School of Computer Science
Carnegie Mellon

10-708 Probabilistic Graphical Models

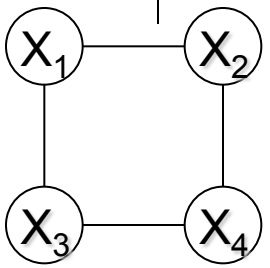
Learning Fully Observed Undirected Graphical Models

Readings:
Jordan Chap. 9.3
Jordan Chap. 20
Sha & Pereira (2003)

Matt Gormley
Lecture 6
February 1, 2016

Housekeeping

- **Homework 1**
  - Was released on Saturday ‘morning’
  - Due date: Feb 15, 12:00 noon



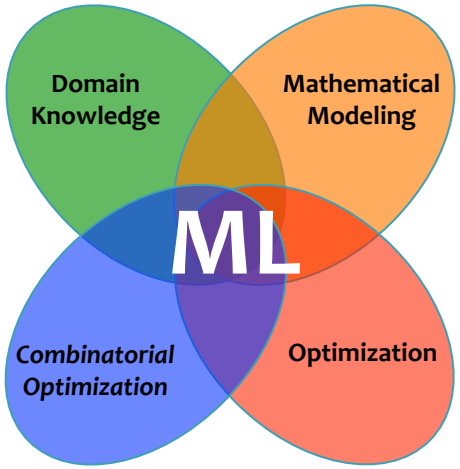
ML Parameter Est. for completely observed MRFs of given structure

Machine Learning

The **data** inspires the structures we want to predict

Our **model** defines a score for each structure

It also tells us what to optimize



Inference finds {best structure, marginals, partition function} for a new observation

(**Inference** is usually called as a subroutine in learning)

Learning tunes the parameters of the model

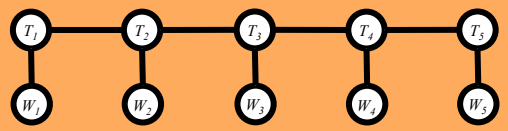
MLE for Undirected GMs

1. Data

$$\mathcal{D} = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$$

2. Model

$$p(\boldsymbol{x} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

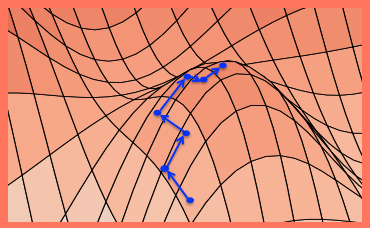


3. Objective

$$\ell(\theta; \mathcal{D}) = \sum_{n=1}^{N} \log p(\boldsymbol{x}^{(n)} \mid \boldsymbol{\theta})$$

4. Learning

$$\boldsymbol{\theta}^* = \operatorname*{argmax}_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}; \mathcal{D})$$



5. Inference

1. Marginal Inference

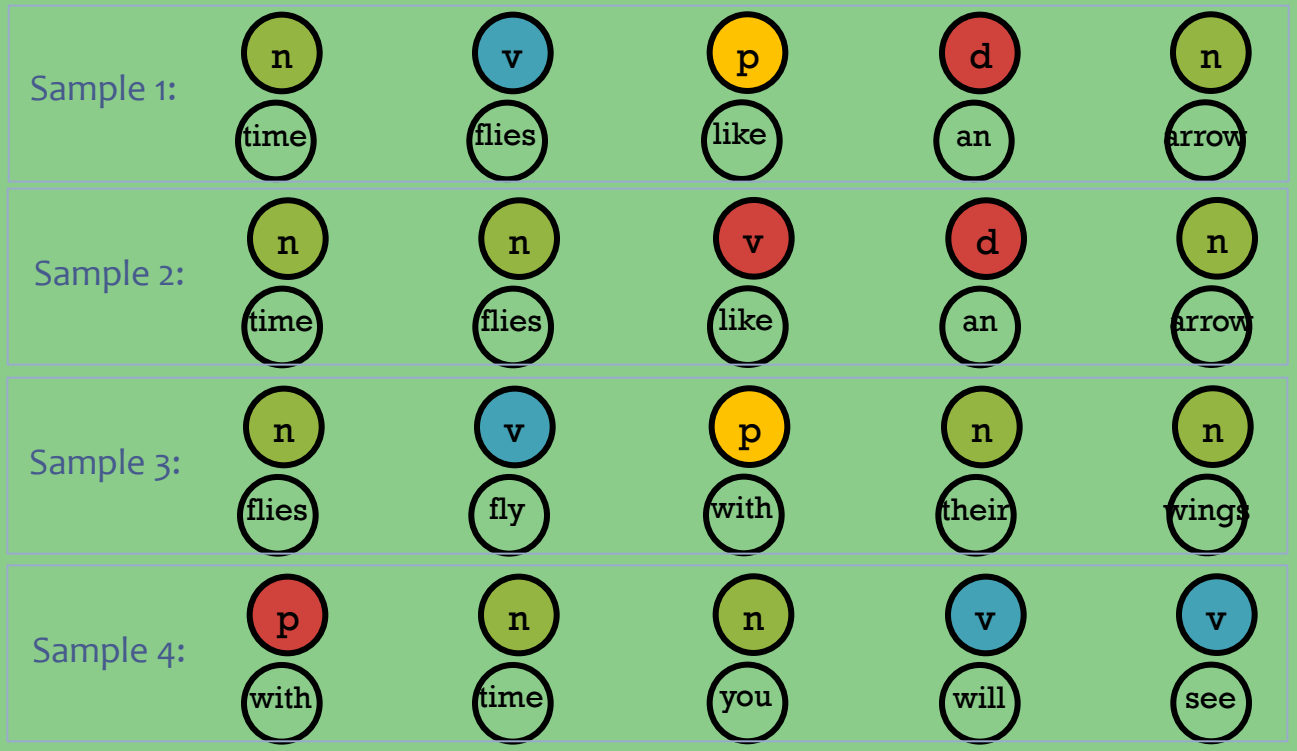
$$p(\boldsymbol{x}_C) = \sum_{\boldsymbol{x}' : \boldsymbol{x}'_C = \boldsymbol{x}_C} p(\boldsymbol{x}' \mid \boldsymbol{\theta})$$

2. Partition Function

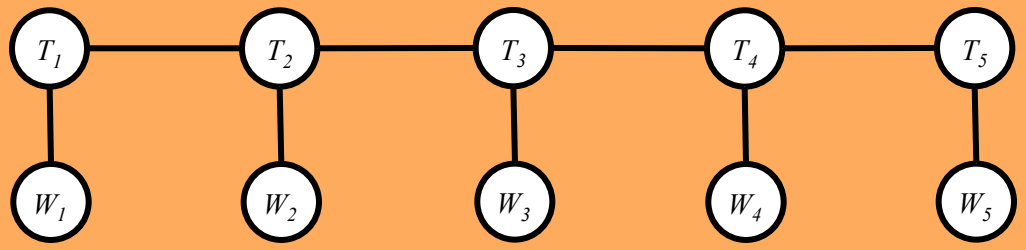
$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

1. Data

Given training examples: $\mathcal{D} = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$



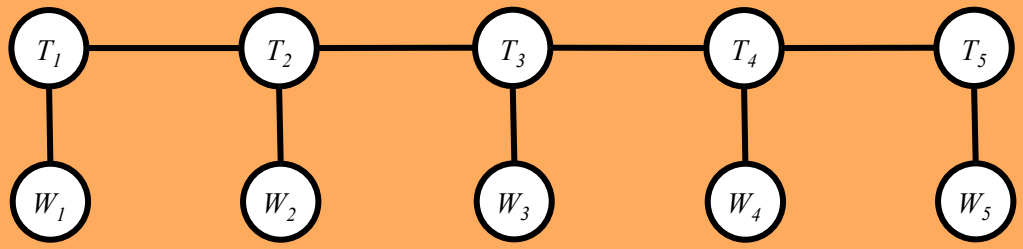
2. Model



2. Model

Define the model to be an MRF:

$$p(\boldsymbol{x} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$



3. Objective

Choose the objective to be log-likelihood:

(Assign high probability to the things we observe and low probability to everything else)

$$\ell(\theta; \mathcal{D}) = \sum_{n=1}^{N} \log p(\boldsymbol{x}^{(n)} \mid \boldsymbol{\theta})$$

3. Objective

Choose the objective to be log-likelihood:

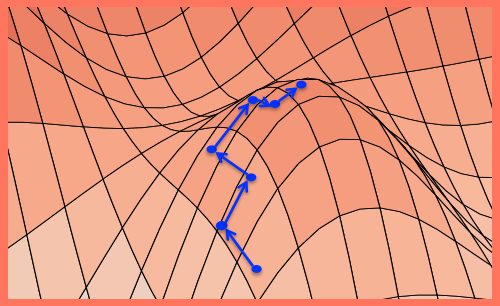
(Assign high probability to the things we observe and low probability to everything else)

$$\ell(\theta; \mathcal{D}) = \sum_{n=1}^{N} \log p(\boldsymbol{x}^{(n)} \mid \boldsymbol{\theta})$$

4. Learning

Tune the parameters to maximize the objective function

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, \ell(\boldsymbol{\theta}; \mathcal{D})$$



3. Objective

Choose the objective to be log-likelihood:

(Assign high probability to the things we observe and low probability to everything else)

$$\ell(\boldsymbol{\theta}; \mathcal{D}) = \sum_{n=1}^{N} \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(n)})$$

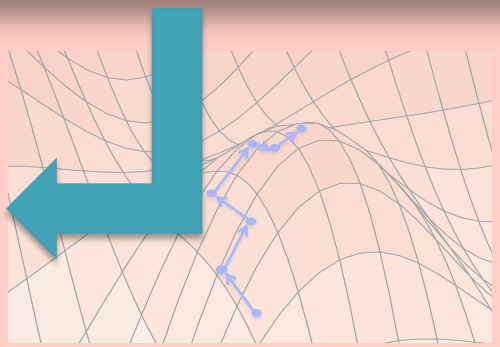
Goals for Today's Lecture

1. Optimize this objective function
2. Characterize the applicability of different optimizers

Tune the parameter

function

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, \ell(\boldsymbol{\theta}; \mathcal{D})$$



5. Inference

Three Tasks:

1. Marginal Inference

Compute marginals of variables and cliques

$$p(x_i) = \sum_{\boldsymbol{x}' : x'_i = x_i} p(\boldsymbol{x}' \mid \boldsymbol{\theta}) \qquad p(\boldsymbol{x}_C) = \sum_{\boldsymbol{x}' : \boldsymbol{x}'_C = \boldsymbol{x}_C} p(\boldsymbol{x}' \mid \boldsymbol{\theta})$$

2. Partition Function

Compute the normalization constant

$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

3. MAP Inference

Compute variable assignment with highest probability

$$\hat{\boldsymbol{x}} = \operatorname*{argmax}_{\boldsymbol{x}} \; p(\boldsymbol{x} \mid \boldsymbol{\theta})$$

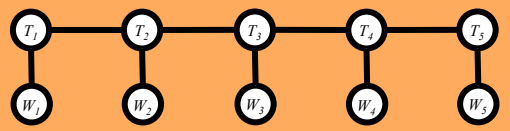
MLE for Undirected GMs

1. Data

$$\mathcal{D} = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$$

2. Model

$$p(\boldsymbol{x} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

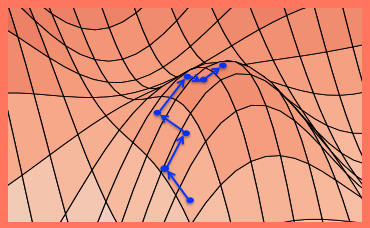


3. Objective

$$\ell(\theta; \mathcal{D}) = \sum_{n=1}^{N} \log p(\boldsymbol{x}^{(n)} \mid \boldsymbol{\theta})$$

4. Learning

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, \ell(\boldsymbol{\theta}; \mathcal{D})$$



5. Inference

1. Marginal Inference

$$p(\boldsymbol{x}_C) = \sum_{\boldsymbol{x}' : \boldsymbol{x}'_C = \boldsymbol{x}_C} p(\boldsymbol{x}' \mid \boldsymbol{\theta})$$

2. Partition Function

$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

MLE for Undirected GMs

- Today's parameter estimation assumptions:
  1. The graphical model structure is given
  2. Every variable appears in the training examples

Questions

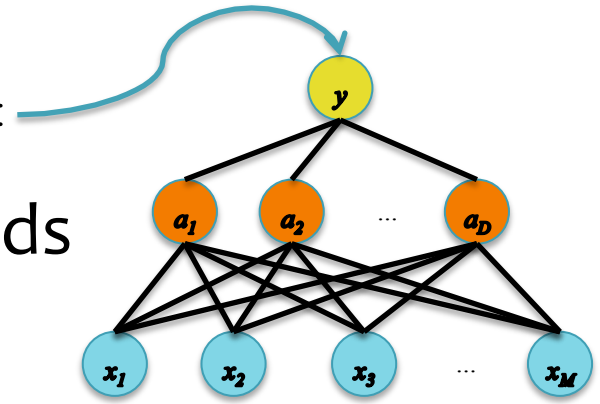
1. What does the **likelihood objective** accomplish?
2. Is likelihood the ***right* objective** function?
3. **How do we optimize** the objective function (i.e. learn)?
4. What **guarantees** does the optimizer provide?
5. (What is the **mapping from data → model**? In what ways can we incorporate our domain knowledge? How does this impact learning?)

Options for MLE of MRFs

- **Setting I:** $\psi_C(\boldsymbol{x}_C) = \theta_{C,\boldsymbol{x}_C}$
  A. MLE by inspection (Decomposable Models)
  B. Iterative Proportional Fitting (IPF)

- **Setting II:** $\psi_C(\boldsymbol{x}_C) = \exp(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}_C))$
  C. Generalized Iterative Scaling
  D. Gradient-based Methods

- **Setting III:** $\psi_C(\boldsymbol{x}_C) =$
  D. Gradient-based Methods

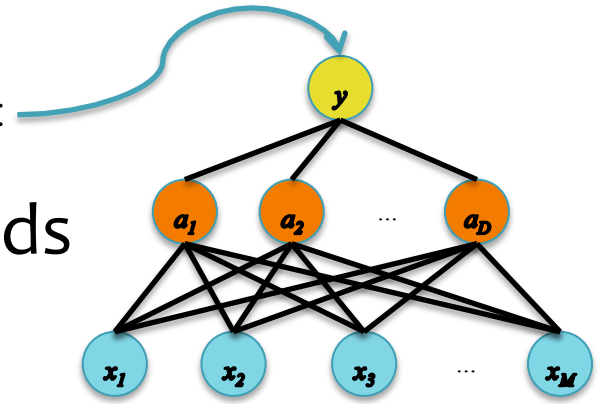


Today's Lecture

- **Setting I:** $\psi_C(\boldsymbol{x}_C) = \theta_{C,\boldsymbol{x}_C}$
  A. MLE by inspection (Decomposable Models)
  B. Iterative Proportional Fitting (IPF)

- **Setting II:** $\psi_C(\boldsymbol{x}_C) = \exp(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}_C))$
  C. Generalized Iterative Scaling
  D. Gradient-based Methods

- **Setting III:** $\psi_C(\boldsymbol{x}_C) =$
  D. Gradient-based Methods

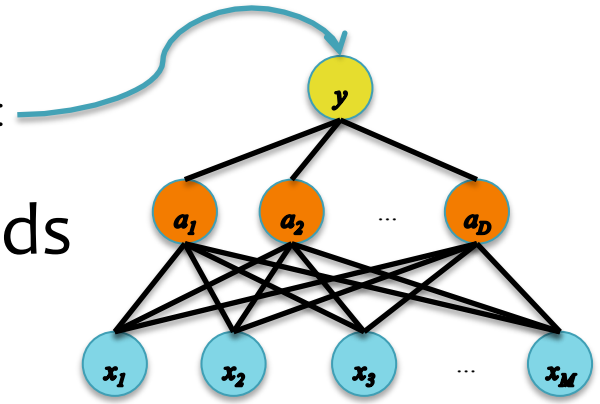


Options for MLE of MRFs

- **Setting I:** $\psi_C(\boldsymbol{x}_C) = \theta_{C,\boldsymbol{x}_C}$
  A. MLE by inspection (Decomposable Models)
  B. Iterative Proportional Fitting (IPF)

- **Setting II:** $\psi_C(\boldsymbol{x}_C) = \exp(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}_C))$
  C. Generalized Iterative Scaling
  D. Gradient-based Methods

- **Setting III:** $\psi_C(\boldsymbol{x}_C) =$
  D. Gradient-based Methods

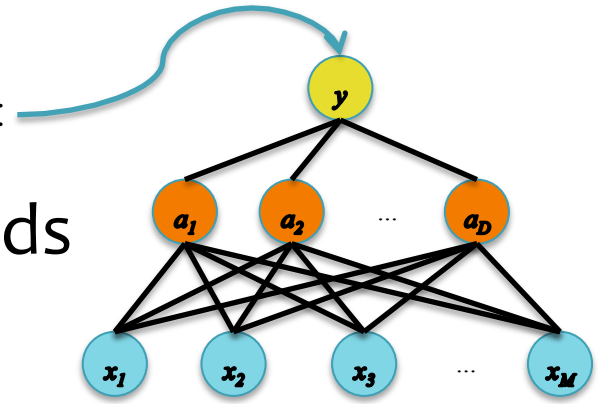


Options for MLE of MRFs

- **Setting I:** $\psi_C(\boldsymbol{x}_C) = \theta_{C,\boldsymbol{x}_C}$
  A. MLE by inspection (Decomposable Models)
  B. Iterative Proportional Fitting (IPF)

- **Setting II:** $\psi_C(\boldsymbol{x}_C) = \exp(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}_C))$
  C. Generalized Iterative Scaling
  D. Gradient-based Methods

- **Setting III:** $\psi_C(\boldsymbol{x}_C) =$
  D. Gradient-based Methods

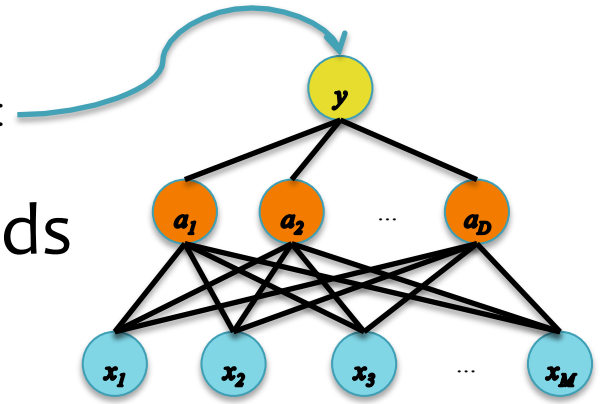


Options for MLE of MRFs

- **Setting I:** $\psi_C(\boldsymbol{x}_C) = \theta_{C,\boldsymbol{x}_C}$
  A. MLE by inspection (Decomposable Models)
  B. Iterative Proportional Fitting (IPF)

- **Setting II:** $\psi_C(\boldsymbol{x}_C) = \exp(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}_C))$
  C. Generalized Iterative Scaling
  D. Gradient-based Methods

- **Setting III:** $\psi_C(\boldsymbol{x}_C) =$
  D. Gradient-based Methods

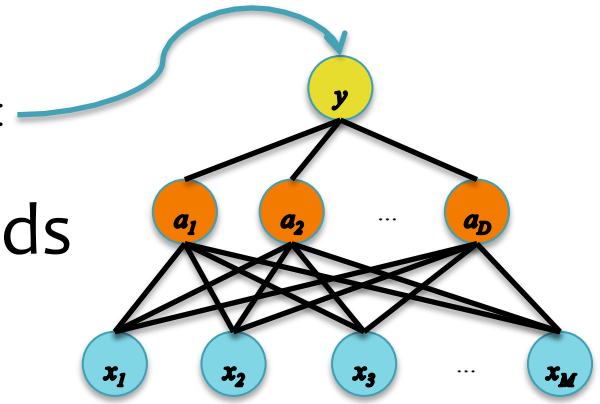


Options for MLE of MRFs

- **Setting I:** $\psi_C(\boldsymbol{x}_C) = \theta_{C,\boldsymbol{x}_C}$
  A. MLE by inspection (Decomposable Models)
  B. Iterative Proportional Fitting (IPF)

- **Setting II:** $\psi_C(\boldsymbol{x}_C) = \exp(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}_C))$
  C. Generalized Iterative Scaling
  D. Gradient-based Methods

- **Setting III:** $\psi_C(\boldsymbol{x}_C) =$
  D. Gradient-based Methods



Whiteboard

- Log-likelihood of MRFs with discrete variables (i.e. tabular clique potentials)
- Derivative of log-likelihood with respect to potentials

Log Likelihood for UGMs with tabular clique potentials

- Sufficient statistics: for a UGM (V,E), the number of times that a configuration $\mathbf{x}$ (i.e., $\mathbf{X}_V=\mathbf{x}$) is observed in a dataset $D=\{\mathbf{x}_1,\ldots,\mathbf{x}_N\}$ can be represented as follows:

$$m(\mathbf{x}) \overset{\text{def}}{=} \sum_n \delta(\mathbf{x},\mathbf{x}_n) \quad \text{(total count)}, \qquad \text{and} \qquad m(\mathbf{x}_c) \overset{\text{def}}{=} \sum_{\mathbf{x}_{V\setminus c}} m(\mathbf{x}) \quad \text{(clique count)}$$

- In terms of the counts, the log likelihood is given by:

$$p(D|\theta) = \prod_n \prod_{\mathbf{x}} p(\mathbf{x}\,|\,\theta)^{\delta(\mathbf{x},\mathbf{x}_n)}$$

$$\log p(D|\theta) = \sum_n \sum_{\mathbf{x}} \delta(\mathbf{x},\mathbf{x}_n) \log p(\mathbf{x}\,|\,\theta) = \sum_{\mathbf{x}} \sum_n \delta(\mathbf{x},\mathbf{x}_n) \log p(\mathbf{x}\,|\,\theta)$$

$$\ell = \sum_{\mathbf{x}} m(\mathbf{x}) \log\left(\frac{1}{Z}\prod_c \psi_c(\mathbf{x}_c)\right)$$

$$= \sum_c \sum_{\mathbf{x}_c} m(\mathbf{x}_c) \log \psi_c(\mathbf{x}_c) - N \log Z$$

- There is a nasty log Z in the likelihood

Log Likelihood for UGMs with tabular clique potentials

- Sufficient statistics: for a UGM (V,E), the number of times that a configuration $\mathbf{x}$ (i.e., $\mathbf{X}_V=\mathbf{x}$) is observed in a dataset $D=\{\mathbf{x}_1,\ldots,\mathbf{x}_N\}$ can be represented as follows:

$$m(\mathbf{x}) \overset{\text{def}}{=} \sum_n \delta(\mathbf{x},\mathbf{x}_n) \quad \text{(total count)}, \qquad \text{and} \qquad m(\mathbf{x}_c) \overset{\text{def}}{=} \sum_{\mathbf{x}_{V\setminus c}} m(\mathbf{x}) \quad \text{(clique count)}$$

- In terms of the counts, the log likelihood is given by:

$$\log p(D|\theta) = \sum_c \sum_{\mathbf{x}_c} m(\mathbf{x}_c) \log \psi_c(\mathbf{x}_c) - N \log Z$$

- There is a nasty log Z in the likelihood

Derivative of log Likelihood

- Log-likelihood: $\ell = \sum_c \sum_{\mathbf{x}_c} m(\mathbf{x}_c) \log \psi_c(\mathbf{x}_c) - N \log Z$
- First term: $\frac{\partial \ell_1}{\partial \psi_c(\mathbf{x}_c)} = m(\mathbf{x}_c) \Big/ \psi_c(\mathbf{x}_c)$
- Second term: $\frac{\partial \log Z}{\partial \psi_c(\mathbf{x}_c)} = \frac{1}{Z} \frac{\partial}{\partial \psi_c(\mathbf{x}_c)} \left(\sum_{\mathbf{x}} \prod_d \psi_d(\tilde{\mathbf{x}}_d) \right)$

Set the value of variables $\mathbf{x}_c$ to $\mathbf{x}$

$$= \frac{1}{Z} \sum_{\mathbf{x}} \delta(\tilde{\mathbf{x}}_c, \mathbf{x}_c) \frac{\partial}{\partial \psi_c(\mathbf{x}_c)} \left(\prod_d \psi_d(\tilde{\mathbf{x}}_d) \right)$$

$$= \sum_{\mathbf{x}} \delta(\tilde{\mathbf{x}}_c, \mathbf{x}_c) \frac{1}{\psi_c(\tilde{\mathbf{x}}_c)} \frac{1}{Z} \prod_d \psi_d(\tilde{\mathbf{x}}_d)$$

$$= \frac{1}{\psi_c(\mathbf{x}_c)} \sum_{\mathbf{x}} \delta(\tilde{\mathbf{x}}_c, \mathbf{x}_c) p(\tilde{\mathbf{x}}) = \frac{p(\mathbf{x}_c)}{\psi_c(\mathbf{x}_c)}$$

Conditions on Clique Marginals

- Derivative of log-likelihood

$$\frac{\partial \ell}{\partial \psi_c(\mathbf{x}_c)} = \frac{m(\mathbf{x}_c)}{\psi_c(\mathbf{x}_c)} - N \frac{p(\mathbf{x}_c)}{\psi_c(\mathbf{x}_c)}$$

- Hence, for the maximum likelihood parameters, we know that:

$$p^*_{MLE}(\mathbf{x}_c) = \frac{m(\mathbf{x}_c)}{N} \overset{\text{def}}{=} \tilde{p}(\mathbf{x}_c)$$

- In other words, at the maximum likelihood setting of the parameters, for each clique, the model marginals must be equal to the observed marginals (empirical counts).

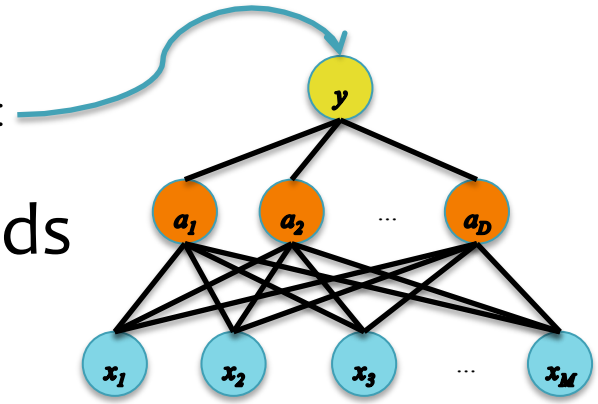
- This doesn't tell us how to get the ML parameters, it just gives us a condition that must be satisfied when we have them.

Options for MLE of MRFs

- **Setting I:** $\psi_C(\boldsymbol{x}_C) = \theta_{C,\boldsymbol{x}_C}$
  A. MLE by inspection (Decomposable Models)
  B. Iterative Proportional Fitting (IPF)

- **Setting II:** $\psi_C(\boldsymbol{x}_C) = \exp(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}_C))$
  C. Generalized Iterative Scaling
  D. Gradient-based Methods

- **Setting III:** $\psi_C(\boldsymbol{x}_C) =$
  D. Gradient-based Methods



Whiteboard

- MLE by Guessing
  - Example 1: linear-chain on three variables
  - Example 2: “decomposable” with four variables

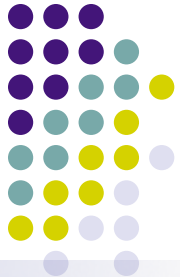
MLE by Guessing

- **Definition:** Graph is **decomposable** if it can be recursively subdivided into sets A, B, and S such that S separates A and B.

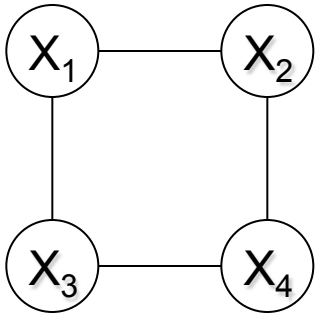
MLE by Guessing

- **Definition:** Graph is **decomposable** if it can be recursively subdivided into sets A, B, and S such that S separates A and B.
- **Recipe for MLE by Guessing:**
  - Three conditions:
    1. Graphical model is *decomposable*
    2. Potentials defined on *maximal cliques*
    3. Potentials are are parameterized as: $\psi_C(\boldsymbol{x}_C) = \theta_{C,\boldsymbol{x}_C}$
  - **Step 1:** set each clique potential to its empirical marginal
  - **Step 2:** divide out every non-empty intersection between cliques *exactly once*

Non-decomposable and/or with non-maximal clique potentials



- If the graph is non-decomposable, and or the potentials are defined on non-maximal cliques (e.g., ψ_{12}, ψ_{34}), we could not equate empirical marginals (or conditionals) to MLE of cliques potentials.



$$p(x_1, x_2, x_3, x_4) = \prod_{\{i,j\}} \psi_{ij}(x_i, x_j)$$

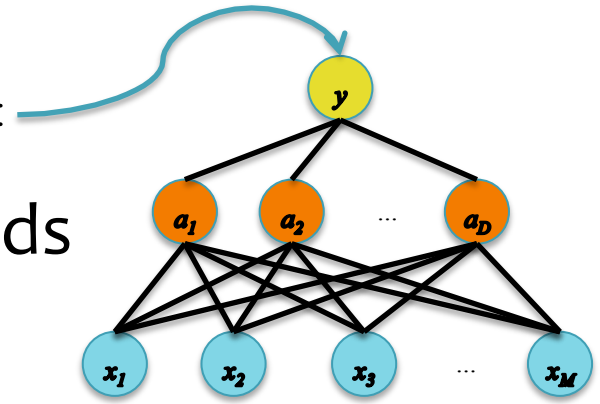
$$\exists (i,j) \quad \text{s.t.} \quad \psi_{ij}^{\text{MLE}}(x_i, x_j) \neq \begin{cases} \tilde{p}(x_i, x_j) \\ \tilde{p}(x_i, x_j) / \tilde{p}(x_i) \\ \tilde{p}(x_i, x_j) / \tilde{p}(x_j) \end{cases}$$

Options for MLE of MRFs

- **Setting I:** $\psi_C(\boldsymbol{x}_C) = \theta_{C,\boldsymbol{x}_C}$
  - A. MLE by inspection (Decomposable Models)
  - B. Iterative Proportional Fitting (IPF)

- **Setting II:** $\psi_C(\boldsymbol{x}_C) = \exp(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}_C))$
  - C. Generalized Iterative Scaling
  - D. Gradient-based Methods

- **Setting III:** $\psi_C(\boldsymbol{x}_C) =$
  - D. Gradient-based Methods



Fixed Point Iteration for Optimization

- Fixed point iteration is a general tool for solving systems of equations
- It can also be applied to optimization.

$$J(\boldsymbol{\theta})$$

$$\frac{dJ(\boldsymbol{\theta})}{d\theta_i} = 0 = f(\boldsymbol{\theta})$$

$$0 = f(\boldsymbol{\theta}) \Rightarrow \theta_i = g(\boldsymbol{\theta})$$

$$\theta_i^{(t+1)} = g(\boldsymbol{\theta}^{(t)})$$

1. Given objective function:
2. Compute derivative, set to zero (call this function f).
3. Rearrange the equation s.t. one of parameters appears on the LHS.
4. Initialize the parameters.
5. For i in $\{1,\ldots,K\}$, update each parameter and increment t:
6. Repeat #5 until convergence

Fixed Point Iteration for Optimization

- Fixed point iteration is a general tool for solving systems of equations
- It can also be applied to optimization.

$$J(x) = \frac{x^3}{3} + \frac{3}{2}x^2 + 2x$$

$$\frac{dJ(x)}{dx} = f(x) = x^2 - 3x + 2 = 0$$

$$\Rightarrow x = \frac{x^2 + 2}{3} = g(x)$$

$$x \leftarrow \frac{x^2 + 2}{3}$$

1. Given objective function:
2. Compute derivative, set to zero (call this function f).
3. Rearrange the equation s.t. one of parameters appears on the LHS.
4. Initialize the parameters.
5. For i in $\{1,\ldots,K\}$, update each parameter and increment t:
6. Repeat #5 until convergence

Fixed Point Iteration for Optimization

We can implement our example in a few lines of python.

$$J(x) = \frac{x^3}{3} + \frac{3}{2}x^2 + 2x$$

$$\frac{dJ(x)}{dx} = f(x) = x^2 - 3x + 2 = 0$$

$$\Rightarrow x = \frac{x^2 + 2}{3} = g(x)$$

$$x \leftarrow \frac{x^2 + 2}{3}$$

```
def f1(x):
    '''f(x) = x^2 - 3x + 2'''
    return x**2 - 3.*x + 2.

def g1(x):
    '''g(x) = \frac{x^2 + 2}{3}'''
    return (x**2 + 2.) / 3.

def fpi(g, x0, n, f):
    '''Optimizes the 1D function g by fixed point iteration
    starting at x0 and stopping after n iterations. Also
    includes an auxiliary function f to test at each value.'''
    x = x0
    for i in range(n):
        print("i=%2d x=%.4f f(x)=%.4f" % (i, x, f(x)))
        x = g(x)
    i += 1
    print("i=%2d x=%.4f f(x)=%.4f" % (i, x, f(x)))
    return x

if __name__ == "__main__":
    x = fpi(g1, 0, 20, f1)
```

Fixed Point Iteration for Optimization

$$J(x) = \frac{x^3}{3} + \frac{3}{2}x^2 + 2x$$

$$\frac{dJ(x)}{dx} = f(x) = x^2 - 3x + 2 = 0$$

$$\Rightarrow x = \frac{x^2 + 2}{3} = g(x)$$

$$x \leftarrow \frac{x^2 + 2}{3}$$

```
$ python fixed-point-iteration.py
i= 0 x=0.0000 f(x)=2.0000
i= 1 x=0.6667 f(x)=0.4444
i= 2 x=0.8148 f(x)=0.2195
i= 3 x=0.8880 f(x)=0.1246
i= 4 x=0.9295 f(x)=0.0755
i= 5 x=0.9547 f(x)=0.0474
i= 6 x=0.9705 f(x)=0.0304
i= 7 x=0.9806 f(x)=0.0198
i= 8 x=0.9872 f(x)=0.0130
i= 9 x=0.9915 f(x)=0.0086
i=10 x=0.9944 f(x)=0.0057
i=11 x=0.9963 f(x)=0.0038
i=12 x=0.9975 f(x)=0.0025
i=13 x=0.9983 f(x)=0.0017
i=14 x=0.9989 f(x)=0.0011
i=15 x=0.9993 f(x)=0.0007
i=16 x=0.9995 f(x)=0.0005
i=17 x=0.9997 f(x)=0.0003
i=18 x=0.9998 f(x)=0.0002
i=19 x=0.9999 f(x)=0.0001
i=20 x=0.9999 f(x)=0.0001
```

Iterative Proportional Fitting (IPF)

IPF applies fixed point iteration to the derivative of the likelihood objective

$$\ell(\theta; \mathcal{D}) = \sum_{n=1}^{N} \log p(\boldsymbol{x}^{(n)} \mid \boldsymbol{\theta})$$

$$\frac{\partial \ell}{\partial \psi_c(\mathbf{x}_c)} = \frac{m(\mathbf{x}_c)}{\psi_c(\mathbf{x}_c)} - N \frac{p(\mathbf{x}_c)}{\psi_c(\mathbf{x}_c)}$$

$$\psi_c(\boldsymbol{x}_c) = \psi_c(\boldsymbol{x}_c) \frac{\tilde{p}(\boldsymbol{x}_c)}{p(\boldsymbol{x}_c)}$$

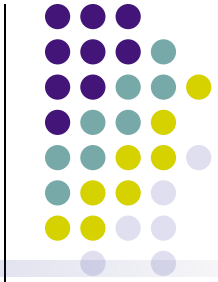
$$\psi_c^{(t+1)}(\mathbf{x}_c) = \psi_c^{(t)}(\mathbf{x}_c) \frac{\tilde{p}(\mathbf{x}_c)}{p^{(t)}(\mathbf{x}_c)}$$

1. Given likelihood objective
2. Compute derivative, set to zero
3. Rearrange the equation s.t. one of potentials appears on the LHS.
4. Initialize the potential tables.
5. For each clique c in C, update each potential table and increment t:
6. Repeat #5 until convergence

Need to do inference here

$$p(\boldsymbol{x}_C) = \sum_{\boldsymbol{x}' : \boldsymbol{x}'_C = \boldsymbol{x}_C} p(\boldsymbol{x}' \mid \boldsymbol{\theta})$$

Properties of IPF Updates



- Applies only when potentials are parameterized as:

$$\psi_C(\boldsymbol{x}_C) = \theta_{C,\boldsymbol{x}_C}$$

- IPF iterates a set of fixed-point equations:

$$\psi_c^{(t+1)}(\mathbf{x}_c) = \psi_c^{(t)}(\mathbf{x}_c)\frac{\tilde{p}(\mathbf{x}_c)}{p^{(t)}(\mathbf{x}_c)}$$

- However, we can prove it is also a coordinate ascent algorithm (coordinates = parameters of clique potentials).

- Hence at each step, it will increase the log-likelihood, and it will converge to a global maximum.

Extra slides from 2015

KL Divergence View

- IPF can be seen as coordinate ascent in the likelihood using the way of expressing likelihoods using KL divergences.
- We can show that maximizing the log likelihood is equivalent to minimizing the KL divergence (cross entropy) from the observed distribution to the model distribution:

$$\max \ell \Leftrightarrow \min KL\left(\tilde{p}(x) \| p(x \mid \theta)\right) = \sum_x \tilde{p}(x) \log \frac{\tilde{p}(x)}{p(x \mid \theta)}$$

- Using a property of KL divergence based on the conditional chain rule: $p(x) = p(x_a)p(x_b|x_a)$:

$$\begin{aligned} KL\left(q(x_a, x_b) \| p(x_a, x_b)\right) &= \sum_{x_a, x_b} q(x_a) q(x_b \mid x_a) \log \frac{q(x_a) q(x_b \mid x_a)}{p(x_a) p(x_b \mid x_a)} \\ &= \sum_{x_a, x_b} q(x_a) q(x_b \mid x_a) \log \frac{q(x_a)}{p(x_a)} + \sum_{x_a, x_b} q(x_a) q(x_b \mid x_a) \log \frac{q(x_b \mid x_a)}{p(x_b \mid x_a)} \\ &= KL\left(q(x_a) \| p(x_a)\right) + \sum_{x_a} q(x_a) KL\left(q(x_b \mid x_a) \| p(x_b \mid x_a)\right) \end{aligned}$$

Extra slides from 2015

IPF minimizes KL divergence

- Putting things together, we have

$$KL\left(\tilde{p}(\mathbf{x}) \,\|\, p(\mathbf{x}\,|\,\theta)\right) = KL\left(\tilde{p}(\mathbf{x}_c) \,\|\, p(\mathbf{x}_c\,|\,\theta)\right) + \sum_{x_a} \tilde{p}(\mathbf{x}_c) KL\left(\tilde{p}(\mathbf{x}_{-c}\,|\,\mathbf{x}_c) \,\|\, p(\mathbf{x}_{-c}\,|\,\mathbf{x}_c)\right)$$

  It can be shown that changing the clique potential ψ_c has no effect on the conditional distribution, so the second term in unaffected.

- To minimize the first term, we set the marginal to the observed marginal, just as in IPF.
  - Note that this is only good when the model is decomposable !

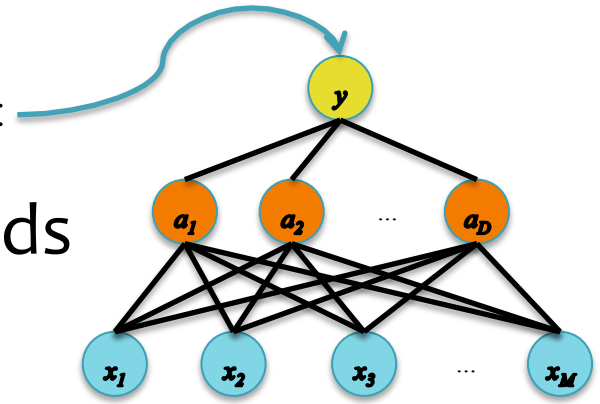
- We can interpret IPF updates as retaining the "old" conditional probabilities $p^{(t)}(\mathbf{x}_{-c}|\mathbf{x}_c)$ while replacing the "old" marginal probability $p^{(t)}(\mathbf{x}_c)$ with the observed marginal $\tilde{p}(\mathbf{x}_c)$.

Options for MLE of MRFs

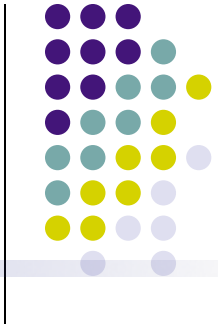
- **Setting I:** $\psi_C(\boldsymbol{x}_C) = \theta_{C,\boldsymbol{x}_C}$
  A. MLE by inspection (Decomposable Models)
  B. Iterative Proportional Fitting (IPF)

- **Setting II:** $\psi_C(\boldsymbol{x}_C) = \exp(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}_C))$
  C. Generalized Iterative Scaling
  D. Gradient-based Methods

- **Setting III:** $\psi_C(\boldsymbol{x}_C) =$
  D. Gradient-based Methods

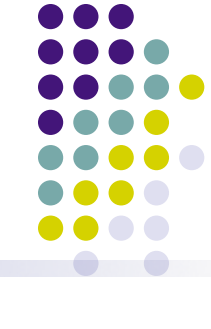


Feature-based Clique Potentials



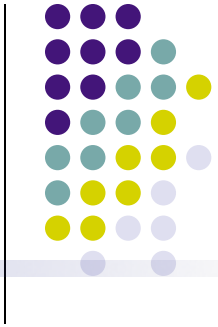
- So far we have discussed the most general form of an undirected graphical model in which cliques are parameterized by general **"tabular"** potential functions $\psi_c(\mathbf{x}_c)$.
- But for large cliques these general potentials are exponentially costly for inference and have exponential numbers of parameters that we must learn from limited data.
- One solution: change the graphical model to make cliques smaller. But this changes the dependencies, and may force us to make more independence assumptions than we would like.
- Another solution: keep the same graphical model, but use a less general parameterization of the clique potentials.
- This is the idea behind feature-based models.

Features



- Consider a clique $\mathbf{x}_c$ of random variables in a UGM, e.g. three consecutive characters $c_1 c_2 c_3$ in a string of English text.
- How would we build a model of $p(c_1 c_2 c_3)$?
  - If we use a single clique function over $c_1 c_2 c_3$, the full joint clique potential would be huge: $26^3 - 1$ parameters.
  - However, we often know that some particular joint settings of the variables in a clique are quite likely or quite unlikely. e.g. ing, ate, ion, ?ed, qu?, jkx, zzz,...
- A "feature" is a function which is vacuous over all joint settings except a few particular ones on which it is high or low.
  - For example, we might have $f_{ing}(c_1 c_2 c_3)$ which is 1 if the string is 'ing' and 0 otherwise, and similar features for '?ed', etc.
- We can also define features when the inputs are continuous. Then the idea of a cell on which it is active disappears, but we might still have a compact parameterization of the feature.

Features as Micropotentials



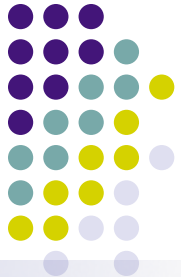
- By exponentiating them, each feature function can be made into a "micropotential". We can **multiply** these micropotentials together to get a **clique potential**.
- Example: a clique potential $\psi(c_1 c_2 c_3)$ could be expressed as:

$$\psi_c(c_1, c_2, c_3) = e^{\theta_{\text{ing}} f_{\text{ing}}} \times e^{\theta_{?\text{ed}} f_{?\text{ed}}} \times \ldots$$

$$= \exp\left\{\sum_{k=1}^{K} \theta_k f_k(c_1, c_2, c_3)\right\}$$

- This is still a potential over 26^3 possible settings, but only uses K parameters if there are K features.
  - By having one indicator function per combination of $\mathbf{x}_c$, we recover the standard tabular potential.

Combining Features



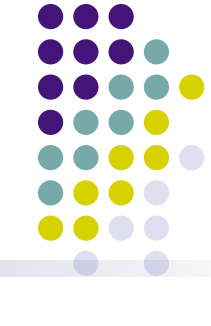
- Each feature has a weight θ_k which represents the numerical strength of the feature and whether it increases or decreases the probability of the clique.
- The marginal over the clique is a generalized exponential family distribution, actually, a **GLIM**:

$$p(c_1,c_2,c_3) \propto \exp\left\{\begin{array}{l}\theta_{\text{ing}} f_{\text{ing}}(c_1,c_2,c_3) + \theta_{?\text{ed}} f_{?\text{ed}}(c_1,c_2,c_3) + \\ \theta_{\text{qu?}} f_{\text{qu?}}(c_1,c_2,c_3) + \theta_{\text{zzz}} f_{\text{zzz}}(c_1,c_2,c_3) + \cdots\end{array}\right\}$$

- In general, the features may be overlapping, unconstrained indicators or any function of any subset of the clique variables:

$$\psi_c(\mathbf{x}_c) \overset{\text{def}}{=} \exp\left\{\sum_{i \in \mathcal{I}_c} \theta_k f_k(\mathbf{x}_{c_i})\right\}$$

Feature Based Model



- We can multiply these clique potentials as usual:

$$p(\mathbf{x}) = \frac{1}{Z(\theta)} \prod_c \psi_c(\mathbf{x}_c) = \frac{1}{Z(\theta)} \exp\left\{ \sum_c \sum_{i \in \mathcal{I}_c} \theta_k f_k(\mathbf{x}_{c_i}) \right\}$$

- However, in general we can forget about associating features with cliques and just use a simplified form:

$$p(\mathbf{x}) = \frac{1}{Z(\theta)} \exp\left\{ \sum_i \theta_i f_i(\mathbf{x}_{c_i}) \right\}$$

- This is just our friend the **exponential family model**, with the features as sufficient statistics!

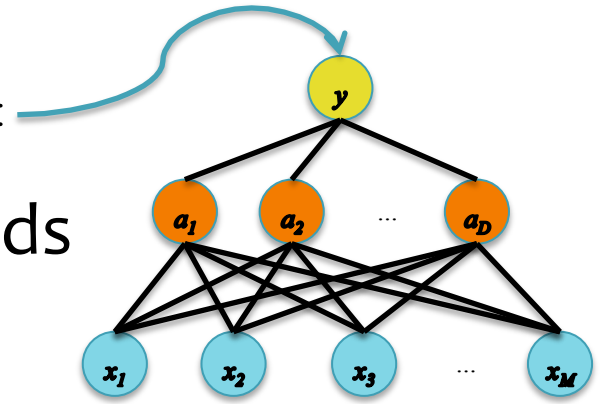
- Learning: recall that in IPF, we have $\psi_c^{(t+1)}(\mathbf{x}_c) = \psi_c^{(t)}(\mathbf{x}_c) \frac{\tilde{p}(\mathbf{x}_c)}{p^{(t)}(\mathbf{x}_c)}$
  - **Not obvious how to use this rule to update the weights and features individually !!!**

Options for MLE of MRFs

- **Setting I:** $\psi_C(\boldsymbol{x}_C) = \theta_{C,\boldsymbol{x}_C}$
  A. MLE by inspection (Decomposable Models)
  B. Iterative Proportional Fitting (IPF)

- **Setting II:** $\psi_C(\boldsymbol{x}_C) = \exp(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}_C))$
  C. Generalized Iterative Scaling
  D. Gradient-based Methods

- **Setting III:** $\psi_C(\boldsymbol{x}_C) =$
  D. Gradient-based Methods



Generalized Iterative Scaling (GIS)

Key idea:

- Define a function which lower-bounds the log-likelihood
- Observe that the bound is tight at current parameters
- **Increase lower-bound** by fixed-point iteration in order to **increase log-likelihood**

Side note: This idea is akin to a standard derivation of the Expectation-Maximization (EM) algorithm

Generalized Iterative Scaling (GIS)

GIS applies fixed point iteration to the derivative of a lower-bound of the likelihood objective

$$\tilde{\ell}(\theta;\mathcal{D}) = \frac{1}{N}\sum_{n=1}^{N}\log p(\boldsymbol{x}^{(n)} \mid \boldsymbol{\theta})$$

$$\tilde{\ell}(\theta;\mathcal{D}) \geq \Lambda(\theta)$$

$$\frac{\partial \Lambda}{\partial \theta_i} = \sum_x \tilde{p}(x) f_i(x) - \exp\left(\Delta\theta_i^{(t)}\right)\sum_x p(x \mid \theta^{(t)}) f_i(x)$$

$$\theta_i^{(t+1)} = \theta_i^{(t)} + \log\left(\frac{\sum_x \tilde{p}(x) f_i(x)}{\sum_x p^{(t)}(x) f_i(x)}\right)$$

1. Given avg. likelihood objective
2. Derive lower bound
3. Compute derivative of bound, set to zero
4. Rearrange the equation s.t. one parameter appears on the LHS.
5. Initialize the parameters.
6. For each i in $\{1,\ldots K\}$, update each parameter and increment t:
7. Repeat #6 until convergence

The lower bound is obtained by linearizing a log and applying Jensen-Shannon.

$$\tilde{\ell}(\theta;\mathcal{D}) \geq \sum_i \theta_i \sum_x \tilde{p}(x) f_i(x) - \sum_x p(x \mid \theta^{(t)}) \sum_i f_i(x) \exp\left(\Delta\theta_i^{(t)}\right) - \log Z(\theta^{(t)}) + 1 \stackrel{\text{def}}{=} \Lambda(\theta)$$

Extra slides from 2015

MLE of Feature Based UGMs

- Scaled likelihood function

$$\tilde{\ell}(\theta;D) = \ell(\theta;D)/N = \frac{1}{N}\sum_n \log p(x_n \mid \theta)$$

$$= \sum_x \tilde{p}(x) \log p(x \mid \theta)$$

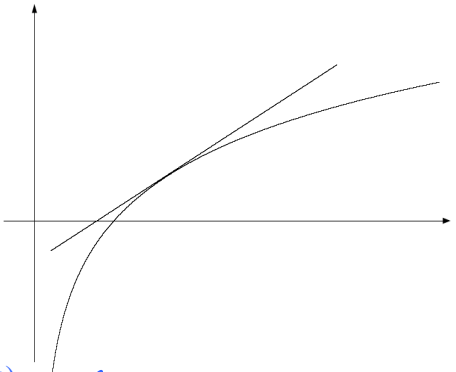
$$= \sum_x \tilde{p}(x) \sum_i \theta_i f_i(x) - \log Z(\theta)$$

- Instead of optimizing this objective directly, we attack its lower bound
  - The logarithm has a linear upper bound …

$$\log Z(\theta) \le \mu Z(\theta) - \log \mu - 1$$

  - This bound holds for all μ, in particular, for

$$\mu = Z^{-1}(\theta^{(t)})$$



  - Thus we have

$$\tilde{\ell}(\theta;D) \ge \sum_x \tilde{p}(x) \sum_i \theta_i f_i(x) - \frac{Z(\theta)}{Z(\theta^{(t)})} - \log Z(\theta^{(t)}) + 1$$

Generalized Iterative Scaling (GIS)

Extra slides from 2015

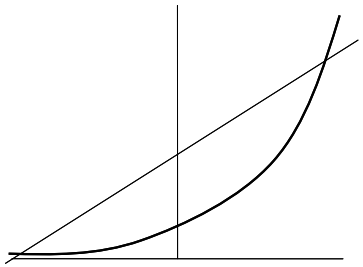
- Lower bound of scaled loglikelihood

$$\tilde{\ell}(\theta;D) \geq \sum_x \tilde{p}(x)\sum_i \theta_i f_i(x) - \frac{Z(\theta)}{Z(\theta^{(t)})} - \log Z(\theta^{(t)}) + 1$$

- Define $\Delta\theta_i^{(t)} \overset{\text{def}}{=} \theta_i - \theta_i^{(t)}$

$$\begin{aligned}
\tilde{\ell}(\theta;D) &\geq \sum_x \tilde{p}(x)\sum_i \theta_i f_i(x) - \frac{1}{Z(\theta^{(t)})}\sum_x \exp\left\{\sum_i \theta_i f_i(x)\right\} - \log Z(\theta^{(t)}) + 1 \\
&= \sum_i \theta_i \sum_x \tilde{p}(x) f_i(x) - \frac{1}{Z(\theta^{(t)})}\sum_x \exp\left\{\sum_i \theta_i^{(t)} f_i(x)\right\}\exp\left\{\sum_i \Delta\theta_i^{(t)} f_i(x)\right\} - \log Z(\theta^{(t)}) + 1 \\
&= \sum_i \theta_i \sum_x \tilde{p}(x) f_i(x) - \sum_x p(x\,|\,\theta^{(t)})\exp\left\{\sum_i \Delta\theta_i^{(t)} f_i(x)\right\} - \log Z(\theta^{(t)}) + 1
\end{aligned}$$

- Relax again
  - Assume $f_i(x) \geq 0, \quad \sum_i f_i(x) = 1$
  - Convexity of exponential: $\exp\left(\sum_i \pi_i x_i\right) \leq \sum_i \pi_i \exp(x_i)$



- We have:

$$\tilde{\ell}(\theta;D) \geq \sum_i \theta_i \sum_x \tilde{p}(x) f_i(x) - \sum_x p(x\,|\,\theta^{(t)}) \sum_i f_i(x)\exp\left(\Delta\theta_i^{(t)}\right) - \log Z(\theta^{(t)}) + 1 \overset{\text{def}}{=} \Lambda(\theta)$$

GIS

Extra slides from 2015

- Lower bound of scaled loglikelihood

$$\tilde{\ell}(\theta;D) \geq \sum_i \theta_i \sum_x \tilde{p}(x) f_i(x) - \sum_x p(x \mid \theta^{(t)}) \sum_i f_i(x) \exp\left(\Delta\theta_i^{(t)}\right) - \log Z(\theta^{(t)}) + 1 \overset{\text{def}}{=} \Lambda(\theta)$$

- Take derivative:

$$\frac{\partial \Lambda}{\partial \theta_i} = \sum_x \tilde{p}(x) f_i(x) - \exp\left(\Delta\theta_i^{(t)}\right) \sum_x p(x \mid \theta^{(t)}) f_i(x)$$

- Set to zero

$$e^{\Delta\theta_i^{(t)}} = \frac{\sum_x \tilde{p}(x) f_i(x)}{\sum_x p(x \mid \theta^{(t)}) f_i(x)} = \frac{\sum_x \tilde{p}(x) f_i(x)}{\sum_x p^{(t)}(x) f_i(x)} Z(\theta^{(t)})$$

  - where $p^{(t)}(x)$ is the unnormalized version of $p(x|\theta^{(t)})$

- Update

$$\theta_i^{(t+1)} = \theta_i^{(t)} + \Delta\theta_i^{(t)} \Rightarrow p^{(t+1)}(x) = p^{(t)}(x) \prod_i e^{\Delta\theta_i^{(t)} f_i(x)}$$

$$p^{(t+1)}(x) = \frac{p^{(t)}(x)}{Z(\theta^{(t)})} \prod_i \left(\frac{\sum_x \tilde{p}(x) f_i(x)}{\sum_x p^{(t)}(x) f_i(x)} Z(\theta^{(t)}) \right)^{f_i(x)}$$

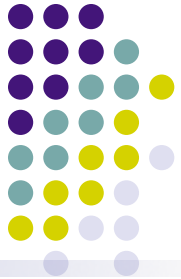
$$\Rightarrow = \frac{p^{(t)}(x)}{Z(\theta^{(t)})} \prod_i \left(\frac{\sum_x \tilde{p}(x) f_i(x)}{\sum_x p^{(t)}(x) f_i(x)} \right)^{f_i(x)} \left(Z(\theta^{(t)}) \right)^{\sum_i f_i(x)}$$

$$= p^{(t)}(x) \prod_i \left(\frac{\sum_x \tilde{p}(x) f_i(x)}{\sum_x p^{(t)}(x) f_i(x)} \right)^{f_i(x)}$$

Recall IPF:

$$\psi_c^{(t+1)}(\mathbf{x}_c) = \psi_c^{(t)}(\mathbf{x}_c) \frac{\tilde{p}(\mathbf{x}_c)}{p^{(t)}(\mathbf{x}_c)}$$

Contrast of IPF and GIS



- IPF is a general algorithm for finding MLE of UGMs.
  - a **fixed-point equation** for ψ_c over single cliques, coordinate ascent
  - Requires the potential to be fully parameterized
  - The clique described by the potentials do not have to be max-clique
  - For fully decomposable model, reduces to a single step iteration
- GIS
  - Iterative scaling on general UGM with feature-based potentials
  - IPF is a special case of GIS which the clique potential is built on features defined as an indicator function of clique configurations.

GIS:

$$p^{(t+1)}(x) = p^{(t)}(x)\prod_i \left(\frac{\sum_x \tilde{p}(x) f_i(x)}{\sum_x p^{(t)}(x) f_i(x)}\right)^{f_i(x)}$$

$$\theta_i^{(t+1)} = \theta_i^{(t)} + \log\left(\frac{\sum_x \tilde{p}(x) f_i(x)}{\sum_x p^{(t)}(x) f_i(x)}\right)$$

IPF:

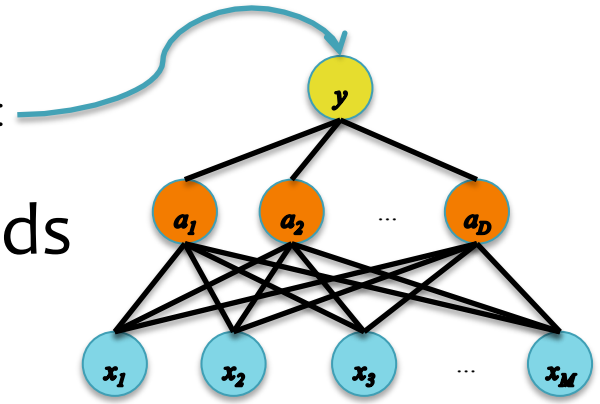
$$\psi_c^{(t+1)}(\mathbf{x}_c) = \psi_c^{(t)}(\mathbf{x}_c)\frac{\tilde{p}(\mathbf{x}_c)}{p^{(t)}(\mathbf{x}_c)}$$

Options for MLE of MRFs

- **Setting I:** $\psi_C(\boldsymbol{x}_C) = \theta_{C,\boldsymbol{x}_C}$
  A. MLE by inspection (Decomposable Models)
  B. Iterative Proportional Fitting (IPF)

- **Setting II:** $\psi_C(\boldsymbol{x}_C) = \exp(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}_C))$
  C. Generalized Iterative Scaling
  D. Gradient-based Methods

- **Setting III:** $\psi_C(\boldsymbol{x}_C) =$
  D. Gradient-based Methods



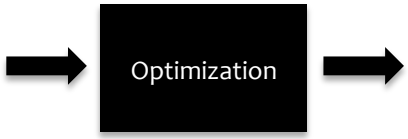
Recipe for Gradient-based Learning

1. Write down the objective function
2. Compute the partial derivatives of the objective (i.e. gradient, and maybe Hessian)
3. Feed objective function and derivatives into black box

4. Retrieve optimal parameters from black box

Optimization Algorithms

What is the black box?



- Newton's method
- Hessian-free / Quasi-Newton methods
  - Conjugate gradient
  - L-BFGS
- Stochastic gradient methods
  - Stochastic gradient descent (SGD)
  - Stochastic meta-descent
  - AdaGrad

Whiteboard

- Gradient of MRF log-likelihood for feature-based potentials
- Gradient of CRF log-likelihood for feature-based potentials [next time]
- L1 and L2 regularization

Practical Considerations for Gradient-based Methods

- Overfitting
  - L2 regularization
  - L1 regularization
  - Regularization by early stopping
- For SGD: Sparse updates

"Empirical" Comparison of Parameter Estimation Methods

- **Example NLP task:** CRF dependency parsing
- **Suppose:** Training time is dominated by inference
- **Dataset:** One million tokens
- **Inference speed:** 1,000 tokens / sec
- ➔ 0.27 hours per pass through dataset

	# passes through data to converge	# hours to converge
GIS	1000+	270
L-BFGS	100+	27
SGD	10	~3

Summary

Setting I: $\psi_C(\boldsymbol{x}_C) = \theta_{C,\boldsymbol{x}_C}$

Setting II: $\psi_C(\boldsymbol{x}_C) = \exp(\boldsymbol{\theta} \cdot \boldsymbol{f}(\boldsymbol{x}_C))$

A. MLE by inspection (Decomposable Models)
- Very limited applicability
- Exemplifies the need for general algorithms

B. Iterative Proportional Fitting (IPF)
- Guaranteed to converge
- Only applies to "tabular" potential functions

C. Generalized Iterative Scaling (GIS)
- Maximizes a lower-bound of log-likelihood
- Iterative algorithm (like IPF), but more broadly applies to exponential family potentials

D. Gradient-based Methods
- Doesn't require fancy optimization algorithms (i.e. SGD works great)
- Faster convergence than GIS
- Applies to arbitrary potentials [later in the course]

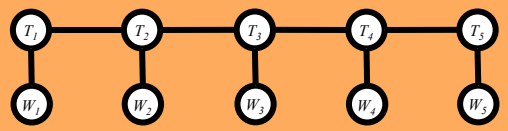
MLE for Undirected GMs

1. Data

$$\mathcal{D} = \{\boldsymbol{x}^{(n)}\}_{n=1}^{N}$$

2. Model

$$p(\boldsymbol{x} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

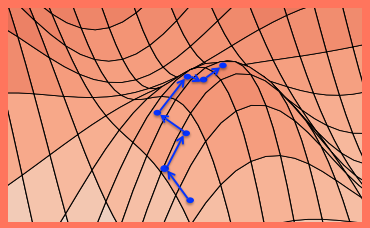


3. Objective

$$\ell(\theta; \mathcal{D}) = \sum_{n=1}^{N} \log p(\boldsymbol{x}^{(n)} \mid \boldsymbol{\theta})$$

4. Learning

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, \ell(\boldsymbol{\theta}; \mathcal{D})$$



5. Inference

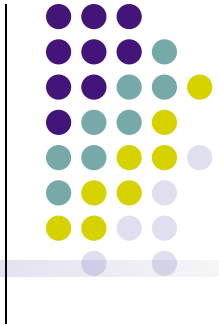
1. Marginal Inference

$$p(\boldsymbol{x}_C) = \sum_{\boldsymbol{x}' : \boldsymbol{x}'_C = \boldsymbol{x}_C} p(\boldsymbol{x}' \mid \boldsymbol{\theta})$$

2. Partition Function

$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

Contrast of MLE for *directed / undirected* GMs



- For directed graphical models, the log-likelihood decomposes into a sum of terms, one per family (node plus parents).
- For undirected graphical models, the log-likelihood does not decompose, because the normalization constant Z is a function of **all** the parameters

$$P(x_1,\ldots,x_n) = \frac{1}{Z}\prod_{c\in C}\psi_c(\mathbf{x}_c) \qquad Z = \sum_{x_1,\ldots,x_n}\prod_{c\in C}\psi_c(\mathbf{x}_c)$$

- In general, we will need to do inference (i.e., marginalization) to learn parameters for undirected models, even in the fully observed case.

5. Inference

Next time:
How to compute these!

Three Tasks:

1. Marginal Inference

Compute marginals of variables and cliques

$$p(x_i) = \sum_{\boldsymbol{x}' : x'_i = x_i} p(\boldsymbol{x}' \mid \boldsymbol{\theta}) \qquad p(\boldsymbol{x}_C) = \sum_{\boldsymbol{x}' : \boldsymbol{x}'_C = \boldsymbol{x}_C} p(\boldsymbol{x}' \mid \boldsymbol{\theta})$$

2. Partition Function

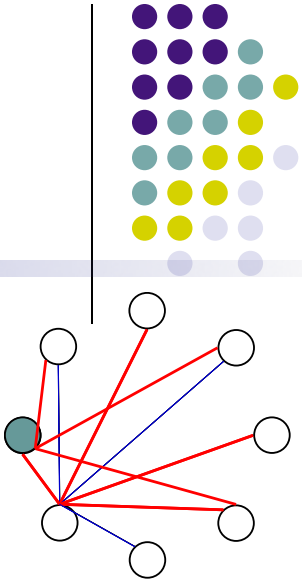
Compute the normalization constant

$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

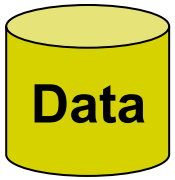
3. MAP Inference

Compute variable assignment with highest probability

$$\hat{\boldsymbol{x}} = \operatorname*{argmax}_{\boldsymbol{x}} \; p(\boldsymbol{x} \mid \boldsymbol{\theta})$$



ML Structural Learning via Neighborhood Selection for **completely observed MRF**



$(x_1^{(1)},\ldots,x_n^{(1)})$
$(x_1^{(2)},\ldots,x_n^{(2)})$
$\ldots$
$(x_1^{(M)},\ldots,x_n^{(M)})$

Gaussian Graphical Models

- Multivariate Gaussian density:

$$p(\mathbf{x} \mid \mu, \Sigma) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left\{-\tfrac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1}(\mathbf{x}-\mu)\right\}$$

- WOLG: let $\mu = 0 \quad Q = \Sigma^{-1}$

$$p(x_1, x_2, \cdots, x_p \mid \mu = 0, Q) = \frac{|Q|^{1/2}}{(2\pi)^{n/2}} \exp\left\{-\tfrac{1}{2}\sum_i q_{ii}(x_i)^2 - \sum_{i<j} q_{ij} x_i x_j\right\}$$

- We can view this as a continuous Markov Random Field with potentials defined on every node and edge:

Pairwise MRF (e.g., Ising Model)

- Assuming the nodes are discrete, and edges are weighted, then for a sample x_d, we have

$$P(\mathbf{x}_d|\Theta) = \exp\left(\sum_{i\in V}\theta_{ii}^t x_{d,i} + \sum_{(i,j)\in E}\theta_{ij}x_{d,i}x_{d,j} - A(\Theta)\right)$$

The covariance and the precision matrices

- Covariance matrix Σ

$$\Sigma_{i,j} = 0 \quad \Rightarrow \quad X_i \perp X_j \quad \text{or} \quad p(X_i, X_j) = p(X_i)p(X_j)$$

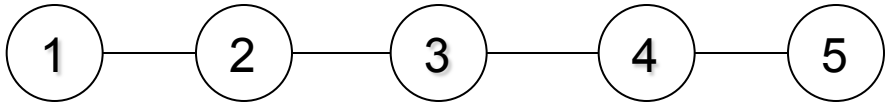
  - Graphical model interpretation?

- Precision matrix $Q = \Sigma^{-1}$

$$Q_{i,j} = 0 \quad \Rightarrow \quad X_i \perp X_j | \mathbf{X}_{-ij} \quad \text{or} \quad p(X_i, X_j | \mathbf{X}_{-ij}) = p(X_i | \mathbf{X}_{-ij}) p(X_j | \mathbf{X}_{-ij})$$

  - Graphical model interpretation?

Sparse precision vs. sparse covariance in GGM



$$\Sigma^{-1} = \begin{pmatrix} 1 & 6 & 0 & 0 & 0 \\ 6 & 2 & 7 & 0 & 0 \\ 0 & 7 & 3 & 8 & 0 \\ 0 & 0 & 8 & 4 & 9 \\ 0 & 0 & 0 & 9 & 5 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} 0.10 & 0.15 & -0.13 & -0.08 & 0.15 \\ 0.15 & -0.03 & 0.02 & 0.01 & -0.03 \\ -0.13 & 0.02 & 0.10 & 0.07 & -0.12 \\ -0.08 & 0.01 & 0.07 & -0.04 & 0.07 \\ 0.15 & -0.03 & -0.12 & 0.07 & 0.08 \end{pmatrix}$$

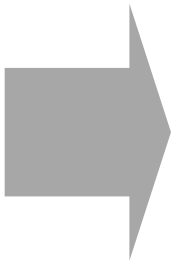
$$\Sigma^{-1}_{15} = 0 \Leftrightarrow X_1 \perp X_5 \,\big|\, X_{nbrs(1) \text{ or } nbrs(5)}$$

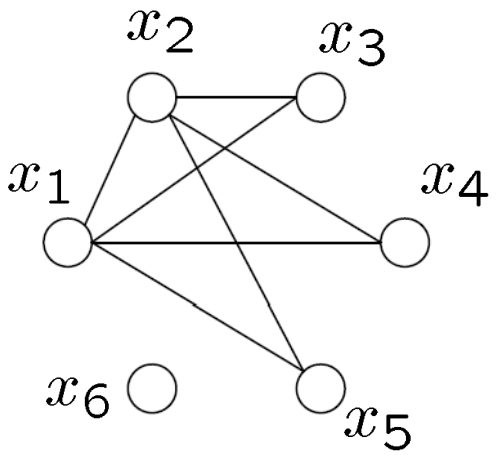
$$\not\Rightarrow$$

$$X_1 \perp X_5 \Leftrightarrow \Sigma_{15} = 0$$

Another example

$$Q = \begin{pmatrix} * & * & * & * & * & 0 \\ * & * & * & * & * & 0 \\ * & * & * & 0 & 0 & 0 \\ * & * & 0 & * & 0 & 0 \\ * & * & 0 & 0 & * & 0 \\ 0 & 0 & 0 & 0 & 0 & * \end{pmatrix}$$





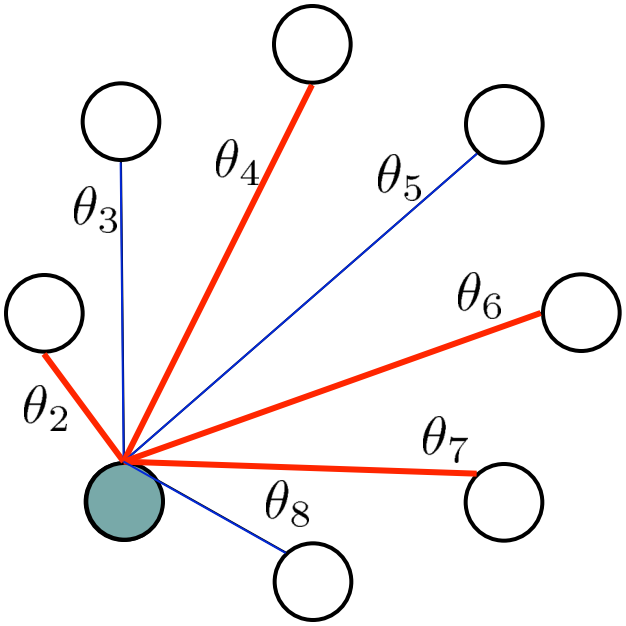
- How to estimate this MRF?
- What if $p >> n$
  - MLE does not exist in general!
  - What about only learning a "sparse" graphical model?
    - This is possible when $s=o(n)$
    - Very often it is the structure of the GM that is more interesting …

Recall lasso

$$\hat{\theta}_i = \arg\min_{\theta_i} l(\theta_i) + \lambda_1 \| \theta_i \|_1$$

$$\text{where } l(\theta_i) = \log P\left(y_i | \mathbf{x}_i, \theta_i\right).$$

Graph Regression

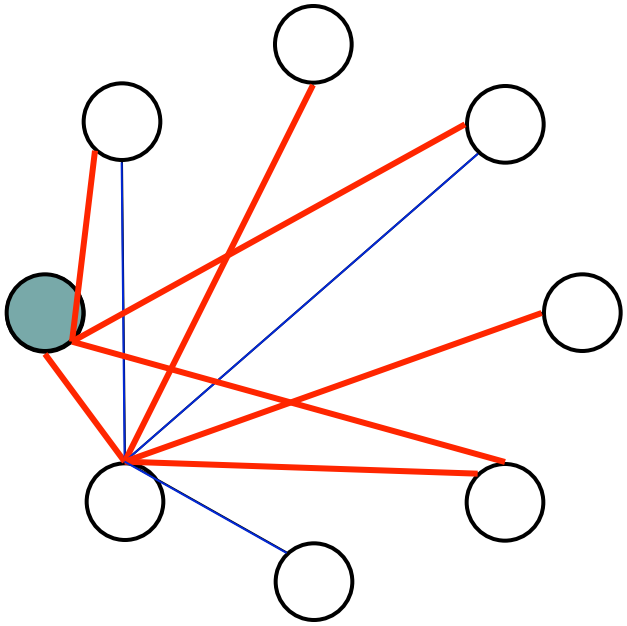


Neighborhood selection

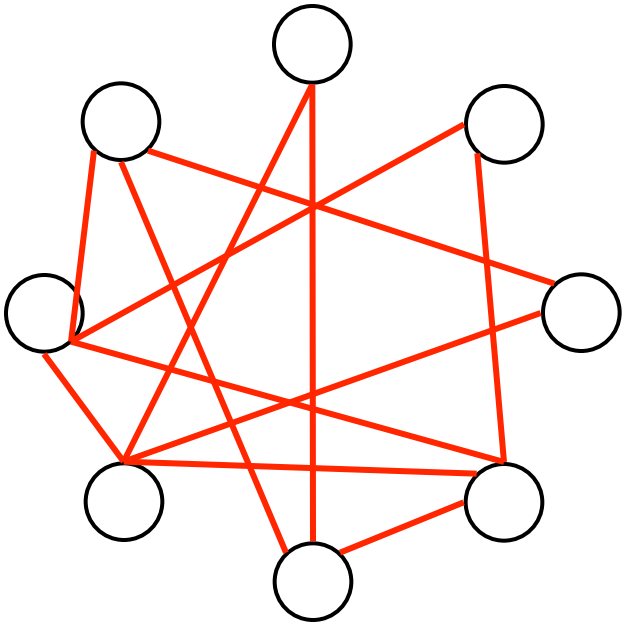
Lasso:

$$\hat{\theta} = \arg\min_{\theta} \sum_{t=1}^{T} l(\theta) + \lambda_1 \| \theta \|_1$$

Graph Regression

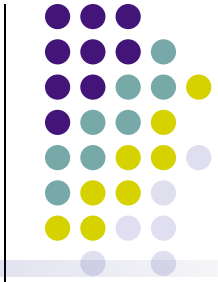


Graph Regression



It can be shown that:
given ***iid*** samples, and under several technical conditions (e.g., "irrepresentable"), the recovered structured is "**sparsistent**" even when p >> n

Learning Ising Model (i.e. pairwise MRF)

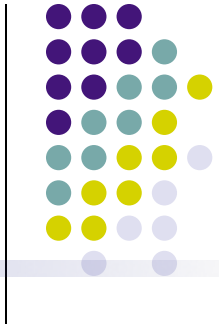


- Assuming the nodes are discrete, and edges are weighted, then for a sample x_d, we have

$$P(\mathbf{x}_d|\Theta) = \exp\left(\sum_{i\in V}\theta_{ii}^t x_{d,i} + \sum_{(i,j)\in E}\theta_{ij}x_{d,i}x_{d,j} - A(\Theta)\right)$$

- It can be shown following the same logic that we can use L_1 regularized logistic regression to obtain a sparse estimate of the neighborhood of each variable in the discrete case.

Consistency



- **Theorem**: for the graphical regression algorithm, under certain verifiable conditions (omitted here for simplicity):

$$\mathbb{P}\left[\hat{G}(\lambda_n) \neq G\right] = \mathcal{O}\left(\exp\left(-Cn^{\epsilon}\right)\right) \to 0$$

Note the from this theorem one should see that the regularizer is not actually used to introduce an "artificial" sparsity bias, but a devise to ensure consistency under finite data and high dimension condition.