Outline

• MAP param estimation for UGMs (20.1-20.4)
• Learning using approximate inference (20.5)
• Alternative objectives (20.6)
Likelihood fn for UGMs

- Log-linear model

\[ P(X_1, \ldots, X_n : \theta) = \frac{1}{Z(\theta)} \exp \left\{ \sum_{i=1}^{k} \theta_i f_i [D_i] \right\}. \]

\[ \ln Z(\theta) = \ln \sum_{\xi} \exp \left\{ \sum_{i} \theta_i f_i [\xi] \right\}. \]

**Concave**

Figure 20.1 Log-likelihood surface for the Markov network A—B—C, as a function of \( \ln \phi_1 [a^1, b^1] \) (x-axis) and \( \ln \phi_2 [b^1, c^1] \) (y-axis); all other parameters in both potentials are set to 1. The data set \( \mathcal{D} \) has \( M = 100 \) instances, for which \( M[a^1, b^1] = 90 \) and \( M[b^1, c^1] = 15 \). (The other sufficient statistics are irrelevant, as all of the other log-parameters are 0.)
Proposition 20.2.3: Let $\Phi$ be a set of features. Then,

$$\frac{\partial}{\partial \theta_i} \ln Z(\theta) = E_\theta[f_i]$$

$$\frac{\partial^2}{\partial \theta_i \partial \theta_j} \ln Z(\theta) = \text{Cov}_\theta[f_i; f_j],$$

where $E_\theta[f_i]$ is a shorthand for $E_{P(\chi; \theta)}[f_i]$.

$$\frac{\partial}{\partial \theta_i} \ln Z(\theta) = \frac{1}{Z(\theta)} \sum_\xi \frac{\partial}{\partial \theta_i} \exp \left\{ \sum_j \theta_j f_j[\xi] \right\}$$

$$= \frac{1}{Z(\theta)} \sum_\xi f_i[\xi] \exp \left\{ \sum_j \theta_j f_j[\xi] \right\}$$

$$= E_\theta[f_i].$$
\[
\frac{\partial^2}{\partial \theta_j \partial \theta_i} \ln Z(\theta) = \frac{\partial}{\partial \theta_j} \left[ \frac{1}{Z(\theta)} \sum_{\xi} f_i[\xi] \exp \left\{ \sum_k \theta_k f_k[\xi] \right\} \right] \\
= -\frac{1}{Z(\theta)^2} \left( \frac{\partial}{\partial \theta_j} Z(\theta) \right) \sum_{\xi} f_i[\xi] \exp \left\{ \sum_k \theta_k f_k[\xi] \right\} \\
+ \frac{1}{Z(\theta)} \sum_{\xi} f_i[\xi] f_j[\xi] \exp \left\{ \sum_k \theta_k f_k[\xi] \right\} \\
= -\frac{1}{Z(\theta)^2} Z(\theta) E_{\theta}[f_i] \sum_{\xi} f_i[\xi] \tilde{P}(\xi : \theta) \\
+ \frac{1}{Z(\theta)} \sum_{\xi} f_i[\xi] f_j[\xi] \tilde{P}(\xi : \theta) \\
= E_{\theta}[f_i] \sum_{\xi} f_i[\xi] P(\xi : \theta) \\
+ \sum_{\xi} f_i[\xi] f_j[\xi] P(\xi : \theta) \\
= E_{\theta}[f_i f_j] - E_{\theta}[f_i] E_{\theta}[f_j] \\
= \text{Cov}_{\theta}[f_i; f_j].
\]
Finding the MLE

At optimum, model moments = empirical moments

\[ \frac{\partial}{\partial \theta_i} \frac{1}{M} \ell(\theta : \mathcal{D}) = E_D[f_i[\mathcal{X}]] - E_{\theta}[f_i]. \]  

(20.4)

This analysis provides us with a precise characterization of the maximum likelihood parameters \( \hat{\theta} \):

**Theorem 20.3.1:** Let \( \Phi \) be a set of features. Then, \( \theta \) is a maximal likelihood parameter assignment if and only if \( E_D[f_i[\mathcal{X}]] = E_{\theta}[f_i] \) for all \( i \).

Must perform inference once per gradient

Just do gradient based optimization, eg stochastic gradient descent. Expensive to compute Hessian explicitly. so use Quasi-Newton.
CRFs

- Conditional density models

\[
\ell_{Y|X}(\theta : D) = \ln P(y[1, \ldots, M] | x[1, \ldots, M], \theta) = \sum_{m=1}^{M} \ln P(y[m] | x[m], \theta).
\]

\[
\frac{\partial}{\partial \theta_i} \ell_{Y|X}(\theta : D) = \sum_{m=1}^{M} [f_i[y[m], x[m]] - E_\theta[f_i | x[m]]].
\]

Must perform inference M times per gradient
MRFs with hidden variables

- Must perform inference $M$ times per gradient

\[
\frac{1}{M} \ln P(\mathcal{D} | \theta) = \frac{1}{M} \ln \left( \sum_{m=1}^{M} \sum_{h[m]} P(o[m], h[m] | \theta) \right) \\
= \frac{1}{M} \ln \left( \sum_{m=1}^{M} \sum_{h[m]} \tilde{P}(o[m], h[m] | \theta) \right) - \ln Z.
\]

\[
\frac{\partial}{\partial \theta_i} \ln \sum_{h[m]} \tilde{P}(o[m], h[m] | \theta) = E_{h[m] \sim P(\mathcal{H}[m] | o[m], \theta)}[f_i],
\]

Proposition 20.3.3: For a data set $\mathcal{D}$

\[
\frac{\partial}{\partial \theta_i} \frac{1}{M} \ell(\theta : \mathcal{D}) = \frac{1}{M} \left[ \sum_{m=1}^{M} E_{h[m] \sim P(\mathcal{H}[m] | o[m], \theta)}[f_i] \right] - E_\theta[f_i].
\]

clamped \hspace{1cm} unclamped
CRFs with hidden variables

• Training is similar to MRFs with hidden variables, except expectations condition on x_n, so need to be redone for each case
\[ \nabla = \sum_i f(x_i) - ME_X[f(X)] \]

\[ \nabla = \sum_i f(x_i) - ME_X[f(X)] \]

\[ \nabla = \sum_i E_H f(H, x_i) - ME_{H,X}[f(H, X)] \]

\[ \nabla = \sum_i E_H f(x_i, y_i, H) - \sum_i E_{H,Y}[f(x_i, Y, H)] \]
ML and MaxEnt

- MLE in the expfam is equivalent to MaxEnt subject to moment constraints

Maximum-Entropy
Find that maximize $Q(\mathcal{X})$ subject to $E_Q[f_i] = E_D[f_i] \quad i = 1, \ldots, k$

Theorem 20.3.4: The distribution $Q^*$ is the maximum entropy distribution satisfying Eq. (20.10) if and only if $Q^* = P_{\hat{\theta}}$, where

$$P_{\hat{\theta}}(\mathcal{X}) = \frac{1}{Z(\hat{\theta})} \exp \left\{ \sum_i \hat{\theta}_i f_i[\mathcal{X}] \right\}$$

and $\hat{\theta}$ is the maximum likelihood parameterization relative to $D$. 
Proof For notational simplicity, let \( P = P_\theta \). From Theorem 20.3.1, it follows that \( E_P[f_i] = E_D[f_i[X]] \) for \( i = 1, \ldots, k \), and hence that \( P \) satisfies the constraints of Eq. (20.10). Therefore, to prove that \( P = Q^* \), we need only show that \( H_P(X) \geq H_Q(X) \) for all other distributions \( Q \) that satisfy these constraints. Consider any such distribution \( Q \).

From Theorem 8.4.1, it follows that:

\[
H_P(X) = -\sum_i \hat{\theta}_i E_P[f_i] + \ln Z(\theta). \tag{20.11}
\]

Thus,

\[
H_P(X) - H_Q(X) = -\left[ \sum_i \theta_i E_P[f_i[X]] \right] + \ln Z_P - E_Q[-\ln Q(X)]
\]

(\( i \))

\[
= -\left[ \sum_i \theta_i E_Q[f_i[X]] \right] + \ln Z_P + E_Q[\ln Q(X)]
\]

\[
= E_Q[-\ln P(X)] + E_Q[\ln Q(X)]
\]

\[
= D(Q \mid P) \geq 0,
\]

where (\( i \)) follows from the fact that both \( P_\theta \) and \( Q \) satisfy the constraints, so that \( E_{P_\theta}[f_i] = E_Q[f_i] \) for all \( i \).

We conclude that \( H_{P_\theta}(X) \geq H_Q(X) \) with equality if and only if \( P_\theta = Q \). Thus, the maximum entropy distribution \( Q^* \) is necessarily equal to \( P_\theta \), proving the result. \( \blacksquare \)
### MAP estimation

- Convex prior + convex likelihood makes objective strictly convex (unique soln)
- Also helps prevent overfitting
- L2 and L1

\[
P(\theta | \sigma^2) = \prod_{i=1}^{k} \frac{1}{\sqrt{2\pi}\sigma} \exp \left\{ -\frac{\theta_i^2}{2\sigma^2} \right\},
\]

\[
P_{\text{Laplacian}}(\theta | \beta) = \frac{1}{2\beta} \exp \left\{ -\frac{|\theta|}{\beta} \right\}.
\]

\[
\ln \frac{P(\xi)}{P(\xi')} = \sum_{i=1}^{k} \theta_i f_i[\xi] - \sum_{i=1}^{k} \theta_i f_i[\xi']
\]

\[
= \sum_{i=1}^{k} \theta_i (f_i[\xi] - f_i[\xi']).
\]
Learning with approximate inference

• Recall that the gradient requires model expectation over the features

\[
\frac{\partial}{\partial \theta_i} \frac{1}{M} \ell(\theta : D) = E_D[f_i[X]] - E_{\theta}[f_i]. \tag{20.4}
\]

• We can use approximate inference to approximate the expectation, but approximate gradients can cause learning to diverge
Pseudo moment matching

• At the optimum, the pseudo marginals must satisfy

\[ E_{\beta_i[C_i]}[f_{C_i}] = E_D[f_i[C_i]]. \]

• Suppose we use tabular features. Then

\[ \beta_i[c_i^j] = \hat{P}(c_i^j). \]

• Hence we don’t need to run inference. There are multiple potentials that can generate these beliefs. We can uniquely recover one set using (for any ordering i<j)

\[ \phi_i \leftarrow \frac{\beta_i}{\mu_{i,j}}. \]
Unified inference and learning

- Pseudo moment matching only works for unconditional, tabular potentials with no tying and no regularizer.
- To combine BP with param optimization, we can optimize:

\[
\text{Approx-Maximum-Entropy}
\]

\[
\text{Find that maximize } \sum_{c_i \in U} H_{\beta_i}(c_i) - \sum_{(c_i, c_j) \in U} H_{\mu_{i,j}}(S_{i,j})
\]

subject to

\[
E_{\beta_i}[f_i] = E_{D}[f_i] \quad i = 1, \ldots, k
\]

\[
Q \in \text{Local}[U]
\]

The model parameters theta are the Lagrange multipliers for \(E[f]\) and the messages are the Lagrange multipliers for the local consistency.
Example

Find that maximize

\[ Q = H_{\beta_1}(A, B) + H_{\beta_2}(B, C) + H_{\beta_3}(A, C) - H_{\mu_{1,2}}(B) - H_{\mu_{2,3}}(C) - H_{\mu_{2,3}}(A) \]

subject to

\[ \sum \beta_i |c_i| = 1 \quad i = 1, 2, 3 \]
Double loop algorithm

- Inner loop optimizes $\delta_{ij}$ by iterating the fixed point eqns
- Outer loop optimizes $\theta$ eg using gradient descent
Approximating Z

- Loglik

\[
\ell(\theta : \xi) = \ln \tilde{P}(\xi | \theta) - \ln Z(\theta)
\]

\[
\ln \tilde{P}(\xi | \theta) - \ln \left( \sum_{\xi'} \tilde{P}(\xi' | \theta) \right)
\]

- We can approximate the sum in different ways
Pseudolikelihood

• Define

\[ P(\xi) = \prod_{j=1}^{n} P(x_j \mid x_1, \ldots, x_{j-1}) \]

\[ P(\xi) \approx \prod_{j} P(x_j \mid x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_n) \]

\[
\ell_{\text{pseudo}}(\theta : \mathcal{D}) = \frac{1}{M} \sum_{m} \sum_{j} \ln P(x_j[m] \mid x_{-j}[m], \theta) 
\]

\[ P(x_j \mid x_{-j}) = \frac{P(x_j, x_{-j})}{P(x_{-j})} = \frac{\tilde{P}(x_j, x_{-j})}{\tilde{P}(x_{-j})} 
= \frac{\tilde{P}(x_j, x_{-j})}{\sum_{x_j'} P(x_j', x_{-j})}. \]
Gradient of PL

\[
\ln P(x_j \mid x_{-j}) = \left( \sum_{i : \text{scopes}[f_i] \ni X_j} \theta_i f_i[x_j, u_j] \right) - \ln \left( \sum_{x'_j} \exp \left\{ \sum_{i : \text{scopes}[f_i] \ni X_j} \theta_i f_i[x'_j, u_j] \right\} \right).
\]

\[
\frac{\partial}{\partial \theta_i} \ln P(x_j \mid x_{-j}) = f_i[x_j, x_{-j}] - E_{x'_j \sim P_\theta(X_j \mid x_{-j})} [f_i[x'_j, x_{-j}]].
\]

Proposition 20.6.1:

\[
\frac{\partial}{\partial \theta_i} \ell_{\text{pseudo}}(\theta : D) = \sum_{j : X_j \in \text{scopes}[f_i]} \left( \frac{1}{M} \sum_m f_i[\xi[m]] - E_{x'_j \sim P_\theta(X_j \mid x_{-j}[m])} [f_i[x'_j, x_{-j}[m]]] \right).
\]

(20.31)
Consistency of PL

• Thm 20.6.2 (Besag). If data is generated from our model with params $\theta^*$, then as $M \to \infty$, argmax $\text{PL}(\theta) \to \theta^*$.

• Pf. The empirical approaches $P(\theta^*)$. Hence

$$\frac{1}{M} \sum_{m} f_i[\xi[m]] \longrightarrow \mathbb{E}_{\xi \sim P_{\theta^*}(X)}[f_i[\xi]].$$

• And

$$\frac{1}{M} \sum_{m} E_{x_j \sim P_{\theta^*}(X_j|x_{-j}[m])}[f_i[x'_j, x_{-j}[m]]] = \sum_{x_{-j}} P_D(x_{-j}) \sum_{x'_j} P_{\theta^*}(x'_j | x_{-j}) f_i[x'_j, x_{-j}]$$

$$\longrightarrow \sum_{x_j} P_{\theta^*}(x_{-j}) \sum_{x'_j} P_{\theta^*}(x'_j | x_{-j}) f_i[x'_j, x_{-j}]$$

$$= \mathbb{E}_{\xi \sim P_{\theta^*}}[f_i[\xi]].$$

• Hence gradient of PL is zero at $\theta^*$. 
Problem with PL

- Ex 20.6.3 (cf Hinton’s greek vase)

\[ X_1 \rightarrow X_2 \]
\[ \downarrow \]
\[ Y \]

Assume \( X_1, X_2 \) are strongly correlated (e.g., mirror images), and \( X_1, Y \) and \( X_2, Y \) are less strongly correlated. PL will learn that \( X_1 \) can be predicted from \( X_2 \), and will ignore \( Y \). At test time, if we observe \( Y \) and want to predict \( X_1 \), we are hosed.
Sample based learning

- Recall loglik is

\[
\frac{1}{M} \ell(\theta : D) = \sum_i \theta_i E_D[f_i[d_i]] - \ln Z(\theta),
\]

\[
Z(\theta) = \sum_\xi \exp \left\{ \sum_i \theta_i f_i[\xi] \right\}
\]

\[
= \sum_\xi \frac{Q(\xi)}{Q(\xi)} \exp \left\{ \sum_i \theta_i f_i[\xi] \right\}
\]

\[
= E_Q \left[ \frac{1}{Q(\mathcal{X})} \exp \left\{ \sum_i \theta_i f_i[\mathcal{X}] \right\} \right].
\]

\[
Z(\theta) = E_{P_{\theta^0}} \left[ \frac{Z(\theta^0) \exp \left\{ \sum_i \theta_i f_i[\mathcal{X}] \right\}}{\exp \left\{ \sum_i \theta_i^0 f_i[\mathcal{X}] \right\}} \right]
\]

\[
= Z(\theta^0) E_{P_{\theta^0}} \left[ \exp \left\{ \sum_i (\theta_i - \theta_i^0) f_i(\mathcal{X}) \right\} \right].
\]

\[
\ln Z(\theta) \approx \ln \left( \frac{1}{K} \sum_{k=1}^K \exp \left\{ \sum_i (\theta_i - \theta_i^0) f_i(\xi^k) \right\} \right) + \ln Z(\theta^0).
\]

Sample K x's given \( \theta \)
Compute \( \ln Z(\theta) \)
Update \( \theta \)
Repeat
Contrastive divergence

• Might need to sample many x’s to accurately approximate Z, but this is slow
• So define a set D- of randomly perturbed neighbors of D, and use

\[
\ell_{CD}(\theta : \mathcal{D} \parallel \mathcal{D}^-) = \left[ E_{\xi \sim \hat{P}_D} [\ln P_\theta(\xi)] - E_{\xi \sim \hat{P}_{D^-}} [\ln P_\theta(\xi)] \right],
\]

\[
\frac{\partial}{\partial \theta_i} \ell_{CD}(\theta : \mathcal{D} \parallel \mathcal{D}^-) = E_{\hat{P}_D} [f_i[X]] - E_{\hat{P}_{D^-}} [f_i].
\]

• Often \( x_{i^-} \) is generated by applying 1 step of Gibbs sampling to \( x_i \)
CD for RBMs

• RBMs have 1 layer of hidden variables, so we need an additional expectation

\[
\nabla_i = E_{x \sim D} E_h f_i(x, h) - E_{x \sim D} E_h f_i(x, h)
\]
\[
\approx \frac{1}{N} \sum_n E_{h_n} f_i(x_n, h_n) - E_{h_n} f_i(x_n^-, h_n)
\]
\[
\approx \frac{1}{N} \sum_n f_i(x_n, h_n^+) - f_i(x_n^-, h_n^-)
\]

Stop learning when your dreams match reality

\[
h_n^+ \sim p(h|x_n, \theta) \quad \text{Interpretation of data}
\]
\[
x_n^- \sim p(x|h_n^+, \theta) \quad \text{Reconstruction/ Fantasy data}
\]
\[
h_n^- \sim p(h|x_n^-, \theta) \quad \text{Interpretation of your fantasies}
\]
MAP approximation (perceptron training)

• Let us approximate $Z$ (sum over all $X$) by the MAP estimate. Objective becomes

$$
\frac{1}{M} \ell(\theta ; \mathcal{D}) = \ln P(\xi^{\text{MAP}}(\theta) | \theta), \quad \frac{1}{M} \sum_{m=1}^{M} \ln \tilde{P}(\xi[m] | \theta) - \ln \tilde{P}(\xi^{\text{MAP}}(\theta) | \theta).
$$

• For a single data term

$$
\ln P(\xi[m] | \theta) - \ln P(\xi^{\text{MAP}}(\theta) | \theta) \\
= \left[ \ln \tilde{P}(\xi[m] | \theta) - \ln Z(\theta) \right] - \left[ \ln \tilde{P}(\xi^{\text{MAP}}(\theta) | \theta) - \ln Z(\theta) \right] \\
= \ln \tilde{P}(\xi[m] | \theta) - \ln \tilde{P}(\xi^{\text{MAP}}(\theta) | \theta) \\
= \sum_{i} \theta_i [f_i(\xi[m])] - f_i[\xi^{\text{MAP}}(\theta)].
$$

• Hence gradient is

$$
E_{\mathcal{D}}[f_i[\mathcal{X}]] - f_i[\xi^{\text{MAP}}(\theta)].
$$
Problem with MAP approximation

- The objective is always negative or 0 since

$$\frac{1}{M} \ell(\theta : D) - \ln P(\xi^{\text{MAP}}(\theta) | \theta),$$

$$\ln P(\xi[m] | \theta) \leq \ln P(\xi^{\text{MAP}}(\theta) | \tilde{\theta}),$$

- We can always achieve the maximum of 0 by setting $\theta = 0$

$$\ln P(\xi[m] | \theta) - \ln P(\xi^{\text{MAP}}(\theta) | \theta)$$

$$= [\ln \tilde{P}(\xi[m] | \theta) - \ln Z(\theta)] - [\ln \tilde{P}(\xi^{\text{MAP}}(\theta) | \theta) - \ln Z(\theta)]$$

$$= \ln \tilde{P}(\xi[m] | \theta) - \ln \tilde{P}(\xi^{\text{MAP}}(\theta) | \theta)$$

$$= \sum_i \theta_i [f_i[\xi[m]] - f_i[\xi^{\text{MAP}}(\theta)]] .$$

- “collapsing” problem
Max-margin training

• For conditional density models, we can change the objective to the following, which prevents collapsing:

\[
\ln P_\theta(y[m] | x[m]) - \left[ \max_{y \neq y[m]} \ln P_\theta(y | x[m]) \right].
\]

Find \( \gamma, \theta \) that maximize \( \gamma \)
subject to
\[
\ln P_\theta(y[m] | x[m]) - \ln P_\theta(y | x[m]) \geq \gamma \quad \text{for all } m, y \neq y[m]
\]
\[
\theta^T(f(y[m], x[m]) - f(y, x[m])) \geq \gamma.
\]

To prevent margin blowing up we bound \( \theta \)

Simple-Max-Margin
Find \( \theta \) that minimize \( ||\theta||^2 \)
subject to
\[
\theta^T(f(y[m], x[m]) - f(y, x[m])) \geq 1 \quad \text{for all } m, y \neq y[m]
\]

QP: quad obj+linear constraints
Slack variables

• We want to minimize $||w||^2$ st

$$\forall_i \forall_{Y'_i \neq Y_i} \log p(Y_i|w, X_i) - \log p(Y'_i|w, X_i) \geq 1.$$  

• But we may not be able to achieve this gap, so we introduce slack variables (results in a Hidden Markov Support Vector Machine)

$$\min_{w, \xi} \sum_i \xi_i + \lambda ||w||^2,$$

s.t. $\forall_i \forall_{Y'_i \neq Y_i} \log p(Y_i|w, X_i) - \log p(Y'_i|w, X_i) \geq 1 - \xi_i, \forall_i \xi_i \geq 0$

Thanks to Mark Schmidt
Margin rescaling

• Intuitively if $Y_i'$ is similar to $Y_i$, we don’t mind if their probabilities are similar, but if they are very different, we want the gap to grow.

• This gives max-margin markov network (M3N) aka structural SVM.

$$\min_{w, \xi} \sum_i \xi_i + \lambda \|w\|_2^2,$$

$$\forall i \forall Y_i' \neq Y_i \log p(Y_i|w, X_i) - \log p(Y_i'|w, X_i) \geq \Delta(Y_i, Y_i') - \xi_i, \forall i \xi_i \geq 0,$$

Thanks to Mark Schmidt.
Unconstrained form

• We can eliminate the slack vars to get

\[ \min_w \sum_i \max_{Y_i' \neq Y_i} (\Delta(Y_i, Y_i') - \log p(Y_i | w, X_i) + \log p(Y_i' | w, X_i)) + \lambda ||w||_2^2, \]

• Requires 2^{nd} best decoding. But since \( \Delta(Y_i, Y_i) = 0 \) we can write

\[ \min_w \sum_i \max_{Y_i'} (\Delta(Y_i, Y_i') + \log p(Y_i' | w, X_i)) - \log p(Y_i | w, X_i) + \lambda ||w||_2^2, \]

• This can use generic MAP decoders that just change the local evidence potentials on Y'.
• For associative markov nets, globally optimal.

Thanks to Mark Schmidt
Cutting plane optimization

- Many possible optimization methods
- Simple approach for QP is cutting planes:
- Maximize quad objective with empty set of constraints – this is an upper bound.
- Add a violated constraint (*)
- Repeat until no violations.
- Thm: only need to add a poly num constraints.
- To find if constraints are violated: define

\[ y_{map} = \arg \max_{y \neq y[m]} \tilde{P}(y, x[m]). \]

- If \( P(y[m], x[m]) < p(y_{map}, x[m]) + 1 \), add this violation. Else all constraints for m’th case are ok

\[ \tilde{P}(y[m], x[m]) > \tilde{P}(y_{map}, x[m]) + 1 \geq \tilde{P}(y, x[m]) + 1, \]