Learning Fully Observed Undirected Graphical Models

Kayhan Batmanghelich

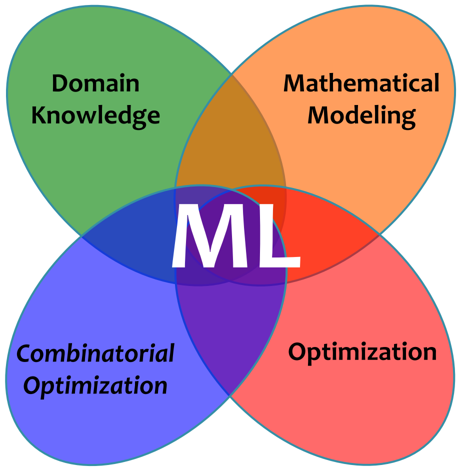
Slides Credit:
Matt Gormley (2016)

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

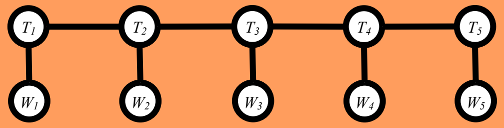
MLE for Undirected GMs

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

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

5. Inference

1. Marginal Inference

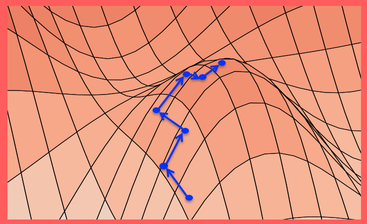
$$p(\boldsymbol{x}_C) = \sum_{\boldsymbol{x}' : \boldsymbol{x}'_C = \boldsymbol{x}_C} p(\boldsymbol{x}' \mid \boldsymbol{\theta})$$

2. Partition Function

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

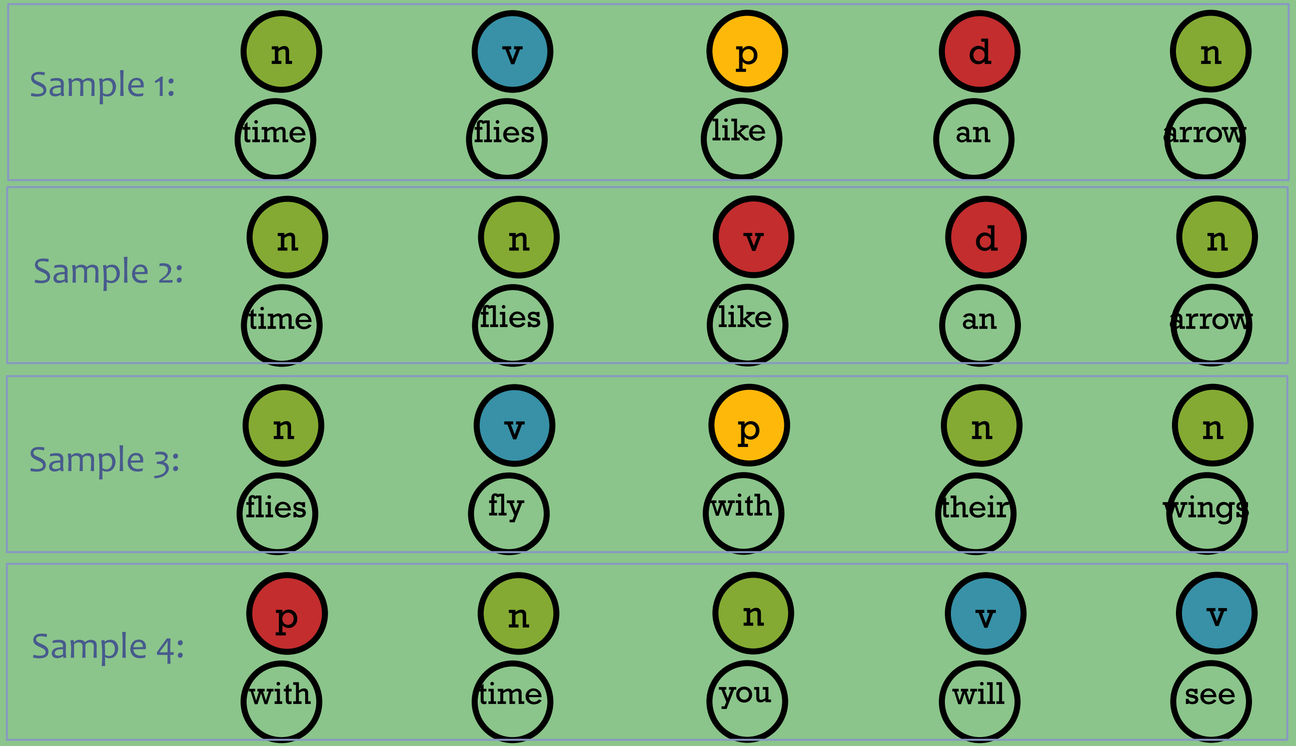
4. Learning

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

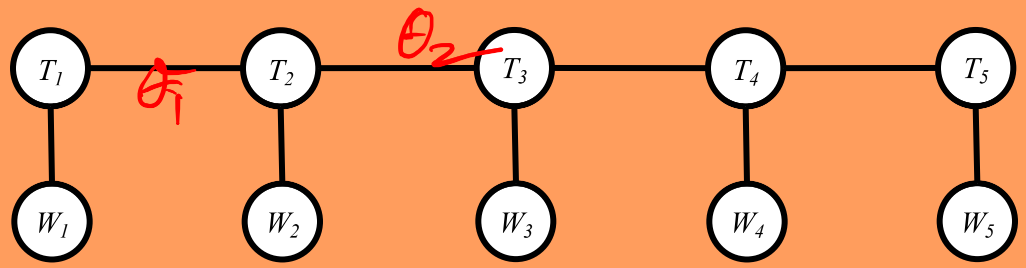


1. Data

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



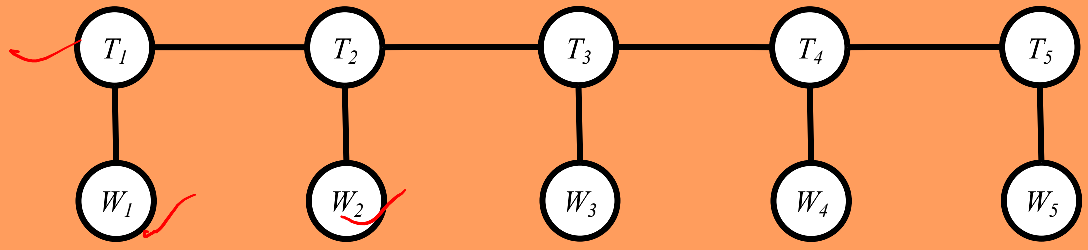
2. Model



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:

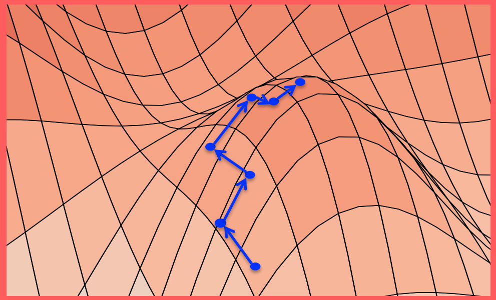
(Assign high probability to the things we observe and low probability to everything else)

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

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:

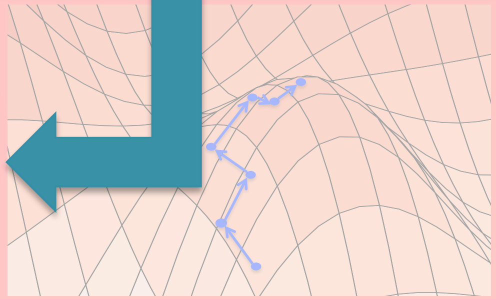
(Assign high probability to the things we observe and low probability to everything else)

Goals for Today's Lecture

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

Tune the parameter function

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



5. Inference

Three Tasks:

1. Marginal Inference
Compute marginals of variables and cliques

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

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

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

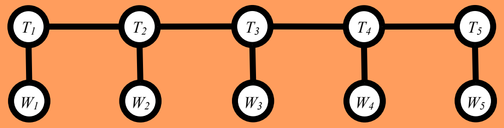
MLE for Undirected GMs

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

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

5. Inference

1. Marginal Inference

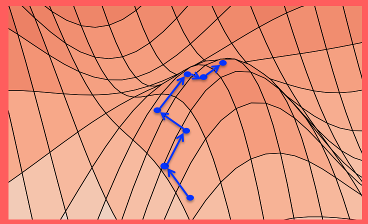
$$p(\boldsymbol{x}_C) = \sum_{\boldsymbol{x}' : \boldsymbol{x}'_C = \boldsymbol{x}_C} p(\boldsymbol{x}' \mid \boldsymbol{\theta})$$

2. Partition Function

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

4. Learning

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

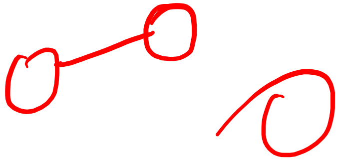


MLE for Undirected GMs

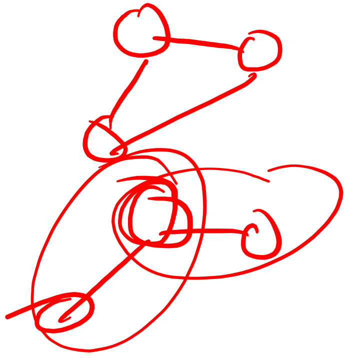
- Today's parameter estimation assumptions:
  1. The graphical model structure is given
  2. Every variable appears in the training examples

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)

θ_{C, x_C}

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

  C. Generalized Iterative Scaling

  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

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

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

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

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

Whiteboard

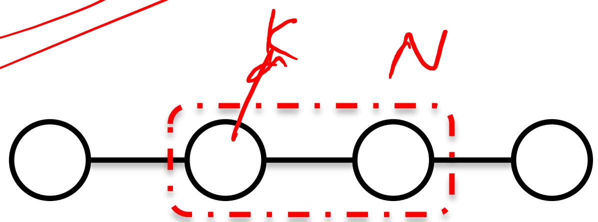
- Derivative of log-likelihood with respect to potentials

Discrete Variables (Tabular clique Potentials)

- Remember categorical distribution

$$p(x = t) \propto \prod_k \theta_k^{\mathbb{I}(x=t)}$$

- Tabular clique potentials look like:



$\phi(\mathcal{Y}_c)$

$$\phi_c(\mathcal{X}_c^n) = \prod_{\mathcal{Y}_c} \phi_c(\mathcal{Y}_c)^{\mathbb{I}[\mathcal{Y}_c = \mathcal{X}_c^n]}$$

Observed values

Lookup table

Param Value for Y_c

- Log likelihood function:

$$L(\phi) = \sum_n \sum_c \sum_{\mathcal{Y}_c} \mathbb{I}[\mathcal{Y}_c = \mathcal{X}_c^n] \log \phi_c(\mathcal{Y}_c) - N \log Z(\phi)$$

Whiteboard

- Derivative of log-likelihood for the tabular clique potentials

$$L(\phi) = \sum_n \sum_c \sum_{\mathcal{Y}_c} \mathbb{I}[\mathcal{Y}_c = \mathcal{X}_c^n] \log \phi_c(\mathcal{Y}_c) - N \log Z(\phi) \qquad Z(\phi) = \sum_{\mathcal{Y}_c} \prod_c \phi_c(\mathcal{Y}_c)$$

Conditions on Clique Marginals

- Derivative of log-likelihood

$$\frac{\partial}{\partial \phi_c(\mathcal{Y}_c)} L(\theta) = \sum_n \mathbb{I}\left[\mathcal{Y}_c = \mathcal{X}_c^n\right] \frac{1}{\phi_c(\mathcal{Y}_c)} - N \frac{p(\mathcal{Y}_c)}{\phi_c(\mathcal{Y}_c)}$$

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

marginal

$$\boxed{p(\mathcal{X}_c) = \epsilon(\mathcal{X}_c)} \qquad \epsilon(\mathcal{X}_c) \equiv \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}\left[\mathcal{X}_c = \mathcal{X}_c^n\right]$$

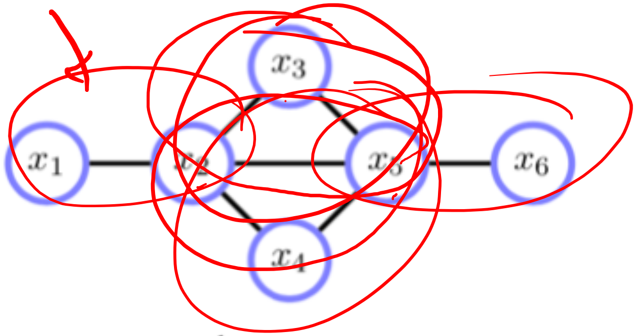
- In other words, at the maximum likelihood setting of the parameters, for each clique, the model marginals must be equal to the observed marginals (empirical counts).
- This doesn't tell us how to get the ML parameters, it just gives us a condition that must be satisfied when we have them.

Options for MLE of MRFs

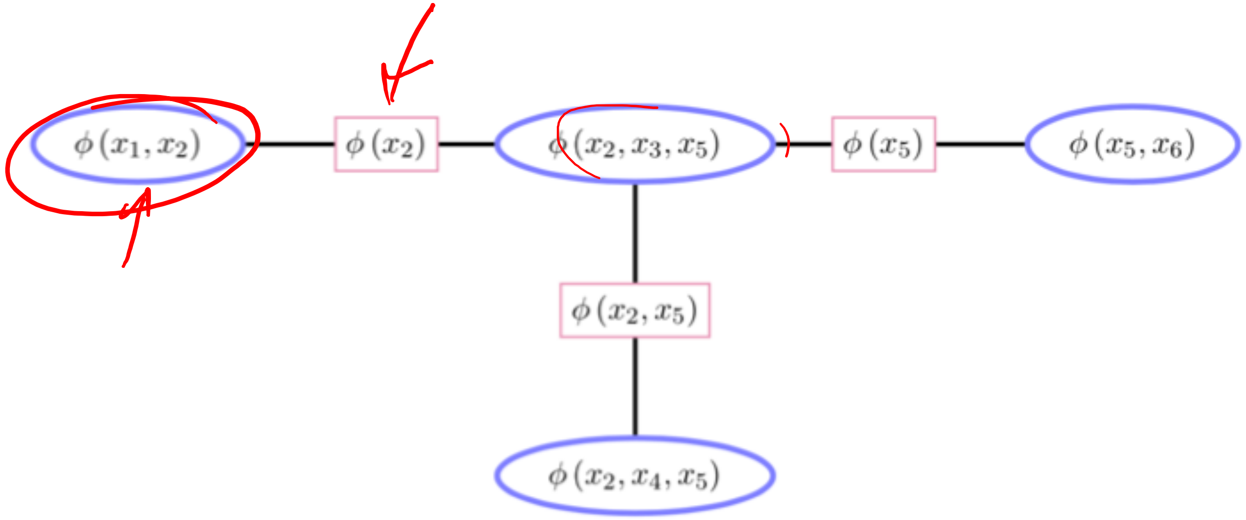
- **Setting I:** $$\psi_C(\boldsymbol{x}_C) = \theta_{C,\boldsymbol{x}_C}$$
  A. MLE by inspection (Decomposable Models) – easy cases
  B. Iterative Proportional Fitting (IPF)

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

Decomposable Graphs

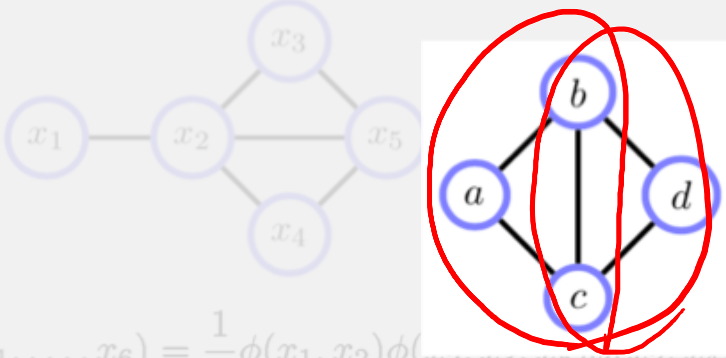


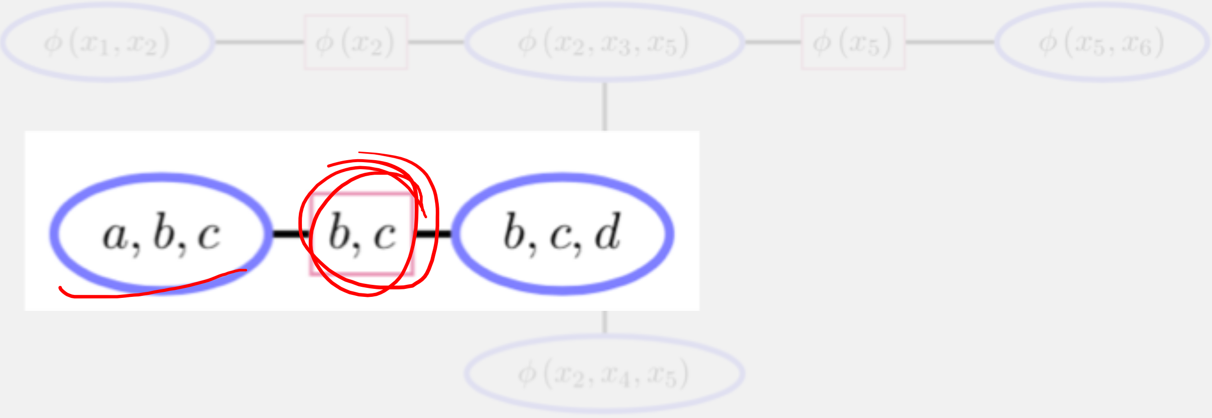
$$p(x_1, \ldots, x_6) = \frac{1}{Z}\phi(x_1, x_2)\phi(x_2, x_3, x_5)\phi(x_2, x_4, x_5)\phi(x_5, x_6)$$



Decomposable Graphs

Remember this from Lectures 6





$$p(a,b,c,d) = \frac{\phi(a,b,c)\phi(b,c,d)}{Z} = \frac{p(a,b,c)p(b,c,d)}{p(c,b)}$$

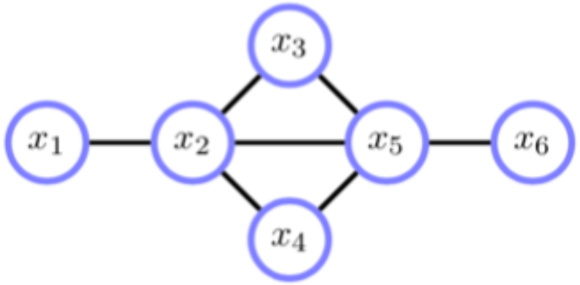
Decomposable Graphs

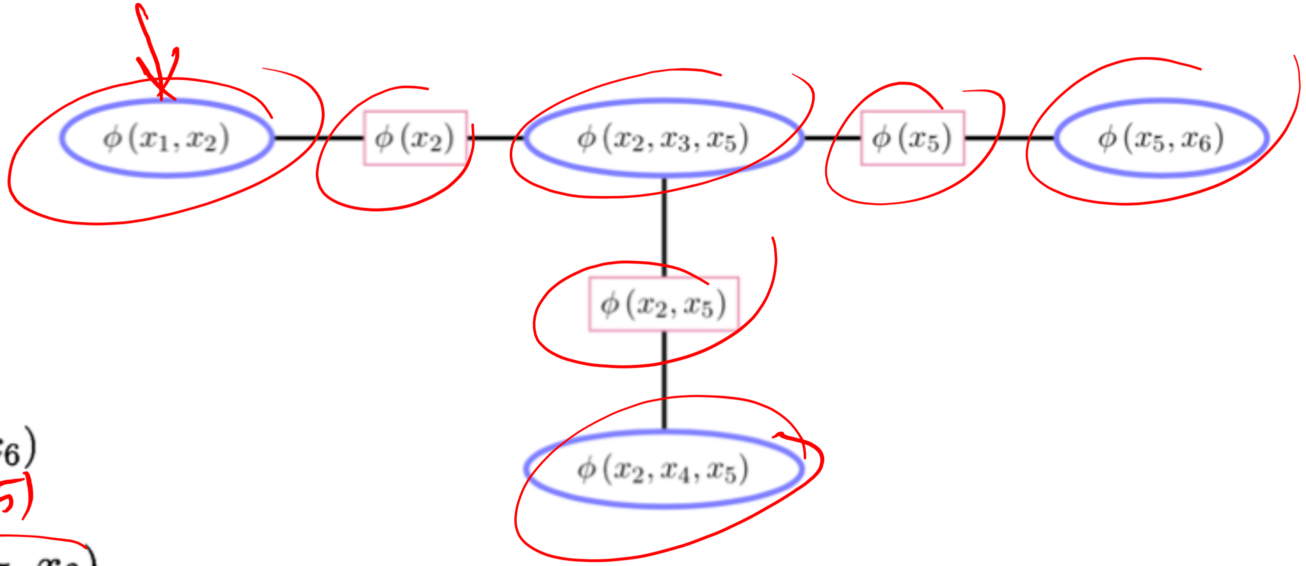
Remember this from Lectures 6

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

$$p(\mathcal{X}) = \frac{\prod_c p(\mathcal{X}_c)}{\prod_s p(\mathcal{X}_s)}$$

Decomposable Graphs





$$p(x_1, \ldots, x_6) = \frac{1}{Z}\phi(x_1, x_2)\phi(x_2, x_3, x_5)\phi(x_2, x_4, x_5)\phi(x_5, x_6)$$

$p(x_1|x_2)$ $\quad$ $p(x_4|x_2,x_5)$

$$p(x_1, \ldots, x_6) = \frac{p(x_1, x_2)p(x_2, x_3, x_5)p(x_2, x_4, x_5)p(x_5, x_6)}{p(x_2)p(x_2, x_5)p(x_5)}$$

$$p(x_1, \ldots, x_6) = \underbrace{p(x_1|x_2)}_{\phi(x_1,x_2)}\underbrace{p(x_2, x_3, x_5)}_{\phi(x_2,x_3,x_5)}\underbrace{p(x_4|x_2, x_5)}_{\phi(x_2,x_4,x_5)}\underbrace{p(x_6|x_5)}_{\phi(x_5,x_6)}$$

$$L = \sum_n \log p(x_1^n|x_2^n) + \log p(x_2^n, x_3^n, x_5^n) + \log p(x_4^n|x_2^n, x_5^n) + \log p(x_6^n|x_5^n)$$

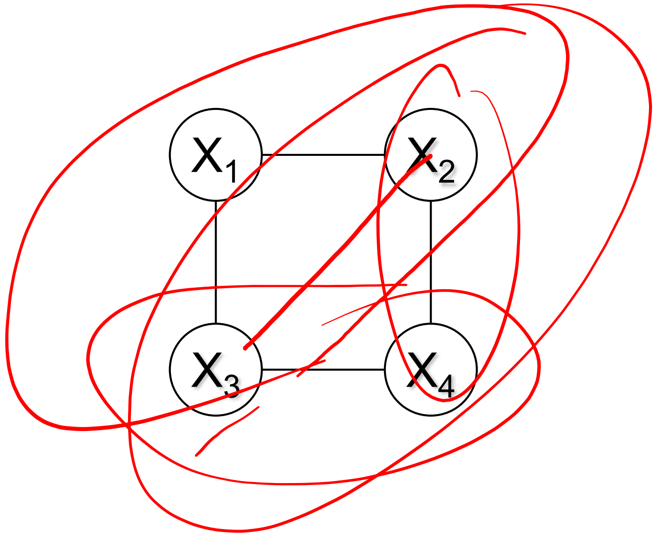
$$\phi(x_1, x_2) = \epsilon(x_1|x_2), \quad \phi(x_2, x_3, x_5) = \epsilon(x_2, x_3, x_5), \quad \phi(x_2, x_4, x_5) = \epsilon(x_4|x_2, x_5), \quad \phi(x_5, x_6) = \epsilon(x_6|x_5)$$

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.



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

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

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

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 $\{1,...,K\}$, 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.

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

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

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

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

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 $\{1,...,K\}$, update each parameter and increment t:
6. Repeat #5 until convergence

Fixed Point Iteration for Optimization

$Av = v\lambda$ $\quad$ $\frac{1}{\lambda} Av = v$

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

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

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

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

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

```python
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)

$\epsilon(y_c) = p(y_c)$

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

$$L(\mathcal{D};\phi) = \sum_{n=1}^{N} \log p(X^n;\phi)$$

$$\frac{\partial}{\partial \phi_c(\mathcal{Y}_c)} L(\theta) = \sum_n \mathbb{I}\left[\mathcal{Y}_c = \mathcal{X}_c^n\right] \frac{1}{\phi_c(\mathcal{Y}_c)} - N \frac{p(\mathcal{Y}_c)}{\phi_c(\mathcal{Y}_c)}$$

$$\phi_c(\mathcal{Y}_c) = \phi_c(\mathcal{Y}_c) \frac{\epsilon(\mathcal{Y}_c)}{p(\mathcal{Y}_c)}$$

$$\phi_c^{(t+1)}(\mathcal{Y}_c) \leftarrow \phi_c^{(t)}(\mathcal{Y}_c) \frac{\epsilon(\mathcal{Y}_c)}{p^{(t)}(\mathcal{Y}_c)}$$

Need to do inference here

$$p^{(t)}(\mathcal{Y}_c) = \sum_{\mathcal{Y}' : \mathcal{Y}'_c = \mathcal{Y}_c} p(\mathcal{Y}'; \theta^{(t)})$$

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

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:

$$\phi_c^{(t+1)}(\mathcal{Y}_c) \leftarrow \phi_c^{(t)}(\mathcal{Y}_c)\frac{\epsilon(\mathcal{Y}_c)}{p^{(t)}(\mathcal{Y}_c)}$$

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

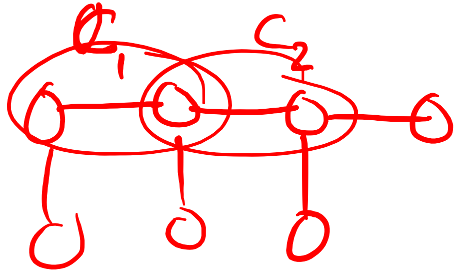
- Hence at each step, it will increase the log-likelihood, and it will converge to a global maximum.

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

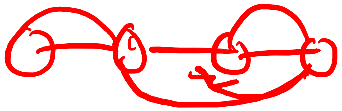
Feature-based Clique Potentials



- *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 is to 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

26 26 brace



- Consider a clique $\mathbf{x}_c$ of random variables in a UGM, e.g. three consecutive characters $c_1c_2c_3$ in a string of English text.
- How would we build a model of $p(c_1c_2c_3)$?
  - If we use a single clique function over $c_1c_2c_3$, 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 $f_{ing}(c_1c_2c_3)$ 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:

$$\psi_c(c_1, c_2, c_3) = e^{\theta_{\text{ing}} f_{\text{ing}}} \times e^{\theta_{\text{?ed}} f_{\text{?ed}}} \times \dots$$
$$= \exp\left\{ \sum_{k=1}^{K} \theta_k f_k(c_1, c_2, c_3) \right\}$$

- 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.

$\frac{1}{Z(\theta)} \prod_c \exp(\theta_c^T f(x_c))$ $\cancel{\frac{1}{Z(\theta)} \exp(\theta^T f(x_c))}$ $\frac{1}{Z(\theta)} \exp(\sum_c \theta_c^T f_c(x_c))$

$\exp(\sum_c \theta_c^T f_c(x_c) - A(\theta))$

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 **GLM**:

$Z(\theta) = e^{A(\theta)}$

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

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

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

Feature Based Model

- We can multiply these clique potentials as usual:

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

- 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 $\cancel{\phi_c^{(t+1)}(\mathcal{Y}_c) \leftarrow \phi_c^{(t)}(\mathcal{Y}_c) \frac{\epsilon(\mathcal{Y}_c)}{p^{(t)}(\mathcal{Y}_c)}}$
  - **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

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**

$\mathcal{L}(D;\theta)$

$\Lambda(\theta)$

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

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

$$L(\mathcal{D};\theta) = \sum_{n=1}^{N} \log p(X^n;\theta)$$

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

$$\frac{\partial \Lambda(\theta_c)}{\partial \theta_c} = \frac{1}{N}\sum_n f_c(\mathcal{X}_n^n) - \mathbb{E}\left[f_c(\mathcal{X}_c) \exp\left((\theta_c - \theta_{old}) \sum_d f_d(\mathcal{X}_d) \right) \right]$$

$$\theta^{t+1} \leftarrow \theta^t + \log\left(\frac{1/N \sum_n f(\mathcal{X}_c^n)}{\mathbb{E}\left[f_c(\mathcal{X}_c) \right]} \right)$$

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,…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.

$$\frac{1}{N}L(\theta) \geq \sum_c \left\{ \frac{1}{N}\sum_n f_c(\mathcal{X}_c^n)\theta_c - \left\langle p_c \exp\left(\alpha_c \sum_d f_d(\mathcal{X}_c) \right) \right\rangle_{p(\mathcal{X}|\theta^{old})} \right\}$$

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:

$$\theta^{t+1} \leftarrow \theta^t + \log\left(\frac{1/N\sum_n f(\mathcal{X}_c^n)}{\mathbb{E}\left[f_c(\mathcal{X}_c)\right]}\right)$$

IPF:

$$\phi_c^{(t+1)}(\mathcal{Y}_c) \leftarrow \phi_c^{(t)}(\mathcal{Y}_c)\frac{\epsilon(\mathcal{Y}_c)}{p^{(t)}(\mathcal{Y}_c)}$$

$$\log \phi_c^{t+1} \leftarrow \log \phi_c^{t} + \log \frac{\epsilon}{p^t(\mathcal{Y}_c)}$$

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

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

Stochastic Gradient Descent

- Suppose we have N training examples s.t. $f(x) = \sum_{i=1}^{N} f_i(x)$.
- This implies that $\nabla f(x) = \sum_{i=1}^{N} \nabla f_i(x)$.

SGD Algorithm:

1. Choose a starting point x.
2. While not converged:
   - Choose a step size t.
   - Choose i so that it sweeps through the training set.
   - Update

$$\vec{x}^{(k+1)} = \vec{x}^{(k)} + t \nabla f_i(\vec{x})$$

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

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

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

A. **Generalized Iterative Scaling (GIS)**
   – Maximizes a lower-bound of log-likelihood
   – Iterative algorithm (like IPF), but more broadly applies to exponential family potentials
   – When $\sum_c f(X_c) = 1$ has an advantage

B. **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]

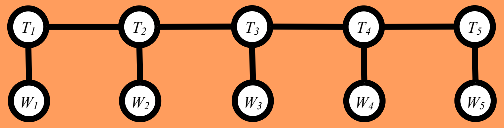
MLE for Undirected GMs

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

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

5. Inference

1. Marginal Inference

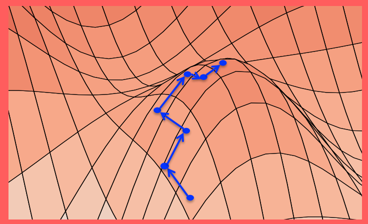
$$p(\boldsymbol{x}_C) = \sum_{\boldsymbol{x}' : \boldsymbol{x}'_C = \boldsymbol{x}_C} p(\boldsymbol{x}' \mid \boldsymbol{\theta})$$

2. Partition Function

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

4. Learning

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



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

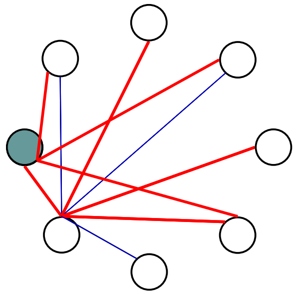
$$p(x_i) = \sum_{\boldsymbol{x}' : x'_i = x_i} p(\boldsymbol{x}' \mid \boldsymbol{\theta}) \qquad \Big| \qquad p(\boldsymbol{x}_C) = \sum_{\boldsymbol{x}' : \boldsymbol{x}'_C = \boldsymbol{x}_C} p(\boldsymbol{x}' \mid \boldsymbol{\theta})$$

2. Partition Function
Compute the normalization constant

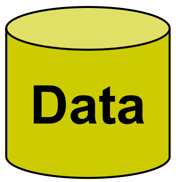
$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}_C)$$

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**



$$
\begin{array}{c}
(x_1^{(1)},\ldots,x_n^{(1)}) \\
(x_1^{(2)},\ldots,x_n^{(2)}) \\
\ldots \\
(x_1^{(M)},\ldots,x_n^{(M)})
\end{array}
$$

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

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

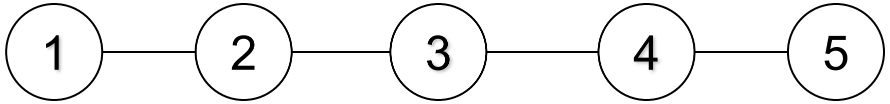
  - Graphical model interpretation?

- 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



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

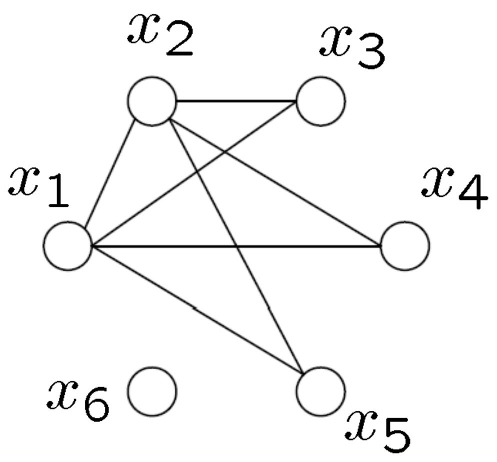
$$\Sigma^{-1}_{15} = 0 \Leftrightarrow X_1 \perp X_5 \,\middle|\, X_{nbrs(1)\text{ or }nbrs(5)}$$

$$\not\Rightarrow$$

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

Another example

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



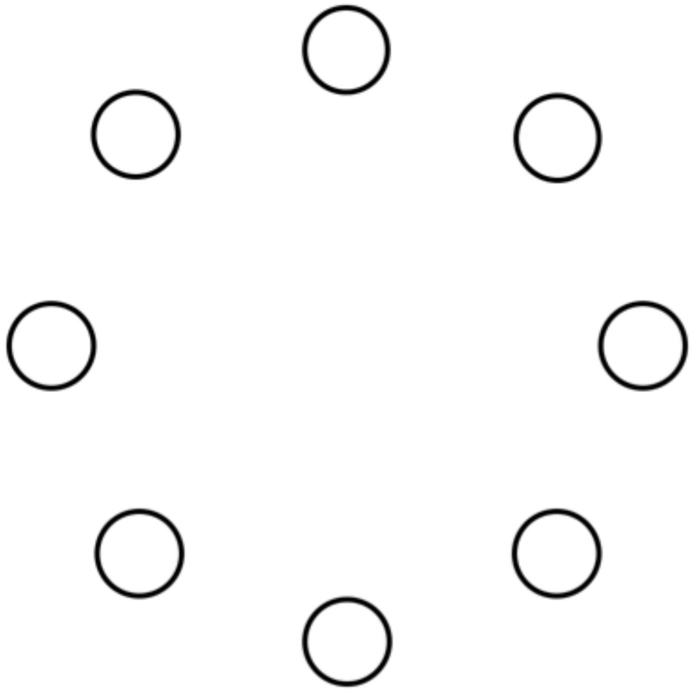
- How to estimate this MRF?
- What if $p >> n$
  - MLE does not exist in general!
  - What about only learning a "sparse" graphical model?
    - This is possible when $s=o(n)$
    - Very often it is the structure of the GM that is more interesting …

Recall lasso

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

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

Graph Regression

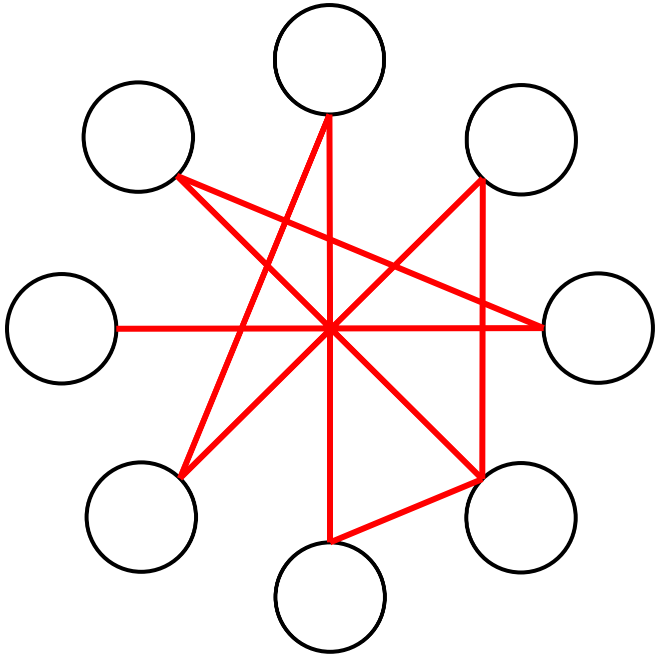


Neighborhood selection

Lasso:

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

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.