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)

Generative Weight Learning

- Maximize likelihood (or posterior)
- Use gradient ascent or L-BFGS
- No local maxima

$$\frac{\partial}{\partial w_i}\log P_w(x) = \boxed{n_i(x)} - \boxed{E_w\left[n_i(x)\right]}$$

No. of times feature *i* is true in data

Expected no. times feature *i* is true according to model

- Requires inference at each step (slow!)

Pseudo-Likelihood

$$PL(x) \equiv \prod_i P(x_i \mid neighbors(x_i))$$

- 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)

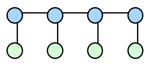
$$\frac{\partial}{\partial w_i} \log P_w(y \mid x) = \underbrace{n_i(x,y)}_{\text{No. of true groundings of clause } i \text{ in data}} - \underbrace{E_w\left[n_i(x,y)\right]}_{\text{Expected no. true groundings according to model}}$$

- 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

```
$w_i \leftarrow 0$
**for** $t \leftarrow 1$ **to** $T$ **do**
    $y_{MAP} \leftarrow$ Viterbi($x$)
    $w_i \leftarrow w_i + \eta$ [count$_i$($y_{Data}$) – count$_i$($y_{MAP}$)]
**return** $\sum_t w_i / T$
```



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

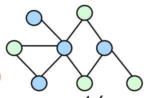
Learning Markov Logic Nets

- 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

```
$w_i \leftarrow 0$
for $t \leftarrow 1$ to $T$ do
    $y_{MAP} \leftarrow \text{MaxWalkSAT}(x)$
    $w_i \leftarrow w_i + \eta\,[\text{count}_i(y_{Data}) - \text{count}_i(y_{MAP})]$
return $\sum_t w_i / T$
```



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