TTIC 31180 — Probabilistic Graphical Models

Problem Set #3 Solutions

Due Date: May 29, 2020

1 Kullback-Leibler Divergence [25pts]

(a) **[5 pts]** Prove that $D(P\|Q) \geq 0$ for all distributions P and Q, with equality iff $Q = P$. (Hint: You may find it useful to use Jensen's inequality: $\mathbb{E}[f(X)] \geq f(\mathbb{E}[X])$ for any convex function $f(X)$).

Solution:

$$\begin{aligned}
D(P\|Q) &= \mathbb{E}_P\left[\log \frac{P(X)}{Q(X)}\right] \\
&= -\mathbb{E}_P\left[\log \frac{Q(X)}{P(X)}\right] \\
&\geq -\log\left(\mathbb{E}_P\left[\frac{Q(X)}{P(X)}\right]\right) \\
&= -\log\left(\sum_X P(X)\frac{Q(X)}{P(X)}\right) \\
&= -\log\left(\sum_X Q(X)\right) \\
&= 0
\end{aligned}$$

where line 3 follows from Jensen's inequality, since $\log$ is convex. Thus, $D(P\|Q) \geq 0$.

Alternative We can also show this as follows:

$$
\begin{aligned}
D(P\|Q) &= \mathbb{E}_P\left[\log\frac{P(X)}{Q(X)}\right] \\
&= \sum_X P(X)\log\frac{P(X)}{Q(X)} \\
&= \sum_X Q(X)\frac{P(X)}{Q(X)}\log\frac{P(X)}{Q(X)} \\
&\geq \sum_X Q(X)\left(\frac{P(X)}{Q(X)} - 1\right) \\
&= \sum_X (P(X) - 1) \\
&= 0
\end{aligned}
$$

where the fourth line follows from the fact that $x\log x \geq x - 1$. ■

(b) **[5 pts]** Consider a factored approximation $Q(X, Y) = Q(X)Q(Y)$ to a joint distribution $P(X, Y)$. Show that minimizing $D(P\|Q)$ (the M-projection) would result in setting $Q(X) = P(X)$ and $Q(Y) = P(Y)$, i.e., the optimal approximation is a product of the true marginals under $P(X, Y)$.

Solution:

$$
\begin{aligned}
D(P\|Q) &= \sum_{x,y} P(x,y)\log\frac{P(x,y)}{Q(x,y)} \\
&= \sum_{x,y} P(x,y)\log\frac{P(x,y)P(x)P(y)}{Q(x)Q(y)P(x)P(y)} \\
&= \sum_{x,y} P(x,y)\log\frac{P(x,y)}{P(x)P(y)} + \sum_{x,y} P(x,y)\log\frac{P(x)}{Q(x)} + \sum_{x,y} P(x,y)\log\frac{P(y)}{Q(y)} \\
&= \sum_{x,y} P(x,y)\log\frac{P(x,y)}{P(x)P(y)} + \sum_{x} P(x)\log\frac{P(x)}{Q(x)} + \sum_{y} P(y)\log\frac{P(y)}{Q(y)} \\
&= D(P(x,y)\|P(x)P(y)) + D(P(x)\|Q(x)) + D(P(y)\|Q(y)) \\
&\geq D(P(x,y)\|P(x)P(y))
\end{aligned}
$$

where the last line follows from the non-negativity of KL-divergence. Equality holds iff $Q(X) = P(X)$ and $Q(Y) = P(Y)$. ■

(c) **[5 pts]** Consider the problem of approximating an unknown distribution $P(X)$ with a parameterized distribution $Q(X;\theta)$, where θ is a parameter of the distribution. Suppose that we have access to a set of M I.I.D. samples $\mathcal{D} = \{X^{(1)}, X^{(2)}, \ldots, X^{(M)}\}$ drawn from

$P(X)$, i.e., $X^{(i)} \sim P(X)$. Maximumum likelihood estimation seeks the setting of the parameter θ^* that maximizes the likelihood of the data

$$\theta^* = \text{arg}\max_\theta \prod_{i=1}^{M} Q(X^{(i)}; \theta)$$

Show that this is equivalent to solving for M-projection, i.e., $Q^* = \text{arg min } D(P\|Q)$.

Solution:

Solving for $Q(X)$ that minimizes the M-projection yields

$$\begin{aligned}
\theta^* &= \text{arg}\min_\theta D(P(X)\|Q(X)) \\
&= \text{arg}\min_\theta \mathbb{E}_P\left[\log\frac{P(X)}{Q(X)}\right] \\
&= \text{arg}\min_\theta \left(-H_P(X) - \mathbb{E}_P\left[\log Q(X)\right]\right) \\
&= \text{arg}\max_\theta \mathbb{E}_P\left[\log Q(X)\right]
\end{aligned}$$

where the third line follows from the fact that $H_P(X)$ is not a function of θ. Without access to P, we can use our IID samples to derive an empirical estimate for the expectation

$$\begin{aligned}
\theta^* &\approx \text{arg}\max_\theta \frac{1}{M}\sum_{i=1}^{M} \log Q(X^{(i)}; \theta) \\
&= \text{arg}\max_\theta \prod_{i=1}^{M} \log Q(X^{(i)}; \theta) \\
&= \text{arg}\max_\theta \prod_{i=1}^{M} Q(X^{(i)}; \theta)
\end{aligned}$$

where the last line follows from the fact that the log is monotonic. ■

(d) **[10 pts]** Consider the following form of the joint distribution $P(X, Y)$:

		X			
		1	2	3	4
Y	1	1/8	1/8	0	0
	2	1/8	1/8	0	0
	3	0	0	1/4	0
	4	0	0	0	1/4

Show that the I-projection for $P(X, Y)$ with $Q(X, Y) = Q(X)Q(Y)$ has three distinct minima. Identify these minima and evaluate $D(Q\|P)$ for each of them. What is the value of $D(Q\|P)$ if we set $Q(X, Y) = P(X)P(Y)$?

Solution:

The I-projection follows as the minimization of $D(Q\|P)$:

$$\begin{aligned}
D(Q\|P) &= \sum_{x,y} Q(x,y)\log\frac{Q(x,y)}{P(x,y)} \\
&= \sum_{x,y} Q(x)Q(y)\log\frac{Q(x)Q(y)}{P(x,y)} \\
&= \sum_{x,y} Q(x)Q(y)\left(\log Q(x) + \log Q(y) - \log P(x,y)\right) \\
&= \sum_{x,y} Q(x)Q(y)\log Q(x) + \sum_{x,y} Q(x)Q(y)\log Q(y) - \sum_{x,y} Q(x)Q(y)\log P(x,y) \\
&= \sum_{x} Q(x)\log Q(x) + \sum_{y} Q(y)\log Q(y) - \sum_{x,y} Q(x)Q(y)\log P(x,y) \\
&= -H(Q(X)) - H(Q(Y)) - \sum_{x,y} Q(x)Q(y)\log P(x,y)
\end{aligned}$$

This gives rise to the following constrained optimization problem:

$$\max_Q \left\{ -H(Q(X)) - H(Q(Y)) - \sum_{x,y} Q(x)Q(y)\log P(x,y) \right\}$$

$$\begin{aligned}
\text{subject to} \quad & Q(x) \geq 0, Q(y) \geq 0 \quad \forall x, y \\
& \sum_x Q(x) = 1, \sum_y Q(y) = 1
\end{aligned}$$

We frame this objective in terms of the following Lagrangian

$$L = -H(Q(X)) - H(Q(Y)) - \sum_{x,y} Q(x)Q(y)\log P(x,y) - \lambda_1\left(\sum_x Q(x) - 1\right) - \lambda_2\left(\sum_y Q(y) - 1\right)$$

The associated partial derivatives are

$$\begin{aligned}
\frac{\partial L}{\partial Q(x)} &= 1 + \log Q(x) - \sum_y Q(y)\log P(x,y) - \lambda_1 \\
\frac{\partial L}{\partial Q(y)} &= 1 + \log Q(y) - \sum_x Q(x)\log P(x,y) - \lambda_2
\end{aligned}$$

Setting these equal to zero yields

$$\begin{aligned}
\log Q(x) &= \lambda_1 + \sum_y Q(y) \log P(x,y) - 1 \\
Q(x) &= e^{\lambda_1 + \sum_y Q(y) \log P(x,y) - 1} \\
&= e^{\lambda_1 - 1} \prod_y P(x,y)^{Q(y)} \\
\log Q(y) &= \lambda_2 + \sum_x Q(x) \log P(x,y) - 1 \\
Q(y) &= e^{\lambda_1 + \sum_x Q(x) \log P(x,y) - 1} \\
&= e^{\lambda_2 - 1} \prod_x P(x,y)^{Q(x)}
\end{aligned}$$

In order for the KL-divergence to be valid, we require that $Q(X,Y) = 0$ whenever $P(X,Y) = 0$. We use this below.

(i) Let's suppose that $Q(X=1) \neq 0$. In this case, we have

$$\begin{aligned}
Q(X=1) &= e^{\lambda_1 - 1} \cdot (1/8)^{Q(Y=1)} \cdot (1/8)^{Q(Y=2)} \cdot 0^{Q(Y=3)} \cdot 0^{Q(Y=4)} \neq 0 \\
&= e^{\lambda_1 - 1} \cdot (1/8)^{Q(Y=1)} \cdot (1/8)^{Q(Y=2)},
\end{aligned}$$

where the second line follows from the requirement that $Q(Y=3) = Q(Y=4) = 0$ in order for this term to be non-zero. This implies that $Q(Y=1) \neq 0$ and/or $Q(Y=2) \neq 0$. If we then consider $X = 2$, we have

$$Q(X=2) = e^{\lambda_1 - 1} \cdot (1/8)^{Q(Y=1)} \cdot (1/8)^{Q(Y=2)}$$

We see that $Q(X=1) = Q(X=2)$.
Meanwhile, we see that $Q(Y=1)$ and $Q(Y=2)$ are equal

$$\begin{aligned}
Q(Y=1) &= e^{\lambda_2 - 1} \cdot (1/8)^{Q(X=1)} \cdot (1/8)^{Q(X=2)} \cdot 0^{Q(X=3)} \cdot 0^{Q(X=4)} \\
Q(Y=2) &= e^{\lambda_2 - 1} \cdot (1/8)^{Q(X=1)} \cdot (1/8)^{Q(X=2)} \cdot 0^{Q(X=3)} \cdot 0^{Q(X=4)}
\end{aligned}$$

in which case both are non-zero, which requires that $Q(X=3) = Q(X=4) = 0$. Since $Q(X=1) = Q(X=2)$ while $Q(X=3) = Q(X=4) = 0$ and similarly that $Q(Y=1) = Q(Y=2)$ while $Q(Y=3) = Q(Y=4) = 0$, we conclude that $Q(X=1) = Q(X=2) = 1/2$ and $Q(Y=1) = Q(Y=2) = 1/2$. This gives rise to the following fixed point

$$\boxed{Q(X) = \begin{cases} 1/2 & X=1, X=2 \\ 0 & X=3, X=4 \end{cases} \qquad Q(Y) = \begin{cases} 1/2 & Y=1, Y=2 \\ 0 & Y=3, Y=4 \end{cases}}$$

The corresponding KL-divergence follows as

$$\boxed{\begin{aligned} D(Q_1 \| P) &= \frac{1}{4} \log \frac{1/4}{1/8} + \frac{1}{4} \log \frac{1/4}{1/8} + \frac{1}{4} \log \frac{1/4}{1/8} + \frac{1}{4} \log \frac{1/4}{1/8} \\ &= \log 2 \end{aligned}}$$

(ii) Now, instead suppose that $Q(X=3) \neq 0$:

$$\begin{aligned} Q(X=3) &= e^{\lambda_1 - 1} \cdot 0^{Q(Y=1)} \cdot 0^{Q(Y=2)} \cdot (1/4)^{Q(Y=3)} \cdot 0^{Q(Y=4)} \neq 0 \\ &= e^{\lambda_1 - 1}(1/4)^{Q(Y=3)} \end{aligned}$$

where the second line follows from the requirement that $Q(Y=1) = Q(Y=2) = Q(Y=4) = 0$, in which case $Q(Y=3) = 1$.
Looking at the expression for $Q(Y=3)$:

$$\begin{aligned} Q(Y=3) &= e^{\lambda_2 - 1} \cdot 0^{Q(X=1)} \cdot 0^{Q(X=2)} \cdot (1/4)^{Q(X=3)} \cdot 0^{Q(X=4)} \neq 0 \\ &= e^{\lambda_1 - 1}(1/4)^{Q(X=3)}, \end{aligned}$$

implies $Q(X=1) = Q(X=2) = Q(X=4) = 0$, in which case $Q(X=3) = 1$.
This gives rise to a second fixed point

$$\boxed{Q(X) = \begin{cases} 1 & X=3 \\ 0 & X=1, X=2, X=4 \end{cases} \qquad Q(Y) = \begin{cases} 1 & Y=3 \\ 0 & Y=1, Y=2, Y=4 \end{cases}}$$

The corresponding KL-divergence follows as

$$\boxed{D(Q_2 \| P) = 1 \log \frac{1}{1/4} = 2}$$

(iii) If we suppose instead that $Q(X=4) \neq 0$, the following fixed point follows as above by symmetry

$$\boxed{Q(X) = \begin{cases} 1 & X=4 \\ 0 & X=1, X=2, X=3 \end{cases} \qquad Q(Y) = \begin{cases} 1 & Y=4 \\ 0 & Y=1, Y=2, Y=3 \end{cases}}$$

The corresponding KL-divergence follows as

$$\boxed{D(Q_2 \| P) = 1 \log \frac{1}{1/4} = 2}$$

■

2 Approximating the Marginal Polytope [10pts]

The local consistency polytope is an approximation to the marginal polytope. This question asks you to investigate the local consistency polytope.

(a) **[5 pts]** Show that for any clique tree, the local consistency polytope is equal to the marginal polytope. You need not prove this from scratch, and are allowed to cite any theorems from the textbook.

Solution:

We will show that they are equivalent by showing that by showing that $\mathit{Marg}[\mathcal{U}] \subseteq \mathit{Local}[\mathcal{U}]$ and then that $\mathit{Local}[\mathcal{U}] \subseteq \mathit{Marg}[\mathcal{U}]$.

$\mathit{Marg}[\mathcal{U}] \subseteq \mathit{Local}[\mathcal{U}]$: For a general cluster graph $\mathcal{U}$, the local consistency polytope only requires that the beliefs be pseudo-marginals. Since this is true for the marginal polytope, then the marginal polytope is contained within the local consistency polytope.

$\mathit{Local}[\mathcal{U}] \subseteq \mathit{Marg}[\mathcal{U}]$: Consider a clique tree $\mathcal{U}$ and some $Q \in \mathit{Local}[\mathcal{U}]$. Since the clique tree is locally consistent, it is also calibrated. Per Theorem 10.4 in Koller & Friedman, that the clique tree induces a measure $Q \propto \tilde{P}_{\Phi}(\mathcal{X})$ and that the beliefs are marginals of $\tilde{P}_{\Phi}(\mathcal{X})$, i.e., $\beta(\boldsymbol{C}_i) = P_{\Phi}(\boldsymbol{C}_i)$. Consequently $Q \in \mathit{Marg}[\mathcal{U}]$.

Thus, for a clique tree $\mathcal{U}$, $\mathit{Marg}[\mathcal{U}] \subseteq \mathit{Local}[\mathcal{U}]$ and $\mathit{Local}[\mathcal{U}] \subseteq \mathit{Marg}[\mathcal{U}]$, thus $\mathit{Marg}[\mathcal{U}] = \mathit{Local}[\mathcal{U}]$ ■

(b) **[5 pts]** Show that for some cluster graph that is *not* a tree, the marginal polytope is *strictly* contained in the local consistency polytope. In other words, provide an example of a parametrization of a graphical model that satisfies the local consistency constraints, but does not correspond to any valid probability distribution.

Solution:

While it was not the intention, the cluster graph given in Problem 3 is an example of a parameterization that satisfies the local consistency constraints, however as the solution to Problem 3(b) shows, beliefs are not valid marginals under a common joint distribution. Thus, this example lies in the local consistency polytope, but does not lie within the marginal polytope. ■

3 Belief Propagation and Bethe Free Energy [10pts]

The fixed points of belief propagation correspond to the minimum stationary points of the Bethe approximation of the free energy (Eqn. 11.19 in Koller & Friedman) for a factor graph. The belief propagation algorithm attains an approximation to the true distribution P when the graph has cycles.

(a) **[5 pts]** This problem asks you to consider a simple example of a distribution that minimizes the Bethe free energy, but is not even a valid distribution.

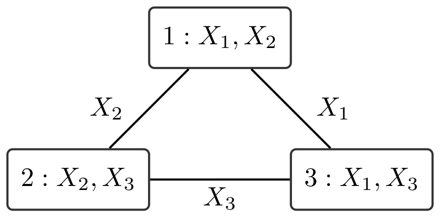


Figure 1: Cluster graph for Problem 3(a).

Consider a simple pairwise Markov random field over three binary variables X_1, X_2, and X_3 with the corresponding cluster graph shown in Figure 1. Suppose that the node and sepset beliefs are given by

$$\mu_{1,3}(X_1) = \mu_{1,2}(X_2) = \mu_{2,3}(X_3) = (0.5, 0.5)$$
$$\beta_1(X_1, X_2) = \begin{bmatrix} 0.4 & 0.1 \\ 0.1 & 0.4 \end{bmatrix} \; \beta_2(X_2, X_3) = \begin{bmatrix} 0.4 & 0.1 \\ 0.1 & 0.4 \end{bmatrix} \; \beta_3(X_1, X_3) = \begin{bmatrix} 0.1 & 0.4 \\ 0.4 & 0.1 \end{bmatrix}$$

Show that there can't be a distribution $P(X_1, X_2, X_3)$ with the above beliefs as its marginals. (Hint: Compute $P(X_1 = 0, X_2 = 0, X_3 = 0), \ldots, P(X_1 = 1, X_2 = 1, X_3 = 1)$ and show that they can't constitute a valid distribution).

Solution:

Let's assume that the beliefs are valid marginals for some distribution $P(X_1, X_2, X_3)$. In this case, we have

$$\beta_1(X_1, X_2) = \sum_{X_3} P(X_1, X_2, X_3) = \begin{bmatrix} P(0,0,0) + P(0,0,1) & P(0,1,0) + P(0,1,1) \\ P(1,0,0) + P(1,0,1) & P(1,1,0) + P(1,1,1) \end{bmatrix} = \begin{bmatrix} 0.4 & 0.1 \\ 0.1 & 0.4 \end{bmatrix}$$
$$\beta_2(X_2, X_3) = \sum_{X_1} P(X_1, X_2, X_3) = \begin{bmatrix} P(0,0,0) + P(1,0,0) & P(0,0,1) + P(1,0,1) \\ P(0,1,0) + P(1,1,0) & P(0,1,1) + P(1,1,1) \end{bmatrix} = \begin{bmatrix} 0.4 & 0.1 \\ 0.1 & 0.4 \end{bmatrix}$$
$$\beta_3(X_1, X_3) = \sum_{X_2} P(X_1, X_2, X_3) = \begin{bmatrix} P(0,0,0) + P(0,1,0) & P(0,0,1) + P(0,1,1) \\ P(1,0,0) + P(1,1,0) & P(1,0,1) + P(1,1,1) \end{bmatrix} = \begin{bmatrix} 0.1 & 0.4 \\ 0.4 & 0.1 \end{bmatrix}$$

From $\beta_3(X_1, X_2)$, we have that $P(1,1,1) \leq 0.1$, and from $\beta_2(X_2, X_3)$, we have that $P(0,1,1) \geq 0.3$. However this is inconsistent with $\beta_1(X_1, X_2)$, which expresses $P(0,1,1) \leq 0.1$. Thus the beliefs are not consistent with a common distribution. ■

(b) **[5 pts]** Consider a distribution $P_\Phi(A, B, C, D)$ associated with the Markov random field shown in Figure 2(a).

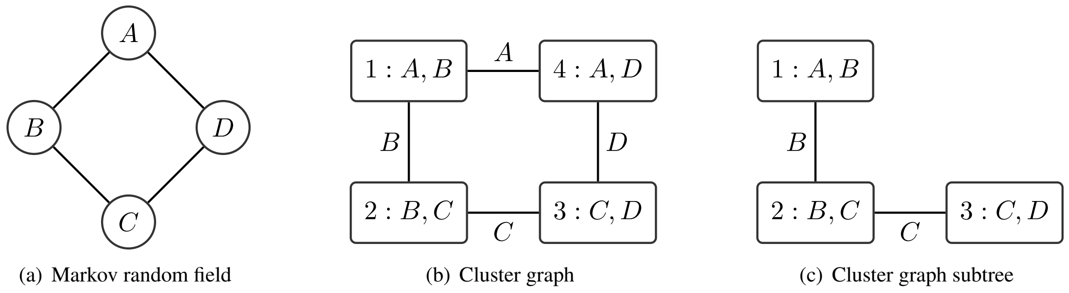


Figure 2: A (a) Markov random field and (b), one choice for the cluster graph, and (c) a potential subtree.

Figure 2(b) depicts one choice for the corresponding cluster graph. As discussed in lecture, belief propagation over the cluster graph results in cluster node and sepset beliefs that are not marginals for the original distribution $P_\Phi(A, B, C, D)$. However, if we remove C_4, they are valid marginals for the reparameterized distribution $P_{\mathcal{T}}(A,B,C,D)$ associated with the resulting subtree $\mathcal{T}$ depicted in Figure 2(c).

The cluster tree distribution and the original distribution obey the following relation

$$P_{\mathcal{T}}(A,B,C,D) = P_{\Phi}(A,B,C,D)\frac{\mu_{3,4}(D)\mu_{1,4}(A)}{\beta_4(A,D)}$$

Show how we can use this residual to bound the error on the estimation of marginals in the cluster graph.

Solution:

Rearranging terms in the expression above, we have

$$\begin{aligned} P_{\Phi}(A,B,C,D) &= P_{\mathcal{T}}(A,B,C,D)\frac{\beta_4(A,D)}{\mu_{3.4}(D)\mu_{1,4}(A)} \\ &= P_{\mathcal{T}}(A,B,C,D)r(A,D) \end{aligned}$$

where $r(A,D) = \frac{\beta_4(A,D)}{\mu_{3.4}(D)\mu_{1,4}(A)}$.

Let's consider the single-node marginal $\beta_1(A,B)$ for the calibrated cluster graph and, equivalently, the cluster subtree. The error between the true marginal $P_{\Phi}(A,B) = \sum_{C,D} P_{\Phi}(A,B,C,D)$ and that computed from the calibrated cluster graph follows as

$$\begin{aligned} P_{\Phi}(A,B) - \beta_1(A,B) &= \sum_{C,D} \left(P_{\Phi}(A,B,C,D) - P_{\mathcal{T}}(A,B,C,D)\right) \\ &= \sum_{C,D} P_{\mathcal{T}}(A,B,C,D)\left(r(A,D) - 1\right) \end{aligned}$$

Let

$$\begin{aligned} r_{\text{min}}(A) &= \min_D r(A,D) \\ r_{\text{max}}(A) &= \max_D r(A,D) \end{aligned}$$

This yields the following bounds on the error

$$\boxed{\beta_1(A,B)r_{\text{min}}(A) \le P_{\Phi}(A,B) - \beta_1(A,B) \le \beta_1(A,B)r_{\text{max}}(A)}$$

The results for other marginals follow similarly.

Another way of looking at this is to consider the error of the marginal for specific instantiations of the random variables, e.g., for $A = a, B = b$

$$\begin{aligned} P_{\Phi}(a,b) - \beta_1(a,b) &= \sum_{C,D} P_{\mathcal{T}}(A,B,C,D)\left(r(a,D) - 1\right) \\ &= \sum_{A,B,C,D} P_{\mathcal{T}}(A,B,C,D)\left(r(a,D) - 1\right) \cdot \delta(A = a) \cdot \delta(B = b) \\ &= \mathbb{E}_{P_{\mathcal{T}}}[(r(A,D) - 1) \cdot \delta(A = a) \cdot \delta(B = b)] \end{aligned}$$

where $\delta(A = a)$ and $\delta(B = b)$ are indicator functions that are 1 when $A = a$ and $B = b$, respectively.

Viewed in this way, the error exactly corresponds to an expectation of $(r(A, D) - 1) \cdot \delta(A = a) \cdot \delta(B = b)$ under the distribution that factorizes according to the subtree. Computing this expectation exactly may be intractable since, as discussed in Koller and Friedman (Section 11.3.3), the residual term involves all variables incident with an edge that was removed to form the subtree. In fact, if it was easy to compute the error, then we could determine the exact marginal by first computing the marginal associated with the subtree and then computing the error, eliminating the need for an approximate inference algorithm. ■

4 Structured Variational Approximations for Factorial HMMs [10pts]

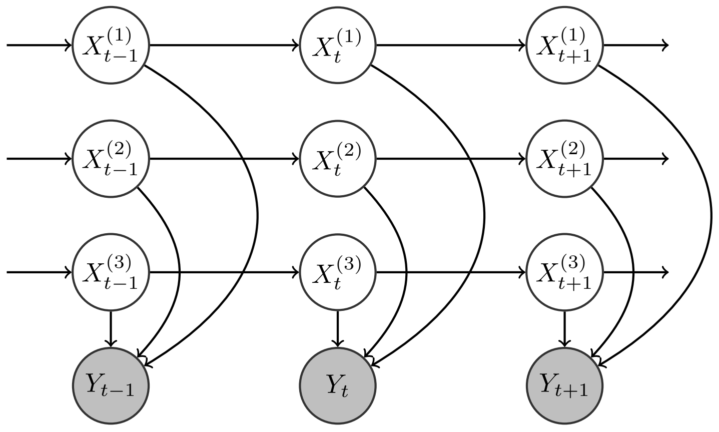


Figure 3: Bayesian network for a factorial HMM .

Consider a dynamic Bayesian network constructed from a hidden Markov model with latent, sequential random variables $\boldsymbol{X}_t = \{X_t^{(1)}, X_t^{(2)}, \ldots X_t^{(n)}\}$ and observed random variables Y_t, where we assume that Y_t is a function of $\boldsymbol{X}_t$. This is known as a factorial HMM and its Bayesian network is visualized in Figure 3.

Find a variational update rule for the structured variational approximation that factorizes as a set of clusters associated with the individual chains, i.e., a cluster $\{X_0^{(i)}, X_1^{(i)}, \ldots, X_T^{(i)}\}$ for each i. Make sure that you simplify the clusters as much as possible, as discussed in Sections 11.5.2.3 and 11.5.2.4 of Koller & Friedman.

Solution:

The original distribution factorizes as

$$P(\boldsymbol{X}_t) \propto \left[\prod_{i=1}^{n}\prod_{t=0}^{T-1} \phi(X_t^{(i)}, X_{t+1}^{(i)})\right] \cdot \left[\prod_{t=0}^{T} \phi(X_t^{(1)}, X_t^{(2)}, \ldots, X_t^{(n)})\right]$$

where we omit the subscript from each factor, since their identify is clear from their scope, i.e., $\phi(X_t^{(i)}, X_{t+1}^{(i)}) = \phi_{t,t+1}(X_t^{(i)}, X_{t+1}^{(i)})$.

Our goal is to derive the update rules the structured variational approximation that assumes that the individual chains are independent:

$$\begin{tikzcd}
{} \arrow[r] & X_{t-1}^{(1)} \arrow[r] & X_t^{(1)} \arrow[r] & X_{t+1}^{(1)} \arrow[r] & {} \\
{} \arrow[r] & X_{t-1}^{(2)} \arrow[r] & X_t^{(2)} \arrow[r] & X_{t+1}^{(2)} \arrow[r] & {} \\
{} \arrow[r] & X_{t-1}^{(3)} \arrow[r] & X_t^{(3)} \arrow[r] & X_{t+1}^{(3)} \arrow[r] & {}
\end{tikzcd}$$

where we have incorporated $\{Y_t\}$ as observed parameters. The corresponding distribution factorizes as

$$\begin{aligned}
Q(\boldsymbol{X}_t) &\propto \prod_{i=1}^{n} Q(X_0^{(i)}) \prod_{t=1}^{T} Q(X_t^{(i)} \mid X_{t-1}^{(i)}) \\
&= \prod_{i=1}^{n} \psi_t(X_0^{(i)}) \prod_{t=1}^{T} \psi_{t-1,t}(X_t^{(i)}, X_{t-1}^{(i)})
\end{aligned}$$

where we leaving the number of chains n unspecified. Note that we will drop the subscripts on the factors when their identity is clear from their scope (i.e., $\psi_{t-1,t}(X_t^{(i)}, X_{t-1}^{(i)}) \to \psi(X_t^{(i)}, X_{t-1}^{(i)})$).

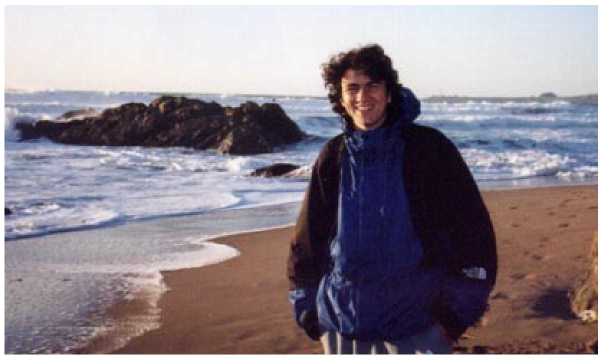
(i) For a clique of the form $\{X_0^{(i)}\}$, the update rule follows as

$$\begin{aligned}
\psi(X_0^{(i)}) &\propto \exp\left\{ \sum_{j>i} \mathbb{E}_Q[\ln \phi(X_0^{(i)}, X_0^{(j)}) | X_0^{(i)}] - \mathbb{E}_Q[\ln \psi(X_0^{(i)}, X_1^{(i)}) | X_0^{(i)}] \right\} \\
&= \exp\left\{ (n-i) \ln \varphi(X_0^{(i)}) - \mathbb{E}_Q[\ln \psi(X_0^{(i)}, X_1^{(i)}) | X_0^{(i)}] \right\}
\end{aligned}$$

where, because $Q \vDash X_0^{(i)} \perp X_0^{(j)}$ for all $j > i$, the first term in the first line becomes

$$\begin{aligned}
\mathbb{E}_Q[\ln \phi(X_0^{(i)}, X_0^{(j)}) | X_0^{(i)}] &= \mathbb{E}_Q[\ln \varphi(X_0^{(i)}) \mid X_0^{(i)}] + \mathbb{E}_Q[\ln \varphi(X_0^{(j)}) | X_0^{(i)}] \\
&= \ln \varphi(X_0^{(i)}) + \mathbb{E}_Q[\varphi(X_0^{(j)})]
\end{aligned}$$

where $\phi(X_0^{(i)}, X_0^{(j)}) = \varphi(X_0^{(i)}) \varphi(X_0^{(j)})$.


(a) Original Image

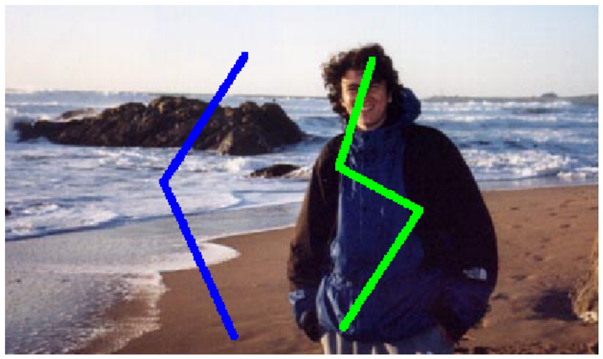

(b) Labeled Image

Figure 4: Interactive foreground/background segmentation of an input image.

(ii) For cliques of the form $\{X_t^{(i)}, X_{t+1}^{(i)}\}$, the update follows as

$$\begin{aligned}
t = 0: \quad \psi(X_t^{(i)}, X_{t+1}^{(i)}) &\propto \exp\left\{\mathbb{E}_Q[\ln \phi(X_{t+1}^{(i)}, X_{t+2}^{(i)}) \mid X_t^{(i)}, X_{t+1}^{(i)}]\right\} \\
&\quad + \sum_{j>i} \mathbb{E}_Q[\ln \phi(X_t^{(i)}, X_t^{(j)}) \mid X_t^{(i)}, X_{t+1}^{(i)}] \\
&\quad + \sum_{j>i} \mathbb{E}_Q[\ln \phi(X_{t+1}^{(i)}, X_{t+1}^{(j)}) \mid X_t^{(i)}, X_{t+1}^{(i)}] \\
&\quad - \mathbb{E}_Q[\ln \psi(X_{t+1}^{(i)}, X_{t+2}^{(i)}) \mid X_t^{(i)}, X_{t+1}^{(i)}] \\
&= \exp\Big\{\mathbb{E}_Q[\ln \phi(X_{t+1}^{(i)}, X_{t+2}^{(i)}) \mid X_{t+1}^{(i)}] \\
&\quad + (n-i) \ln \varphi(X_t^{(i)}) + (n-1) \ln \varphi(X_{t+1}^{(i)}) \\
&\quad - \mathbb{E}_Q[\ln \psi(X_{t+1}^{(i)}, X_{t+2}^{(i)}) \mid X_{t+1}^{(i)}]\Big\}
\end{aligned}$$

where the simplification in the second-to-last line follows from $Q \vDash X_t^{(i)} \perp X_t^{(j)}$ for all $j > i$.

$$\begin{aligned}
t > 0: \quad \psi(X_t^{(i)}, X_{t+1}^{(i)}) \propto \exp\Big\{&\mathbb{E}_Q[\ln \phi(X_{t-1}^{(i)}, X_t^{(i)}) \mid X_t^{(i)}] + \mathbb{E}_Q[\ln \phi(X_{t+1}^{(i)}, X_{t+2}^{(i)}) \mid X_{t+1}^{(i)}] \\
&+ (n-i) \ln \varphi(X_t^{(i)}) + (n-1) \ln \varphi(X_{t+1}^{(i)}) \\
&- \mathbb{E}_Q[\ln \psi(X_{t+1}^{(i)}, X_{t+2}^{(i)}) \mid X_{t+1}^{(i)}]\Big\}
\end{aligned}$$

■

5 Image Segmentation with Loopy Belief Propagation [45pts]

In lecture, we used image processing tasks to motivate the need for approximate inference strategies. One such task is foreground/background estimation as a specific form of K-ary segmentation, where the goal is to label each pixel in an image as belonging to the foreground or background. This problem asks you to implement loopy belief propagation to perform semi-supervised foreground/background segmentation based on user-provided examples of each class. For example, suppose that Carlos (a famous machine learning researcher) wants to cut himself out of a picture (Fig.4(a)) so that he can paste it against a different background (maybe a wintery scene given his jacket). Carlos is lazy and the most effort that he is willing to put in is to draw two scribbles on the image (Fig. 4(b)), one labeling some foreground pixels (green) and the other labeling background pixels (blue).

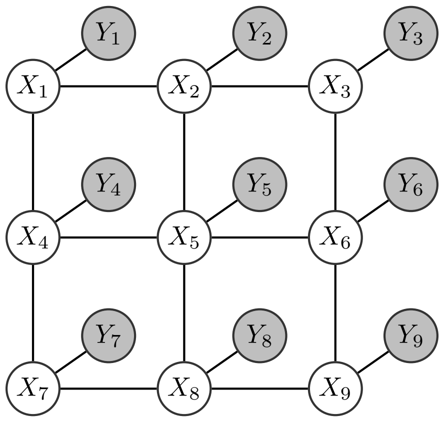


Figure 5: A Markov random field for image segmentation.

This problem can be represented with a pairwise Markov random field $\mathcal{U}$ (Fig. 5), where $X_i \in \boldsymbol{X}$ is the binary-valued segment label for (super)pixel i, and $Y_i \in \boldsymbol{Y}$ is an observed feature for that (super)pixel (e.g., it could be the intensity). This gives rise to a joint distribution of the form

$$P(\boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{Z} \prod_i \phi_i(X_i, Y_i) \prod_{(i-j)\in E_{\mathcal{U}}} \phi_{i,j}(X_i, X_j), \tag{1}$$

where the factor $\phi_i(X_i, Y_i)$ relates a pixel's class to its feature representation Y_i, and $\phi_{i,j}(X_i, X_j)$ expresses the influence among neighboring pixels.

A well-known problem with reasoning densely over individual pixels is that the MRFs tend to be quite large (e.g., even a small 300×300 image results in an MRF with 9×10^4 nodes). As we discussed in one of the first few lectures, a common way to deal with this is to generate a more compact MRF over superpixels (Fig. 6(b)), i.e., groups of neighboring pixels that result from an over-segmentation of the image. With this approach, we will modify our problem formulation slightly, letting X_i denote the binary label associated with superpixel i and Y_i be an observed feature of all the pixels assigned to superpixel i. Having performed inference, we then assign the resulting label for superpixel i to all of its corresponding pixels in the original image.

Foreground/Background Segmentation

Your task is to implement semi-supervised foreground/background segmentation via loopy belief propagation. You should provide a function `segment.py` that can be called as

```
$ python segment.py orig-img.png super-map.mat scribble-mask.mat beta
```

where `orig-img.png` is the original image, `super-map.mat` and `scribble-mask.mat` are paths to `mat` files for the superpixel map and scribble mask (defined below), respectively, and `beta` is a parameter that we will define shortly. This function should save the resulting segmented image as `segmented-img.png`.

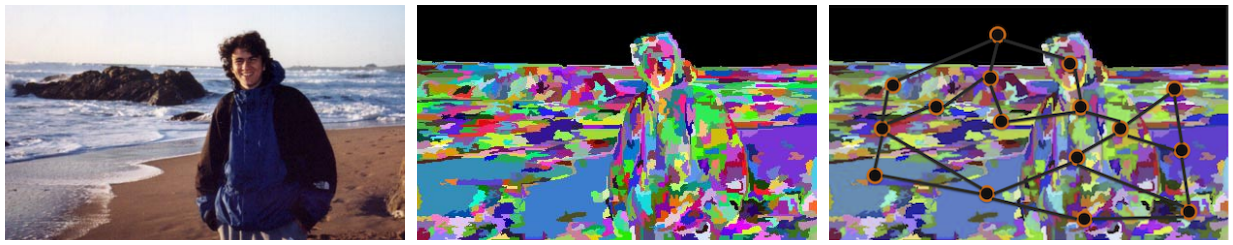

(a) Original Image (b) Superpixels (c) Neighborhood Graph

Figure 6: Superpixels and corresponding neighborhood graph.

Given an image, the foreground/background image sketch, and the superpixels for the image, your implementation will need to (i) generate the features Y_i for each superpixel based upon appearance statistics (we will use Gaussian mixture models); (ii) generate the adjacency matrix for the superpixel image; and (iii) perform loopy belief propagation. The following questions step you through this process.

(a) **[10 pts]** We will define the features Y_i for each superpixel in terms a separate Gaussian mixture model (GMM) for the foreground and background superpixels. Each mixture model will be defined over the pixel intensities in the CIELUV color space. Note that the original image is in the RGB color space, and you will need to convert it to the CIELUV color space (the problem set includes a `Rgb2Luv.py` class that will do this conversion for you.

The problem set includes a a scribble mask (`carlos-beach-scribble-mask.mat`) that is the same size as the original image, where a value of 1 indicates that the corresponding pixel in the original image lies in the foreground, 2 if it is part of the background, and 0 otherwise. Use the foreground and background pixels in the original image to learn a Gaussian mixture model over CIELUV intensities for each class with 5 components per mixture. You may find scikit-learn (link) useful, particularly the `sklearn.mixture.GaussianMixture` (link (note: if you get a 404 error, try changing `%23` in the URL to `#`)) class, which provides a `fit()` function that fits the model (link (note: if you get a 404 error, try changing `%23` in the URL to `#`)). If you use `sklearn.mixture.GaussianMixture`, you may want to set `covariance_type` to `diag`, but the defaults for the other parameters are fine.

What to turn in: Report the mean vectors and diagonal of the covariance matrices (as two 3×5 tables each) for the foreground and background GMMs.

Solution:

The results vary with different versions of Python (presumably due to different versions of scikit-learn). In some cases, I've seen foreground parameters that match those for Python 2.7 and background parameters that match those from Python 3.

Using a diagonal covariance matrix for each component, you get the following

Python 2.7

Foreground Means

1:	28.39596503	10.6230484	-23.32150524
2:	54.73947190	-1.72193471	-27.84652448
3:	84.63740754	17.15899049	16.56317764
4:	54.50065485	22.40060389	4.85031228
5:	36.52520608	-0.12727288	-46.72315787

Foreground Covariance (diagonal)

1:	44.91839799	15.07363376	99.46434611
2:	108.34387870	5.39853917	70.84211236
3:	35.33265174	10.64629835	12.81092907
4:	131.64964184	77.91813844	75.32319417
5:	7.04491018	10.15429068	27.43684325

Background Means

1:	97.85477618	0.98057927	0.88588536
2:	67.91295776	18.51847214	13.04651211
3:	44.28240071	5.19183086	-9.3172613
4:	87.88612379	1.36002163	-3.50898318
5:	75.98499128	-2.30883509	-11.15103079

Background Covariance (diagonal)

1:	0.17657164	0.44547257	1.49001094
2:	6.27412289	1.65730431	4.69503092
3:	25.89589439	12.02964117	18.17568327
4:	19.86005735	3.90382752	10.53124074
5:	67.36390510	1.74035412	6.58456978

Python 3

Foreground Means

1:	64.62573353	21.18150835	9.07370055
2:	37.07983392	-2.16458367	-51.59718824
3:	28.72076187	10.90250160	-21.89361188
4:	54.61138812	-1.72414664	-27.75309043
5:	36.12459619	1.05021833	-44.18398521

Foreground Covariance (diagonal)

1:	288.77301504	60.40836037	78.60514198
2:	3.93759138	0.88488078	5.29163999
3:	52.67915358	14.60144466	94.02390134
4:	110.52502939	5.39811575	73.59512034
5:	9.08181228	12.26073171	21.17716859

Background Means

1:	67.91295776	18.51847214	13.04651210
2:	87.88808724	1.36075670	-3.50705665
3:	75.99833049	-2.30743842	-11.14675711
4:	97.85525946	0.98045310	0.88582829
5:	44.28218909	5.19187859	-9.31727507

Background Covariance (diagonal)

1:	6.27412297	1.65730436	4.69503093
2:	19.86848691	3.90191699	10.52888548
3:	67.40760423	1.74099656	6.59021322
4:	0.17594932	0.44553764	1.48958106
5:	25.89261304	12.02960441	18.17585428

■

(b) **[10 pts]** Next, we will generate the superpixel graph structure. The problem set includes a superpixel map file (`carlos-beach-sp.mat`, variable `labels`) that assigns a superpixel index to each pixel in the original image. Using this map, generate an adjacency matrix for the superpixel neighborhood graph. If the image is divided into S superpixels, this matrix will be of size $S \times S$, and will contain a 1 whenever two superpixels are adjacent (and 0 otherwise).

Hint: Consider scanning rows and columns of the superpixel map to find transitions.

What to turn in: Provide a visualization of the adjacency matrix as an image.

(c) **[20 pts]** The features at superpixels Y_i will be the average color (CIELUV vectors) of the pixels associated each superpixel. The node potentials will be defined as the likelihood of these feature vectors under the foreground and background GMMs from above,

$$\phi_i(X_i = \text{fg}, Y_i) = P(Y_i \,|\, \text{GMM}_{\text{fg}})$$
$$\phi_i(X_i = \text{bg}, Y_i) = P(Y_i \,|\, \text{GMM}_{\text{bg}})$$

Note that `sklearn.mixture.GaussianMixture` provides a `score_samples` function that returns the log probability of a given sample under the associated mixture, which you can take the exponential of to get the probability.

Meanwhile, the edge potential will be defined as

$$\phi_{i,j}(X_i, X_j) = \exp\left(-\beta \times \mathbb{1}(X_i \neq X_j)\right)$$

where $\mathbb{1}$ is an indicator function.

Generate a cluster graph using these factors according to the adjacency matrix. Feel free to use the structure depicted in Figure 11.6 of Koller & Friedman, which defines a node for each unary and pairwise potential.

Implement loopy belief propagation (see Algorithm 11.1 in Koller & Friedman for a sketch) to calibrate the cluster graph with $\beta = 2$. Stop running loopy belief propagation once the maximum absolute difference between an old and new message is less than 10^{-5} (i.e., use this as a indication that the graph is calibrated). Remember to initialize each message to 1 and to normalize them. Hint: compute messages in log-space to avoid numerical instabilities. (You do not need to dampen messages to ensure convergence.)

What to turn in: In addition to your code (see below), an image showing the result with $\beta = 2$.

(d) **[5 pts]** Vary $\beta \in [0, 10]$ with a step size of 2 and propagate superpixel labels back to their constituent pixel segmentations. Why is $\beta = 0$ special? What would happen as $\beta \to \infty$? What behavior do you see as β increases?

What to turn in: Include the resulting images (e.g., assigning 1 (or 128) to foreground pixels, 2 (or 255) to background pixels, and 0 to all others) for each setting of β (make it clear which setting each corresponds to).

What is included: An archive file (`ps3-segmentation.tar`) that contains the original image (`carlos-beach.png`), segmentation map (`carlos-beach-sp.mat`), scribble mask (`carlos-beach-scribble-mask.mat`), and a Python file (`Rgb2Luv.py`) that converts RGB to CIELUV.

What to turn in: In addition to the items requested above, provide all the code necessary to run your `segment.py` via the command-line call given earlier. If you use Python3, please indicate it somehow (e.g., you can name your file `segment3.py`).