Advanced Machine Learning
Lecture 21

Structured Output Learning II

23.01.2013

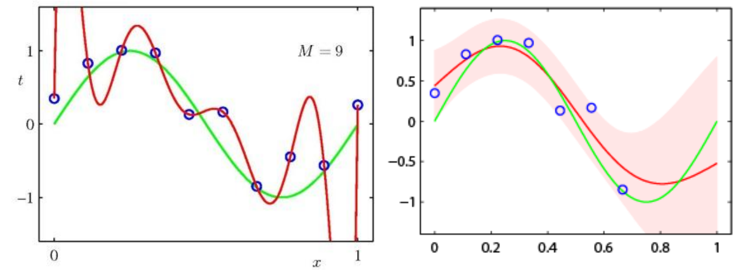
Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de/

leibe@vision.rwth-aachen.de

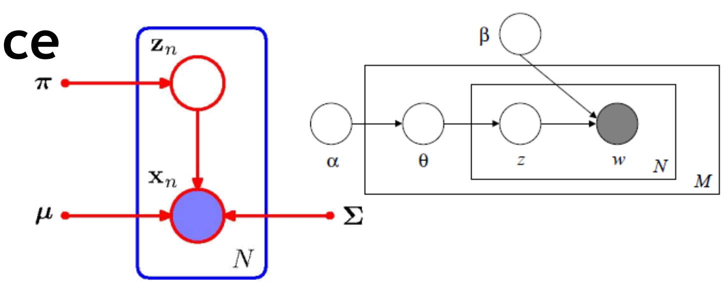
This Lecture: *Advanced Machine Learning*

- Regression Approaches
  - Linear Regression
  - Regularization (Ridge, Lasso)
  - Kernels (Kernel Ridge Regression)
  - Gaussian Processes

$$f : \mathcal{X} \rightarrow \mathbb{R}$$



- Bayesian Estimation & Bayesian Non-Parametrics
  - Prob. Distributions, Approx. Inference
  - Mixture Models & EM
  - Dirichlet Processes
  - Latent Factor Models
  - Beta Processes



- SVMs and Structured Output Learning
  - SVMs, SVDD, SV Regression
  - Structured Output Learning

$$f : \mathcal{X} \rightarrow \mathcal{Y}$$

Topics of This Lecture

- Recap: Structured Output Learning
  - General structured prediction
  - Structured Output SVM
  - Cutting plane training
  - Limitations
  - One-slack formulation
- Application: Multi-class SVMs
  - Crammer-Singer formulation
- Kernels in S-SVMs
  - Joint kernel function
  - Kernelized S-SVM
  - Application examples

Recap: Grand Unified View

Predict structured output by maximization

$$\mathbf{y} = \arg\max_{\mathbf{y}\in\mathcal{Y}} F(\mathbf{x},\mathbf{y})$$

of a compatibility function

$$F(\mathbf{x},\mathbf{y}) = \langle \mathbf{w}, \phi(\mathbf{x},\mathbf{y}) \rangle$$

that is linear in a parameter vector $\mathbf{w}$.

Recap: Generic Structured Prediction

- A generic structured prediction problem
  - $\mathcal{X}$: arbitrary input domain
  - $\mathcal{Y}$: structured output domain, decompose $\mathbf{y} = (y_1, \ldots, y_K)$
  - Prediction function $f : \mathcal{X} \to \mathcal{Y}$ given by

$$f(\mathbf{x}) = \arg\max_{\mathbf{y} \in \mathcal{Y}} F(\mathbf{x}, \mathbf{y})$$

  - Compatibility function (or negative of "energy")

$$\begin{aligned} F(\mathbf{x}, \mathbf{y}) &= \langle \mathbf{w}, \phi(\mathbf{x}, \mathbf{y}) \rangle \\ &= \sum_{i=1}^{K} \mathbf{w}_i^\top \phi_i(y_i, \mathbf{x}) && \text{unary terms} \\ &+ \sum_{i,j=1}^{K} \mathbf{w}_{ij}^\top \phi_{ij}(y_i, y_j, \mathbf{x}) && \text{binary terms} \\ &+ \ldots && \text{higher-order terms} \end{aligned}$$

Recap: Learning in Structured Models

- **Problem statement**
  - Given: parametric model (family): $F(\mathbf{x},\mathbf{y}) = \langle \mathbf{w}, \phi(\mathbf{x},\mathbf{y}) \rangle$
    prediction method: $f(\mathbf{x}) = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} F(\mathbf{x},\mathbf{y})$
    training example pairs $\{(\mathbf{x}_1,\mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\} \subset \mathcal{X} \times \mathcal{Y}$.
  - Goal: determine „good" parameter vector $\mathbf{w}$.

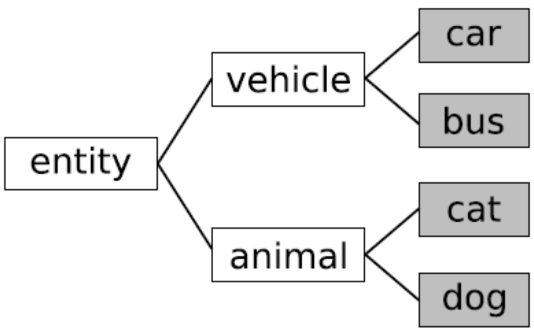
- **What make a solution "good"?**
  - Define a loss function

$$\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$$

such that $\Delta(\mathbf{y}, \mathbf{y}')$ measures the loss/cost incurred by predicting $\mathbf{y}'$ when $\mathbf{y}$ is correct.

Recap: Popular Structured Loss Functions

- **Zero-one loss**
  - Definition: $\Delta(\mathbf{y},\mathbf{y}') = \delta(\mathbf{y} \neq \mathbf{y}')$
  - *"Every prediction that is not identical to the intended one is considered a mistake, and all mistakes are penalized equally."*
  - Most common loss for multi-class problems.
  - Less frequently used for structured prediction tasks.



- **Hierarchical multi-class loss**
  - Definition: $\Delta(\mathbf{y},\mathbf{y}') = \frac{1}{2} dist_H(\mathbf{y},\mathbf{y}')$

    where H is a hierarchy over the classes in $\mathcal{Y}$ and $dist_H(\mathbf{y},\mathbf{y}')$ measures the distance of $\mathbf{y}$ and $\mathbf{y}'$.
  - Common way to incorporate information about label hierarchies in multi-class prediction problems

Image from [Nowozin & Lampert, FTCGV'11]

Recap: Popular Structured Loss Functions

- **Hamming loss**
  - Definition: $\Delta(\mathbf{y}, \mathbf{y}') = \frac{1}{m} \sum_{i=1}^{m} \delta(\mathbf{y}_i \neq \mathbf{y}'_i)$
  - Frequently used loss for image segmentation and other tasks in which the output $\mathbf{y}$ consists of multiple part labels $\mathbf{y}_1, \ldots, \mathbf{y}_m$.
  - Each part label is judged independently and the average number of labeling errors is determined.

- **Area overlap loss**
  - Definition: $\Delta(\mathbf{y}, \mathbf{y}') = 1 - \frac{area(\mathbf{y} \cap \mathbf{y}')}{area(\mathbf{y} \cup \mathbf{y}')}$
  - Standard loss in object localization, e.g., the PASCAL VOC detection challenges.
  - $\mathbf{y}$ and $\mathbf{y}'$ are bounding box coordinates, and $\mathbf{y} \cap \mathbf{y}'$ and $\mathbf{y} \cup \mathbf{y}'$ are their intersection and union, respectively.

Recap: Structured Output SVM

- **Slack formulation** of S-SVM
  - Solve
$$\min_{\mathbf{w}\in\mathbb{R}^D,\,\xi_n\in\mathbb{R}^+} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{N}\sum_{n=1}^{N}\xi_n$$
    subject to
$$\langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_n, \mathbf{y}_n)\rangle \;\geq\; \Delta(\mathbf{y}_n, \mathbf{y}) + \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}_n, \mathbf{y})\rangle - \xi_n$$
$$\text{for all } \mathbf{y} \in \mathcal{Y} \setminus \{\mathbf{y}_n\}$$
- Optimization problem very similar to normal SVM
  - Quadratic in $\mathbf{w}$, linear in ξ.
  - Constraints linear in $\mathbf{w}$ and ξ.
  - **Convex!**
- But there are $N(|\mathcal{Y}| - 1)$ constraints!

  $\Rightarrow$ Numeric optimization needs some tricks, will be expensive.

Recap: Solving S-SVM Training

- Solving the S-SVM optimization
  - There are $N(|\mathcal{Y}| - 1)$ constraints!
  - But: Weight vector has only D degrees of freedom.
    Slack variables have only N degrees of freedom.

  $\Rightarrow$ $D+N$ constraints suffice to determine the optimal solution.
  - If we knew the set of relevant constraints in advance, we could solve the optimization efficiently.

  $\Rightarrow$ Approximate the solution iteratively.

- Cutting Plane training
  - Delayed constraint generation technique
  - Search for the best weight vector and the set of active constraints simultaneously in an iterative manner.
  - Approximate solution with much faster runtime.

Recap: Cutting Plane Training

- **Cutting Plane algorithm**
  1. Start from an empty working set.
  2. In each iteration, solve the optimization problem for $(\mathbf{w}^*, \xi^*)$ with only the constraints in the working set.
  3. Check for each sample if any of the $|\mathcal{Y}|$ constraints are violated.
  4. If not, we have found the optimal solution.
  5. Otherwise, add most violated constraints to the working set.

- **Speed-ups**
  - To achieve faster convergence, choose a tolerance $\epsilon > 0$ and require a constraint to be violated by at least ϵ.

  $\Rightarrow$ Possible to prove convergence after $\mathcal{O}(\frac{1}{\epsilon^2})$ steps with the guarantee that objective value at the solution differs only at most by ϵ from the global minimum.

Cutting Plane Training: Limitations

- Cutting plane training
  - Attractive, since it allows us to reuse existing components:
  - Ordinary SVM solvers
  - Algorithms for (loss-adapted) MAP prediction

- However...
  - Convergence rate can be unsatisfactory, in particular for large values of C.
  - Convergence after $\mathcal{O}(\frac{1}{\epsilon^2})$ steps means: for a value of $\epsilon = 0.1$, we already need on the order of 100 steps...
  - This can be improved to $\mathcal{O}(\frac{1}{\epsilon})$ with the recently introduced one-slack formulation.

Back to S-SVMs

- **One-slack S-SVM formulation**
  - Solve $$(\mathbf{w}^*, \xi^*) = \operatorname*{argmin}_{\mathbf{w}\in\mathbb{R}^D,\, \xi\in\mathbb{R}_+} \frac{1}{2}\|\mathbf{w}\|^2 + C\xi$$

    subject to $\forall(\bar{\mathbf{y}}_1, ..., \bar{\mathbf{y}}_N) \in \mathcal{Y}^N$:
    $$\sum_{n=1}^{N} [\Delta(\mathbf{y}_n, \bar{\mathbf{y}}_n) + \langle \mathbf{w}, \phi(\mathbf{x}_n, \bar{\mathbf{y}}_n)\rangle - \langle \mathbf{w}, \phi(\mathbf{x}_n, \mathbf{y}_n)\rangle] \leq N\xi$$

- **Equivalent to n-Slack S-SVM formulation**
  - But only one common slack variable ξ.
  - We now have $|\mathcal{Y}|^N$ constraints, so even more than with n-slack.
  - However, cutting-plane optimization now achieves a solution ϵ-close to the optimum in $\mathcal{O}(\frac{1}{\epsilon})$ steps.

  $\Rightarrow$ *Significant reduction in training time for practical problems.*

Topics of This Lecture

- Recap: Structured Output Learning
  - General structured prediction
  - Structured Output SVM
  - Cutting plane training
  - Limitations
  - One-slack formulation
- Application: Multi-class SVMs
  - Crammer-Singer formulation
- Kernels in S-SVMs
  - Joint kernel function
  - Kernelized S-SVM
  - Application examples

Example: Crammer-Singer Multiclass SVM

- **Procedure**
  - **Define the joint feature space**

$$\mathcal{Y} = \{1, 2, \ldots, K\}, \quad \Delta(y, y') = \begin{cases} 1 & \text{for } y \neq y' \\ 0 & \text{otherwise} \end{cases}.$$

$$\phi(x, y) = \Big([\![y = 1]\!]\phi(x),\ [\![y = 2]\!]\phi(x),\ \ldots,\ [\![y = K]\!]\phi(x) \Big)$$

  - **Solve** $$\min_{w,\xi} \frac{1}{2}\|w\|^2 + \frac{C}{N}\sum_{n=1}^{N} \xi^n$$

subject to , for $n = 1, \ldots, N$**,**

$$\langle w, \phi(x^n, y^n)\rangle - \langle w, \phi(x^n, y)\rangle \geq 1 - \xi^n \qquad \text{for all } y \in \mathcal{Y} \setminus \{y^n\}$$

  - **Classification:** $f(x) = \operatorname{argmax}_{y \in \mathcal{Y}} \ \langle w, \phi(x, y)\rangle$

Topics of This Lecture

- Recap: Structured Output Learning
  - General structured prediction
  - Structured Output SVM
  - Cutting plane training
  - Limitations
  - One-slack formulation
- Application: Multi-class SVMs
  - Crammer-Singer formulation
- **Kernels in S-SVMs**
  - **Joint kernel function**
  - **Kernelized S-SVM**
  - **Application examples**

Kernels in S-SVMs

- Joint kernel function
  - The S-SVM formulation is based on a joint feature map $\phi(\mathbf{x},\mathbf{y})$., i.e., on pairs of (*input*, *output*).
  - We can now also define a joint kernel function for such mappings k: $(\mathcal{X} \times \mathcal{Y}) \times (\mathcal{X} \times \mathcal{Y}) \rightarrow \mathbb{R}$ as follows

$$k\left((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')\right) = \langle \phi(\mathbf{x},\mathbf{y}), \phi(\mathbf{x}',\mathbf{y}') \rangle$$

  - k measures similarities between (*input*, *output*) pairs.
- Same advantages as for regular SVMs
  - One does not need an explicit expression for the feature map ϕ.
  - It suffices if we can evaluate the kernel function for arbitrary arguments.

  $\Rightarrow$ Specifically advantageous if the feature map is very high-dimensional.

Joint Kernel Functions

- What do joint kernel functions look like?

$$k\left((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')\right) = \langle \phi(\mathbf{x},\mathbf{y}), \phi(\mathbf{x}',\mathbf{y}') \rangle$$

  - As in graphical models: easier if ϕ decomposes w.r.t. factors

$$\phi(\mathbf{x},\mathbf{y}) = \left(\phi_F(\mathbf{x},\mathbf{y}_F)\right)_{F\in\mathcal{F}}$$

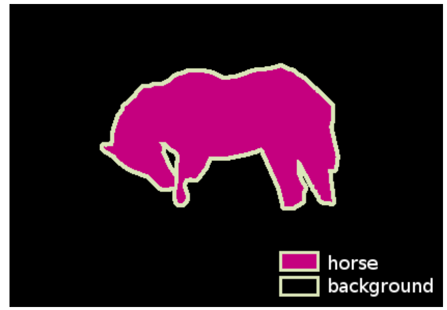
  - Then the kernel k decomposes into a sum over factors

$$\begin{aligned} k\left((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}')\right) &= \langle (\phi_F(\mathbf{x},\mathbf{y}_F))_{F\in\mathcal{F}}, (\phi_F(\mathbf{x}',\mathbf{y}'_F))_{F\in\mathcal{F}} \rangle \\ &= \sum_{F\in\mathcal{F}} \langle (\phi_F(\mathbf{x},\mathbf{y}_F)), (\phi_F(\mathbf{x}',\mathbf{y}'_F)) \rangle \\ &= \sum_{F\in\mathcal{F}} k_F((\mathbf{x},\mathbf{y}_F),((\mathbf{x}',\mathbf{y}'_F)) \end{aligned}$$

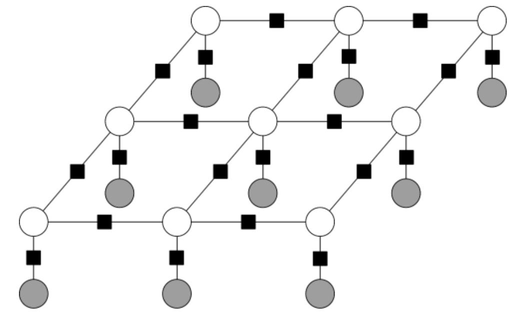
$\Rightarrow$ We can define kernels for each object type.

Example: Figure-Ground Segmentation

- Task with a grid structure

$(\mathbf{x}, \mathbf{y}) = ($  , 

$)$



- Typical kernels: arbitrary in $\mathbf{x}$, linear w.r.t. $\mathbf{y}$:
  - Unary factors

$$k_p\left((x_p, y_p), (x'_p, y'_p)\right) = k(x_p, x'_p)\delta(y_p = y'_p)$$

with $k(x_p, x'_p)$ local image kernel, e.g. χ^2 or hist. intersection.

  - Pairwise factors

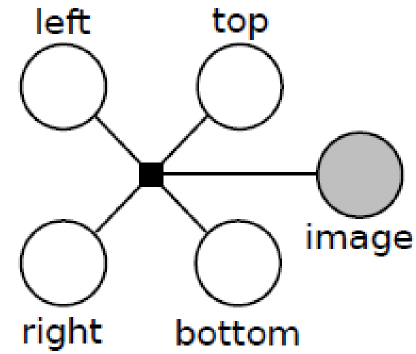
$$k_{pq}\left((y_p, y_q), (y'_p, y'_q)\right) = \delta(y_p = y'_p)\delta(y_q = y'_q)$$

  - More powerful than all-linear and argmax prediction still possible.

Example: Object Localization

- **Object detection task**

$(\mathbf{x}, \mathbf{y}) = ($  ,  $)$



  - **Only one factor that includes all $\mathbf{x}$ and $\mathbf{y}$:**

$$k\left((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')\right) = k_{image}(\mathbf{x}|_{\mathbf{y}}, \mathbf{x}'|_{\mathbf{y}'})$$

with k_{image} the image kernel and $\mathbf{x}|_{\mathbf{y}}$ is image region within box $\mathbf{y}$.

$\Rightarrow$ **argmax-prediction is as difficult here as object localization with k_{image}-SVM!**

Kernelized S-SVM

- **Dual** formulation with **kernels**
  - Solve $$\alpha^* = \arg\max_{\alpha \in \mathbb{R}_+^{N \times \mathcal{Y}}} \sum_{\substack{n=1 \\ \mathbf{y} \in \mathcal{Y}}}^{N} \alpha_{n\mathbf{y}} - \frac{1}{2} \sum_{\substack{n=1 \\ \mathbf{y}' \in \mathcal{Y}}}^{N} \sum_{\substack{n'=1 \\ \mathbf{y} \in \mathcal{Y}}}^{N} \alpha_{n\mathbf{y}} \alpha_{n'\mathbf{y}'} \overline{K}_{\mathbf{y}\mathbf{y}'}^{nn'}$$

    subject to, for $n = 1, \ldots, N$,
    $$\sum_{\mathbf{y} \in \mathcal{Y}} \alpha_{n\mathbf{y}} \leq \frac{C}{N}$$

    where $$\overline{K}_{\mathbf{y}\mathbf{y}'}^{nn'} = K_{\mathbf{y}_n \mathbf{y}'_{n'}}^{nn'} - K_{\mathbf{y}_n \mathbf{y}'}^{nn'} - K_{\mathbf{y}\mathbf{y}'_n}^{nn'} + K_{\mathbf{y}\mathbf{y}'}^{nn'}$$

    and $$K_{\mathbf{y}\mathbf{y}'}^{nn'} = k\left((\mathbf{x}_n, \mathbf{y}), (\mathbf{x}_{n'}, \mathbf{y}')\right).$$

  - Decision function
    $$f(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} \sum_{n=1}^{N} \sum_{\mathbf{y}' \in \mathcal{Y}} \alpha_{n\mathbf{y}'} k\left((\mathbf{x}_n, \mathbf{y}'), (\mathbf{x}, \mathbf{y})\right)$$

Discussion and Analysis

- Analysis
  - Prediction function

$$f(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y}\in\mathcal{Y}} \sum_{n=1}^{N} \sum_{\mathbf{y}'\in\mathcal{Y}} \alpha_{n\mathbf{y}'} k\left((\mathbf{x}_n, \mathbf{y}'), (\mathbf{x}, \mathbf{y})\right)$$

  - In principle, this function might become infeasible to compute, since it contains a potentially exponential number of summands.
  - However, this is not a problem in practice, since the constraints enforce sparsity in the coefficients.

$$\sum_{\mathbf{y}\in\mathcal{Y}} \alpha_{n\mathbf{y}} \leq \frac{C}{N}$$

$\Rightarrow$ For every $n=1,\ldots,N$, most coefficients $\alpha_{n\mathbf{y}}$ for $\mathbf{y}\in\mathcal{Y}$ will be zero.

$\Rightarrow$ Possible to keep a working set over non-zero coefficients during optimization.

Summary

- **Given**
  - Training set $\{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\} \to \mathcal{X} \times \mathcal{Y}$
  - Loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$.
- **Task:**
  - Learn parameter $\mathbf{w}$ for $f(\mathbf{x}) := \operatorname{argmax}_{\mathbf{y}} \langle \mathbf{w}, \phi(\mathbf{x}, \mathbf{y}) \rangle$ that minimizes expected loss on future data.
- **S-SVM solution derived by maximum margin framework:**
  - Enforce correct output to be better than others by a margin :

$$\langle \mathbf{w}, \phi(\mathbf{x}_n, \mathbf{y}_n) \rangle \geq \Delta(\mathbf{y}_n, \mathbf{y}) + \langle \mathbf{w}, \phi(\mathbf{x}_n, \mathbf{y}) \rangle \quad \text{for all } \mathbf{y} \in \mathcal{Y}$$

  - Convex optimization problem, but non-differentiable
  - Many equivalent formulations $\to$ different training algorithms
  - Training needs repeated argmax prediction, no probabilistic inference

References and Further Reading

- Structured SVMs were first introduced here
  - I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun, Large Margin Methods for Structured and Interdependent Output Variables, Journal of Machine Learning Research, Vol. 6, pp. 1453-1484, 2005.

- Additional details on Structured SVMs can be found in Chapter 6 of the following tutorial on Structured Learning
  - S. Nowozin, C. Lampert, Structured Learning and Prediction in Computer Vision, Foundations and Trends in Computer Graphics and Vision, Vol. 6(3-4), pp. 185-365, 2011.