Inference and Representation

Rachel Hodos

New York University

Lab 6, October 21, 2015

Outline

10,000 foot view

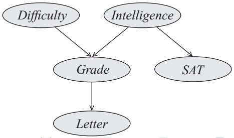
Graphical models define *local* relationships between variables and their neighbors, in order to efficiently model and make inferences about the *global* system.

Outline

1. Midterm review
   - Representation
   - Inference
   - Learning

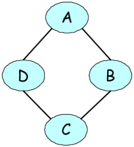
Bayesian Networks

- Parametrized by CPDs
- Structure implies independence via d-separation
- P-maps between some BN and P often exist but not always
- Markov equivalence = same skeleton and immoralities
- MLE parameters for discrete variables easy to compute (parameter for each CPD entry is just fraction of corresponding cases in observed data)
- Examples: Naive Bayes, Logistic regression, QMR-DT (disease and symptoms), HMM, LDA, gene regulatory networks



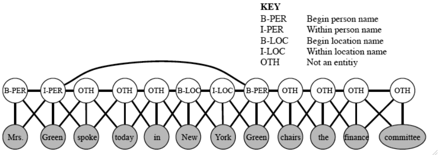
Markov Random Fields

- Parametrized by potential functions
- Structure implies independence via graph-separation
- There are also distributions that MRFs can not perfectly represent (e.g. v-structures)
- Partition function difficult to compute
- Examples: Grid-structured (Boltzmann machines, Ising), Gaussian MRF (= multivariate normal)
- Can moralize a BN to become an MRF by keeping same edges and marrying the parents



Conditional Random Fields

- MRFs that model some conditional distribution $P(Y|X)$
- Potential functions and even graph structure might vary based on X
- Partition function (Z) is now a function of X.
- Examples: Skip-chain CRF for named entity recognition; Grid-structured CRF for image segmentation



Outline

1. Midterm review
   - Representation
   - Inference
   - Learning

Exact Inference

- Can be performed by marginalization, i.e.

$$p(Y|E=e) = \frac{p(Y,e)}{p(e)}$$

- NP-hard (both MAP and marginal inference, for both BN and MRF)
- However, this is only worst case. Sometimes tractable, e.g. HMMs.
- Algorithm: variable elimination. Just a more efficient way to compute all the sums.
- Variable elimination ordering: also NP-hard, but there are greedy heuristics, e.g. minimize # of induced edges.
- Runtime is exponential in the treewidth (i.e. width of the *induced graph*)

Belief Propagation (exact for tree-structured MRFs)

- Sum-product BP (from lab 5) can be used to compute all marginals in linear time
- Sum-product message:

$$m_{j\to i}(x_i) = \sum_{x_j} \phi_j(x_j)\phi_{ij}(x_i, x_j) \prod_{k\in N(j)\setminus i} m_{k\to j}(x_j)$$

- Max-product BP (see next slide) just replace sum with max and you can do MAP inference
- Exact for tree-structured graphs, otherwise no guarantees
- When used on non-tree structures (i.e. graphs with loops), sometimes called *Loopy Belief Propagation*

Max-product BP

- Same as sum-product BP except that the messages are now:

$$m_{j\to i}(x_i) = \max_{x_j} \phi_j(x_j)\phi_{ij}(x_i, x_j) \prod_{k\in N(j)\backslash i} m_{k\to j}(x_j)$$

- After passing all messages, can compute single node *max-marginals*,

$$m_i(x_i) = \phi_i(x_i) \prod_{j\in N(i)} m_{j\to i}(x_i) \qquad \propto \qquad \max_{\mathbf{x}_{V\backslash i}} p(\mathbf{x}_{V\backslash i}, x_i)$$

- If the MAP assignment $\mathbf{x}^*$ is **unique**, can find it by locally decoding each of the single node max-marginals, i.e.

$$x_i^* = \arg\max_{x_i} m_i(x_i)$$

Approximate Inference

- Since exact inference is NP-hard, we often resort to approximate inference
- Two main approaches: 1) Monte Carlo methods, and 2) Variational inference. We will discuss 2) later.
- Monte Carlo methods generate samples from the posterior
- These samples can then be used to approximate, e.g.:
    - The full posterior
    - Marginals
    - Expectations

Sampling for Monte Carlo Methods (part I)

- Unconditional sampling in BNs is straightforward
- Accurate conditional sampling in BNs is trickier since denominator $p(E = e)$ could be very small
- Sampling in MRFs is also not straightforward (Z)
- *Normalized importance sampling* uses samples from $q(x)$ (easy to sample from) to approximate samples from $p(x)$ (hard to sample from)
- *Likelihood reweighting* is a technique to generate conditional samples from a BN, can be viewed as a specific case of normalized importance sampling

Sampling for Monte Carlo Methods (part II)

- Markov Chain Monte Carlo methods (MCMC) use *adaptive* proposal distribution
- Metropolis Hastings (MH) is a popular example of MCMC
- Gibbs sampling is a special case of MH

Outline

1. Midterm review
   - Representation
   - Inference
   - Learning

EM algorithm

- For MLE parameter estimation with hidden variables
- Not exact because: 1) iterative, so results depends on when you stop, and 2) can have local minima
- However, guaranteed to converge (proved this in lab)
- Algorithm:
  1. Write down the **complete log-likelihood** $\log p(\mathbf{x}, \mathbf{z}; \theta)$ in such a way that it is linear in $\mathbf{z}$

EM algorithm

- For MLE parameter estimation with hidden variables
- Not exact because: 1) iterative, so results depends on when you stop, and 2) can have local minima
- However, guaranteed to converge (proved this in lab)
- Algorithm:
  1. Write down the **complete log-likelihood** $\log p(\mathbf{x}, \mathbf{z}; \theta)$ in such a way that it is linear in $\mathbf{z}$
  2. Initialize θ_0, e.g. at random or using a good first guess

EM algorithm

- For MLE parameter estimation with hidden variables
- Not exact because: 1) iterative, so results depends on when you stop, and 2) can have local minima
- However, guaranteed to converge (proved this in lab)
- Algorithm:
  1. Write down the **complete log-likelihood** $\log p(\mathbf{x}, \mathbf{z}; \theta)$ in such a way that it is linear in $\mathbf{z}$
  2. Initialize θ_0, e.g. at random or using a good first guess
  3. Repeat until convergence:

EM algorithm

- For MLE parameter estimation with hidden variables
- Not exact because: 1) iterative, so results depends on when you stop, and 2) can have local minima
- However, guaranteed to converge (proved this in lab)
- Algorithm:
  1. Write down the **complete log-likelihood** $\log p(\mathbf{x}, \mathbf{z}; \theta)$ in such a way that it is linear in $\mathbf{z}$
  2. Initialize θ_0, e.g. at random or using a good first guess
  3. Repeat until convergence:

$$\theta_{t+1} = \arg\max_{\theta} \sum_{m=1}^{M} E_{p(\mathbf{z}_m|\mathbf{x}_m;\theta_t)}[\log p(\mathbf{x}_m, \mathbf{Z}; \theta)]$$

EM algorithm

- For MLE parameter estimation with hidden variables
- Not exact because: 1) iterative, so results depends on when you stop, and 2) can have local minima
- However, guaranteed to converge (proved this in lab)
- Algorithm:
    1. Write down the **complete log-likelihood** $\log p(\mathbf{x}, \mathbf{z}; \theta)$ in such a way that it is linear in $\mathbf{z}$
    2. Initialize θ_0, e.g. at random or using a good first guess
    3. Repeat until convergence:

$$\theta_{t+1} = \arg\max_{\theta} \sum_{m=1}^{M} E_{p(\mathbf{z}_m | \mathbf{x}_m; \theta_t)}[\log p(\mathbf{x}_m, \mathbf{Z}; \theta)]$$