Recap: MLE for BNs

- If we assume the parameters for each CPD are globally independent, and all nodes are fully observed, then the log-likelihood function decomposes into a sum of local terms, one per node:

$$\ell(\theta; D) = \log p(D | \theta) = \log \prod_{n} \left( \prod_{i} p(x_{n,i} | x_{e_{i}}, \theta_{i}) \right) = \sum_{n} \left( \sum_{i} \log p(x_{n,i} | x_{e_{i}}, \theta_{i}) \right)$$

$$\theta_{ijk}^{ML} = \frac{n_{ijk}}{\sum_{i,j,k} n_{ijk}}$$
MLE for undirected graphical models

- 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.

$$
\mathcal{P}(x_1, \ldots, x_n) = \frac{1}{Z} \prod_{c \in C} \psi_c(x_c) \\
Z = \sum_{x_1, \ldots, x_n, c \in C} \psi_c(x_c)
$$

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

Log Likelihood for UGMs with tabular clique potentials

- Sufficient statistics: for a UGM $(V, \mathcal{E})$, the number of times that a configuration $x$ (i.e., $x_V = x$) is observed in a dataset $D = \{x_1, \ldots, x_N\}$ can be represented as follows:

$$
m(x) = \sum_n \delta(x, x_n) \text{ (total count),} \quad \text{and} \quad m(x_c) = \sum_{x_{\bar{c}}} m(x) \text{ (clique count)}
$$

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

$$
\mathcal{P}(D | \theta) = \prod_N \prod_x \mathcal{P}(x | \theta)^{\delta(x, x_n)} \\
\log \mathcal{P}(D | \theta) = \sum_N \sum_x \delta(x, x_n) \log \mathcal{P}(x | \theta) = \sum_N \sum_x \delta(x, x_n) \log \mathcal{P}(x | \theta)
$$

$$
\ell = \sum_x m(x) \log \left( \frac{1}{Z} \prod_c \psi_c(x_c) \right) \\
= \sum_x \sum_{x_c} m(x_c) \log \psi_c(x_c) - N \log Z
$$

- There is a nasty $\log Z$ in the likelihood.
Derivative of log Likelihood

- Log-likelihood: \( \ell = \sum_{c} \sum_{x_c} m(x_c) \log \psi_c(x_c) - N \log Z \)

- First term:
  \[
  \frac{\partial \ell}{\partial \psi_c(x_c)} = \frac{m(x_c)}{\psi_c(x_c)}
  \]

- Second term:
  \[
  \frac{\partial \log Z}{\partial \psi_c(x_c)} = \frac{1}{Z} \frac{\partial}{\partial \psi_c(x_c)} \left( \sum_x \prod_d \psi_d(\bar{x}_d) \right) = \frac{1}{Z} \sum_x \delta(\bar{x}_c, x_c) \frac{1}{\psi_c(x_c)} \prod_d \psi_d(\bar{x}_d) = \frac{1}{\psi_c(x_c)} \sum_x \delta(\bar{x}_c, x_c) p(\bar{x}) \]

Set the value of variables to \( \bar{x} \)

Conditions on Clique Marginals

- Derivative of log-likelihood
  \[
  \frac{\partial \ell}{\partial \psi_c(x_c)} = \frac{m(x_c)}{\psi_c(x_c)} - \frac{N}{Z} \frac{p(x_c)}{\psi_c(x_c)}
  \]

- Hence, for the maximum likelihood parameters, we know that:
  \[
  p_{\text{MLE}}^*(x_c) = \frac{m(x_c)}{N} = \frac{\hat{p}(x_c)}{Z}
  \]

- 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.
MLE for undirected graphical models

- Is the graph decomposable (triangulated)?
- Are all the clique potentials defined on maximal cliques (not sub-cliques)? e.g., $\psi_{123}, \psi_{234}$ not $\psi_{12}, \psi_{23}, \ldots$
- Are the clique potentials full tables (or Gaussians), or parameterized more compactly, e.g. $\psi_{123}(x) = \exp(\sum_k \phi_k(x_i))$?

<table>
<thead>
<tr>
<th>Decomposable?</th>
<th>Max clique?</th>
<th>Tabular?</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>Direct</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>✔️</td>
<td>IPF</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>Gradient</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>GIF</td>
</tr>
</tbody>
</table>

MLE for decomposable undirected models

- Decomposable models:
  - $G$ is decomposable $\iff$ $G$ is triangulated $\iff$ $G$ has a junction tree
  - Potential based representation: $p(x) = \prod_i \psi_i(x_i) / \prod_i \phi_i(x_i)$

- Consider a chain $X_1 - X_2 - X_3$. The cliques are $(X_1, X_2)$ and $(X_2, X_3)$; the separator is $X_2$
  - The empirical marginals must equal the model marginals.

- Let us guess that $\hat{\rho}_{MLE}(x_1, x_2, x_3) = \frac{\hat{p}(x_1, x_2, x_3)}{\hat{p}(x_2)}$

  - We can verify that such a guess satisfies the conditions:
    $\hat{p}_{MLE}(x_1, x_2) = \sum_{x_3} \hat{p}_{MLE}(x_1, x_2, x_3) = \hat{p}(x_1 | x_2) \sum_{x_3} \hat{p}(x_2, x_3) = \hat{p}(x_1, x_2)$
    and similarly $\hat{p}_{MLE}(x_2, x_3) = \hat{p}(x_2, x_3)$
MLE for decomposable undirected models (cont.)

- Let us guess that $\hat{p}_{MLE}(x_1, x_2, x_3) = \frac{\hat{p}(x_1, x_2) \hat{p}(x_2, x_3)}{\hat{p}(x_3)}$

- To compute the clique potentials, just equate them to the empirical marginals (or conditionals), i.e., the separator must be divided into one of its neighbors. Then $Z = 1$.

$$\psi_{12}^{MLE}(x_1, x_2) = \hat{p}(x_1, x_2) \quad \psi_{23}^{MLE}(x_2, x_3) = \frac{\hat{p}(x_2, x_3)}{\hat{p}(x_2)} = \hat{p}(x_2 | x_3)$$

- One more example:

$$\hat{p}_{MLE}(x_1, x_2, x_3, x_4) = \frac{\hat{p}(x_1, x_2, x_3) \hat{p}(x_2, x_3, x_4)}{\hat{p}(x_3)}$$

$$\psi_{123}^{MLE}(x_2, x_3) = \frac{\hat{p}(x_1, x_2, x_3)}{\hat{p}(x_2, x_3)} = \hat{p}(x_1 | x_2, x_3)$$

$$\psi_{234}^{MLE}(x_2, x_3, x_4) = \hat{p}(x_2, x_3, x_4)$$

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., $\psi_{12}$, $\psi_{34}$), we could not equate empirical marginals (or conditionals) to MLE of cliques potentials.

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

$$\exists (i,j) \text{ s.t. } \psi_j^{MLE}(x_i, x_j) = \begin{cases} \frac{\hat{p}(x_i, x_j)}{\hat{p}(x_i)} & \text{if } \hat{p}(x_i, x_j) > 0 \\ \frac{\hat{p}(x_i, x_j)}{\hat{p}(x_j)} & \text{if } \hat{p}(x_i, x_j) < 0 \end{cases}$$

Homework!
Iterative Proportional Fitting (IPF)

- From the derivative of the likelihood:
  \[
  \frac{\partial \mathcal{L}}{\partial \psi_c(x_c)} = m(x_c) - \mu_c \frac{p(x_c)}{\psi_c(x_c)}
  \]

- we can derive another relationship:
  \[
  \frac{\tilde{p}(x_c)}{\psi_c(x_c)} = \frac{p(x_c)}{\psi_c(x_c)}
  \]
  in which \( \psi_c \) appears implicitly in the model marginal \( p(x_c) \).

- This is therefore a fixed-point equation for \( \psi_c \).
  - Solving \( \psi_c \) in closed-form is hard, because it appears on both sides of this implicit nonlinear equation.

- The idea of IPF is to hold \( \psi_c \) fixed on the right hand side (both in the numerator and denominator) and solve for it on the left hand side. We cycle through all cliques, then iterate:
  \[
  \psi^{(i+1)}_c(x_c) = \psi^{(i)}_c(x_c) \frac{\tilde{p}(x_c)}{\tilde{p}^{(i)}(x_c)}
  \]

Properties of IPF Updates

- IPF iterates a set of fixed-point equations.

- 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.

- I-projection: finding a distribution with the correct marginals that has the maximal entropy
IPF can be seen as coordinate ascent in the likelihood using the way of expressing likelihoods using KL divergences.

Recall that we have shown maximizing the log likelihood is equivalent to minimizing the KL divergence (cross entropy) from the observed distribution to the model distribution:

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

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

$$KL(q(x_a, x_b) \parallel p(x_a, x_b)) = \sum_{x_a, x_b} q(x_a, x_b) q(x_b \mid x_a) \log \frac{q(x_a, x_b) q(x_b \mid x_a)}{p(x_a)p(x_b \mid x_a)}$$

$$= \sum_{x_a} q(x_a) q(x_b \mid x_a) \log \frac{q(x_a)}{p(x_a)} + \sum_{x_a} q(x_a) q(x_b \mid x_a) \log \frac{q(x_b \mid x_a)}{p(x_b \mid x_a)}$$

$$= KL(q(x_a) \parallel p(x_a)) + \sum_{x_a} q(x_a) KL(q(x_b \mid x_a) \parallel p(x_b \mid x_a))$$

Putting things together, we have

$$KL(\tilde{p}(x) \parallel p(x \mid \theta)) = KL(\tilde{p}(x_c) \parallel p(x_c \mid \theta)) + \sum_{x_c} \tilde{p}(x_c) KL(\tilde{p}(x_{-c} \mid x_c) \parallel p(x_{-c} \mid x_c))$$

It can be shown that changing the clique potential $\psi_c$ 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.

We can interpret IPF updates as retaining the “old” conditional probabilities $p^{(t)}(x_{-c} \mid x_c)$ while replacing the “old” marginal probability $p^{(t)}(x_c)$ with the observed marginal $\tilde{p}(x_c)$. 
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 potential functions $\psi_c(x_c)$.
- 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 $x_c$ 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 $f_{\text{ing}}(c_1, c_2, c_3)$ 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:

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

- This is still a potential over $2^3$ possible settings, but only uses $K$ parameters if there are $K$ features.
  - By having one indicator function per combination of $x_c$, we recover the standard tabular potential.

Combining Features

- Each feature has a weight $\theta_k$ 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:

  $$p(c_1, c_2, c_3) \propto \exp \left\{ \theta_{c_1} f_{c_1}(c_1, c_2, c_3) + \theta_{c_2} f_{c_2}(c_1, c_2, c_3) + \ldots \right\}$$

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

  $$\psi_c(x_c) \overset{\text{def}}{=} \exp \left\{ \sum_{x_c \in I_c} \theta_k f_k (x_c) \right\}$$

- How can we combine feature into a probability model?
Feature Based Model

- We can multiply these clique potentials as usual:
  \[ p(x) = \frac{1}{Z(\theta)} \prod_z \psi_z(x_z) = \frac{1}{Z(\theta)} \exp \left\{ \sum_z \sum_{i \in L_z} \theta_k f_k(x_z) \right\} \]

- However, in general we can forget about associating features with cliques and just use a simplified form:
  \[ p(x) = \frac{1}{Z(\theta)} \exp \left\{ \sum f_i(x_z) \right\} \]

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

- Learning: recall that in IPF, we have \[ \psi^{(r+1)}_z(x_z) = \psi^{(r)}_z(x_z) \frac{\tilde{p}(x_z)}{p^{(r)}(x_z)} \]
  - Not obvious how to update the weights and features individually

MLE of Feature Based UGMs

- Scaled likelihood function
  \[ \tilde{L}(\theta; D) = \frac{L(\theta; D)}{N} = \frac{1}{N} \sum_n \log p(x_n | \theta) \]
  \[ = \sum_x \tilde{p}(x) \log p(x | \theta) \]
  \[ = \sum_x \tilde{p}(x) \sum f_i(x) - \log Z(\theta) \]

- Instead of optimizing this objective directly, we attack its lower bound
  - The logarithm has a linear upper bound …
    \[ \log Z(\theta) \leq \mu Z(\theta) - \log \mu - 1 \]
  - This bound holds for all \( \mu \), in particular, for \( \mu = Z^{-1}(\theta^{(r)}) \)
  - Thus we have
    \[ \tilde{L}(\theta; D) \geq \sum_x \tilde{p}(x) \sum f_i(x) - \frac{Z(\theta)}{Z(\theta^{(r)})} \log Z(\theta^{(r)}) + 1 \]
**Generalized Iterative Scaling (GIS)**

- Lower bound of scaled loglikelihood
  \[ \tilde{\ell} (\theta; D) \geq \sum_x \tilde{p}(x) \sum_j \theta_j f_j(x) - \frac{Z(\theta)}{Z(\theta^{(r)})} \log Z(\theta^{(r)}) + 1 \]

- Define \( \Delta \theta^{(r)} \) via
  \[ \tilde{\ell} (\theta; D) \geq \sum_x \sum_j \frac{1}{Z(\theta^{(r)})} \sum \exp \left[ \sum \theta_j f_j(x) \right] - \log Z(\theta^{(r)}) + 1 \]

- Relax again
  - Assume \( f_j(x) \geq 0, \sum f_j(x) = 1 \)
  - Convexity of exponential: \( \exp \left( \sum \pi_i x_j \right) \leq \sum \pi_i \exp (x_j) \)

- We have:
  \[ \tilde{\ell} (\theta; D) \geq \sum_i \theta_i \sum_x \tilde{p}(x) f_i(x) - \sum_x p(x | \theta^{(r)}) \sum_j f_j(x) \exp (\Delta \theta^{(r)} j) - \log Z(\theta^{(r)}) + 1 \overset{\text{def}}{=} \Lambda(\theta) \]
Where does the exponential form come from?

- Review: Maximum Likelihood for exponential family
  \[ \ell(\theta; D) = \sum_x m(x) \log p(x | \theta) \]
  \[ = \sum_x m(x) \left( \sum_i \theta_i f_i(x) - \log Z(\theta) \right) \]
  \[ = \sum_x m(x) \sum_i \theta_i f_i(x) - N \log Z(\theta) \]
  \[ \frac{\partial}{\partial \theta_i} \ell(\theta; D) = \sum_x m(x) f_i(x) - N \frac{\partial}{\partial \theta_i} \log Z(\theta) \]
  \[ = \sum_x m(x) f_i(x) - N \sum_x p(x | \theta) f_i(x) \]
  \[ \Rightarrow \quad \sum_x p(x | \theta) f_i(x) = \sum_x m(x) f_i(x) = \sum_x \hat{p}(x | \theta) f_i(x) \]

- i.e., At ML estimate, the expectations of the sufficient statistics under the model must match empirical feature average.

Maximum Entropy

- We can approach the modeling problem from an entirely different point of view. Begin with some fixed feature expectations:
  \[ \sum_x p(x) f_i(x) = \alpha_i \]

- Assuming expectations are consistent, there may exist many distributions which satisfy them. Which one should we select?
  - The most uncertain or flexible one, i.e., the one with maximum entropy.

- This yields a new optimization problem:
  \[ \max_p \quad H(p(x)) = -\sum_x p(x) \log p(x) \]
  \[ \text{s.t.} \quad \sum_x p(x) f_i(x) = \alpha_i \]
  \[ \sum_x p(x) = 1 \]
  This is a variational definition of a distribution!
Solution to the MaxEnt Problem

- To solve the MaxEnt problem, we use Lagrange multipliers:

\[
L = -\sum_x p(x) \log p(x) - \sum_i \theta_i \left( \sum_x p(x)f_i(x) - \alpha_i \right) - \mu \left( \sum_x p(x) - 1 \right)
\]

\[
\frac{\partial L}{\partial p(x)} = 1 + \log p(x) - \sum_i \theta_i f_i(x) - \mu
\]

\[
p^*(x) = e^{\mu} \exp \left\{ \sum_i \theta_i f_i(x) \right\}
\]

\[
Z(\theta) = e^{\mu} = \sum_x \exp \left\{ \sum_i \theta_i f_i(x) \right\} \quad \text{(since } \sum_x p^*(x) = 1)\]

\[
p(x|\theta) = \frac{1}{Z(\theta)} \exp \left\{ \sum_i \theta_i f_i(x) \right\}
\]

- So feature constraints + MaxEnt ⇒ exponential family.
- Problem is strictly convex w.r.t. \( p \), so solution is unique.

A more general MaxEnt problem

\[
\min_p \quad \text{KL}(p(x) \parallel h(x))
\]

\[
def = \sum_x p(x) \log \frac{p(x)}{h(x)} = -\mathcal{H}(p) - \sum_x p(x) \log h(x)
\]

s.t. \( \sum_x p(x)f_i(x) = \alpha_i \)

\[
\sum_x p(x) = 1
\]

\[
\Rightarrow \quad p(x|\theta) = \frac{1}{Z(\theta)} h(x) \exp \left\{ \sum_i \theta_i f_i(x) \right\}
\]
Constraints from Data

- Where do the constraints $\alpha_i$ come from?
- Just as before, measure the empirical counts on the training data:
  \[ \alpha_i = \sum_x \frac{m(x)}{N} f_i(x) = \sum_x \tilde{p}(x) f_i(x) \]
- This also ensures consistency automatically.
- Known as the “method of moments”. (c.f. law of large numbers)
- We have seen a case of convex duality:
  - In one case, we assume exponential family and show that ML implies model expectations must match empirical expectations.
  - In the other case, we assume model expectations must match empirical feature counts and show that MaxEnt implies exponential family distribution.
- No duality gap $\Rightarrow$ yield the same value of the objective

Geometric interpretation

- All exponential family distribution:
  \[ \mathcal{E} = \left\{ p(x) : p(x|\theta) = \frac{1}{Z(\theta)} h(x) \exp \left\{ \sum_i \theta_i f_i(x) \right\} \right\} \]
- All distributions satisfying moment constraints
  \[ \mathcal{M} = \left\{ p(x) : \sum_x p(x) f_i(x) = \sum_x \tilde{p}(x) f_i(x) \right\} \]
- Pythagorean theorem
  \[ \text{KL}(q \parallel p) = \text{KL}(q \parallel p_M) + \text{KL}(p_M \parallel q) \]

MaxEnt:
\[
\begin{align*}
\text{min}_p &\quad \text{KL}(q \parallel h) \\
\text{s.t.} &\quad q \in \mathcal{M} \\
\text{KL}(q \parallel h) &= \text{KL}(q \parallel p_M) + \text{KL}(p_M \parallel h)
\end{align*}
\]

MaxLik:
\[
\begin{align*}
\text{min}_p &\quad \text{KL}(\hat{p} \parallel p) \\
\text{s.t.} &\quad q \in \mathcal{E} \\
\text{KL}(\hat{p} \parallel p) &= \text{KL}(\hat{p} \parallel p_M) + \text{KL}(p_M \parallel p)
\end{align*}
\]
Conditional Random Fields

- So far we have focussed on maxent models for density estimation.
- We can also formulate such models for classification and regression (conditional density estimation).

\[ p_{\theta}(y|x) = \frac{1}{Z(\theta, x)} \exp \left( \sum_{c} \theta_{c}(x, y_c) \right) \]

- The model above is like doing logistic regression on the features. Now features can be very complex, nonlinear functions of the data.

- Allow arbitrary dependencies on input
- Clique dependencies on labels
- Use approximate inference for general graphs