Learning
Markov Networks
Learning Markov Networks

- Learning parameters (weights)
  - Generatively
  - Discriminatively
- Learning with incomplete data
- Learning structure (features)
Generative Weight Learning

- Maximize likelihood or posterior probability
- Numerical optimization (gradient or 2\textsuperscript{nd} order)
- No local maxima
  \[ \frac{\partial}{\partial w_i} \log P_w(x) = n_i(x) - E_w[n_i(x)] \]
  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!)
Priors for Markov networks

- As in BNs, posterior probability is proportional to the likelihood times the prior:
  \[ P(w|x) = \alpha P(x,w) = P(x|w) P(w) \]
  \[ \log P(x,w) = \log P(x|w) + \log P(w) + \log \alpha \]

- Common priors
  - L₂ norm: Encourages small weights
  Univariate Gaussian over each weight \( w_i \), with mean zero and standard deviation \( \sigma_i \).
  - L₁ norm: Encourages sparse weights (many zeroes)
  \[ P(w) \sim \exp(-w/\lambda) \]
Pseudo-Likelihood

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

- Likelihood of each variable given its neighbors in the data
- Does not require inference at each step
- Consistent estimator
- Widely used in vision, spatial statistics, etc.
- But PL parameters may not work well for long inference chains
Discriminative Weight Learning (a.k.a. Conditional Random Fields)

- Maximize conditional likelihood of query \((y)\) given evidence \((x)\)

\[
\frac{\partial}{\partial w_i} \log P_w(y \mid x) = n_i(x, y) - E_w[n_i(x, y)]
\]

- Key difference:
  Expectation is only over \(y\)
- Inference is easier, but still hard

Number of times feature \(f_i\) is true in the data

Expected number according to model
Other Weight Learning Approaches

- **Generative**: Iterative scaling
- **Discriminative**: Max margin
Learning with Missing Data

- Gradient of likelihood is now difference of expectations

\[
\frac{\partial}{\partial w_i} \log P_w(x) = E_w[n_i(y \mid x)] - E_w[n_i(x, y)]
\]

- First expectation is over all possible y (given x)
- Second expectation is over all possible x and y
- Can use gradient descent or EM
Structure Learning

- Start with atomic features
- Greedily conjoin features to improve score
- Problem: Need to reestimate weights for each new candidate
- Approximation: Keep weights of previous features constant
Other Structure Learning Approaches

- Search for conditional independencies, then select a graph consistent with those independencies.
- Start with example features, then generalize (“bottom-up” learning).
- Include all pairwise features, run weight learning with an $L_1$ norm, and keep all features that remain non-zero.