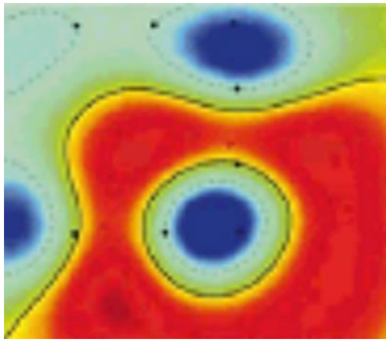
Technische Universität Berlin

Structured Output Prediction

Support Vector Machines and Kernels



From Simple to Structured Prediction

Standard machine learning models:

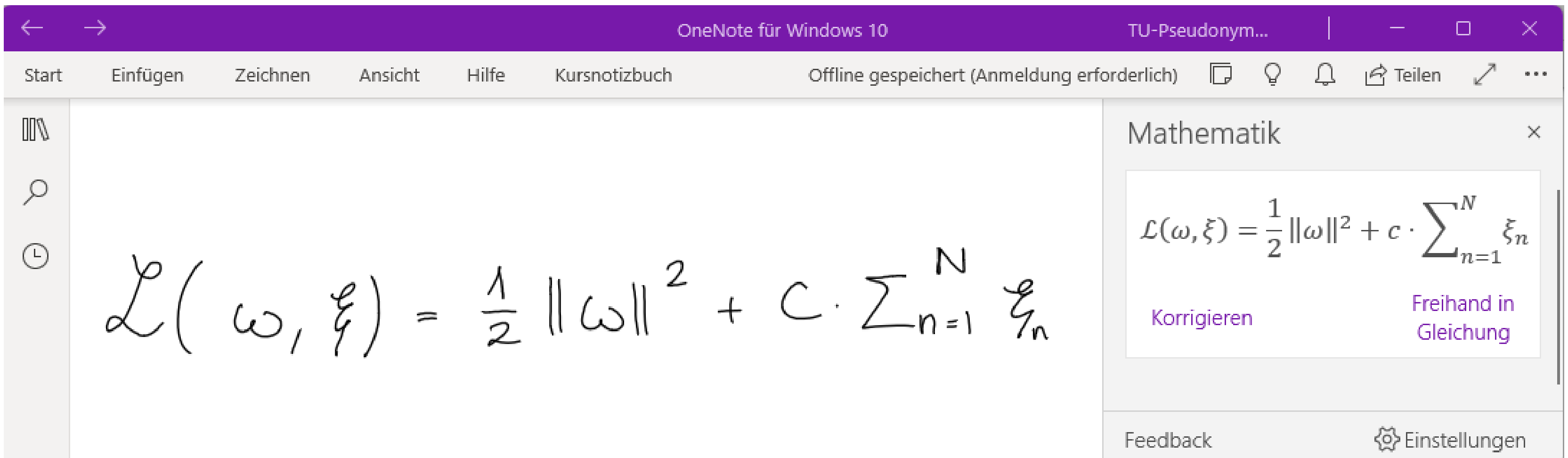
$h: \mathcal{X} \to \mathcal{Y}$ maps inputs to outputs where

- Regression: $\mathcal{Y} = \mathbb{R}$
- Classification: $\mathcal{Y} = \{1, \dots, K\}$
- Multivariate regression: $\mathcal{Y} = \mathbb{R}^K$
- (Multi-class) logistic regression: $\mathcal{Y} = \Delta^K$ (Δ: simplex)
- …

Today's Lecture

Structured Output Prediction:
$h: \mathcal{X} \to \mathcal{Y}$ maps inputs $x \in \mathcal{X}$ to **structured** outputs $y \in \mathcal{Y}$

Example:
Handwritten Character Sequences

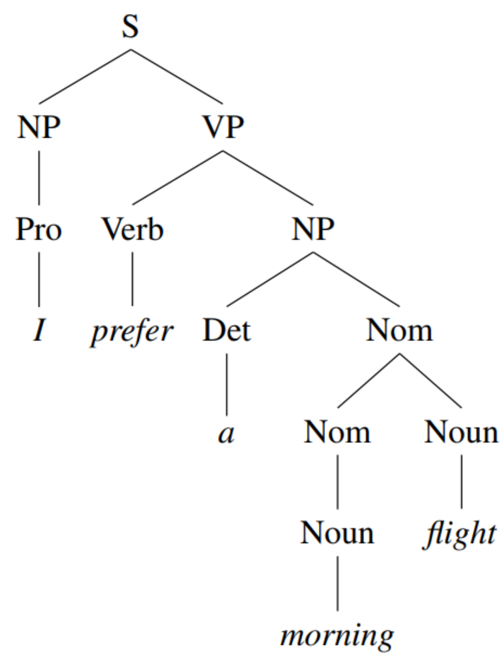


$$x \xrightarrow[h(x)]{} y$$

Example created with:
Microsoft. OneNote. www.onenote.com

Example:
Context Free Grammar Parsing

"I prefer a morning flight"

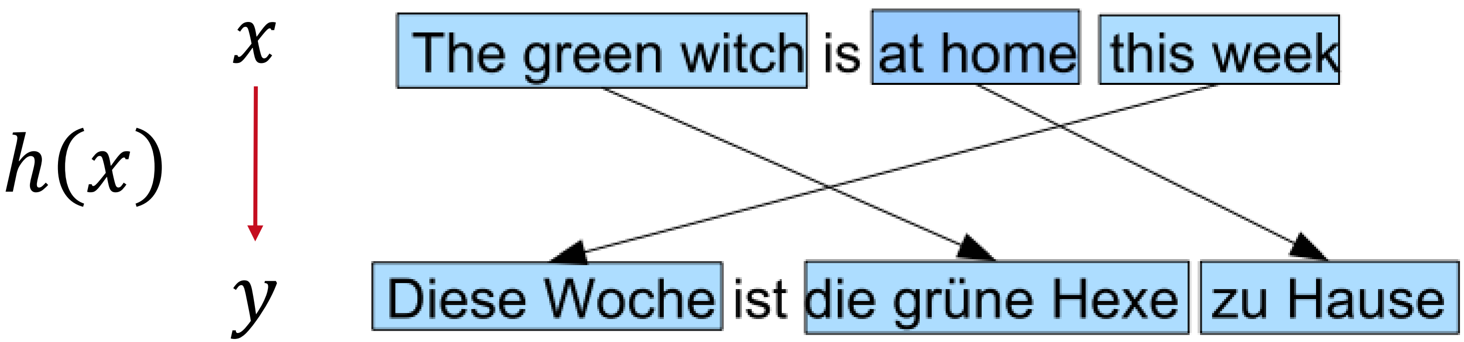


Grammar Rules		
S	→	*NP VP*
NP	→	*Pronoun*
	|	*Proper-Noun*
	|	*Det Nominal*
Nominal	→	*Nominal Noun*
	|	*Noun*
VP	→	*Verb*
	|	*Verb NP*
	|	*Verb NP PP*
	|	*Verb PP*
PP	→	*Preposition NP*

$$x \xrightarrow{\quad h(x) \quad} y$$

Example taken from:
D. Jurafsky, J.H. Martin. (2024) "Speech and Language Processing – An Introduction to Natural Language Processing, Computer Linguistics, and Speech Recognition" 3rd ed. Chapter 17.

Example:
Bilingual Word Alignment



Example taken from:
D. Jurafsky, J.H. Martin. (2024) “Speech and Language Processing – An Introduction to Natural Language Processing, Computer Linguistics, and Speech Recognition” 3rd ed. Chapter 13.

Isis

Predicting Structured Objects with Support Vector Machines

Thorsten Joachims
Dept. of Computer Science
Cornell University
Ithaca, NY 14853, USA
tj@cs.cornell.edu

Thomas Hofmann
Google Inc.
Brandschenkestrasse 110
8002 Zürich, Switzerland
thofmann@google.com

Yisong Yue
Dept. of Computer Science
Cornell University
Ithaca, NY 14853, USA
yyue@cs.cornell.edu

Chun-Nam Yu
Dept. of Computer Science
Cornell University
Ithaca, NY 14853, USA
cnyu@cs.cornell.edu

ABSTRACT

Machine Learning today offers a broad repertoire of methods for classification and regression. But what if we need to predict complex objects like trees, orderings, or alignments? Such problems arise naturally in natural language processing, search engines, and bioinformatics. The following explores a generalization of Support Vector Machines (SVMs) for such complex prediction problems.

Structured Output Prediction = Multi-Class Learning ?

Similarities to multi-class learning:

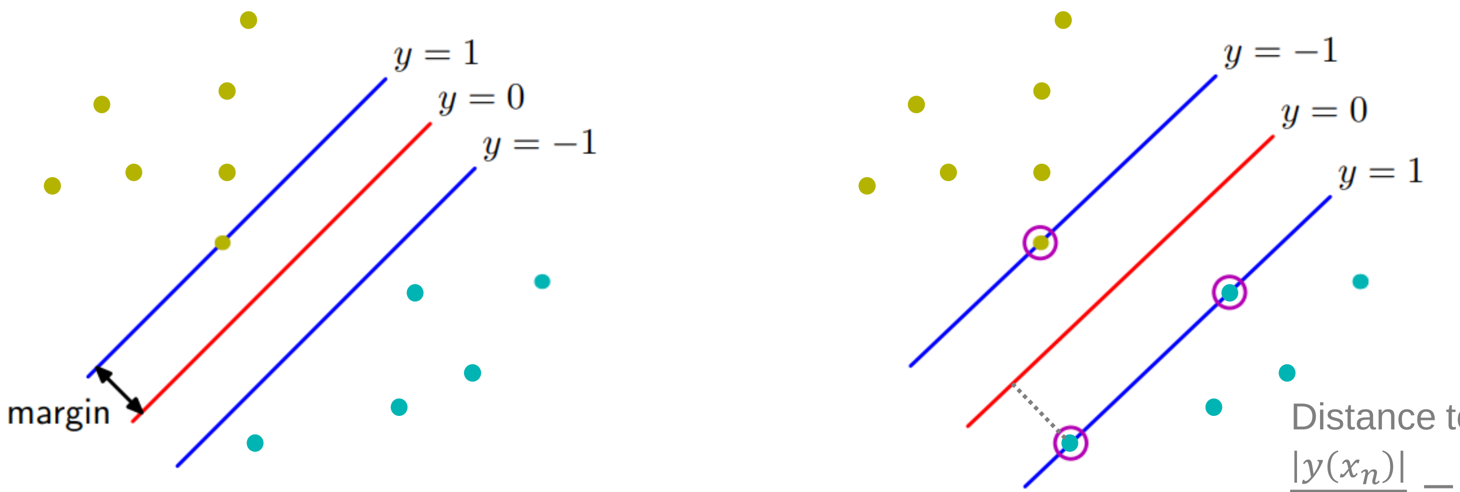
✓ Every possible structured output $y \in \mathcal{Y}$ corresponds to one “class”

✓ Predicting the output corresponds to finding the best fitting “class”

$\Rightarrow$ Multi-Class Support Vector Machines

Recap
SVMs for Binary Classification

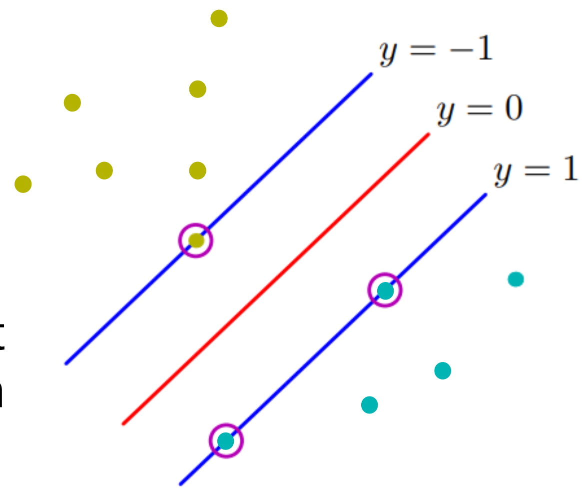
Data set $\{x_n, y_n\}_{n=1}^{N}$ with $y_n \in \{-1, +1\}$
Find $y(x_n) \coloneqq w^T \phi(x_n)$ with $y(x_n) = y_n$ such that the margin is maximized



Distance to hyperplane
$$\frac{|y(x_n)|}{||w||} = \frac{y_n \cdot w^T \phi(x_n)}{||w||}$$

Figure 7.1 The margin is defined as the perpendicular distance between the decision boundary and the closest of the data points, as shown on the left figure. Maximizing the margin leads to a particular choice of decision boundary, as shown on the right. The location of this boundary is determined by a subset of the data points, known as support vectors, which are indicated by the circles.

Recap
SVMs for Binary Classification



The objective of the maximum margin classifier is

$$\max_{w} \left\{ \frac{1}{\|w\|} \min_{n} \{ y_n \cdot w^T \phi(x_n) \} \right\}$$

Because the distance to the boundary is invariant against scaling $w \to a \cdot w$, we can fix $y_n \cdot w^T \phi(x_n) \geq 1$ and obtain

$$\min_{w} \left\{ \frac{1}{2} \|w\|^2 \right\} \quad \text{s.t.} \;\; y_n \cdot w^T \phi(x_n) \geq 1$$

Primal:

$$\mathcal{L}(w, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{n=1}^{N} \alpha_n (y_n \cdot w^T \phi(x_n) - 1)$$

Dual:

$$\tilde{\mathcal{L}}(\alpha) = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{n,m=1}^{N} \alpha_n \alpha_m y_n y_m k(x_n, x_m)$$

$$s.t. \qquad \forall n\!: \; 0 \leq \alpha_n \quad and \quad \sum_{n} \alpha_n y_n = 0$$

C. Bishop. (2006) “Pattern Recognition and Machine Learning” 8th ed. Chapter 7.

Recap
SVMs for Binary Classification

For overlapping class distributions, a perfect fit is not possible.

→ Training points are allowed to be misclassified but with a penalty that increases linearly with the distance to the decision boundary.

For every data point a slack variable $\xi_n \geq 0$ is introduced

$$\xi_n = \begin{cases} 0, & if\ y_n = y(x_n) \\ |y_n - y(x_n)|, & if\ y_n \neq y(x_n) \end{cases}$$

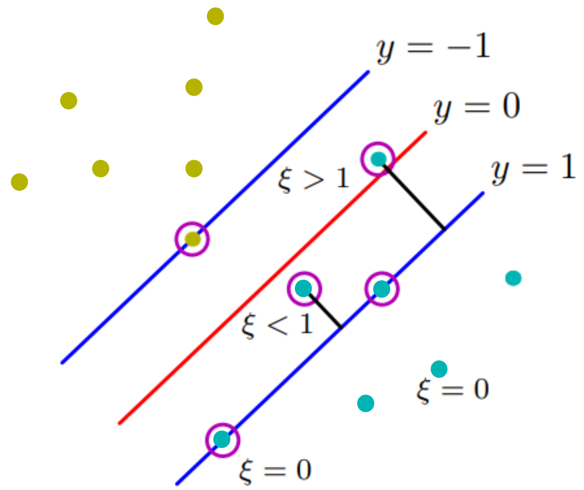


Figure 7.3 Illustration of the slack variables $\xi_n \geqslant 0$. Data points with circles around them are support vectors.

Recap
SVMs for Binary Classification

The slack variables are incorporated into the objective

$$\min_{w}\left\{\frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\xi_n\right\} \quad \text{s.t.} \quad y_n \cdot w^T\phi(x_n) \geq 1 - \xi_n\,, \qquad \xi_n \geq 0$$

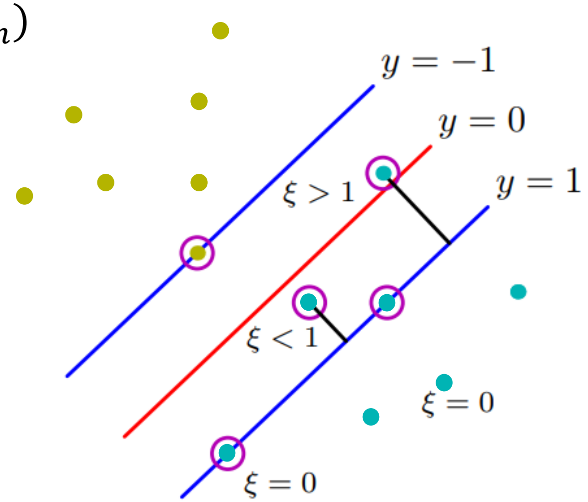
Primal:

$$\mathcal{L}(w,b,\alpha,\beta) = \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N}\alpha_n(y_n w^T\phi(x_n) - 1 + \xi_n) - \sum_{n=1}^{N}\beta_n\xi_n$$

Dual:

$$\tilde{\mathcal{L}}(\alpha) = \sum_{n=1}^{N}\alpha_n - \frac{1}{2}\sum_{n,m=1}^{N}\alpha_n\alpha_m y_n y_m k(x_n, x_m)$$

$$s.t. \qquad \forall n\colon\ 0 \leq \alpha_n \leq C \quad and \quad \sum_{n}\alpha_n y_n = 0$$



Prediction:

$$y(x) = \sum_{n=1}^{N}\alpha_n y_n k(x, x_n)$$

Multi-Class SVMs

Define one weight vector w_y for each class y such that the prediction rule $h(x)$ chooses the class with highest score

$$h(x) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, w_y^T \phi(x)$$

Enforce $w_{\bar{y}}^T \phi(x_n) < w_{y_n}^T \phi(x_n)$ for all incorrect outputs $y_n \neq \bar{y} \in \mathcal{Y} \setminus \{y_n\}$. This leads to a convex optimization problem with $N(k-1)$ linear constraints

$$\underset{w}{\operatorname{argmin}} \frac{1}{2} \|w\|^2 \ \ s.t. \quad \forall n. \forall \bar{y} \neq y_n \colon \, w_{y_n}^T \phi(x_n) - w_{\bar{y}}^T \phi(x_n) \geq 1$$

– there is no generalization across outputs

– $N(k-1)$ many constraints become infeasible to solve if the number of classes becomes large or even infinite

Structured Output Prediction $\neq$ Multi-Class Learning ?

Homework

prediction: $h(x) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, w_y^T \phi(x)$

Similarities to multi-class learning:

✓ Every possible structured output $y \in \mathcal{Y}$ corresponds to one "class"

✓ Predicting $h(x)$ corresponds to finding the correct "class"

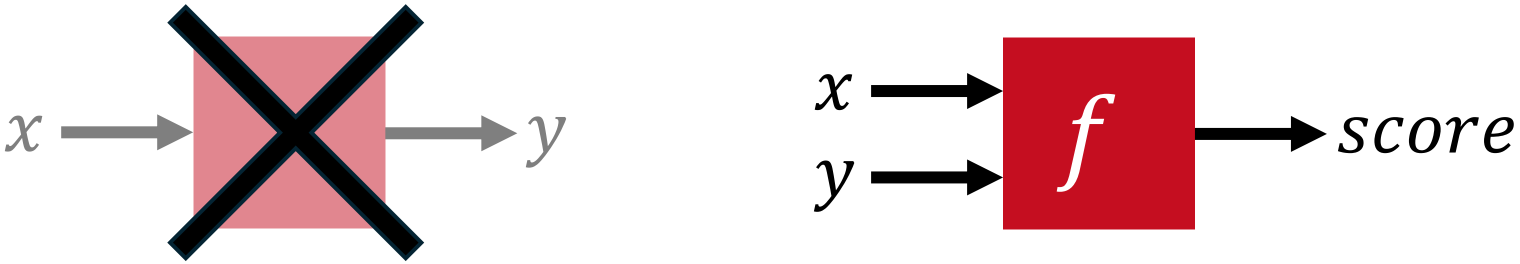
Differences make direct application of multi-class approaches impractical:

– Brute-force prediction is infeasible due to large (or infinite) number of possible "classes" $|\mathcal{Y}|$.

– Number of parameters and runtime of learning algorithm shall not increase with $|\mathcal{Y}|$.

– Structured outputs require a more refined notion of correct vs. incorrect.

Compatibility Score



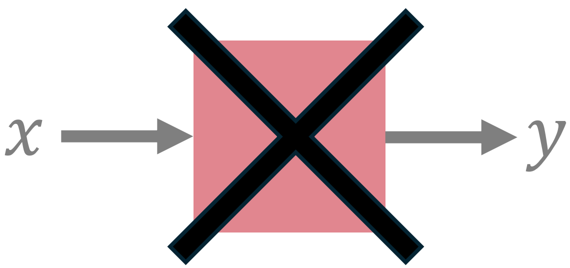
To learn a structured prediction model, we train a function that **scores the compatibility** between input x and output y:

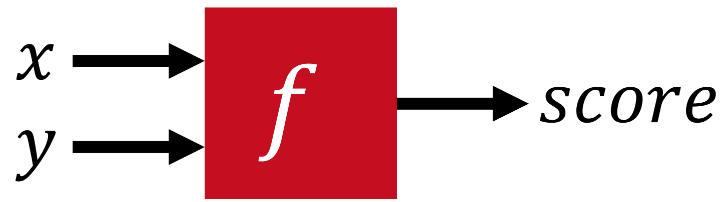
$$f: \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}.$$

The final prediction is then given by the best possible output structure according to the scoring function

$$h(x) \coloneqq \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\, f(x, y) \,.$$

Joint Feature Map





Idea:

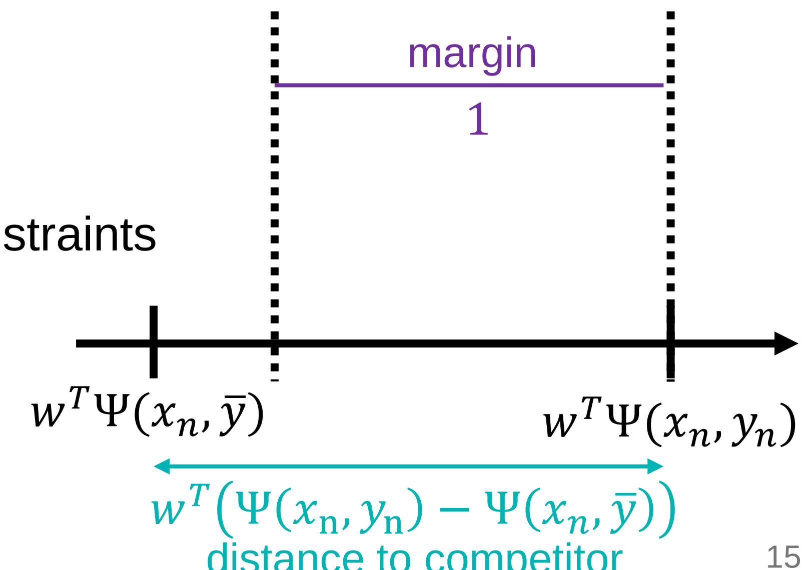
→ extract features from input-output pairs using a joint feature map $\Psi(x, y)$ instead of $\phi(x)$

→ take $f(x, y) = w^T\Psi(x, y)$

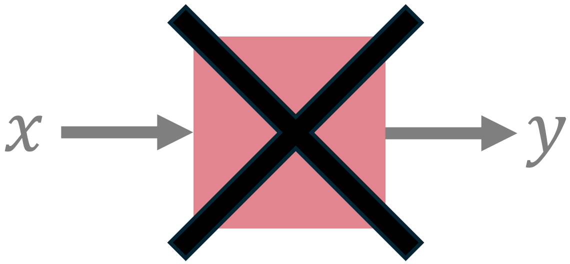
Convex optimization problem with linear constraints

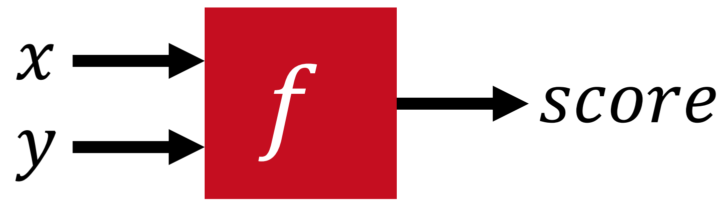
$$\min_w \left\{\frac{1}{2}\|w\|^2\right\} \quad s.t. \ \ (\forall n. \ \forall \bar{y} \neq y_n)$$

$$w^T\Psi(x_n, y_n) - w^T\Psi(x_n, \bar{y}) \geq 1$$



Joint Feature Map





$$f(x,y) = w^T \Psi(x,y)$$

$$\min_w \frac{1}{2} \|w\|^2 \quad s.t. \quad w^T \Psi(x_n, y_n) - w^T \Psi(x_n, \bar{y}) \geq 1 \quad (\forall n.\ \forall \bar{y} \neq y_n)$$

- ✓ $\Psi(x,y)$ combines properties of x and y
- ✓ design of Ψ is flexible and problem-specific
- ✓ enables generalization across outputs
- ✓ nr. of parameters $|w|$ is equal to the number of features extracted by Ψ
- – number of constraints is still $N(k-1)$

Efficient Prediction

prediction: $h(x) \coloneqq \underset{y \in \mathcal{Y}}{\operatorname{argmax}}\ w^T \Psi(x, y)$

– prediction requires brute-force exhaustive search over $\mathcal{Y}$ which is not feasible

Idea:

→ decompose $\mathcal{Y}$ into non-overlapping parts

$$\mathcal{Y} = \mathcal{Y}_1 \times \cdots \times \mathcal{Y}_m$$

→ features in Ψ do not combine properties of different parts.

✓ final output simply combines the compatibilities that were computed on each part separately

Efficient Prediction
Dynamic Programming (Viterbi)

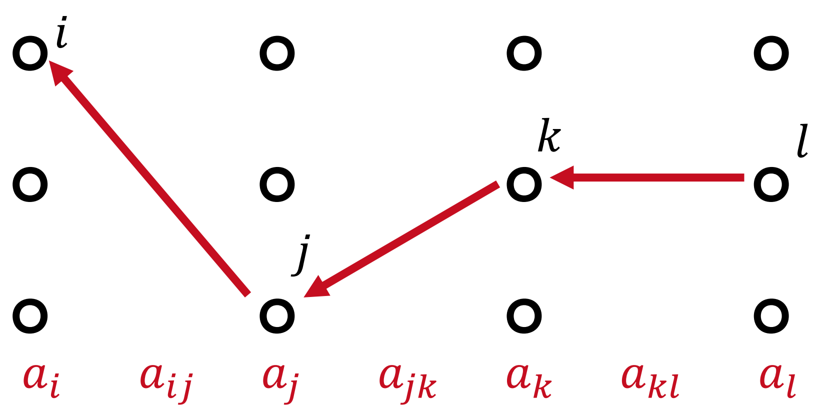
Homework

Assume the function $w^T\Psi(x,y)$ decomposes into a sum of subfunctions involving only **separate parts of $\boldsymbol{y}$**.

$$w^T\Psi(x,y) = w^T\Psi\left(x,\left(y_i,y_j,y_k,y_l\right)\right) = a_i + a_{ij} + a_j + a_{jk} + a_k + a_{kl} + a_l$$

The maximization of $w^T\Psi(x,y)$ can be done by pushing the $\max$ into the sum

$$\max_{ijkl}\left\{a_i + a_{ij} + a_j + a_{jk} + a_k + a_{kl} + a_l\right\}$$
$$= \max_i\left\{a_i + \max_j\left\{a_{ij} + a_j + \max_k\left\{a_{jk} + a_k + \max_l\{a_{kl} + a_l\}\right\}\right\}\right\}$$



Efficient Prediction
Dynamic Programming (Viterbi)

Homework

Assume the function $w^T\Psi(x,y)$ decomposes into a sum of subfunctions involving only separate parts of y.

$$w^T\Psi(x,y) = w^T\Psi\left(x,\left(y_i,y_j,y_k,y_l\right)\right) = a_i + a_{ij} + a_j + a_{jk} + a_k + a_{kl} + a_l$$

$$\max_i\left\{a_i + \max_j\left\{a_{ij} + a_j + \max_k\left\{a_{jk} + a_k + \max_l\{a_{kl} + a_l\}\right\}\right\}\right\}$$

Examples:

→ Sum form $\Psi(x,y) = \Psi_{ij}(x,y) + \Psi_{jk}(x,y) + \Psi_{kl}(x,y)$

→ Concatenation form $\Psi(x,y) = \left[\underbrace{\Psi_{ij}(x,y),}_{a_i,a_{ij}}\quad \underbrace{\Psi_{jk}(x,y),}_{a_j,a_{jk}}\quad \underbrace{\Psi_{kl}(x,y)}_{a_k,a_{kl},a_l}\right]$

- ✓ Linear instead of exponential complexity.
- ✓ The maximizing element can be recovered by backtracking $\max$.
- ✓ Can be done for Ψ that are of sum or concatenation form.

Soft-Margin Structural SVM

$$\min_w \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N} \xi_n$$

$$s.t. \quad w^T\Psi(x_n, y_n) - w^T\Psi(x_n, \bar{y}) \geq 1 - \xi_{n\bar{y}}, \quad \xi_{n\bar{y}} \geq 0 \quad (\forall n.\ \forall \bar{y} \neq y_n)$$

A loss function $\ell: \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ enables us to quantify the mismatch between $\bar{y}$ and y_n (incorrect predictions can vary in quality)

✓ design of ℓ is flexible and problem-specific

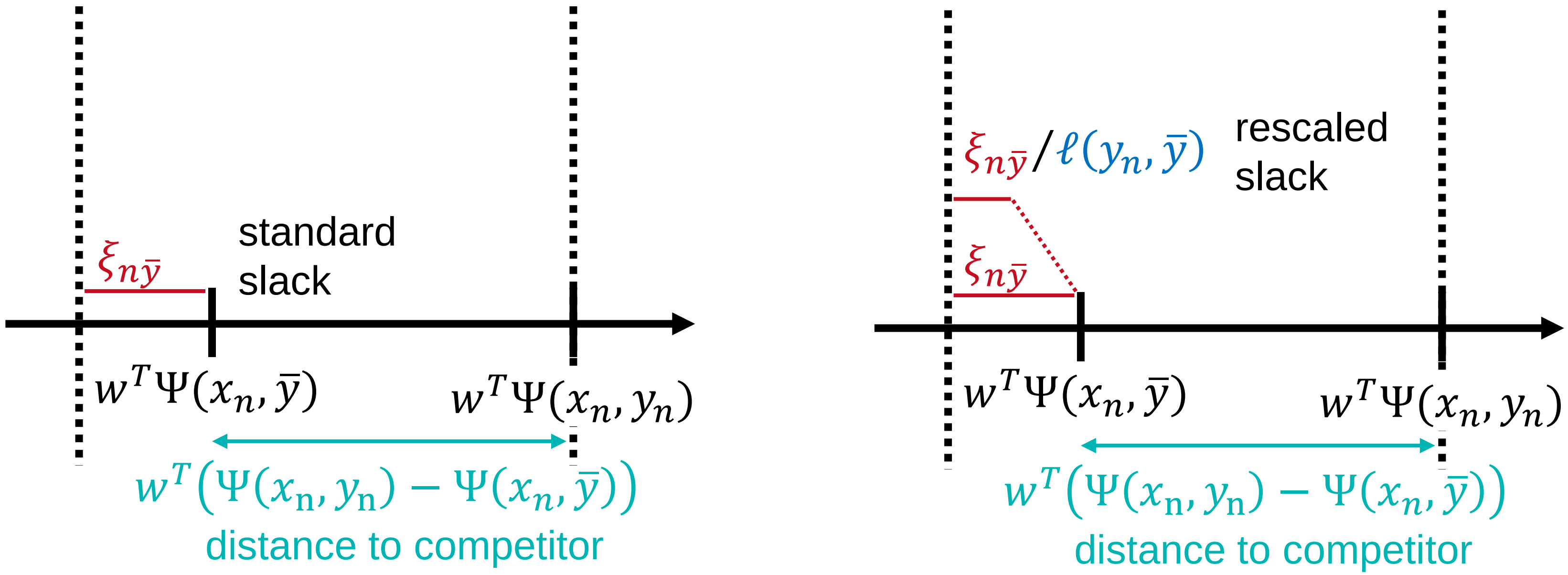
Idea: penalize violations of margin constraint, $\xi_{n\bar{y}} > 0$, more heavily if this mismatch $\bar{y}$ has a high loss $\ell(y_n, \bar{y})$

$$1/_N \sum_{n=1}^{N} \max_{\bar{y}\in\mathcal{Y}} \{\ell(y_n, \bar{y}) \cdot \xi_{n\bar{y}}\}$$

→ Slack rescaling

→ Margin rescaling

Slack Rescaling

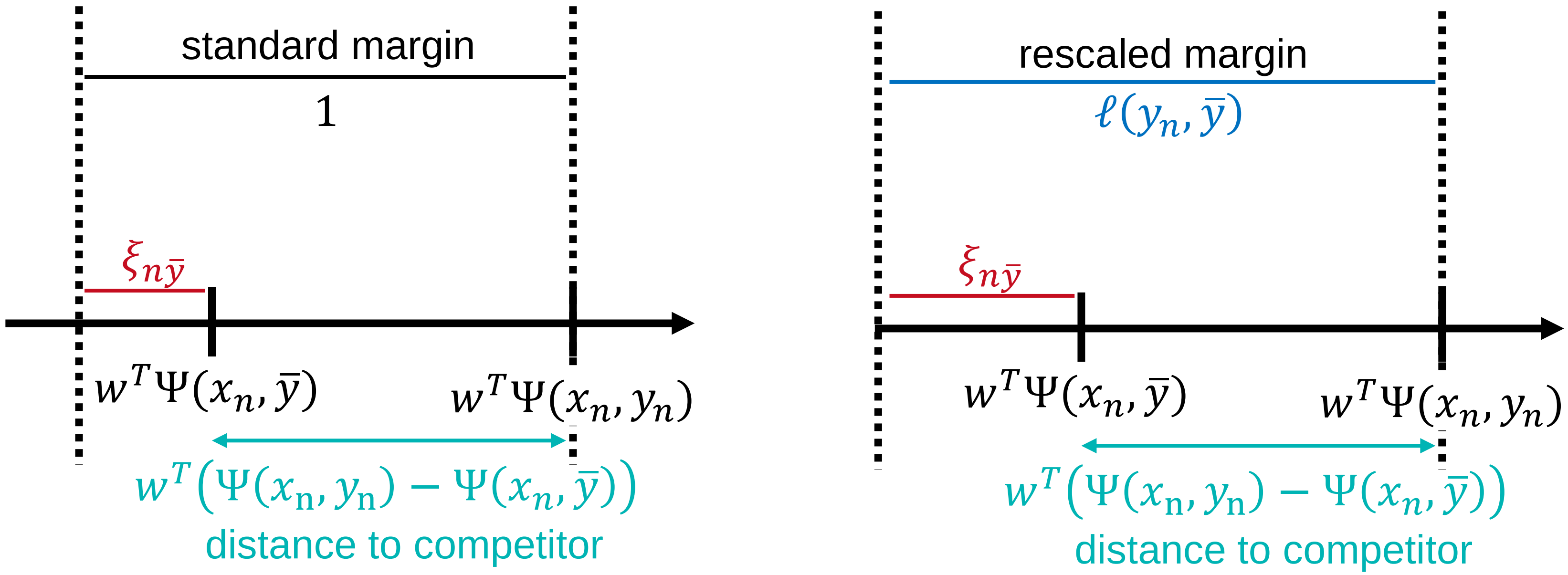


$$\underset{w}{\operatorname{argmin}} \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\xi_n$$

$$s.t. \quad w^T\Psi(x_n,y_n) - w^T\Psi(x_n,\bar{y}) \geq 1 - \frac{\xi_{n\bar{y}}}{\ell(y_n,\bar{y})},$$

$$\xi_{n\bar{y}} \geq 0 \quad (\forall n.\ \forall \bar{y} \neq y_n)$$

Margin Rescaling



$$\underset{w}{\operatorname{argmin}} \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N} \xi_n$$

$$s.t. \quad w^T\Psi(x_n, y_n) - w^T\Psi(x_n, \bar{y}) \geq \ell(y_n, \bar{y}) - \xi_{n\bar{y}},$$

$$\xi_{n\bar{y}} \geq 0 \quad (\forall n.\ \forall \bar{y} \neq y_n)$$

Dual Formulation

Homework

Primal:

$$\mathcal{L}(w,\alpha,\beta) = \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\sum_{\bar{y}\neq y_n}\xi_{n\bar{y}} - \sum_{n=1}^{N}\sum_{\bar{y}\neq y_n}\left(\beta_{n\bar{y}}\cdot\xi_{n\bar{y}}\right) - \sum_{n=1}^{N}\sum_{\bar{y}\neq y_n}\alpha_{n\bar{y}}\left(w^T\left(\Psi(x_n,y_n)-\Psi(x_n,\bar{y})\right)-1+\xi_{n\bar{y}}\right)$$

Dual:

$$\tilde{\mathcal{L}}(\alpha) = \sum_{n=1}^{N}\sum_{\bar{y}\neq y_n}\alpha_{n\bar{y}} - \frac{1}{2}\sum_{n,n'}^{N}\sum_{\substack{\bar{y}\neq y_n\\ \bar{y}'\neq y_{n'}}}\alpha_{n\bar{y}}\alpha_{n'\bar{y}'}\cdot\underbrace{\left(\Psi(x_n,y_n)-\Psi(x_n,\bar{y})\right)^T\left(\Psi(x_{n'},y_{n'})-\Psi(x_{n'},\bar{y}')\right)}_{k\left((x_n,y_n),(x_{n'},y_{n'})\right)}$$

$s.t. \quad \forall n.\forall\bar{y}\neq y_n\colon\ 0\leq\alpha_{n\bar{y}} \quad and \quad \sum_n\sum_{\bar{y}\neq y_n}\alpha_{n\bar{y}}\leq C$

Kernel:

$k\colon(\mathcal{X}\times\mathcal{Y})\times(\mathcal{X}\times\mathcal{Y})\to\mathbb{R}$ such that

$$k\left((x_n,y_n),(x_{n'},y_{n'})\right) = \langle\Psi(x_n,y_n)-\Psi(x_n,\bar{y}),\Psi(x_{n'},y_{n'})-\Psi(x_{n'},\bar{y}')\rangle$$

→ design of k is flexible and problem-specific

Constructing Kernels

Techniques for Constructing New Kernels.

Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following new kernels will also be valid:

$$\begin{aligned}
k(\mathbf{x}, \mathbf{x}') &= ck_1(\mathbf{x}, \mathbf{x}') && (6.13)\\
k(\mathbf{x}, \mathbf{x}') &= f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}') && (6.14)\\
k(\mathbf{x}, \mathbf{x}') &= q\left(k_1(\mathbf{x}, \mathbf{x}')\right) && (6.15)\\
k(\mathbf{x}, \mathbf{x}') &= \exp\left(k_1(\mathbf{x}, \mathbf{x}')\right) && (6.16)\\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') && (6.17)\\
k(\mathbf{x}, \mathbf{x}') &= k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}') && (6.18)\\
k(\mathbf{x}, \mathbf{x}') &= k_3\left(\boldsymbol{\phi}(\mathbf{x}), \boldsymbol{\phi}(\mathbf{x}')\right) && (6.19)\\
k(\mathbf{x}, \mathbf{x}') &= \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}' && (6.20)\\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b) && (6.21)\\
k(\mathbf{x}, \mathbf{x}') &= k_a(\mathbf{x}_a, \mathbf{x}'_a)k_b(\mathbf{x}_b, \mathbf{x}'_b) && (6.22)
\end{aligned}$$

where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\boldsymbol{\phi}(\mathbf{x})$ is a function from $\mathbf{x}$ to $\mathbb{R}^M$, $k_3(\cdot, \cdot)$ is a valid kernel in $\mathbb{R}^M$, $\mathbf{A}$ is a symmetric positive semidefinite matrix, $\mathbf{x}_a$ and $\mathbf{x}_b$ are variables (not necessarily disjoint) with $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, and k_a and k_b are valid kernel functions over their respective spaces.

Training Structural SVMs

– still $N(k-1)$ many constraints

Idea:

→ find an ϵ-approximate solution by considering only a subset of most violated constraints

→ subset of constraints $\mathcal{W}$ is equivalent to full set of constraints up to a precision ϵ

Training Structural SVMs

Algorithm 1 for training structural SVMs (margin-rescaling).

Input: $S = ((\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n))$, C, ϵ
$\mathcal{W} \leftarrow \emptyset$, $\boldsymbol{w} = \mathbf{0}$, $\xi_i \leftarrow 0$ for all $i = 1, ..., n$
repeat
    for i=1,...,n **do**
        $\hat{\mathbf{y}} \leftarrow \text{argmax}_{\hat{\mathbf{y}} \in \mathcal{Y}} \{\Delta(\mathbf{y}_i, \hat{\mathbf{y}}) + \boldsymbol{w} \cdot \Psi(\mathbf{x}_i, \hat{\mathbf{y}})\}$
        if $\boldsymbol{w} \cdot [\Psi(\mathbf{x}_i, \mathbf{y}_i) - \Psi(\mathbf{x}_i, \hat{\mathbf{y}})] < \Delta(\mathbf{y}_i, \hat{\mathbf{y}}) - \xi_i - \epsilon$ **then**
            $\mathcal{W} \leftarrow \mathcal{W} \cup \{\boldsymbol{w} \cdot [\Psi(\mathbf{x}_i, \mathbf{y}_i) - \Psi(\mathbf{x}_i, \hat{\mathbf{y}})] \geq \Delta(\mathbf{y}_i, \hat{\mathbf{y}}) - \xi_i\}$
            $(\boldsymbol{w}, \boldsymbol{\xi}) \leftarrow \underset{\boldsymbol{w}, \boldsymbol{\xi} \geq \mathbf{0}}{\text{argmin}} \frac{1}{2} \boldsymbol{w} \cdot \boldsymbol{w} + \frac{C}{n} \sum_{i=1}^{n} \xi_i$ s.t. $\mathcal{W}$
        end if
    end for
until $\mathcal{W}$ has not changed during iteration
return$(\boldsymbol{w}, \boldsymbol{\xi})$

Summary

- ✓ Structured output prediction: $h\colon \mathcal{X} \to \mathcal{Y}$ maps inputs $x \in \mathcal{X}$ to **structured** outputs $y \in \mathcal{Y}$
- ✓ Multi-class approaches cannot be directly applied for structured output prediction because of large number of possible output structures $|\mathcal{Y}|$
- ✓ Structural SVMs are a generalization of multi-class SVMs
- ✓ Instead of mapping x to y directly, we define a compatibility score $w^T\Psi(x, y)$ between x and y and maximize this score
- ✓ The dual problem enables the use of kernels $k\colon (\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \times \mathcal{Y}) \to \mathbb{R}$
- ✓ For efficient prediction we must assume that the structured output space can be decomposed into non-overlapping parts so that we can apply DP with linear runtime
- ✓ Loss ℓ let's us enumerate the mismatch between different outputs
- ✓ Structural SVMs can be trained in polynomial time but finds only an ϵ-accurate solution