This Lecture: Advanced Machine Learning

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

Recap: Grand Unified View

Predict structured output by maximization of a compatibility function

\[ y = \arg \max_{y \in Y} F(x, y) \]

that is linear in a parameter vector \( w \).

Recap: Learning in Structured Models

- Problem statement
  - Given: parametric model (family): \( F(x, y) = \langle w, \phi(x, y) \rangle \)
  - prediction method: \( f(x) = \arg \max_{y \in Y} F(x, y) \)
  - training example pairs \( \{ (x_1, y_1), \ldots, (x_n, y_n) \} \subset \mathcal{X} \times \mathcal{Y} \)
  - Goal: determine „good“ parameter vector \( w \).
- What make a solution „good“?
  - Define a loss function
    \[ \Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+ \]
    such that \( \Delta(y', y) \) measures the loss/cost incurred by predicting \( y' \) when \( y \) is correct.
Recap: Popular Structured Loss Functions

- **Zero-one loss**
  - Definition: $\Delta(y, y') = \delta(y \neq y')$
  - “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: $\Delta(y, y') = \frac{1}{2} \text{dist}_H(y, y')$
  - where $H$ is a hierarchy over the classes in $Y$ and $\text{dist}_H(y, y')$ measures the distance of $y$ and $y'$.
  - Common way to incorporate information about label hierarchies in multi-class prediction problems.

Recap: Structured Output SVM

- **Slack formulation of S-SVM**
  - Solve $\min \mathbf{w} \in \mathbb{R}^D, \xi \in \mathbb{R}^N \frac{1}{2} \|\mathbf{w}\|^2 + \frac{C}{N} \sum_{n=1}^{N} \xi_n$
  - subject to $(\mathbf{w}, \phi(x_n, y_n)) \geq \Delta(y_n, y) + (\mathbf{w}, \phi(x_n, y)) - \xi_n$
  - for all $y \in \mathcal{Y} \setminus \{y_n\}$
  - Optimization problem very similar to normal SVM
    - Quadratic in $\mathbf{w}$, linear in $\xi$.
    - Constraints linear in $\mathbf{w}$ and $\xi$.
    - Convex!
  - But there are $N|\mathcal{Y}| - 1$ constraints!
  - 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.
  - $D+N$ constraints suffice to determine the optimal solution.
  - If we knew the set of relevant constraints in advance, we could solve the optimization efficiently.
  - 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 $\epsilon$.
  - Possible to prove convergence after $O(\frac{1}{\epsilon})$ steps with the guarantee that objective value at the solution differs only at most by $\epsilon$ 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 $\epsilon$.
    - Convergence after $O(\frac{1}{\epsilon^2})$ steps means: for a value of $\epsilon = 0.1$, we already need on the order of 100 steps.
    - This can be improved to $O(\frac{1}{\epsilon^3})$ with the recently introduced one-slack formulation.
**Back to S-SVMs**

- **One-slack S-SVM formulation**
  - Solve \( \min_{w, \xi} \frac{1}{2} ||w||^2 + C \xi \)
  - subject to \( \sum_{n=1}^{N} \Delta(y_n) + \langle w, \phi(x_n, y_n) \rangle - \langle w, \phi(x_n, y_n) \rangle \leq N \xi \)
  - Equivalent to \( n \)-slack \( S \)-SVM formulation
  - But only one common slack variable \( \xi \).
  - We now have \( |Y| \) constraints, so even more than \( n \)-slack.
  - However, cutting-plane optimization now achieves a solution \( \epsilon \)-close to the optimum in \( O(n) \) steps.
  - Significant reduction in training time for practical problems.

**Example: Crammer-Singer Multiclass SVM**

- **Procedure**
  - Define the joint feature space \( \mathcal{Y} = \{1, 2, \ldots, N\} \), \( \Delta(y, y') = \begin{cases} 1 & \text{for } y \neq y' \\ 0 & \text{otherwise} \end{cases} \)
  - \( \phi(x, y) = \begin{cases} 1 & \text{if } y = 1 \phi(x) \\ 2 & \text{if } y = 2 \phi(x) \\ \vdots & \text{if } y = N \phi(x) \end{cases} \)
  - Solve \( \min_{w, \xi} \frac{1}{2} ||w||^2 + \frac{C}{N} \sum_{n=1}^{N} \Delta(y_n, y') \)
  - subject to, for \( n = 1, \ldots, N \),
  - \( \langle w, \phi(x, y) \rangle - \langle w, \phi(x', y') \rangle \geq 1 - \xi \) for all \( y \in \mathcal{Y} \setminus \{y'\} \)
  - Classification: \( f(x) = \arg\max_{y \in \mathcal{Y}} \langle w, \phi(x, y) \rangle \)

**Kernels in S-SVMs**

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

- **Same advantages as for regular SVMs**
  - One does not need an explicit expression for the feature map \( \phi \).
  - It suffices if we can evaluate the kernel function for arbitrary arguments.
  - Specifically advantageous if the feature map is very high-dimensional.

**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

- **Joint Kernel Functions**
  - What do joint kernel functions look like?
    - \( k((x, y), (x', y')) = \langle \phi(x, y), \phi(x', y') \rangle \)
    - As in graphical models: easier if \( \phi \) decomposes w.r.t. factors
    - \( \phi(x, y) = \phi_F(x, y_F) \)
    - Then the kernel \( k \) decomposes into a sum over factors
    - \( k((x, y), (x', y')) = \sum_{F \in \mathcal{F}} \langle \phi_F(x, y_F), \phi_F(x', y'_F) \rangle \)
    - \( \sum_{F \in \mathcal{F}} k_F((x, y_F), (x', y'_F)) \)
    - We can define kernels for each object type.
Example: Figure-Ground Segmentation

- Task with a grid structure
  \[(x, y) = (\text{top, right}), (\text{bottom, left})\]

- Typical kernels: arbitrary in \(x\), linear w.r.t. \(y\):
  - Unary factors
    \[k_p ((x_p, y_p), (x_p', y_p')) = k(x_p, x_p') \delta(y_p = y_p')\]
    with \(k(x, x')\) local image kernel, e.g. \(x^2\) or hist. intersection.
  - Pairwise factors
    \[k_{pq} ((y_p, y_q), (y_p', y_q')) = \delta(y_p = y_p') \delta(y_q = y_q')\]
    More powerful than all linear and argmax prediction still possible.

Example: Object Localization

- Object detection task
  \[(x, y) = (\text{left, bottom}), (\text{right, top})\]

- Only one factor that includes all \(x\) and \(y\):
  \[k ((x, y), (x', y')) = k_{\text{image}}(x, y)\]
  with \(k_{\text{image}}\) the image kernel and \(x_y\) is image region within box \(y\).

  \[\text{argmax-prediction is as difficult here as object localization with } k_{\text{image}}-\text{SVM!}\]

Kernelized S-SVM

- Dual formulation with kernels
  - Solve \(\alpha^* = \arg\max_{\alpha \in [0, 1]^N} \sum_{n=1}^{N} \alpha_n y_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{y \in Y} \alpha_n y_n K_{yy'}^{n}\)
    subject to, for \(n = 1, ..., N\),
    \[\sum_{y \in Y} \alpha_n y_n \leq C + \frac{y_n \alpha_n}{N}\]
    where \(K_{yy'} = K_{yy'}^{n} - K_{yy'}^{n'} + K_{yy}^{n'}\)
    and \(K_{yy}^{n} = k((x_n, y_n), (x_n', y_n'))\).
  - Decision function
    \[f(x) = \arg\max_{y \in Y} \sum_{n=1}^{N} \alpha_n y_n k((x_n, y_n'), (x, y))\]

Discussion and Analysis

- Analysis
  - Prediction function
    \[f(x) = \arg\max_{y \in Y} \sum_{n=1}^{N} \alpha_n y_n k((x_n, y_n'), (x, y))\]
  - 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.
  \[\sum_{y \in Y} \alpha_n y_n \leq C + \frac{y_n \alpha_n}{N}\]
  \[\Rightarrow \text{For every } n = 1, ..., N, \text{most coefficients } \alpha_n \text{ for } y \in Y \text{ will be zero.}\]
  \[\Rightarrow \text{Possible to keep a working set over non-zero coefficients during optimization.}\]

Summary

- Given
  - Training set \(\{(x_1, y_1), ..., (x_N, y_N)\} \rightarrow X \times Y\)
  - Loss function \(\Delta : X \times Y \rightarrow R\).
- Task:
  - Learn parameter \(w\) for \(f(x) := \arg\max_{y \in Y} \phi(x, y)\) that minimizes expected loss on future data.
- S-SVM solution derived by maximum margin framework:
  - Enforce \textit{correct output} to be better than others by a margin :
    \[<w, \phi(x_n, y_n), y_n> - <w, \phi(x_n, y)> \text{ for all } y \in Y\]
  - Convex optimization problem, but non-differentiable
  - Many equivalent formulations \(\Rightarrow\) different training algorithms
  - Training needs repeated \(\arg\max\) prediction, no probabilistic inference

References and Further Reading

- Structured SVMs were first introduced here
- Additional details on Structured SVMs can be found in Chapter 6 of the following tutorial on Structured Learning