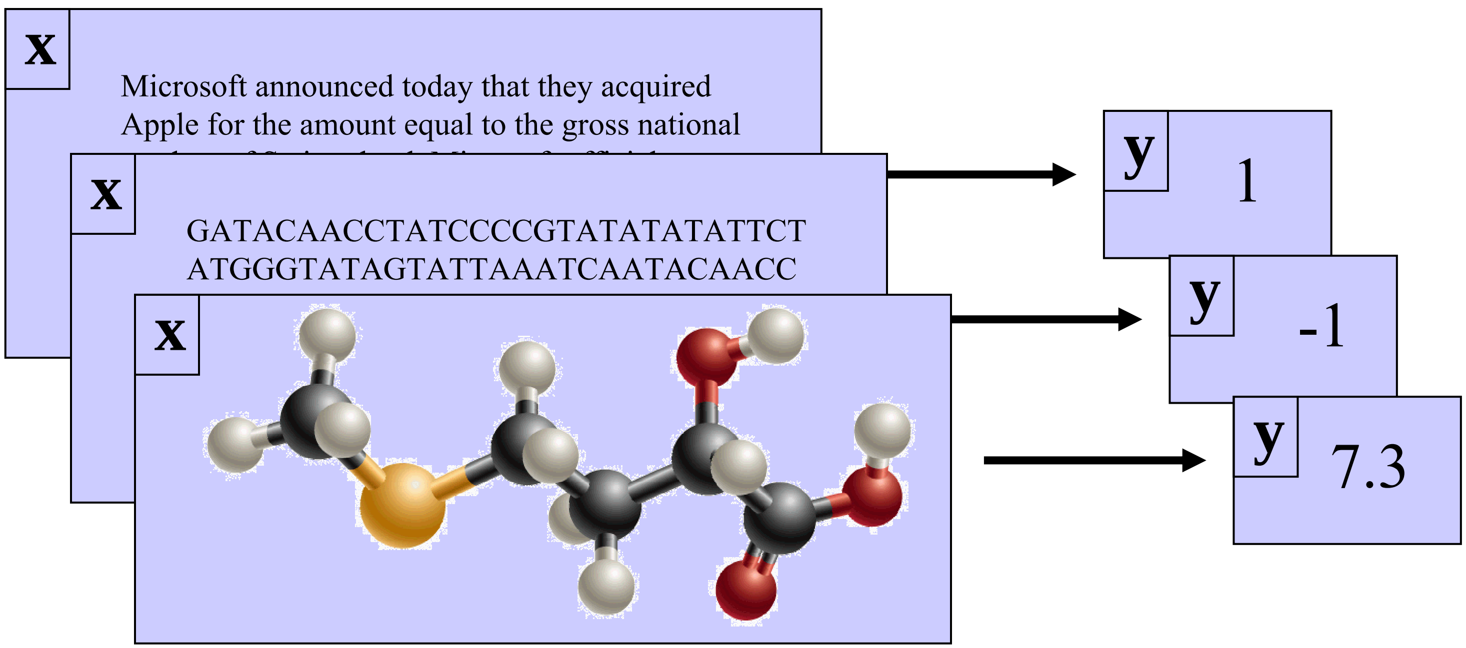


Supervised Learning

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

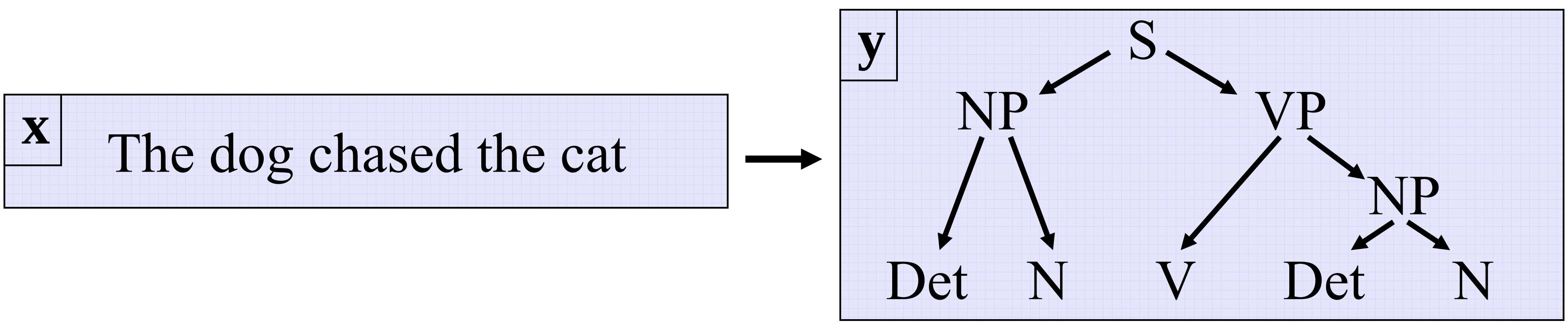
$$h : X \longrightarrow Y$$

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

- **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)

x

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

→

y

-1	antarctica
-1	benelux
-1	germany
+1	iraq
+1	oil
-1	coal
-1	trade
-1	acquisitions

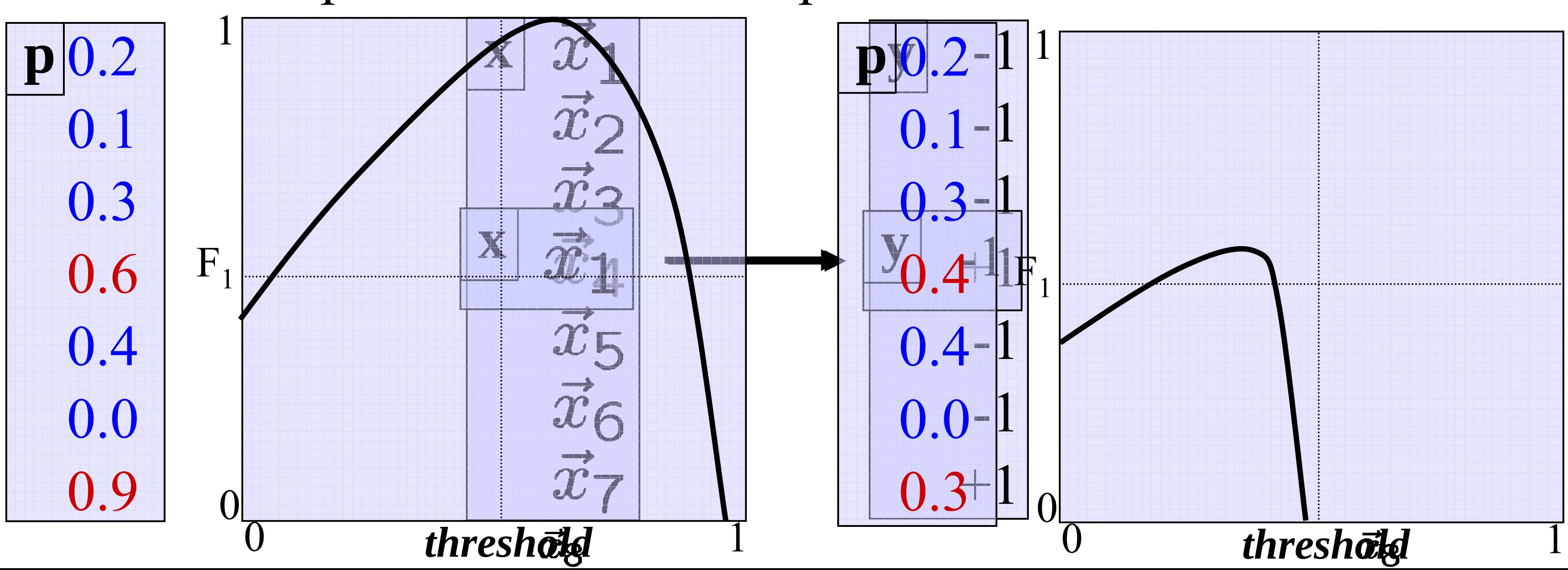
Examples of Complex Output Spaces

- **Non-Standard Performance Measures (e.g. F_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

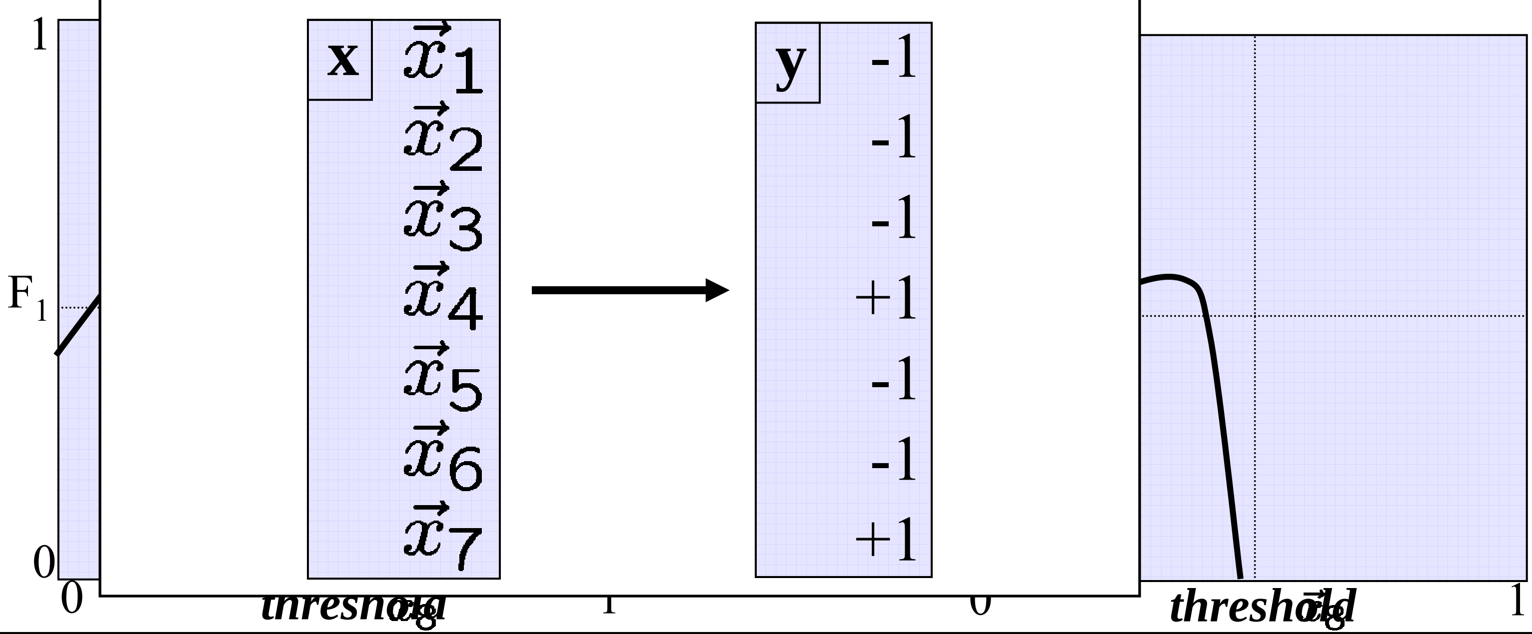
- **Non-Standard Performance Measures (e.g. F_1-score, Lift)**
  - F_1-score: harmonic average of precision and recall

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

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

    ➔ Depends on other examples!

p
0.2
0.1
0.3
0.6
0.4
0.0
0.9



x	y
$\vec{x}_1$	-1
$\vec{x}_2$	-1
$\vec{x}_3$	-1
$\vec{x}_4$	+1
$\vec{x}_5$	-1
$\vec{x}_6$	-1
$\vec{x}_7$	+1

$\longrightarrow$

Examples of Complex Output Spaces

- **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)

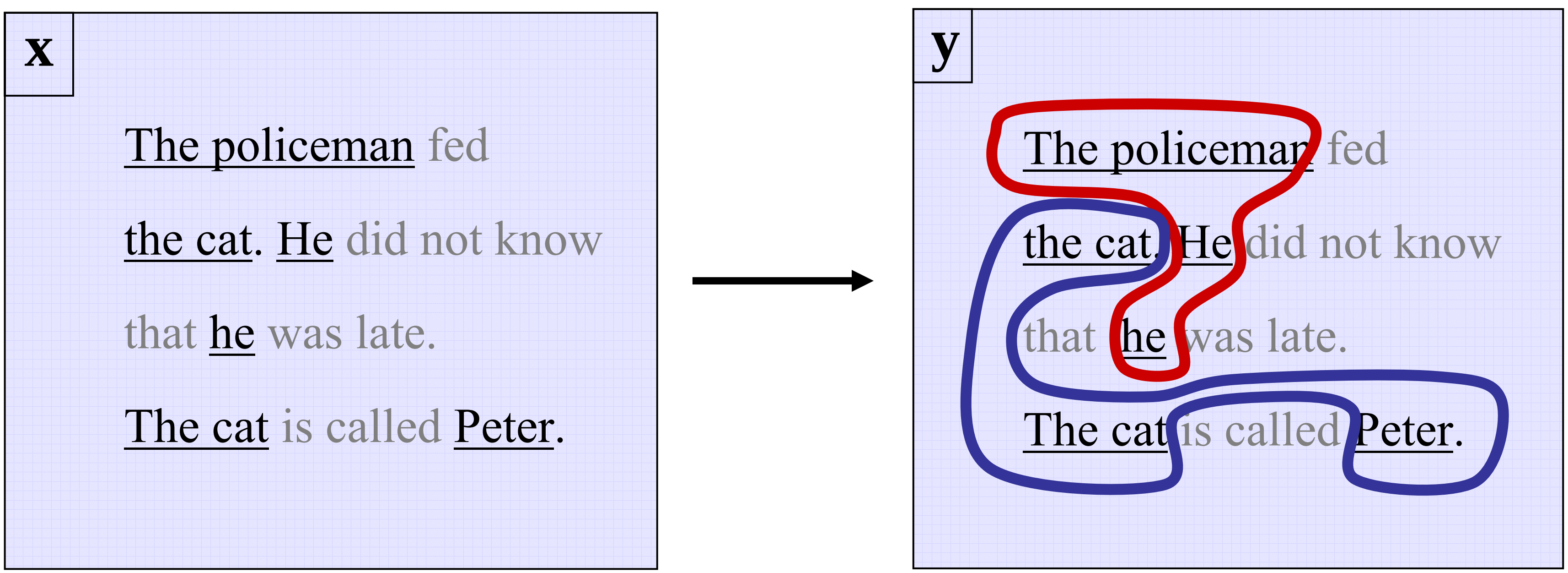
x SVM $\longrightarrow$ **y**

1. Kernel-Machines
2. SVM-Light
3. Learning with Kernels
4. SV Meppen Fan Club
5. Service Master & Co.
6. School of Volunteer Management
7. SV Mattersburg Online

…

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.



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
```

Outline: Structured Output Prediction with SVMs

- **Task: Learning to predict complex outputs**
- **SVM algorithm for complex outputs**
  - Formulation as convex quadratic program
  - General algorithm
  - Sparsity bound
- **Example 1: Learning to parse natural language**
  - Learning weighted context free grammar
- **Example 2: Learning to align proteins**
  - Learning to predict optimal alignment of homologous proteins for comparative modelling

Why do we Need Research on Complex Outputs?

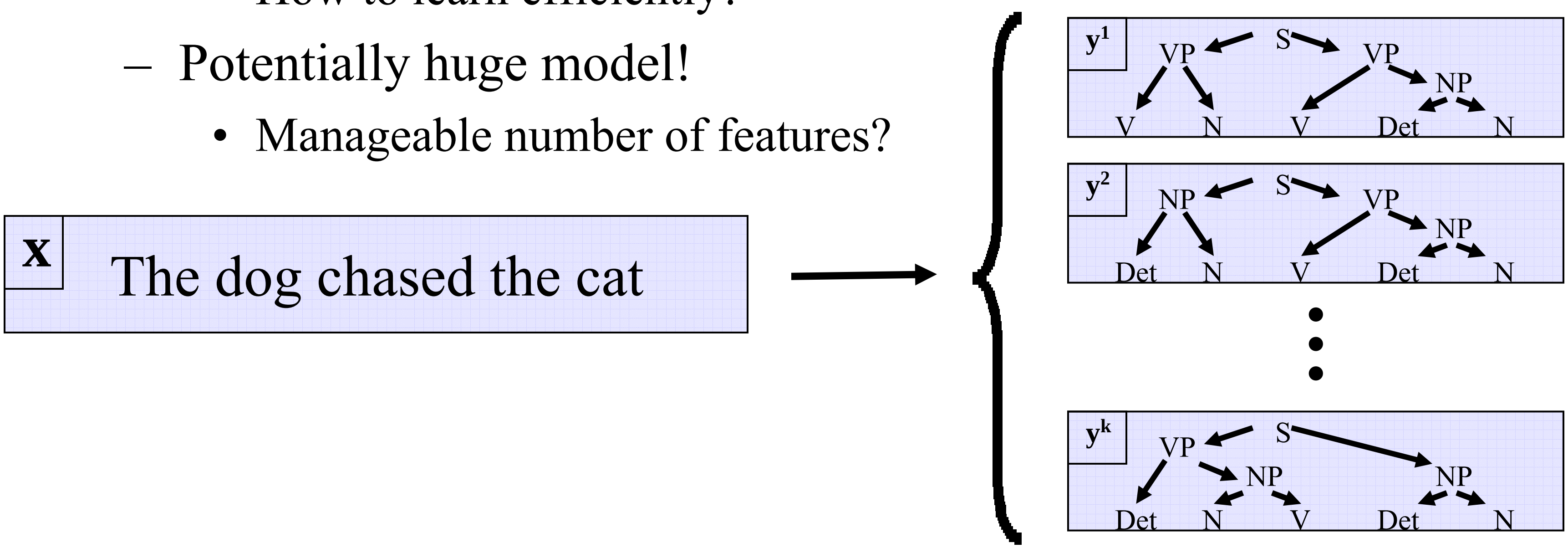
- **Important applications for which conventional methods don't fit!**
  - Noun-phrase co-reference: two step approaches of pair-wise classification and clustering as postprocessing, e.g [Ng & Cardie, 2002]
  - Directly optimize complex loss functions (e.g. F1, AvgPrec)
- **Improve upon existing methods!**
  - Natural language parsing: generative models like probabilistic context-free grammars
  - SVM outperforms naïve Bayes for text classification [Joachims, 1998] [Dumais et al., 1998]
- **More flexible models!**
  - Avoid generative (independence) assumptions
  - Kernels for structured input spaces and non-linear functions
- **Transfer what we learned for classification and regression!**
  - Boosting
  - Bagging
  - Support Vector Machines

Related Work

- **Generative training (i.e. learn P(Y,X))**
  – Hidden-Markov models
  – Probabilistic context-free grammars
  – Markov random fields
  – Etc.
- **Discriminative training (i.e. learn P(Y|X))**
  – 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]
  – Perceptron training of HMM [Collins, 2002]
  – Maximum-margin Markov networks [Taskar et al., 2003]

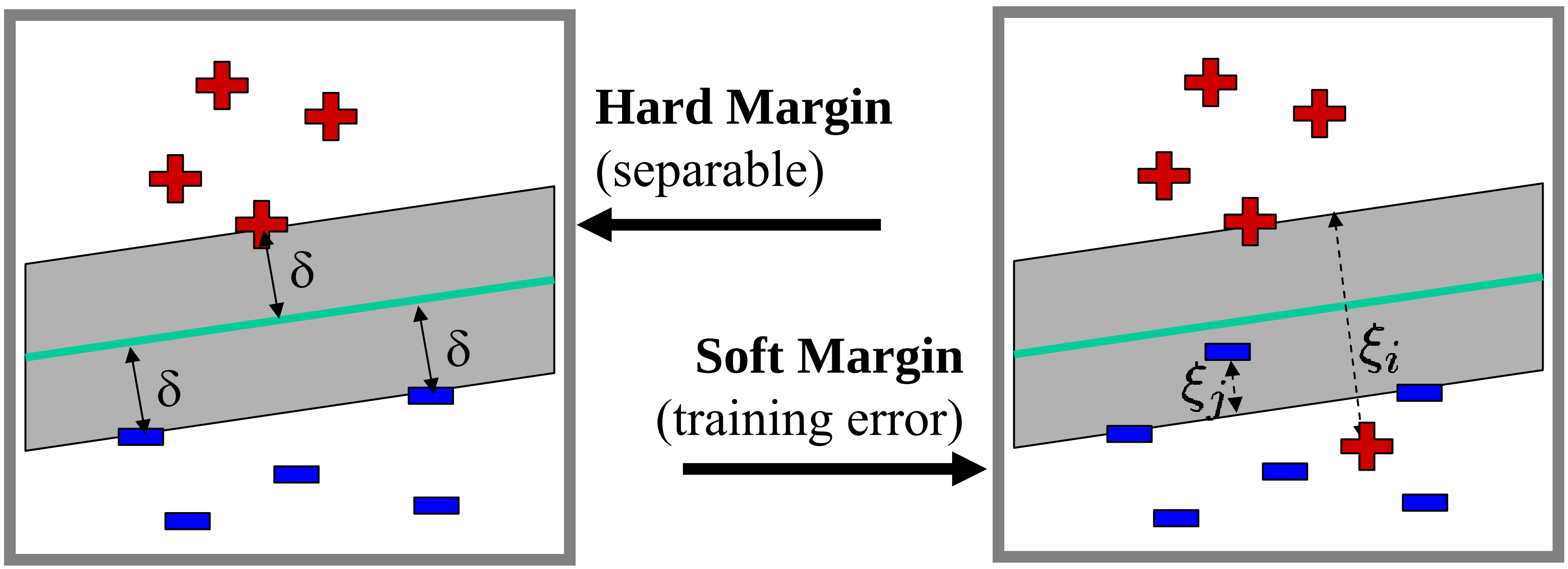
Challenges in Discriminative Learning with Complex Outputs

- **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?



Support Vector Machine [Vapnik et al.]

- **Training Examples:** $(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n) \quad \vec{x} \in \Re^N \quad y \in \{+1, -1\}$
- **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 $\frac{1}{\delta^2} + C \sum_{i=1}^{n} \xi_i$



Support Vector Machine [Vapnik et al.]

- **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 $\frac{1}{\delta^2} + C \sum_{i=1}^{n} \xi_i$

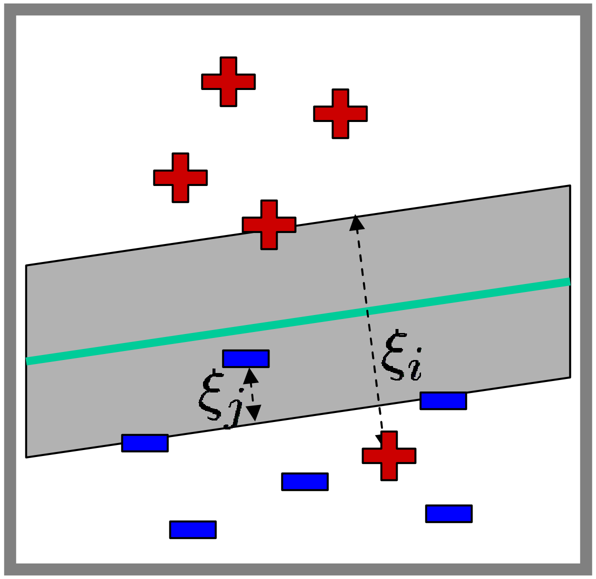
Optimization Problem:

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

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

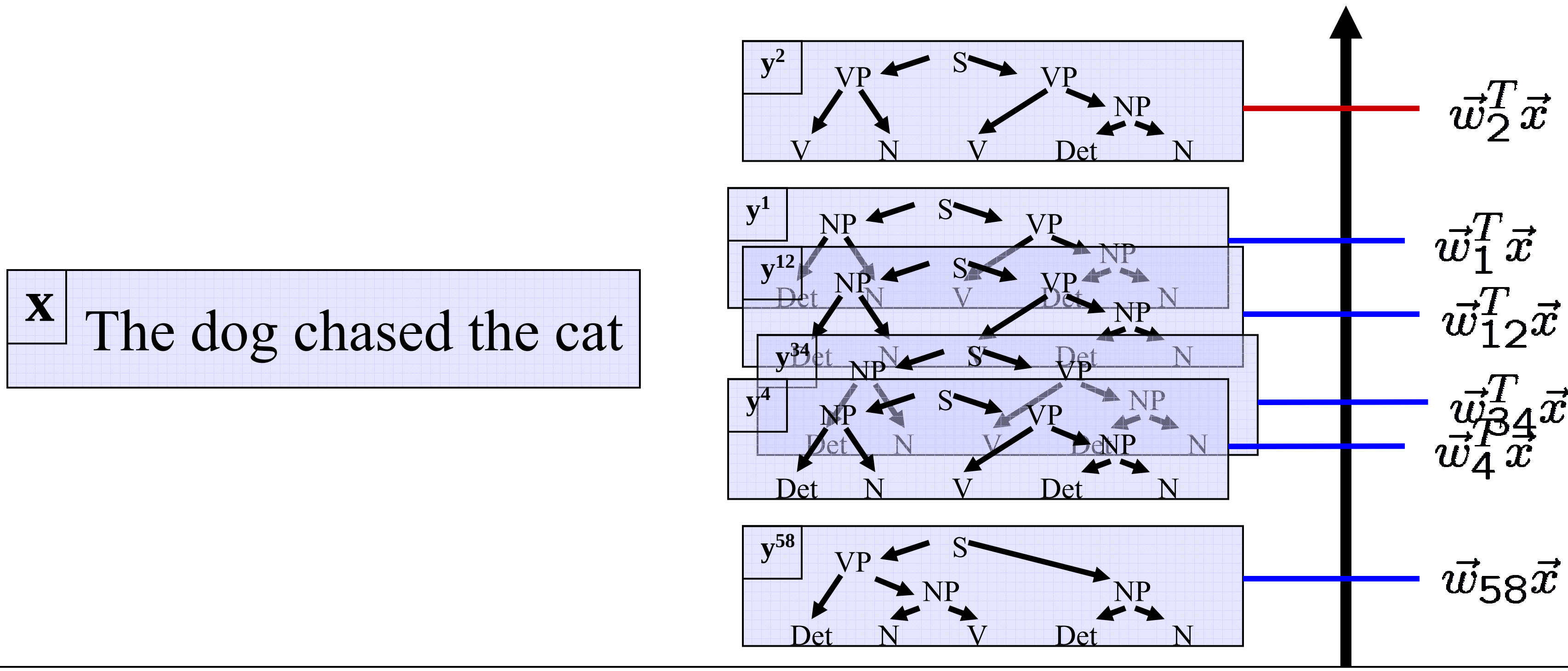
$$\ldots$$

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



Multi-Class SVM [Crammer & Singer]

- **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:** $h(\vec{x}) = argmax_{i \in \{1,\ldots,k\}} \left[\vec{w}_i^T \vec{x} \right]$



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

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

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

$$\ldots$$

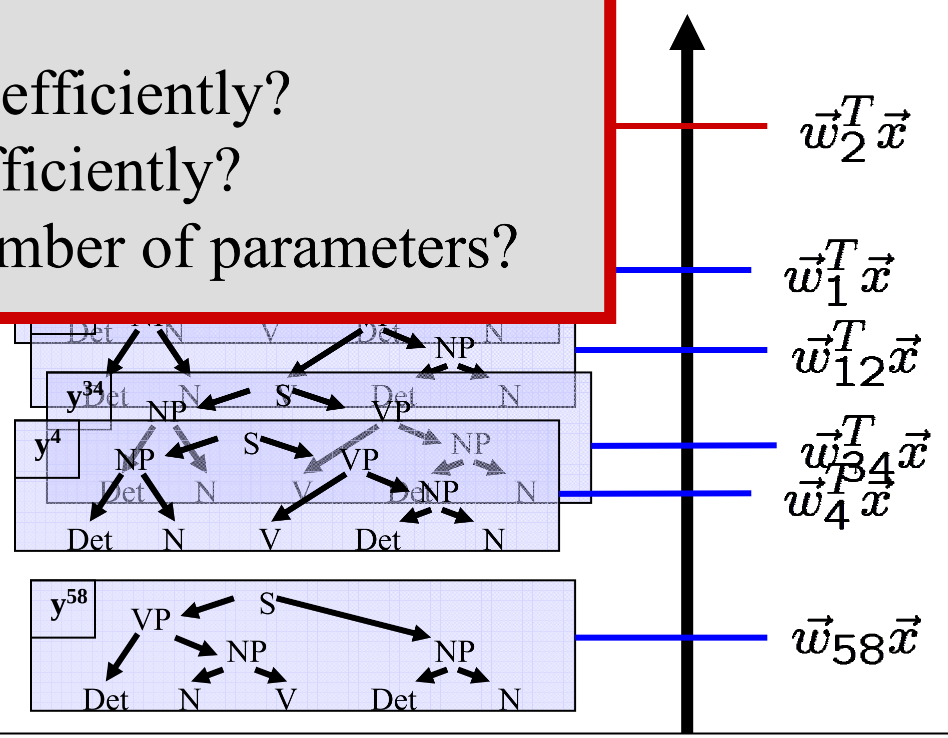
$$\forall j \neq y_n : \vec{w}_{y_n}^T \vec{x}_n \geq \vec{w}_j^T \vec{x}_n + 1 - \xi_n$$

- …, $k\}$
-

Problems

- How to predict efficiently?
- How to learn efficiently?
- Manageable number of parameters?

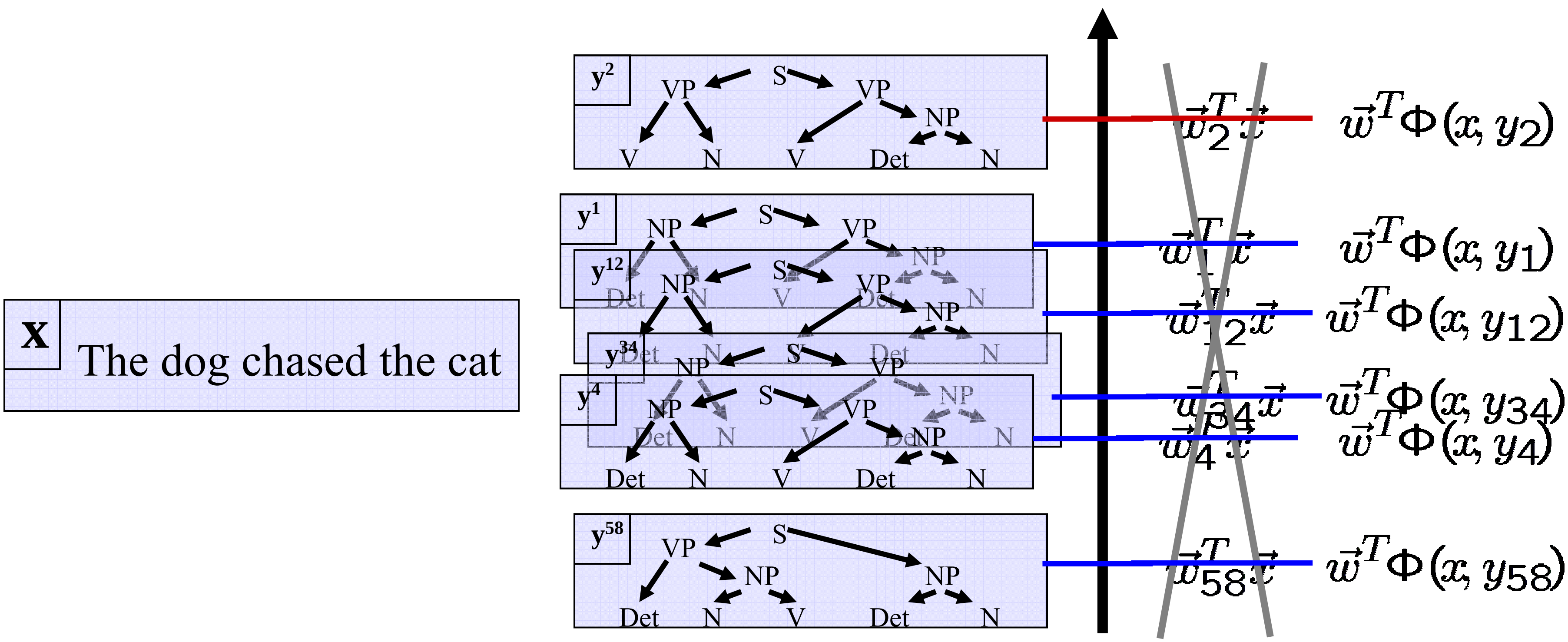
x The dog chased the cat



Joint Feature Map

- **Feature vector $\Phi(x, y)$ that describes match between *x* and *y***
- **Learn single weight vector and rank by $\vec{w}^T \Phi(x, y)$**

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

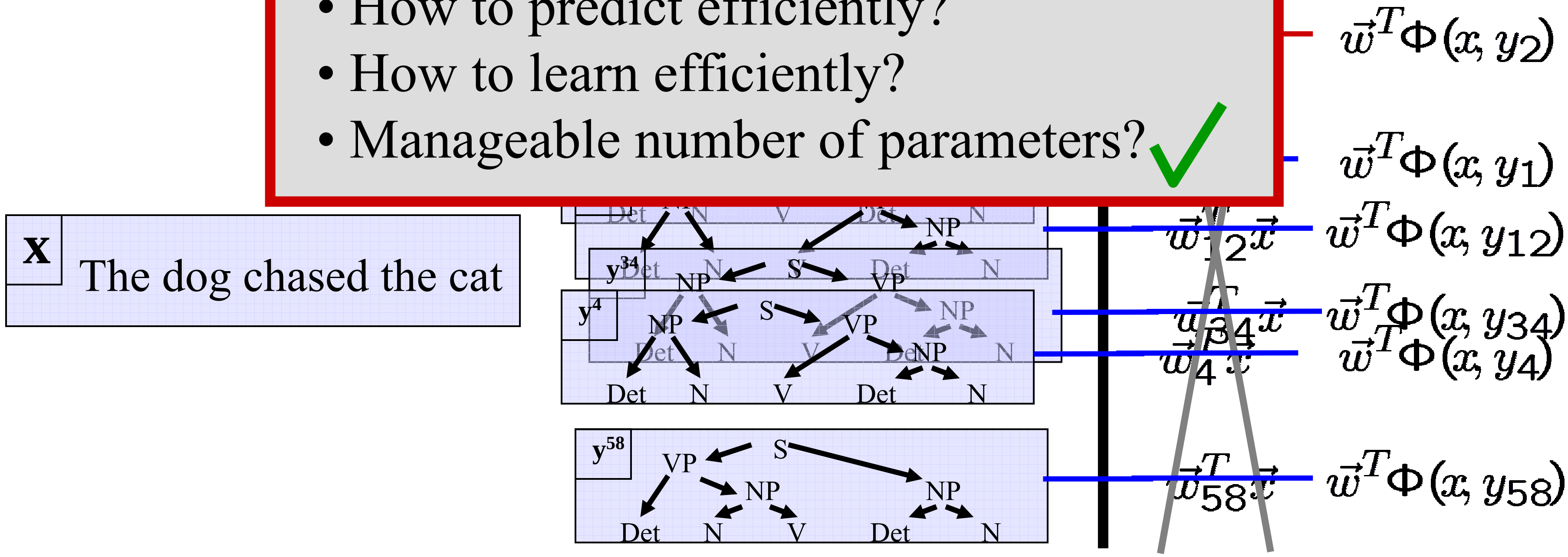


Joint Feature Map

- **Feature vector $\Phi(x, y)$ that describes match between *x* and *y***
- **Learn single weight vector and rank by $\vec{w}^T \Phi(x, y)$**

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

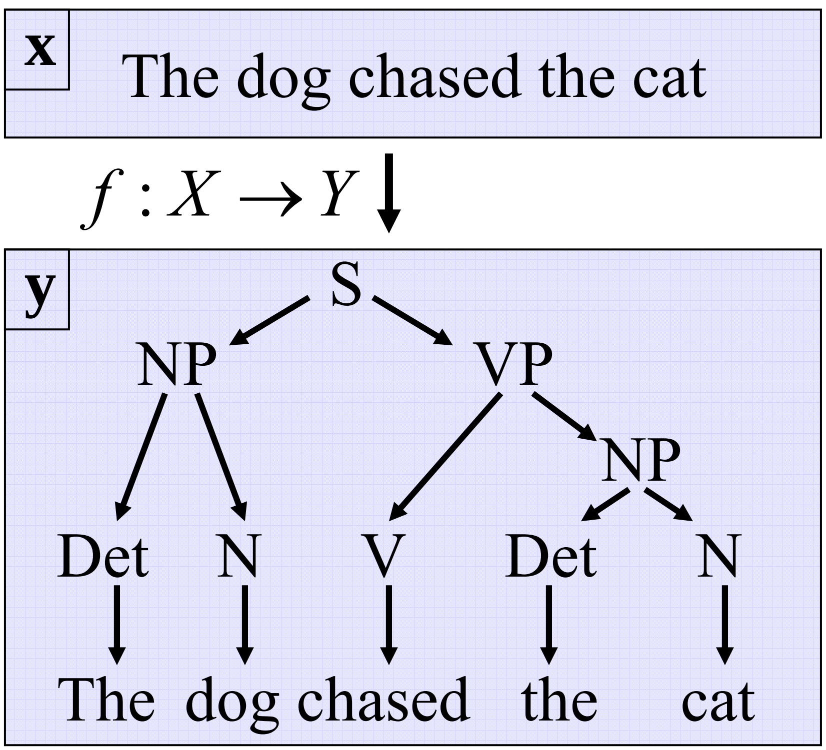
<u>Problems</u>
- How to predict efficiently?
- How to learn efficiently?
- Manageable number of parameters? ✓



Joint Feature Map for Trees

CKY Parser

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



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

Joint Feature Map for Trees

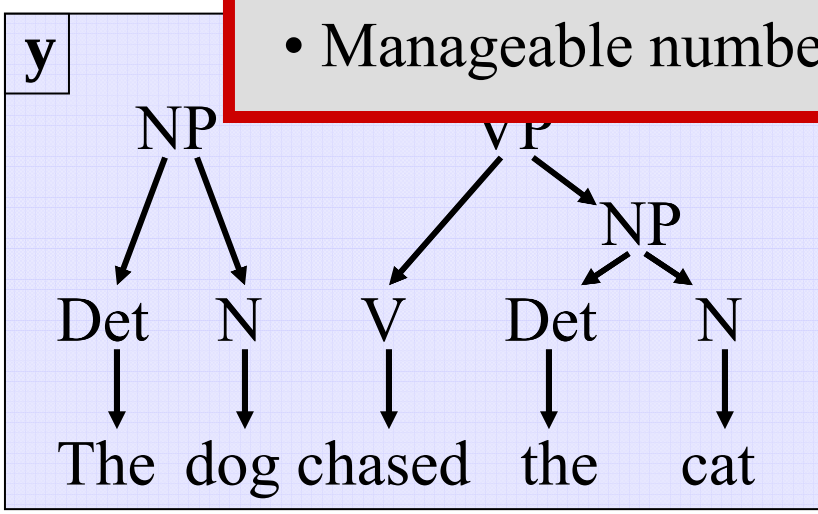
- **Weighted Context Free Grammar**
  - Each rule r_i (e.g. $S \rightarrow NP\ VP$) has a weight w_i
  - Score of a tree is the sum of its weights
  - Find highest scoring tree $h(\vec{x}) = argmax_{y \in Y} \left[\vec{w}^T \Phi(x,y)\right]$

CKY Parser

x The

$f : X$

y



Problems

- How to predict efficiently? ✓
- How to learn efficiently?
- Manageable number of parameters? ✓

$\rightarrow NP\ VP$

$\rightarrow NP$

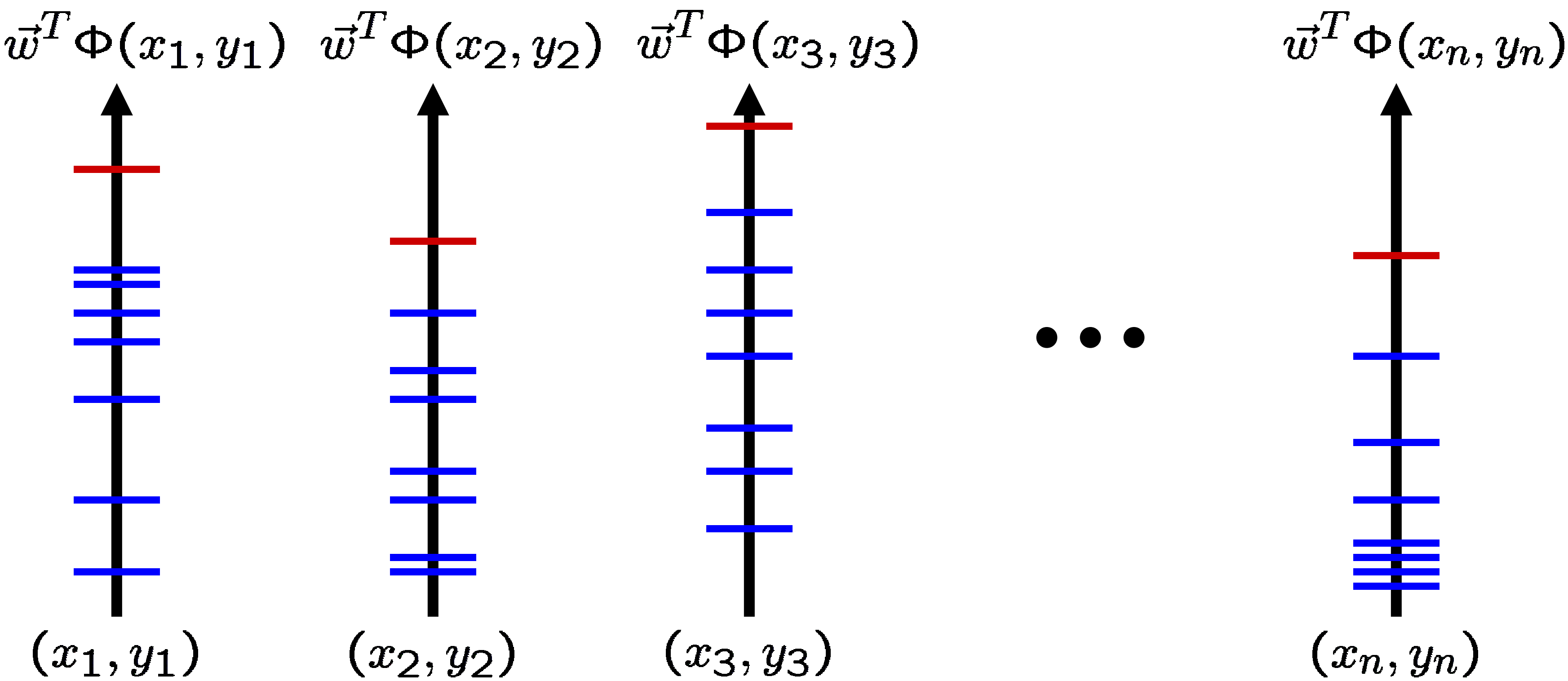
$\rightarrow Det\ N$

$\rightarrow V\ NP$

$$\Phi(\mathbf{x},\mathbf{y}) = \begin{pmatrix} \vdots & \\ 0 & Det \rightarrow dog \\ 2 & Det \rightarrow the \\ 1 & N \rightarrow dog \\ 1 & V \rightarrow chased \\ 1 & N \rightarrow cat \end{pmatrix}$$

Structural Support Vector Machine

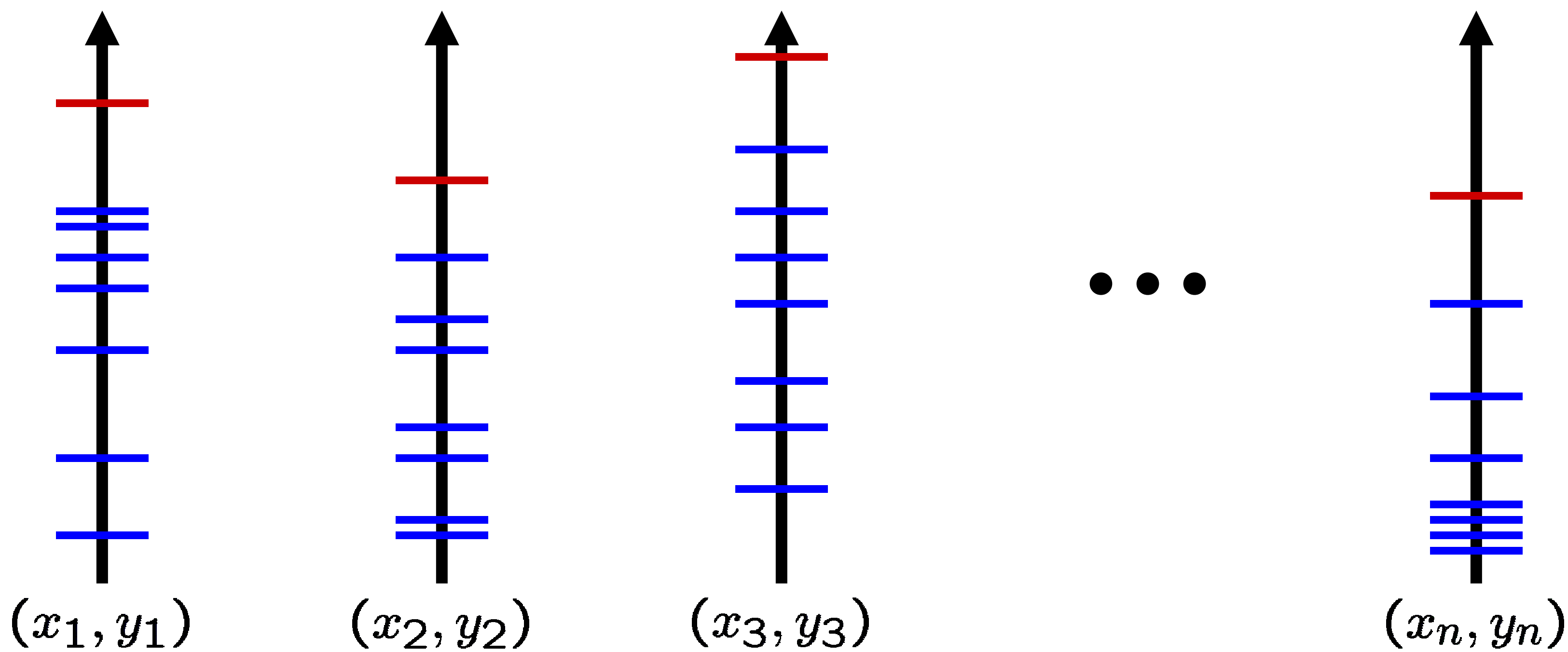
- **Joint features $\Phi(x, y)$ describe match between x and y**
- **Learn weights $\vec{w}$ so that $\vec{w}^T\Phi(x, y)$ is max for correct y**



Structural Support Vector Machine

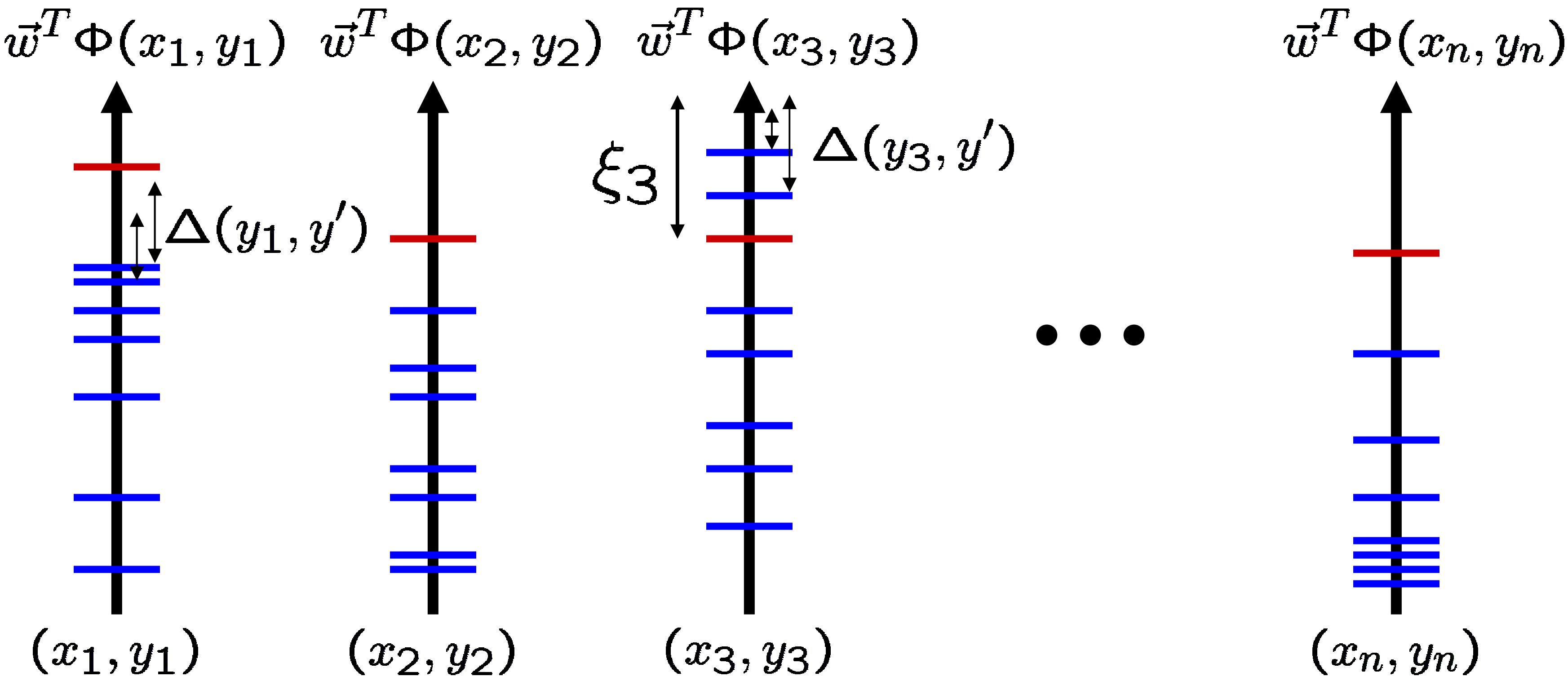
Hard-margin optimization problem:

$$
\begin{aligned}
\min_{\vec{w}} \quad & \frac{1}{2}\vec{w}^T\vec{w} \\
s.t. \quad & \forall y \in Y \setminus y_1 : \vec{w}^T\Phi(x_1, y_1) \geq \vec{w}^T\Phi(x_1, y) + 1 \\
& \dots \\
& \forall y \in Y \setminus y_n : \vec{w}^T\Phi(x_n, y_n) \geq \vec{w}^T\Phi(x_n, y) + 1
\end{aligned}
$$



Loss Functions: Soft-Margin Struct SVM

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



Loss Functions: Soft-Margin Struct SVM

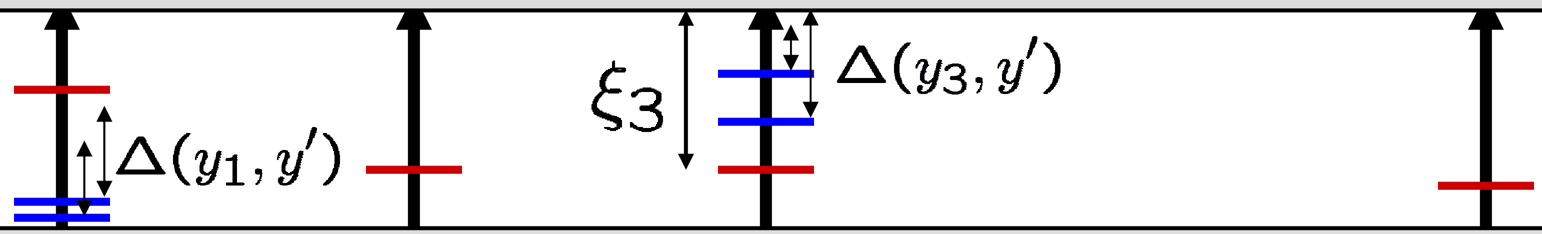
Soft-margin optimization problem:

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

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

$$\dots$$

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



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)

Sparse Approximation 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

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

[Jo03] [TsoJoHoAl05]

Polynomial Sparsity Bound

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

Problems

- How to predict efficiently? ✓
- How to learn efficiently? ✓
- Manageable number of parameters? ✓

constr … Tucker conditions are fulfilled up to a precision ϵ . The loss has to be bounded $0 \leq \Delta(y_i, y) \leq A$, and $||\Phi(x,y)|| \leq R$.

[Jo03] [TsoJoHoAl05]

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

[TsoJoHoAl05]

More Expressive Features

- **Linear composition:** $\Phi(x, \vec{y}) = \sum_{j=1}^{k} \phi(x, y_j)$

- **So far:** $\phi(x, y_i) = \begin{pmatrix} 0 \\ \ldots \\ 0 \\ 1 \\ 0 \\ \ldots \\ 0 \end{pmatrix}$ if $y_i = \text{'S} \leftarrow \text{NP VP'}$

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

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

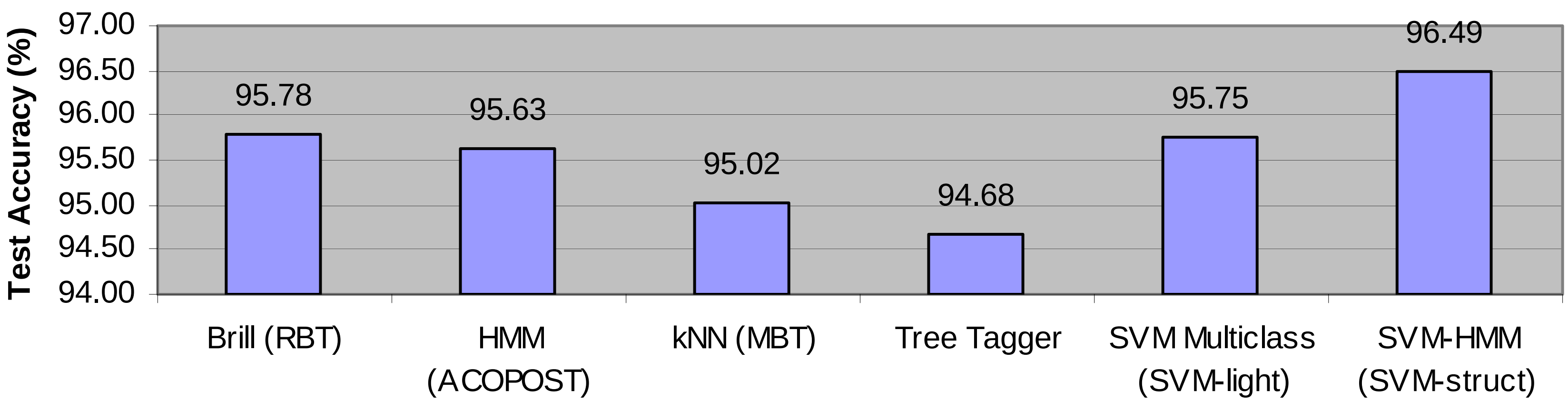
- **Example:** $\phi(x, y_i) = \begin{pmatrix} 1 & \text{if } x_{start} = \text{'while'} \wedge x_{end} = \text{'.'} \\ (start - end)^2 & \\ 1 & \text{span contains } x_{start} = \text{'and'} \\ \ldots & \end{pmatrix}$

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 StructSVM to New Problem

- **Basic algorithm implemented in SVM-struct**
  - `http://svmlight.joachims.org`

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

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

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

$\Rightarrow$ **Generic structure that covers OMM, MPD, Finite-State Transducers, MRF, etc. (polynomial time inference)**

Outline: Structured Output Prediction with SVMs

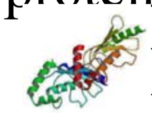
- **Task: Learning to predict complex outputs**
- **SVM algorithm for complex outputs**
  - Formulation as convex quadratic program
  - General algorithm
  - Sparsity bound
- **Example 1: Learning to parse natural language**
  - Learning weighted context free grammar
- **Example 2: Learning to align proteins**
  - Learning to predict optimal alignment of homologous proteins for comparative modeling

Comparative Modeling of Protein Structure

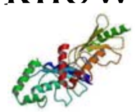
- **Goal: Predict structure from sequence**

  h(“APPGEAYLQV”) → 

- **Hypothesis:**
  - Amino Acid sequences for into structure with lowest engery
  - Problem: Huge search space (> 2^{100} states)
- **Approach: Comparative Modeling**
  - Similar protein sequences fold into similar shapes
    → use known shapes as templates
  - Task 1: Find a similar known protein for a new protein

    h(“APPGEAYLQV”, ) → yes/no

  - Task 2: Map new protein into known structure

    h(“APPGEAYLQV”, ) → [A→3,P→4,P→7,…]

Predicting an Alignment

- **Protein Sequence to Structure Alignment (Threading)**
  - Given a pair *x=(s,t)* of new sequence *s* and known structure *t*, predict the alignment *y*.
  - Elements of *s* and *t* are described by features, not just character identity.

x

```
( BBBLLBBLLHHHHH )
( 32401450143520 )
( ABJLHBNJYAUGAI )

( BHJKBNYGU )
( BBLLBBLLH )
```

→

y

```
BB-BLLBBLLHHHHH
32-401450143520
AB-JLHBNJYAUGAI
 | || || |||
 BHJK-BN-YGU
 BBLL-BB-LLH
```

Linear Score Sequence Alignment

Method: Find alignment *y* that maximizes linear score

Example:

- Sequences:

  `s=(A B C D)`

  `t=(B A C C)`

- Alignment y_1:

A	B	C	D
B	A	C	C

  → score = 0+0+10-10 = 0

- Alignment y_2:

-	A	B	C	D
B	A	C	C	-

  → score = -5+10+5+10-5 = 15

	A	B	C	D	-
A	10	0	-5	-10	-5
B	0	10	5	-10	-5
C	-5	5	10	-10	-5
D	-10	-10	-10	10	-5
-	-5	-5	-5	-5	-5

Algorithm: Dynamic programming

How to Estimate the Scores?

- **General form of linear scoring function:**

$$score\,(\mathbf{x}{=}(\mathbf{s},\mathbf{t}),\mathbf{y}) \;=\; \sum_i score(y_i^{\mathbf{s}}, y_i^{\mathbf{t}})$$

- **Estimation:**
  - Generative estimation of $score(y_i^{\mathbf{s}}, y_i^{\mathbf{t}})$ via
    - Log-odds
    - Hidden Markov Model
  - Discriminative estimation of complex models via SVM

$$\begin{aligned} score\,(\mathbf{x}{=}(\mathbf{s},\mathbf{t}),\mathbf{y}) &= \sum_i score(y_i^{\mathbf{s}}, y_i^{\mathbf{t}}) \\ &= \sum_i \mathbf{w}^T \phi(\mathbf{s},\mathbf{t},y_i) \\ &= \mathbf{w}^T \sum_i \phi(\mathbf{s},\mathbf{t},y_i) \\ &= \mathbf{w}^T \Phi(\mathbf{x},\mathbf{y}) \end{aligned}$$

→ match/gap score can be arbitrary linear function

Expressive Scoring Functions

- **Conventional substitution matrix** $score(y_i^{\mathbf{s}}, y_i^{\mathbf{t}})$
  - Poor performance at low sequence similarity, if only amino acid identity is considered
  - Difficult to design generative models that take care of the dependencies between different features.
  - Would like to make use of structural features like secondary structures, exposed surface area, and take into account the interactions between these features
- **General feature-based scoring function** $\mathbf{w}^T \phi(\mathbf{s}, \mathbf{t}, y_i)$
  - Allows us to describe each character by feature vector (e.g. secondary structure, exposed surface area, contact profile)
  - Learn w vector of parameters
  - Computation of argmax still tractable via dynamic program

Loss Function

- **Q loss: fraction of incorrect alignments**
  - Correct alignment **y**=

-	A	B	C	D
B	A	C	C	-

  - Alternate alignment **y'**=

A	-	B	C	D
B	A	C	C	-

→ $\Delta_Q(y,y')=1/3$

- **Q4 loss: fraction of incorrect alignments outside window**
  - Correct alignment **y**=

-	A	B	C	D
B	A	C	C	-

  - Alternate alignment **y'**=

A	-	B	C	D
B	A	C	C	-

→ $\Delta_{Q4}(y,y')=0/3$

➔ **Model how "bad" different types of mistakes are for structural modelling.**

Experiment

- **Train set [Qiu & Elber]:**
  - 5119 structural alignments for training, 5169 structural alignments for validation of regularization parameter C
- **Test set:**
  - 29764 structural alignments from new deposits to PDB from June 2005 to June 2006.
  - All structural alignments produced by the program CE by superposing the 3D coordinates of the proteins structures. All alignments have CE Z-score greater than 4.5.
- **Features (known for structure, predicted for sequence):**
  - Amino acid identity (A,C,D,E,F,G,H,I,K,L,M,N,P,Q,R,S,T,V,W,Y)
  - Secondary structure (α,β,λ)
  - Exposed surface area (0,1,2,3,4,5)

Results: Model Complexity

Feature Vectors:

- **Simple:** $\Phi(s,t,y_i) \Leftrightarrow$ (A|A; A|C; …;-|Y; α|α; α|β…; 0|0; 0|1;…)
- **Anova2:** $\Phi(s,t,y_i) \Leftrightarrow$ (Aα|Aα…; α0|α0…; A0|A0;…)
- **Tensor:** $\Phi(s,t,y_i) \Leftrightarrow$ (Aα0|Aα0; Aα0|Aα1; …)
- **Window:** $\Phi(s,t,y_i) \Leftrightarrow$ (AAA|AAA; …; ααααα|ααααα; …; 00000|00000;…)

Q-Score	# Features	Training	Validation	Test
Simple	1020	26.83	27.79	39.89
Anova2	49634	42.25	35.58	44.98
Tensor	203280	52.36	34.79	42.81
Window	447016	51.26	38.09	46.30

Q-score when optimizing to Q-loss

Results: Comparison

Q4-score	**Test**
SVM (Window, Q4-loss)	70.71
SSALN [Qiu & Elber]	67.30
BLAST	28.44
TM-align [Zhang & Skolnick]	(85.32)

Methods:

- SVM: train on Window feature vector with Q4-loss
- SSALN: generative method using same training data
- BLAST: lower baseline
- TM-align: upper baseline (disagreement between two structural alignment methods

Conclusions: Structured Output Prediction

- **Learning to predict complex output**
  - Predict structured objects
  - Optimize loss functions over multivariate predictions
- **An SVM method for learning with complex outputs**
  - Learning to predict trees (natural language parsing) [Tsochantaridis et al. 2004 (ICML), 2005 (JMLR)] [Taskar et al., 2004 (ACL)]
  - Optimize to non-standard performance measures (imbalanced classes) [Joachims, 2005 (ICML)]
  - Learning to cluster (noun-phrase coreference resolution) [Finley, Joachims, 2005 (ICML)]
  - Learning to align proteins [Yu et al., 2005 (ICML Workshop)]
- **Software: SVM^{struct}**
  - **`http://svmlight.joachims.org/`**

Reading: Structured Output Prediction

- **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] [Tsochantaridis et al., 2004, 2005] [Taskar 2004]

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]
- **More flexible models!**
  - Avoid generative (independence) assumptions
  - Kernels for structured input spaces and non-linear functions
- **Transfer what we learned for classification and regression!**
  - Boosting
  - Bagging
  - Support Vector Machines

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**
  - A
  - F
- **Tra**
  - F
  - F
  - 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