CPSC 540: Machine Learning
Structured SVMs

Mark Schmidt
University of British Columbia

Winter 2019
3 main approaches to structured prediction (predicting object $y$ given features $x$):

1. **Generative models** use $p(y \mid x) \propto p(y, x)$ as in naive Bayes.
   - Turns structured prediction into density estimation.
   - But remember how hard it was just to model images of digits?
   - We have to model features and solve supervised learning problem.

2. **Discriminative models** directly fit $p(y \mid x)$ as in logistic regression (next topic).
   - View structured prediction as conditional density estimation.
   - Just focuses on modeling $y$ given $x$, not trying to model features $x$.
   - Lets you use complicated features $x$ that make the task easier.

3. **Discriminant functions** just try to map from $x$ to $y$ as in SVMs.
   - Now you don’t even need to worry about calibrated probabilities.
SVMs and Likelihood Ratios

- **Logistic regression** optimizes a likelihood of the form
  \[ p(y^i \mid x^i, w) \propto \exp(y^i w^T x^i). \]

- But if we only want correct decisions it’s sufficient to have
  \[ \frac{p(y^i \mid x^i, w)}{p(-y^i \mid x^i, w)} \geq \kappa, \]
  for any \( \kappa > 1 \).

- Taking logarithms and plugging in probabilities gives
  \[ y^i w^T x^i + \log Z - (-y^i w^T x^i) - \log Z \geq \log \kappa \]

- Since \( \kappa \) is arbitrary let’s use \( \log(\kappa) = 2 \),
  \[ y^i w^T x^i \geq 1. \]
SVMs and Likelihood Ratios

- So to classify all $i$ correctly it’s sufficient that
  \[ y^i w^T x^i \geq 1, \]
  but this linear program may have no solutions.

- To give solution, allow non-negative “slack” $r_i$ and penalize size of $r_i$,
  \[ \arg\min_{w,r} \sum_{i=1}^{n} r_i \quad \text{with} \quad y^i w^T x^i \geq 1 - r_i \quad \text{and} \quad r_i \geq 0. \]

- If we apply our Day 2 linear programming trick in reverse this minimizes
  \[ f(w) = \sum_{i=1}^{n} [1 - y^i w^T x^i]^+ \]
  and adding an L2-regularizer gives the standard SVM objective.
  - The notation $[\alpha]^+$ means $\max\{0, \alpha\}$. 
Multi-Class SVMs: \( nk \)-Slack Formulation

- With multi-class logistic regression we use
  \[
p(y^i = c \mid x^i, w) \propto \exp(w_c^T x^i).
\]

- If want correct decisions it's sufficient for all \( y' \neq y^i \) that
  \[
  \frac{p(y^i \mid x^i, w)}{p(y' \mid x^i, w)} \geq \kappa.
  \]

- Following the same steps as before, this corresponds to
  \[
  w_{y^i}^T x^i - w_{y'}^T x^i \geq 1.
  \]

- Adding slack variables our linear programming trick gives
  \[
  f(W) = \sum_{i=1}^{n} \sum_{y' \neq y^i} [1 - w_{y^i}^T x^i + w_{y'}^T x^i]^+,
  \]
  which with L2-regularization we'll call the \( nk \)-slack multi-class SVM.
Multi-Class SVMs: \(n\)-Slack Formulation

- If we want correct decisions it’s also sufficient that
  \[
  \frac{p(y^i \mid x^i, w)}{\max_{y' \neq y^i} p(y' \mid x^i, w)}.
  \]

- This leads to the constraints
  \[
  \max_{y' \neq y^i} \{w^T_{y^i} x^i - w^T_{y'} x^i\} \geq 1.
  \]

- Following the same steps gives an alternate objective
  \[
  f(W) = \sum_{i=1}^{n} \max_{y' \neq y^i} [1 - w^T_{y^i} x^i + w^T_{y'} x^i]^+,
  \]
  which with L2-regularization we’ll call the \(n\)-slack multi-class SVM.
Multi-Class SVMs: $nk$-Slack vs. $n$-Slack

- Our two formulations of multi-class SVMs:

  \[
  f(W) = \sum_{i=1}^{n} \sum_{y' \neq y^i} [1 - w_{y^i}^T x^i + w_{y'}^T x^i]^+ + \frac{\lambda}{2} \|W\|_F^2, \\
  f(W) = \sum_{i=1}^{n} \max_{y' \neq y^i} [1 - w_{y^i}^T x^i + w_{y'}^T x^i]^+ + \frac{\lambda}{2} \|W\|_F^2.
  \]

- The $nk$-slack loss penalizes based on all $y'$ that could be confused with $y^i$.
- The $n$-slack loss only penalizes based on the “most confusing” alternate example.

- While $nk$-slack often works better, $n$-slack can be used for structured prediction...
Hidden Markov Support Vector Machines

For **decoding in conditional random fields** to get the entire labeling correct we need

$$\frac{p(y^i | x^i, w)}{p(y' | x^i, w)} \geq \gamma,$$

for all alternative configurations $y'$.

Following the same steps as before we obtain

$$f(w) = \sum_{i=1}^{n} \max_{y' \neq y_i} [1 - \log p(y^i | x^i, w) + \log p(y' | x^i, w)]^+ + \frac{\lambda}{2} \|w\|^2,$$

the **hidden Markov support vector machine** (HMSVM).

Tries to make log-probability of true $y^i$ greater than for other $y'$ by more than 1.
Hidden Markov Support Vector Machines

Two problems with the HMSVM:
1. It requires finding second-best decoding, which is harder than decoding.
2. It views any alternative labeling $y'$ as equally bad.

Suppose that $y^i = [1 \ 1 \ 1 \ 1]$, and predictions of two models are

$$y' = [1 \ 1 \ 0 \ 1], \quad y' = [0 \ 0 \ 0 \ 0],$$

should both models receive the same loss on this example?
Adding a Loss Function

- We can fix both HMSVM issues by replacing the “correct decision” constraint,

\[ \log p(y^i | x^i, w) - \log p(y' | x^i, w) \geq 1, \]

with a constraint containing a loss function \( g \),

\[ \log p(y^i | x^i, w) - \log p(y' | x^i, w) \geq g(y^i, y'). \]

- Usually we take \( g(y^i, y') \) to be the difference between \( y^i \) and \( y' \).

- If \( g(y^i, y^i) = 0 \), you can maximize over all \( y' \) instead of \( y' \neq y^i \).
  - Further, if \( g \) is written as sum of functions depending on the graph edges, finding “most violated” constraint is equivalent to decoding.
Structured SVMs

- These constraints lead to the **max-margin Markov network** objective,

\[
f(w) = \sum_{i=1}^{n} \max_{y'} [g(y^i, y') - \log p(y^i \mid x^i, w) + \log p(y' \mid x^i, w)]^+ + \frac{\lambda}{2} \|w\|^2,
\]

which is also known as a **structured SVM**.

- **Beyond learning principle, key differences between CRFs and SSVMs:**
  - SSVMs **require decoding**, not inference, for learning:
    - Exact SSVMs in cases like graph cuts, matchings, rankings, etc.
  - SSVMs have **loss function** for complicated accuracy measures:
    - But loss needs to decompose over parts for tractability.
    - Could also formulate ‘loss-augmented’ CRFs.

- We can also train with approximate decoding methods.
  - State of the art training: block-coordinate Frank Wolfe (bonus slides).
SVMs for Ranking with Pairwise Preference

Suppose we want to rank examples.

A common setting is with features $x^i$ and pairwise preferences:

- List of objects $(i, j)$ where we want $y^i > y^j$.

Assuming a log-linear model,

$$p(y^i \mid x^i, w) \propto \exp(w^T x^i),$$

we can derive a loss function based on the pairwise preference decision,

$$\frac{p(y^i \mid x^i, w)}{p(y^j \mid x^j, w)} \geq \gamma,$$

which gives a loss function of the form

$$f(w) = \sum_{(i, j) \in R} [1 - w^T x^i + w^T x^j]^+. $$
Fitting Structured SVMs

Overview of progress on training SSVMs:

- **Cutting plane and bundle methods** (e.g., `svmStruct` software):
  - Require $O(1/\epsilon)$ iterations.
  - Each iteration requires **decoding on every training example**.

- **Stochastic sub-gradient methods**:
  - Each iteration requires **decoding on a single training example**.
  - **Still requires** $O(1/\epsilon)$ iterations.
  - Need to choose step size.

- **Dual Online exponentiated gradient (OEG)**:
  - Allows **line-search for step size** and has $O(1/\epsilon)$ rate.
  - Each iteration requires **inference on a single training example**.

- **Dual block-coordinate Frank-Wolfe (BCFW)**:
  - Each iteration requires **decoding on a single training example**.
  - Requires $O(1/\epsilon)$ iterations.
  - Closed-form **optimal step size**.
  - Theory allows approximate decoding.
Block Coordinate Frank Wolfe

Key ideas behind BCFW for SSVMs:

- Dual problem has as the form
  \[
  \min_{\alpha_i \in M_i} F(\alpha) = f(A\alpha) - \sum_i f_i(\alpha_i).
  \]
  where \(f\) is smooth.

- Problem structure where we can use **block coordinate descent**:
  - Normal coordinate updates intractable because \(\alpha_i \in |\mathcal{Y}|\).
  - But Frank-Wolfe block-coordinate update is equivalent to decoding
    \[
    s = \arg\min_{s' \in M_i} F(\alpha) + \langle \nabla_i F(\alpha), s' - \alpha_i \rangle.
    \]
    \[
    \alpha_i = \alpha_i - \gamma(s - \alpha_i).
    \]
  - Can implement algorithm in terms of primal variables.

- Connections between Frank-Wolfe and other algorithms:
  - Frank-Wolfe on dual problem is subgradient step on primal.
  - ‘Fully corrective’ Frank-Wolfe is equivalent to cutting plane.