Statistical Learning

Learning Bayes Net Structure
- **Initial state**: Empty or prior network
- **Operators**: Add, delete, reverse arc (avoiding cycles)
- **Evaluation function**: Posterior probability
- **Search**: Hill-climbing, simulated annealing, etc.

Learning Markov Networks
- Learning parameters (weights)
  - Generatively
  - Discriminatively
- Learning structure (features)

Learning Markov Networks
- Learning parameters (weights)
  - Generatively
  - Discriminatively
- Learning structure (features)

Generative Weight Learning
- Maximize likelihood (or posterior)
- Use gradient ascent or L-BFGS
- No local maxima
  \[
  \frac{\partial}{\partial w} \log p_i(x) = \frac{n_i(x)}{E_i [n_i(x)]} - \frac{n_i(x)}{\sum_j n_j(x)}
  \]
- Requires inference at each step (slow!)
Pseudo-Likelihood

\[ PL(x) = \prod_i P(x_i \mid \text{neighbors}(x_i)) \]

- Likelihood of each variable given its neighbors in the data
- Does not require inference at each step
- Widely used in vision, spatial statistics, etc.
- But PL parameters may not work well for long inference chains

Discriminative Weight Learning (aka Conditional Random Fields)

- Maximize conditional likelihood of query (y) given evidence (x)
  \[ \frac{\partial}{\partial w_i} \log P_y(y \mid x) = \eta_i(x,y) - E_y \eta_i(x,y) \]
  
  No. of true groundings of clause i in data
  Expected no. true groundings according to model

- Approximate expected counts by counts in MAP state of y given x

Voted Perceptron

- Originally proposed for training HMMs discriminatively
- Assumes network is linear chain
- Can be generalized to arbitrary networks

\[
\begin{align*}
  &w_i \leftarrow 0 \\
  &\text{for } t \leftarrow 1 \text{ to } T \text{ do} \\
  &\quad y_{MAP} \leftarrow \text{Viterbi}(x) \\
  &\quad w_i \leftarrow w_i + \eta \left[ \text{count}(y_{Data}) - \text{count}(y_{MAP}) \right] \\
  &\quad \text{return } \sum_i w_i / T
\end{align*}
\]

Structure Learning

- Feature search
  1. Start with atomic features
  2. Form conjunctions of current and atomic features
  3. Select best new feature and add to feature set
  4. Repeat until no improvement

- Evaluation
  - Likelihood, K-L divergence
  - Approximation: Previous weights don’t change
Learning Markov Logic Nets

- Data is a relational database
- Closed world assumption (if not: EM)
- Learning parameters (weights)
  - Generatively: Pseudo-likelihood
  - Discriminatively: Voted perceptron
- Learning structure (formulas)

Voted Perceptron for MLNs

- HMMs are special case of MLNs
- Replace Viterbi by MaxWalkSAT
- Network can now be arbitrary graph

\[
\begin{align*}
  w_i &\leftarrow 0 \\
  \text{for } t &\leftarrow 1 \text{ to } T \text{ do} \\
  y_{MAP} &\leftarrow \text{MaxWalkSAT}(x) \\
  w_i &\leftarrow w_i + \eta \left[ \text{count}(y_{Data}) - \text{count}(y_{MAP}) \right] \\
  \text{return } &\frac{\sum_i w_i}{T}
\end{align*}
\]

Structure Learning

- Generalizes feature induction in Markov nets
- Any inductive logic programming approach can be used, but . . .
- Goal is to induce any clauses, not just Horn
- Evaluation function should be likelihood
- Requires learning weights for each candidate
- Turns out not to be bottleneck
- Bottleneck is counting clause groundings
- Solution: Subsampling

Structure Learning

- Initial state: Unit clauses or hand-coded KB
- Operators: Add/remove literal, flip sign
- Evaluation function: Pseudo-likelihood + Structure prior
- Search: Beam search, shortest-first search