Sequence Labeling (IV)
Viterbi and Struct. Perceptron/SVM

CS 690N, Spring 2018
Advanced Natural Language Processing
http://people.cs.umass.edu/~brenocon/anlp2018/

Brendan O’Connor
College of Information and Computer Sciences
University of Massachusetts Amherst
• Seq. labeling as log-linear **structured prediction**

\[
\hat{y}_{1:M} = \arg\max_{y_{1:M} \in \mathcal{Y}(w_{1:M})} \theta^T f(w_{1:M}, y_{1:M}),
\]

\[
p(w, y) = \prod_{t} p(y_t \mid y_{t-1}) p(w_t \mid y_t)
\]

HMM

CRF \( c = \text{pairs of RVs} \)

\[
p(y \mid w) \propto \exp \left( \sum_c \theta^T f_c(w, y_c) \right)
\]

• Local Markovian assumptions => efficient dynamic programming inference

• \( P(w) \): Likelihood (only generative model)
  - Forward algorithm

• \( P(y_m \mid w) \): Predicted tag marginals
  - Forward-Backward algorithm
  - for EM for unsup HMM .. gradients for sup CRF .. or direct usage in applications (e.g. high recall noun finder: get all with \( \geq 20\% \) prob)

• \( P(y \mid w) \): Predicted sequence (“decoding”)
  - **Viterbi algorithm**
Viterbi

- Max-product belief propagation, analogous to forward-backward as sum-product BP
- Key idea: summarize the maximal prefix path so far ... up to all possibilities for the next to last state
- Why not select a single best path so far?

- Viterbi worksheet!
Structured Perceptron

- Viterbi is very common for decoding. Inconvenient that you also need forward-backward for CRF learning
- Collins 2002: actually you can directly train only using Viterbi: **structured perceptron**
  - Theoretical results hold from the usual perceptron...
- Important extension in NLP: **Structured SVM**
  - a.k.a. **Structured large-margin/hinge-loss energy network**
  - a.k.a. **Cost-augmented perceptron**
- SP, SSVM, CRF training are variants of highly related objective functions and SSGD updates
Comparisons

- CRF vs. SP/SSVM
  - Only need an argmax decoder. Don’t need to calculate the normalizer.
  - Sometimes algorithms are fundamentally similar (Markov models: FB~Viterbi) but sometimes very different (e.g. graph matching: often sum/counting is #P-complete but argmax is polynomial)
  - Use tools from discrete optimization (e.g. off-the-shelf ILP decoders, typically using simplex and interior point .. or other algorithms, e.g. (alternating direction) dual decomposition)
  - (What if dynamic programming doesn’t work?)
  - Latent variables ~basically work better in a probabilistic framework

- SP vs. SSVM
  - Averaging vs. Regularization
  - Cost function: can customize (e.g. FP vs FN tradeoffs)

- CRF and SSVM most common today; use the SP if you’re implementing yourself, at least to get started!