Lab 12: Structured Prediction

Yacine Jernite

December 4, 2014
Lecture plan

- structured perceptron
- application: confused messages
- application: dependency parsing
- structured SVM
What does learning mean?

- **Density estimation:**
  - Accurately represent the distribution of data point
  - Most straightforward loss for a family of distributions $\mathcal{F}$:
    \[
    \min_{p \in \mathcal{F}} D(p, \hat{p})
    \]

  - $D(p, \hat{p}) = D_{KL}(p\|\hat{p})$ gives I-projection
  - $D(p, \hat{p}) = D_{KL}(\hat{p}\|p)$ gives M-projection

- Becomes (see lecture 3, slide 7):
  \[
  \max_{p \in \mathcal{F}} p(D)
  \]
What does learning mean?

- Density estimation

- **Structure or Knowledge discovery:**
  - Use learned model structure and parameters to get insights into data
  - For example LDA gives topics of a corpus
  - QMR-DT: identify syndromes
  - No *a priori* best loss, maximizing likelihood is a good idea.
What does learning mean?

- Density estimation

- Structure or Knowledge discovery

Specific prediction tasks:

- Predict the most likely value of a latent variable
- Becomes a classification problem
- Classification loss:

\[
\min_{p \in \mathcal{F}} \sum_{(x,y)} \mathbb{I}[\exists y' \neq y; p(y'|x) > p(y|x)]
\]
Classification loss

- Equivalently:

\[
\min_{p \in \mathcal{F}} \sum_{(x,y)} \mathbb{I}[\exists y' \neq y; \log(p(y'|x)) - \log(p(y|x)) > 0]
\]

- Recall log-linear formulation:

\[
\log(p(y'|x)) - \log(p(y|x)) = \mathbf{w} \cdot \left( \sum_c f_c(x, y'_c) - \sum_c f_c(x, y_c) \right)
\]

- Perfect solution: solve LP with $|\mathcal{D}| \times |\mathcal{Y}|$

- $|\mathcal{Y}|$ combinatorial: instead, structured algorithms
Why structured?

Starts from:

\[ \mathbb{I}[\exists y' \neq y; \log(p(y'|x)) - \log(p(y|x)) > 0] = \mathbb{I}[y' = \arg \max_u p(u|x) \wedge \log(p(y'|x)) - \log(p(y|x)) > 0] \]

\[ \arg \max_u p(u|x) \] obtained by marginal inference
Structured perceptron:

- Same as regular perceptron, where the prediction for $y$ is "structured"
- In case of violation, update:
  \[ w = w + f(x_t, y_t) - f(x_t, y) \]
  
- If the data is separable with marging $\gamma$, converges after:
  \[ \left( \frac{2 \max_{m,y} \| f(x_m, y) \|_2}{\gamma} \right)^2 \]

- Average weights as regularization
Simple example: Confusing messages
Simple example: Confusing messages

- Three impatient individuals fight for a token (right to speak)
- Gets token, says a word, loses token.
- All talking about different things
- Get to observe who did what
- Train a perceptron to track token (un-mix sentences)
Simple example: Confusing messages

Example:

- This / birds ate painting meat is fly beautiful

- [1, 2, 3, 2, 1, 2, 1, 3, 1]
Simple example: Confusing messages

- What model do we use?
- What are the features?
- How do we learn?
Simple example: Confusing messages

- This *birds* ate painting *meat* is *fly* beautiful
- Max Entropy Markov Model

Features?
Simple example: Confusing messages

- This *birds* ate painting *meat* is *fly* beautiful
- More information

- Features?
Simple example: Confusing messages

Features:
- \( f(y_t, y_{t+1}) \): tags
- \( f(y_t, x_t) \): tag, word, topic
- \( f(y_t, y_{t+2}, x_t, x_{t+2}) \): tags, POS, topics, (words?)

Prediction:
- Dynamic programming (complexity?)

Learning: perceptron
Simple example: Confusing messages

One iteration:
- Training datum:
  - This *birds* ate painting *meat* is *fly* beautiful
  - [1, 2, 3, 2, 1, 2, 1, 3, 1]

- Prediction:
  - *This* *birds* ate painting *meat* is *fly* beautiful
  - [2, 2, 3, 2, 1, 1, 3, 1]

- Update:

  \[ w = w + f(1, 2) - f(2, 2) + f(1, \text{This}, \text{Topic}_\text{this}) \]
  \[ - f(2, \text{This}, \text{Topic}_\text{this}) + f(1, 3, \text{DET}, \text{NNS}, \text{Topic}_\text{This}, \text{Topic}_\text{birds}) \]
  \[ - f(2, 3, \text{DET}), \text{NNS}, \text{Topic}_\text{This}, \text{Topic}_\text{birds}) + \ldots \]
Simple example: Confusing messages

Repeat until convergence.
Example of structured perceptron: dependency parsing

- Dependency parsing
- Incremental parsing
- Features, perceptron
- Exact search, Beam search
Example of structured perceptron: dependency parsing

Figure: A dependency (head-modifier) and constituency (grammar) parse.
Example of structured perceptron: dependency parsing

- For each pair of words, is one the “head” of the other?
- Edge classification problem?
Example of structured perceptron: dependency parsing

- Each word only has one “head”.
- Resulting graph is a tree
- Structured prediction
Example of structured perceptron: dependency parsing

First approach:

- Suppose the presence of an edge only depends on the sentence

- Example features for $f(e_{i,j}, \mathbf{w})$:
  - words $w_i, w_j$
  - POS tags of $w_i, w_j$
  - POS tags of $w_{i-1}, w_{i+1}, w_{j-1}, w_{j+1}$

- Inference is reduced to a Maximum Spanning Tree problem
Example of structured perceptron: dependency parsing

First approach:

MST followed by perceptron update.
Example of structured perceptron: dependency parsing

Straightforward algorithm, but does not support dependence between edges.
Example of structured perceptron: dependency parsing

- For projective grammars, can be done incrementally.
- Map the set of edges to a sequence of actions.
- Perceptron: predicts next action given partial parse tree.

http://stp.lingfil.uu.se/~nivre/docs/eacl3.pdf
Example of structured perceptron: dependency parsing

Possible actions:
- SHIFT: Moves a word from the buffer to the stack
- LEFT-ARC: First stack item points to second stack item
- RIGHT-ARC: Second stack item points to first stack item

Features:
- Words (lexicalized)
- POS
- Surrounding POS
- head of the current word
- Edge types

The black car hit the big motorcycle
Example of structured perceptron: dependency parsing

- How do we do supervised training?

- need an inference algorithm
  - Greedy search: take highest scoring action
  - Exact inference: complexity $O(n^5)$, can be reduced to $O(n^3)$ with some restrictions
  - Beam search: middle ground

- Choose which weights to update

http://www.aclweb.org/anthology/N12-1015
Example of structured perceptron: dependency parsing

- **Beam Search**
  - keep $k$ sequences of actions with highest score

- **Early Updates**
  - Only update perceptron weights for action which “falls off the beam”

- **Max-Violation**
  - Update perceptron weights for action that fall off the beam until farthest from the beam
Example of structured perceptron: dependency parsing

Figure: Choosing what to update
Example of structured perceptron: dependency parsing

Figure: Effect of Max-Violation
Structured SVM: slack variables

- We have:
  \[ w \cdot (f(x_t, y_t) - f(x_t, y)) > 0 \Leftrightarrow w' \cdot (f(x_t, y_t) - f(x_t, y)) \geq 1 \]

- If the data is separable, perceptron

- If not, minimize violations:
  \[
  \min_{\xi \geq 0} \sum_t \xi_t \text{ s.t. } w \cdot (f(x_t, y_t) - f(x_t, y)) \geq 1 - \xi_t
  \]

- Regularized version gives structural SVM:
  \[
  \min_{w, \xi \geq 0} \sum_t \xi_t + C \|w\|^2
  \]
Structured algorithms: SVM

- Using ALL constraints:

\[ \xi^*_t = \max(0, \max_{y \in \mathcal{Y}} 1 - w \cdot (f(x_t, y_t) - f(x_t, y))) \]

- Hence, whole objective becomes:

\[ \min_w \sum_t \max(0, \max_{y \in \mathcal{Y}} 1 - w \cdot (f(x_t, y_t) - f(x_t, y))) + C\|w\|^2 \]

- As before, \( \max_{y \in \mathcal{Y}} \) is a MAP inference problem, hence structured method

- Can easily replace uniform 1 loss with other metric, as long as inference remains easy.

- Example, Hamming loss.
Next time, pseudo-likelihood

Preparation exercises