Notes on Variational Inference

Variational methods are a general family of techniques that make deterministic approximations to complex probabilistic models. Variational methods can be used to approximate posteriors in Bayesian models, and also to estimate marginal probabilities and MAP assignments in graphical models. While variational methods and stochastic simulation methods such as MCMC address many of the same problems, they differ greatly in their approach. Variational methods are based on deterministic approximation and numerical optimization, while simulation methods are based on random sampling. Variational methods have been successfully applied to a wide range of problems, but they come with weak theoretical guarantees.

1. Introduction

Variational methods are increasingly being used in the machine learning community as alternatives to *Markov chain Monte Carlo* methods for complex probabilistic models. In a variational method, one forms an approximate model that depends on free "variational" parameters, and them optimizes over those parameters with respect to some objective function. Of course, when viewed this way, the notion of a variational approximation is very broad.

In these notes we focus on variational methods fo approximate posterior inference for some simple Bayesian mixture models, and also on methods that have been developed for *graphical model*s. In a graphical model over a joint random vector (X, Y), where both X and Y are multidimensional, it is of interest to make statistical inferences including the following:

- marginal probabilities $\mathbb{P}(X_i = x \,|\, Y)$,
- most probable assignments $x^\star = \arg\max_x \mathbb{P}(\{X_i = x_i\} \,|\, Y)$,
- maximum marginals $x_i^\star = \arg\max_{x_i} \mathbb{P}(X_i = x_i \,|\, Y)$,
- likelihoods $\mathbb{P}(X, Y)$.

Each of these quantities is intractable to calculate exactly for general graphical models, so it is essential to develop approximation strategies.

2. Variational Inference for a Mixture Model

Fix two distributions F_0 and F_1, with densities $f_0(x)$ and $f_1(x)$, and form the mixture model

$$\begin{aligned} \theta &\sim \text{Beta}(\alpha, \beta) && (1) \\ X \,|\, \theta &\sim \theta F_1 + (1-\theta) F_0. && (2) \end{aligned}$$

Then the likelihood for data $x_1, \ldots, x_n$ is

$$p(x_{1:n}) = \int_0^1 \text{Beta}(\theta \,|\, \alpha, \beta) \prod_{i=1}^n \left(\theta f_1(x_i) + (1-\theta) f_0(x_i)\right) d\theta. \tag{3}$$

Our goal, for now, is to compute or approximate the posterior distribution $p(\theta \,|\, x_{1:n})$ over the mixing weight θ.

2.1. The Gibbs sampler

In the natural *Gibbs sampler* for this model, we introduce a latent binary variable z_i to indicate which mixture component item x_i was generated from, and iterate the following two steps:

1. Sample $Z_i \,|\, \theta, x_{1:n}$
2. Sample $\theta \,|\, z_{1:n}, x_{1:n}$

The first step is carried out by sampling $u_i \sim \text{Uniform}(0,1)$, independently for each i, and selecting

$$z_i = \begin{cases} 1 & \text{if } u_i \leq \dfrac{\theta f_1(x_i)}{\theta f_1(x_i) + (1-\theta) f_0(x_i)} \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

The second step is then carried out by sampling

$$\theta \sim \text{Beta}\left(\sum_{i=1}^{n} z_i + \alpha, n - \sum_{i=1}^{n} z_i + \beta\right). \tag{5}$$

As the iterations proceed, a histogram is tabulated for θ, which is then the approximate posterior distribution. Note that the posterior is approximated as a *mixture* of Beta distributions, where the potential number of mixing components is $n+1$.

2.2. The mean-field variational algorithm

A simple *variational algorithm* is based on a use of *Jensen's inequality* to bound the likelihood from below. We'll use the inequality in the following form:

Jensen's inequality

$$\sum_{j=1}^{k} \alpha_j x_j \;\geq\; \prod_{j=1}^{k} \left(\frac{\alpha_j x_j}{q_j}\right)^{q_j} \tag{6}$$

where $q_j \geq 0$ and $q_1 + \cdots + q_k = 1$.

To see this just take logarithms and use concavity of $\log$. We then have

$$\begin{aligned} p(x_{1:n}) = \int_0^1 \text{Beta}(\theta \,|\, \alpha, \beta) \prod_{i=1}^n \left(\theta f_1(x_i) + (1-\theta) f_0(x_i)\right) \, d\theta \qquad (7) \\ \geq \int_0^1 \text{Beta}(\theta \,|\, \alpha, \beta) \prod_{i=1}^n \left(\frac{\theta f_1(x_i)}{q_i}\right)^{q_i} \left(\frac{(1-\theta) f_0(x_i)}{1-q_i}\right)^{1-q_i} d\theta \qquad (8) \\ = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \prod_{i=1}^n \left(\frac{f_1(x_i)}{q_i}\right)^{q_i} \left(\frac{f_0(x_i)}{1-q_i}\right)^{1-q_i} \int_0^1 \theta^{\alpha+\sum_i q_i - 1} (1-\theta)^{\beta+n-\sum_i q_i - 1} d\theta \qquad (9) \\ = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \frac{\Gamma(\alpha+\sum_i q_i)\Gamma(\beta+n-\sum_i q_i)}{\Gamma(\alpha+\beta+n)} \prod_{i=1}^n \left(\frac{f_1(x_i)}{q_i}\right)^{q_i} \left(\frac{f_0(x_i)}{1-q_i}\right)^{1-q_i}. \qquad (10) \end{aligned}$$

This inequality holds for any probabilities q_i, which are now treated as variational parameters; q_i is an estimate of the posterior probability $p(Z_i = 1 \,|\, x_{1:n})$.

The strategy is to maximize this lower bound as a function of these parameters. A convenient way to do this is with the EM algorithm, where the latent variable is θ. Note that conditioned on $q_{1:n}$, equation (9) says that the distribution of θ is $\text{Beta}(\alpha + \sum_{i=1}^n q_i, \beta + n - \sum_{i=1}^n q_i)$. Using the fact that

$$\int \log \theta \, \text{Beta}(\theta \,|\, \gamma_1, \gamma_2) \, d\theta = \Psi(\gamma_1) - \Psi(\gamma_1 + \gamma_2) \qquad (11)$$

where $\Psi(x) = \frac{d}{dx} \log \Gamma(x)$ is the digamma function, some calculation shows that the EM algorithm takes the following form:

1. Holding q_i fixed, set $\gamma = (\gamma_1, \gamma_2)$ to

$$\begin{aligned} \gamma_1 &= \alpha + \sum_{i=1}^n q_i \qquad (12) \\ \gamma_2 &= \beta + n - \sum_{i=1}^n q_i \qquad (13) \end{aligned}$$

2. Holding γ_1 and γ_2 fixed, set q_i to

$$q_i = \frac{f_1(x_i) \exp \Psi(\gamma_1)}{f_1(x_i) \exp \Psi(\gamma_1) + f_0(x_i) \exp \Psi(\gamma_2)} \qquad (14)$$

The approximate posterior distribution over θ is then taken to be

$$\widehat{p}(\theta \,|\, x_{1:n}) = \text{Beta}(\theta \,|\, \gamma_1, \gamma_2) \qquad (15)$$

and the estimated posterior probability of Z_i is $\widehat{p}(Z_i = 1 \,|\, x_{1:n}) = q_i$. This algorithm converges quickly, with a well defined numerical convergence criterion. The convergence of the Gibbs sampler, on the other hand, may not be as easy to assess.

Note that the number of variational parameters is $O(n)$, and the variational approximation to the posterior over θ is a single Beta distribution. Thus, this approximation tends to underestimate the variance in the posterior, and is in addition biased. However, we have traded off the estimation accuracy for computational efficiency. Several other more powerful variational algorithms have been developed, including the *expectation-propagation* algorithm (Minka, 2003) which makes an effort to match the variance of the approximation to the actual variance.

3. Variational Methods for Random Fields

To illustrate the qualitative difference between an MCMC approximation and a variational approximation for graphical models, consider the example of a undirected graphical model for binary random variables $X_1, \dots, X_n$, with pairwise interactions over a graph $G = (V, E)$, taking the form

$$\mathbb{P}_\beta(x_1, \dots, x_n) \propto \exp\left(\sum_{s \in V} \beta_s x_s + \sum_{(s,t) \in E} \beta_{st} x_s \, x_t \right). \tag{16}$$

This is also called an "Ising model" and is central to statistical physics. The simplest Gibbs sampling algorithm for this model is the following:

```
Gibbs sampler

1. Choose vertex $s \in V$ at random
2. Sample $u \sim \text{Uniform}(0,1)$ and update
       $x_s = \begin{cases} 1 & u \le \left(1 + \exp\left(-\beta_s - \sum_{t \in N(s)} \beta_{st} x_t\right)\right)^{-1} \\ 0 & \text{otherwise} \end{cases}$    (17)
3. Iterate
```

In contrast, a simple mean-field variational algorithm for this model takes the following form.

```
Mean field variational algorithm

1. Choose vertex $s \in V$ at random
2. Update
       $\mu_s = \left(1 + \exp\left(-\beta_s - \sum_{t \in N(s)} \beta_{st} \mu_t\right)\right)^{-1}$    (18)
3. Iterate
```

The parallel structure of these two algorithms shows that the MCMC is stochastic, with a random variable X_s sampled in each iteration, while the variational algorithm is deterministic, with an expectation parameter μ_s updated in each iteration. Accordingly, the convergence of the Gibbs sampler must be assessed in terms of stochastic convergence, while the variational algorithm can be assessed by simpler numerical measures.

The derivation of this algorithm comes from considering the structured variational approximation strategy of (59). In particular, we use one of the simplest families of tractable distributions possible—the family that treats each random variable as independent. Graphically, this corresponds to completely disconnecting the underlying graph by throwing out all of the edges.

In this case, assuming a graphical model with pairwise interactions and binary variables, of the form (16), the variational approximation becomes

$$\Psi(\theta) \quad \geq \quad \sup_{\mu} \sum_{s \in V} \mu_s \theta_s + \sum_{(s,t) \in E} \mu_s \mu_t \theta_{st} - \sum_{s \in V} \big\{ \mu_s \log \mu_s + (1 - \mu_s) \log(1 - \mu_s) \big\}.$$

Maximizing the lower bound using a coordinate ascent scheme, optimizing each μ_s in turn, holding the others fixed, leads to the updates

$$\mu_s = \Big(1 + \exp \Big(-\theta_s - \textstyle\sum_{t \in N(s)} \theta_{st} \mu_t \Big) \Big)^{-1}. \tag{19}$$

4. General Approaches to Variational Inference

The above variational approximation can be viewed as an instance of the following general strategy.

Variational inference strategy

1. Assume a simplified form for the approximate posterior

$$p(z, \theta \,|\, x) = q(z, \theta \,|\, x) \tag{20}$$

   where q depends on various variational parameters.
2. Maximize a lower bound on the marginal likelihood by numerically optimizing the variational parameters.

In the example above, we assume a factored approximation to the posterior distribution $p(\theta, z_{1:n} \,|\, x_{1:n})$ as

$$q(\theta, z_{1:n}) = \text{Beta}(\theta \,|\, \gamma_1, \gamma_2) \prod_{i=1}^{n} q_i^{z_i} (1 - q_i)^{1 - z_i}. \tag{21}$$

The optimization problem is then to minimize the Kullback-Liebler divergence

$$D(q \,\|\, p) = \sum_{z_{1:n}} \int_0^1 q(\theta, z_{1:n}) \log \frac{q(\theta, z_{1:n})}{p(z_{1:n}, \theta \,|\, x_{1:n})} \, d\theta. \tag{22}$$

To see this, note that the log-likelihood can be written as

$$
\begin{aligned}
\log p(x_{1:n}) &= \int \sum_{z} q(z_{1:n}, \theta) \log p(x_{1:n}) \, d\theta && (23) \\
&= \sum_{z_{1:n}} \int q(z_{1:n}, \theta) \log \frac{p(x_{1:n}, z_{1:n}, \theta)\, q(z_{1:n}, \theta)}{p(z_{1:n}, \theta \,|\, x_{1:n})\, q(z_{1:n}, \theta)} \, d\theta && (24) \\
& && (25) \\
&= \sum_{z_{1:n}} \int q(z_{1:n}, \theta) \log \frac{p(x_{1:n}, z_{1:n}, \theta)}{q(z_{1:n}, \theta)} d\theta && (26) \\
&\quad + \sum_{z_{1:n}} \int q(z_{1:n}, \theta) \log \frac{q(z_{1:n}, \theta)}{p(z_{1:n}, \theta \,|\, x_{1:n})} \, d\theta && (27) \\
&\geq \sum_{z_{1:n}} \int q(z_{1:n}, \theta) \log \frac{p(x_{1:n}, z_{1:n}, \theta)}{q(z_{1:n}, \theta)} d\theta. && (28)
\end{aligned}
$$

Minimizing over the parameters γ_1, γ_2, and $q_{1:n}$ leads to the algorithm presented above.

Selecting the form of the approximation $q(\cdot)$ is analogous to choosing the proposal distribution in a Metropolis-Hastings algorithm. However, the parameters of this distribution are then optimized numerically, rather than used in a sampling procedure.

5. Variational EM

A close relation exists between mean-field variational inference and the expectation-maximization algorithm. Consider equations we derived for the EM algorithm, but now replace $p(y \,|\, x, \theta')$ by a variational distribution $q(y \,|\, x)$. Then we have that

$$
\begin{aligned}
\ell(\theta) &= \int q(y \,|\, x) \log p(x \,|\, \theta) \, dy && (29) \\
&= \int q(y \,|\, x) \log \frac{q(y \,|\, x)}{p(y \,|\, x, \theta)} + \int q(y \,|\, x) \log \frac{p(x, y;\, \theta)}{q(y \,|\, x)} \, dy && (30) \\
&\geq \int q(y \,|\, x) \log \frac{p(x, y;\, \theta)}{q(y \,|\, x)} \, dy && (31) \\
&\equiv Q(\theta; q). && (32)
\end{aligned}
$$

The lower bound is maximized by *minimizing* the KL divergence

$$
D\left(q(Y \,|\, x) \,\|\, p(Y \,|\, x, \theta) \right). \tag{33}
$$

In a *variational EM algorithm* we replace the conditional distribution $p(y \,|\, x, \theta')$ by a variational approximation $q(y \,|\, x)$, where the approximation is estimated using the current parameters θ'. Then, the E-step is carried out with the variational approximation, and the M-step is the calculated as usual. This analysis shows that variational EM has the same monotonicity property as standard EM, which is very useful in checking the correctness of an implementation.

When the complete data model is in the exponential family, this leads to the following variational EM algorithm.

Variational EM

Let the complete data distribution be an exponential family

$$p(z;\theta) = b(z)\exp\left(\theta^T\phi(z) - \Psi(\theta)\right). \tag{34}$$

Then the variational EM algorithm iterates between the following steps:

Variational E-step: Compute the variational distribution $q(z \mid x)$, maximizing

$$\theta'^T \mathbb{E}_q(\phi(Z) \mid x) + H(q) \tag{35}$$

where θ' is the current parameter estimate.
M-step: Match the (approximate) expected sufficient statistics, choosing θ so that

$$\mathbb{E}_q[\phi(Z) \mid x] = \mathbb{E}[\phi(Z) \mid \theta] \tag{36}$$

Update: Set $\theta' = \theta$ and iterate to convergence.

Note that the variational approximation $q(z \mid x)$ is conditioned on the observed data x; the expectation $\mathbb{E}_q(\phi(Z) \mid x)$ is only averaging over the latent data. The variational E-step is itself an iterative, numerical algorithm to optimize (35).

Example 5.1. [Bayesian mixture] Returning to our earlier example, suppose that $f_1 = \mathcal{N}(\mu_1, \sigma^2)$ and $f_0 = \mathcal{N}(\mu_0, \sigma^2)$ with fixed variance. Then the variational updates to the means in the M-step are given by

$$\mu_1 = \frac{\sum_{i=1}^n q_i\, x_i}{\sum_{i=1}^n q_i} \tag{37}$$

$$\mu_0 = \frac{\sum_{i=1}^n (1-q_i)\, x_i}{\sum_{i=1}^n (1-q_i)}. \tag{38}$$

By comparison, when using the Gibbs sampler we carry out a stochastic E-step where indicator variables $Z_i^{(b)}$, $b = 1, \ldots, B$ are sampled for each data point x_i. The means are then updated as

$$\mu_1 = \frac{\sum_{i=1}^n \sum_{b=1}^B Z_i^{(b)}\, x_i}{\sum_{i=1}^n \sum_{b=1}^B Z_i^{(b)}} \tag{39}$$

$$\mu_0 = \frac{\sum_{i=1}^n \sum_{b=1}^B \left(1 - Z_i^{(b)}\right) x_i}{\sum_{i=1}^n \sum_{b=1}^B \left(1 - Z_i^{(b)}\right)} \tag{40}$$

in the M-step. Thus, the sampling procedure samples random variables Z_i which indicate which mixture component example i was generated from, while the variational procedure estimates the probability q_i that the ith example comes from the first mixture component.

6. Sampling and Variational Inference for LDA

Recall the latent Dirichlet allocation (LDA) model of Blei et al. (2003) from our discussion of mixtures. This topic model has been widely used for text documents, images, and other data types. Let us think of the model applied to text documents, where document d is made up of n_d words $x_{di} \in \{1, \ldots, V\}$, for $i = 1, \ldots, n_d$. The model assigns a latent topic $Z_{di} \in \{1, 2, \ldots, K\}$ to each word x_{di} in document d. The generative process underlying LDA is the following, where Mult denotes the multinomial distribution.

1. Draw $\beta_k \sim$ Dirichlet(ζ) for $k = 1, \ldots, K$
2. For each document d,
   (a) Draw $\theta_d \sim$ Dirichlet(α).
   (b) For $i \in \{1, \ldots, n_d\}$:
      i. Draw topic assignment $Z_{di} \,|\, \theta_d \sim$ Mult(θ_d).
      ii. Draw word $X_{di} \,|\, Z_{di} = z_{di},\ \beta\ \sim$ Mult$(\beta_{z_{di}})$.

Note that this is essentially the same as the working example of Section 2, but with K rather than two mixture components, and where we assume the mixing distributions F_k are multinomial and random.

For simplicity, we assume symmetric Dirichlet priors, so that

$$\zeta = \underbrace{(\zeta, \ldots, \zeta)}_{V \text{ times}} \qquad \alpha = \underbrace{(\alpha, \ldots, \alpha)}_{K \text{ times}}. \tag{41}$$

As some additional notation, we let $n_{dk\bullet} = \sum_w n_{dkw}$ denote the number of times a word in document d is generated from topic k, and similarly let $n_{\bullet kw} = \sum_d n_{dkw}$ denote the number of times over all documents that word w is generated from topic k. In general, $\bullet$ indicates that an index is summed out.

Let us contrast some advanced simulation and variational methods for this model. First, note that the full joint distribution over the topic proportions θ, topic assignments Z, topic parameters β and observed words x is

$$p(x, z, \theta, \beta \,|\, \alpha, \zeta) \;=\; \prod_{d=1}^{D} \frac{\Gamma(K\alpha)}{\Gamma(\alpha)^K} \prod_{k=1}^{K} \theta_{dk}^{\alpha - 1 + n_{dk\bullet}} \prod_{k=1}^{K} \frac{\Gamma(V\zeta)}{\Gamma(\zeta)^V} \prod_{w=1}^{V} \beta_{kw}^{\zeta - 1 + n_{\bullet kw}}. \tag{42}$$

The key observation in deriving an efficient Gibbs sampler is that it is possible to "collapse" this distribution by analytically integrating out θ and β:

$$p(z, x \,|\, \alpha, \zeta) = \tag{43}$$

$$\prod_{d=1}^{D} \left[\frac{\Gamma(K\alpha)}{\Gamma(K\alpha + n_{d\bullet\bullet})} \prod_{k=1}^{K} \frac{\Gamma(\alpha + n_{dk\bullet})}{\Gamma(\alpha)} \right] \prod_{k=1}^{K} \left[\frac{\Gamma(V\zeta)}{\Gamma(V\zeta + n_{\bullet k\bullet})} \prod_{w=1}^{V} \frac{\Gamma(\zeta + n_{\bullet kw})}{\Gamma(\zeta)} \right].$$

This leads to the *collapsed Gibbs sampler* algorithm that iteratively samples a topic for each word according to the conditional distribution

$$p\left(z_{di} = k \,|\, z^{\setminus di}, x, \alpha, \zeta\right) \propto \frac{\Gamma\left(\alpha + n_{dk\bullet}^{\setminus di}\right)\,\Gamma\left(\zeta + n_{\bullet k x_{di}}^{\setminus di}\right)}{\Gamma\left(V\zeta + n_{\bullet k \bullet}^{\setminus di}\right)} \tag{44}$$

where the superscript $\setminus di$ indicates that the counts are over the variables with x_{di} and z_{di} removed. This Gibbs sampler is implemented without great difficulty, and has been observed to converge quickly even on large document collections.

Teh et al. (2007) show that similar calculations can be exploited in a collapsed variational algorithm. In particular, suppose that the variational approximation is factored as

$$\begin{aligned} q(z, \theta, \beta) &= q(z)\, q(\theta, \beta \,|\, z) & (45)\\ &= \prod_{d=1}^{D} \prod_{i=1}^{n_d} q_{di}(z_{di})\, q(\theta, \beta \,|\, z). & (46) \end{aligned}$$

If no restrictions are placed on $q(\theta, \beta \,|\, z)$, then the optimal q must be given by the true posterior given z. But this, we observed above, has an analytic form. Optimizing for the remaining variational parameters leads to the updates

$$\phi_{dik} \;=\; q(z_{di} = k) \;\propto\; \exp\left(\mathbb{E}_{q(z^{\setminus di})}\left[\log(\alpha + n_{dk\bullet}^{\setminus di}) + \log(\zeta + n_{\bullet k x_{di}}^{\setminus di}) - \log(V\zeta + n_{\bullet k \bullet}^{\setminus di})\right]\right). \tag{47}$$

Teh et al. (2007) propose a Gaussian approximation to this distribution that is an effective variational approximation in practice.

The remainder of these notes are on more advanced topics.

7. Structured variational approximations

Here is another view that leads to different insights. Recall that the conjugate of a function f is $f^*(y) = \sup_x \langle y, x\rangle - f(x)$ which defines a convex function. If the original function f is convex and lower semicontinuous then $f = f^{**}$, and thus $f(x) = \sup_y \langle x, y\rangle - f^*(y)$.

Now, for an exponential family model, the log-normalization constant

$$\Psi(\theta) = \log \int_{\mathcal{X}} \exp(\langle \theta, \phi(x)\rangle) d\sigma(x) \tag{48}$$

defines a convex function of the natural parameter θ. Define the *marginal polytope* $\mathcal{M}$ as

$$\mathcal{M} = \{\mu \,|\, \mu = \mathbb{E}_\theta[\phi(X)], \text{ for some } \theta\}. \tag{49}$$

Then it is not difficult to show that the conjugate is given by a negative entropy:

$$\Psi^*(\mu) = \begin{cases} -H(p(x \,|\, \theta(\mu))) & \mu \in \operatorname{int} \mathcal{M} \\ \infty & \mu \notin \operatorname{cl} \mathcal{M}. \end{cases} \tag{50}$$

To see this, let $q = p_{\theta(\mu)}$, where $\theta(\mu)$ is the natural parameter corresponding to mean parameter μ. Then

$$\begin{aligned} \Psi^*(\mu) &= \sup_\theta \langle \mu, \theta \rangle - \Psi(\theta) && (51)\\ &= \sup_\theta \langle \mathbb{E}_q(\phi), \theta \rangle - \Psi(\theta) && (52)\\ &= \sup_\theta -D(q \,\|\, p_\theta) - H(q) && (53)\\ &= \inf_\theta D(q \,\|\, p_\theta) - H(q) && (54)\\ &= -H(q) && (55) \end{aligned}$$

in case $\mu \in \operatorname{int} \mathcal{M}$. This gives the representation

$$\begin{aligned} \Psi(\theta) &= \sup_{\mu \in \mathcal{M}} \langle \mu, \theta \rangle - \Psi^*(\mu) && (56)\\ &= \sup_{\mu \in \mathcal{M}} \langle \mu, \theta \rangle + H(\theta(\mu)). && (57) \end{aligned}$$

However, this representation may not be practical for two reasons. First, the model $\theta(\mu)$ having mean parameters μ may not be explicitly obtained. But even if the model could be determined, the entropy $H(\theta(\mu))$ is in general difficult to calculate; it involves a similar sum or integral over the state space that the log-normalizing constant requires.

Structured mean-field variational algorithm

This forms a *lower* bound on the log-normalizer

$$\begin{aligned} \Psi(\theta) &= \sup_{\mu \in \mathcal{M}} \langle \mu, \theta \rangle - \Psi^*(\mu) && (58)\\ &\geq \sup_{\mu \in \mathcal{M}_{\text{tr}}} \langle \mu, \theta \rangle - \Psi^*(\mu) && (59) \end{aligned}$$

where $\mathcal{M}_{\text{tr}} \subset \mathcal{M}$ is the set of mean parameters for a "tractable" family of distributions.

8. Exact MAP Inference

When Gibbs distributions and random fields were first proposed for image modeling in the early 1980s, it wasn't recognized that computationally efficient and exact methods were available for certain inference calculations.

Suppose that $G = (V, E)$ is a graph, and $X_i \in \{0, 1\}$ is a labeling of each node. In an image setting, this corresponds to, say, a picture with black or white pixels. Consider model over pixels that is given by a Ising form Gibbs distribution:

$$\begin{aligned} p(x) &\propto \exp\left(\frac{1}{2}\sum_{i,j}\beta_{ij}(x_i x_j + (1-x_i)(1-x_j))\right) && (60)\\ &= \exp\left(\frac{1}{2}\sum_{ij}\beta_{ij}\delta(x_i,x_j)\right). && (61)\end{aligned}$$

If $\beta_{ij} \geq 0$ for all i, j then this model encourages neighboring pixels to have the same color. In a Bayesian view, this can be interpreted as a prior distribution over colorings.

For a given an image x, suppose that it is actually observed with the pixel colors corrupted by noise. In particular, suppose that we observe the value $y_i \in \mathbf{R}$ in position i with probability

$$p(y_i \,|\, x_i) \;=\; p(y_i \,|\, 1)^{x_i} p(y_i \,|\, 0)^{(1-x_i)}. \tag{62}$$

Then, given y, the log-posterior satisfies

$$\begin{aligned} \log p(x \,|\, y) &\overset{c}{=} \log p(x) + \log p(y \,|\, x) && (63)\\ &= \log p(x) + \sum_i x_i \log p(y_i \,|\, 1) + \sum_i (1-x_i)\log p(y_i \,|\, 0) && (64)\\ &\overset{c}{=} \log p(x) + \sum_i x_i \log\frac{p(y_i \,|\, 1)}{p(y_i \,|\, 0)} && (65)\end{aligned}$$

where $\overset{c}{=}$ means equal up to an additive constant. We can then write this as

$$\log p(x \,|\, y) \;\overset{c}{=}\; \frac{1}{2}\left(\sum_{i,j}\beta_{ij}(x_i x_j + (1-x_i)(1-x_j))\right) + \sum_i \lambda_i x_i \tag{66}$$

where $\lambda_i = \log\left(\frac{p(y_i \,|\, 1)}{p(y_i \,|\, 0)}\right)$.

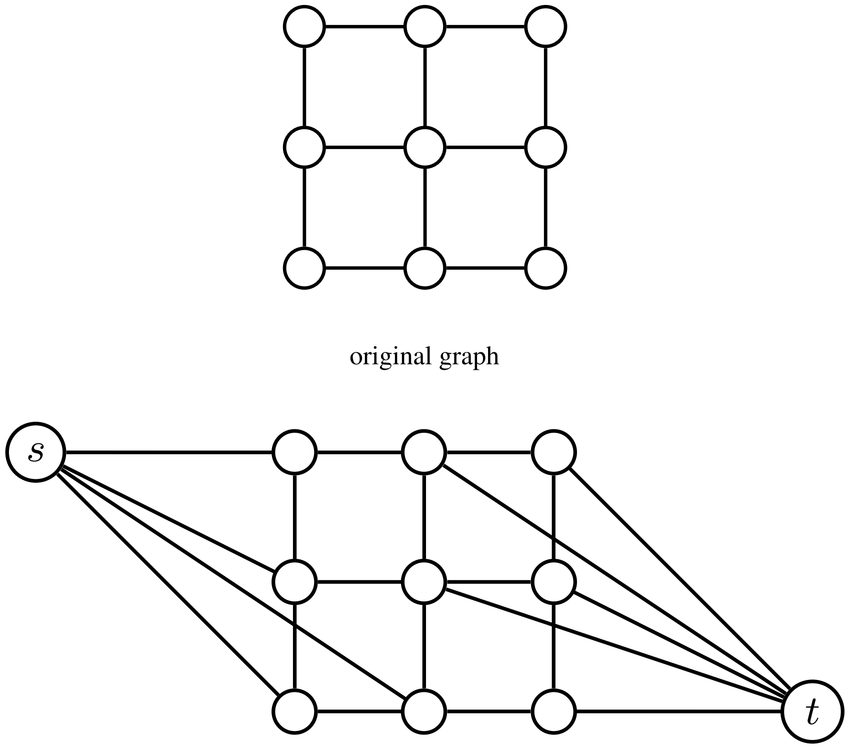
While maximizing this posterior involves, in principle, a search over 2^n possible states, it is possible to compute the best assignment more efficiently. Construct a network with two additional nodes, a source s and a sink t. Create an edge between s and node i with cost $c_{si} = \lambda_i$ if $\lambda_i > 0$; otherwise create an edge between t and node i with cost $c_{it} = -\lambda_i$. For each pair of nodes i, j, create an edge between i and j with cost $\beta_{ij} = c_{ij}$ if $\beta_{ij} > 0$.

Now, if $x \in \{0,1\}^n$ is a labeling, let

$$\begin{aligned} B &= \{i \,|\, x_i = 1\} \cup \{s\} && (67)\\ W &= \{i \,|\, x_i = 0\} \cup \{t\} && (68)\end{aligned}$$

and define the *cost* of the labeling as

$$c(x) = \sum_{i=B}\sum_{j\in W} c_{ij}. \tag{69}$$



original graph

graph with source and sink

FIG 1. *Network representing a pairwise Gibbs distribution with binary labels.*

The partition B, W forms a *cut* of the graph, and $c(x)$ is the *capacity* of the cut. It is given by

$$c(x) = \sum_{i=1}^{n} x_i \max(0, -\lambda_i) + \sum_{i=1}^{n} (1 - x_i) \max(0, \lambda_i) + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \beta_{ij} (x_i - x_j)^2. \tag{70}$$

Moreover, $\min_x c(x)$ is the maximum flow through the network from source to sink, subject to the given edge capacity constraints. This flow is found, for example, by the Ford-Fulkerson algorithm.

Unfortunately, this algorithm does not generalize to $k > 2$ colors, nor to models that have mixed-sign coefficients or larger than pairwise interaction potentials.

9. Message Passing Algorithms

In the chapter on information theory we discussed how the belief propagation procedure, applied to sparse graph representations of codes, revolutionized coding theory in the mid 1990s, by effectively solving the fifty year-old channel coding problem framed by Shannon. In this section we'll describe some of the details of belief propagation for approximate inference, and its relation to variational methods.

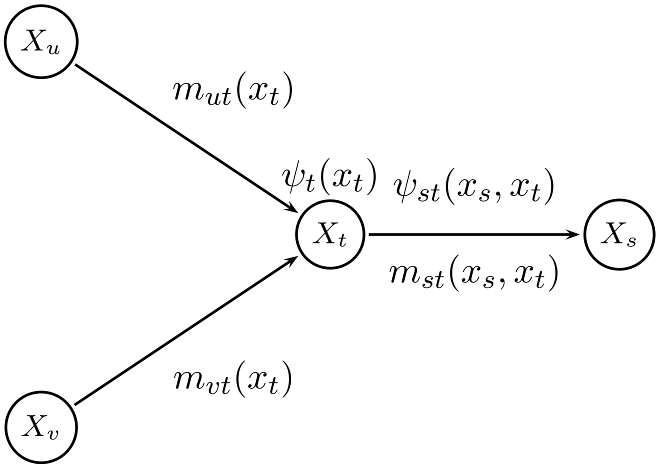
To begin, we'll restrict to directed or undirected trees. Assume that the graphical model has the form

$$p(x) \propto \prod_{s \in V} \psi_s(x_s) \prod_{(s,t) \in E} \psi_{st}(x_s, x_t) \tag{71}$$

$$= \exp\left(\sum_{s \in V} \theta_s(x_s) + \sum_{(s,t) \in E} \theta_{st}(x_s, x_t) \right). \tag{72}$$

For directed trees we take $\psi(x_r) = p(x_r)$ for a root node, and $\psi_{ij}(x_i, x_j) = p(x_j \mid x_i)$ for a node $x_i \to x_j$.

The idea behind belief propagation is to send "messages" between the nodes along edges in the graph. The message that node t sends to node s is a probability mass function, and can be thought of as node t's estimate of the distribution of node s, given all of the evidence that node t has collected from its other neighbors in the graph. The messages passed between nodes are determined by the sum-product algorithm.

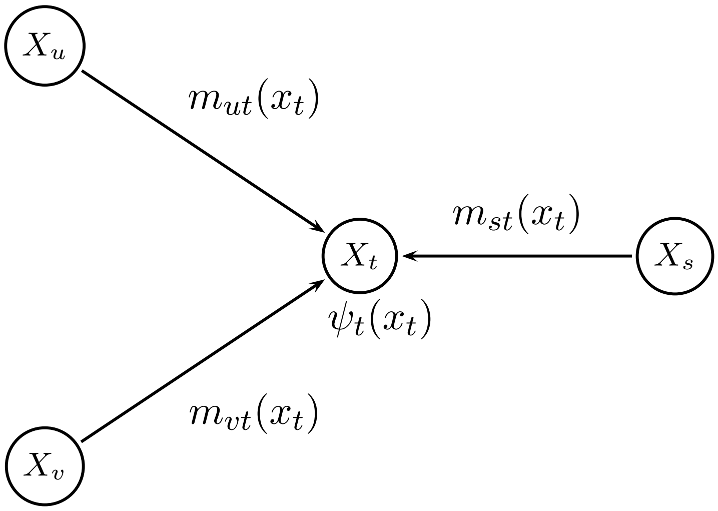


$$m_{ts}(x_s) \propto \textstyle\sum_{x_t} \psi_t(x_t)\, \psi_{st}(x_s, x_t)\, m_{ut}(x_t) m_{vt}(x_t)$$

FIG 2. *The belief propagation (sum-product) algorithm passes "messages" along edges in the graph. The messages depending on x_t incoming to node t are multiplied together with the vertex and edge potentials $\psi_t(x_t)$ and $\psi_{st}(x_s, x_t)$. The sum over all x_t of all such products is then the message sent from t to s.*

In order to compute a MAP or most probable configuration, we replace the sum by a $\max$, leading to the max-product algorithm.

More generally, we can carry out the sum-product algorithm for a general *commutative semiring*, that is, a ring where the operations $+$ and $\cdot$ are commutative and associative, and where the multiplication $\cdot$ is distributive over $+$. For example, $a \cdot \max(b, c) = \max(a \cdot b, a \cdot c)$. If we work in the log domain, with $\psi_{st}(x_s, x_t) \to -\log \psi_{st}(x_s, x_t)$ then the max-product algorithm becomes the min-sum algorithm.



$$\mu_t(x_t) \propto \psi_t(x_t)\, m_{st}(x_t)\, m_{ut}(x_t) m_{vt}(x_t)$$

FIG 3. *After the message passing algorithm has converged, the "beliefs" at each node t are obtained by multiplying the vertex potential $\psi_t(x_t)$ by the incoming messages depending on x_t, and then normalizing. This is an estimate of the marginal probability $\mathbb{P}(X_t = x_t)$, which is exact in the case the graph is a tree.*

Various optimality properties are known for the sum-product and max-product algorithm; however, the max-product algorithm has stronger guarantees. For example, it is known that on a graph with a single cycle, belief propagation either converges to a stable fixed point, or has periodic oscillations. If it converges, then it gives the correct MAP assignments. One of the most elegant results known is the following.

Theorem 9.2. *[Freeman and Weiss, 2003] If m^* is a fixed point of the max-product algorithm and x^* is the maximizing assignment, so that*

$$x^*_s = \arg\max_{x_s} \mu_s(x_s) \tag{83}$$

then $p(x^* \mid E) > p(x \mid E)$ for all $x \neq x^*$ that differ from x^* in a set of positions that forms a subgraph of G consisting of at most one cycle.

This is a strong optimality guarantee, which is stronger than a local maximum but weaker than a global maximum. As a corollary to this result, we have **Corollary 9.3.** *For Gaussian random*

fields, the max-product algorithm, if it converges, computes the correct posterior mean.

This result follows since for a Gaussian model, the mean equals the mode. But if the algorithm converges, the theorem implies that it is a local maximum. It must therefore be the global maximum, and determine the posterior mean.

Sum-Product Algorithm

The messages sent from node t to node s are

$$\begin{aligned} m_{ts}(x_s) &= c_{ts} \sum_{x_t} \psi_t(x_t) \psi_{st}(x_s, x_t) \prod_{v \in \mathcal{N}(t) \setminus s} m_{vt}(x_t) && (73) \\ &= c_{ts} \sum_{x_t} \exp\left(\theta_t(x_t) + \theta_{st}(x_s, x_t)\right) \prod_{v \in \mathcal{N}(t) \setminus s} m_{vt}(x_t) && (74) \end{aligned}$$

where c_{ts} is a scaling constant (to prevent numerical underflow), typically chosen so that the messages sum to one, $\sum_{x_s} m_{ts}(x_s) = 1$.
A node can send a message to a neighboring node as soon as it has received messages from all of its other neighbors. In particular, leaf nodes can send messages immediately. After all messages have been exchanged, the *belief* $\mu_s(x_s)$ at node s is given by

$$\begin{aligned} \mu_s(x_s) &\propto \psi_s(x_s) \prod_{t \in \mathcal{N}(s)} m_{ts}(x_s) && (75) \\ &= \exp(\theta_s(x_s)) \prod_{t \in \mathcal{N}(s)} m_{ts}(x_s). && (76) \end{aligned}$$

Note that the scaling constants c_{ts} do not affect the values of the beliefs. When the graph is a tree, $m_s(x_s)$ is the exact marginal probability $p(x_s \mid E)$, where E denotes the observed data. The beliefs on edges $\mu_{st}(x_s, x_t)$ are given by

$$\mu_{st}(x_s, x_t) \propto \psi_s(x_s)\, \psi_t(x_t)\, \psi_{st}(x_s, x_t) \prod_{u \in \mathcal{N}(s) \setminus t} m_{us}(x_s) \prod_{u \in \mathcal{N}(t) \setminus s} m_{ut}(x_t) \qquad (77)$$

Max-Product Algorithm

The messages sent from node t to node s are

$$m_{ts}(x_s) = c_{ts} \max_{x_t} \psi_t(x_t)\psi_{st}(x_s, x_t) \prod_{v \in \mathcal{N}(t) \backslash s} m_{vt}(x_t) \tag{78}$$

$$= c_{ts} \max_{x_t} \exp\left(\theta_t(x_t) + \theta_{st}(x_s, x_t)\right) \prod_{v \in \mathcal{N}(t) \backslash s} m_{vt}(x_t). \tag{79}$$

After all messages have been exchanged, the *max marginal* $\mu_s(x_s)$ at node s is given by

$$\mu_s(x_s) \propto \max_{x_s} \psi_s(x_s) \prod_{t \in \mathcal{N}(s)} m_{ts}(x_s) \tag{80}$$

$$= \max_{s} \exp(\theta_s(x_s)) \prod_{t \in \mathcal{N}(s)} m_{ts}(x_s). \tag{81}$$

The max marginal on edge (s, t) is given by

$$\mu_{st}(x_s, x_t) \propto \psi_s(x_s)\,\psi_t(x_t)\,\psi_{st}(x_s, x_t) \prod_{u \in \mathcal{N}(s) \backslash t} m_{us}(x_s) \prod_{u \in \mathcal{N}(t) \backslash s} m_{ut}(x_t). \tag{82}$$

When the graph is a tree, μ_s is the most probable assignment in the model, conditioned on the observed data.

9.1. Iterative Conditional Modes

It is of some historical interest to note the following heuristic procedure to approximate the posterior mode of an undirected model. First, set each node to the most probable value given only the single node potentials:

$$x_t = \arg\max_x \psi_t(x). \tag{84}$$

Next, cycle through the nodes and for node s set x_s to the most probable value conditioned on the neighboring nodes:

$$x_s \,|\, x_{t \in \mathcal{N}(s)} = \operatorname*{argmax}_x \psi_s(x) \prod_{t \in \mathcal{N}(s)} \psi_{st}(x_s, x_t). \tag{85}$$

This process is continued, cycling through all the nodes in the graph until convergence (which is not guaranteed). This is the *iterated conditional modes* algorithm of Besag (1986). The algorithm can be viewed as a greedy precursor to the max-product algorithm. An example of this procedure used to denoise an image is given in Figure 4.

9.2. What does belief propagation optimize?

Recall that the mean-field variational algorithm optimizes

$$\sum_{s,t} \mu_s \mu_t \theta_{s,t} + \sum_s \mu_s \theta_s - \sum_s \left\{(\mu_s \log \mu_s + (1 - \mu_s) \log(1 - \mu_s)\right\} \tag{86}$$

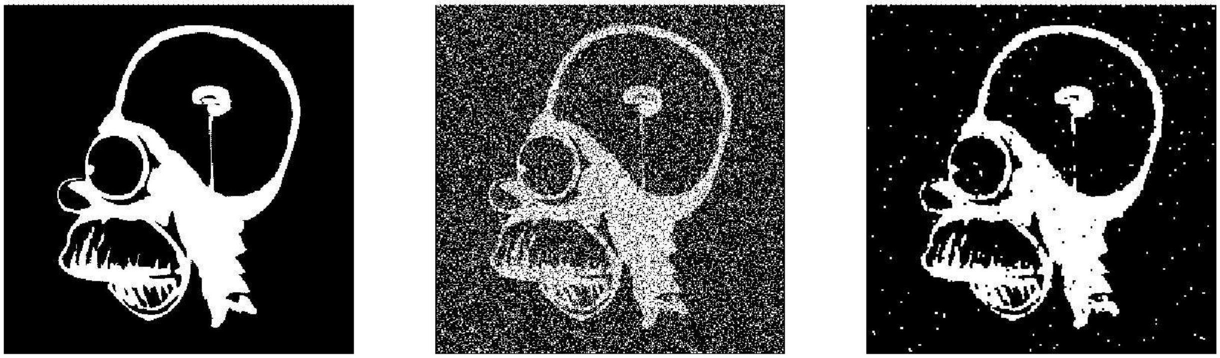


FIG 4. *Denoising a Homer Simpson medical image using iterative conditional modes (ICM). Left: Original black/white image. Center: Image with 20% random binary noise (a bit is flipped with probability* 0.2*). Right: Image decoded using ICM, with* $\beta = 10$.

in the binary case, or more generally

$$\begin{aligned}\sum_{s,t}\sum_{x_s,x_t}\mu_s(x_s)\mu_t(x_t)\log\psi_{s,t}(x_s,x_t)+\sum_s\sum_{x_s}\mu_s(x_s)\log\psi_s(x_s)\\ -\sum_s\sum_{x_s}\mu_s(x_s)\log\mu_s(x_s).\end{aligned}\tag{87}$$

Now, for any tree-structured graph, we can write

$$p(x) = \prod_{s\in V}p(x_s)\prod_{(s,t)\in E}\frac{p(x_s,x_t)}{p(x_s)p(x_t)} \tag{88}$$

$$= \frac{\prod_{(s,t)\in E}p(x_s,x_t)}{\prod_s p(x_s)^{n_s-1}} \tag{89}$$

where n_s is the number of nodes neighboring s in the graph. More generally, the junction tree represents an arbitrary graphical model in this factored form; however, the size of clique nodes in the junction tree grows exponentially.

The *Bethe approximation* to the entropy is given by

$$H_{\text{Bethe}}(\mu) = \tag{90}$$

$$-\sum_s\sum_{x_s}\mu_s(x_s)\log\mu_s(x_s)-\sum_{s,t}\sum_{x_s,x_t}\mu_{s,t}(x_s,x_t)\log\frac{\mu_{s,t}(x_s,x_t)}{\mu_s(x_s)\mu_t(x_t)} \tag{91}$$

$$= \sum_s H(\mu_s)-\sum_{s,t}I(\mu_{s,t}) \tag{92}$$

where $I(\mu_{s,t})$ is the mutual information. When the graph is a tree, the Bethe approximation to the entropy is exact; however, when there are cycles in the graph, it is an approximation. A basic fact about belief propagation is the following.

A set of marginal estimates μ is a fixed point of the belief propagation message passing algorithm if and only if they form a stationary point of the *Bethe free energy*

$$\mathcal{E}_{\text{Bethe}}(\mu) = -\sum_{s,t}\sum_{x_s,x_t} \mu_{s,t}(x_s,x_t)\log\psi_{s,t}(x_s,x_t) - \sum_{s}\sum_{x_s}\mu_s \log\psi_s(x_s) - H_{\text{Bethe}}(\mu). \tag{93}$$

With the view that belief propagation is attempting to minimize the Bethe free energy, one might consider alternative numerical optimization schemes for minimizing this energy directly.

Extensions to belief propagation based on clustering nodes together can also be viewed in terms of a kind of energy minimization, called the *Kikuchi free energy* in statistical physics.

10. LP Relaxations

Consider a graph $G = (V, E)$, where V denotes the set of nodes and E denotes the set of edges. Let X_s be a random variable associated with node s, for $s \in V$, yielding a random vector $X = \{X_1, \ldots, X_n\}$, and let $\phi = \{\phi_\alpha, \alpha \in I\}$ denote the set of potential functions (or sufficient statistics) for a set I of cliques in G. Associated with ϕ is a vector of parameters $\theta = \{\theta_\alpha, \alpha \in I\}$. With this notation, the exponential family of distributions of X, associated with ϕ and G is given by

$$p(x;\theta) \;=\; \exp\left(\sum_\alpha \theta_\alpha\phi_\alpha - \Psi(\theta)\right).$$

As discussed in Yedidia et al. (2001), at the expense of increasing the state space one can assume without loss of generality that the graphical model is a pairwise Markov random field, *i.e.*, the set of cliques I is the set of edges $\{(s,t) \in E\}$, so that

$$p(x;\theta) \propto \exp\left(\sum_{s\in V}\theta_s\phi_s(x_s) + \sum_{(s,t)\in E}\theta_{st}\phi_{st}(x_s,x_t)\right).$$

If each X_s takes values in a discrete set $\mathcal{X}_s$, we can represent any potential function as a linear combination of indicator functions, $\phi_s(x_s) = \sum_j \phi_s(j)\,\mathcal{I}_j(x_s)$ and $\phi_{st}(x_s,x_t) = \sum_{jk}\phi_{st}(j,k)\,\mathcal{I}_{j,k}(x_s,x_t)$ where

$$\mathcal{I}_j(x_s) \;=\; \begin{cases} 1 & x_s = j \\ 0 & \text{otherwise} \end{cases} \tag{94}$$

and

$$\mathcal{I}_{j,k}(x_s,x_t) \;=\; \begin{cases} 1 & x_s = j \text{ and } x_t = k \\ 0 & \text{otherwise}\,. \end{cases} \tag{95}$$

$$\tag{96}$$

We thus consider pairwise MRFs with indicator potential functions as

$$p(x|\theta) \propto \exp\left(\sum_{s,j}\theta_{s;j}\mathcal{I}_j(x_s) + \sum_{s,t;j,k}\theta_{s,j;t,k}\mathcal{I}_{j,k}(x_s,x_t)\right).$$

The MAP problem is then given by

$$x^* = \underset{x}{\operatorname{argmax}}\sum_{s,j}\theta_{s;j}\mathcal{I}_j(x_s) + \sum_{s,t;j,k}\theta_{s,j;t,k}\mathcal{I}_{j,k}(x_s,x_t). \tag{97}$$

10.1. Linear Relaxations

MAP estimation in the discrete case is essentially a combinatorial optimization problem, and it can be cast as an integer program. Recent work has studied approximate MAP estimation using linear program relaxations Bertsimas and Tsitsiklis (1997). Letting variables $\mu(s;j)$ and $\mu(s,j;t,k)$ correspond to the indicator variables $\mathcal{I}_j(x_s)$ and $\mathcal{I}_{j,k}(x_s,x_t)$, we obtain the following integer linear program (ILP),

$$\begin{aligned} \text{max} \quad & \sum_{s;j}\theta_{s;j}\,\mu_1(s;j) + \sum_{s,t;j,k}\theta_{s,j;t,k}\,\mu_2(s,j;t,k) \\ \text{such that} \quad & \sum_k \mu_2(s,j;t,k) = \mu_1(s;j) \\ & \sum_j \mu_1(s;j) = 1 \\ & \mu_1(s;j) \in \{0,1\} \\ & \mu_2(s,j;t,k) \in \{0,1\}. \end{aligned} \tag{98}$$

This ILP can then be relaxed to the following linear program (LP),

$$\begin{aligned} \text{max} \quad & \sum_{s;j}\theta_{s;j}\,\mu_1(s;j) + \sum_{s,t;j,k}\theta_{s,j;t,k}\,\mu_2(s,j;t,k) \\ \text{such that} \quad & \sum_k \mu_2(s,j;t,k) = \mu_1(s;j) \\ & \sum_j \mu_1(s;j) = 1 \\ & 0 \le \mu_1(s;j) \le 1 \\ & 0 \le \mu_2(s,j;t,k) \le 1. \end{aligned} \tag{99}$$

Chekuri et al. (2005) propose the above LP relaxation as an approximation algorithm for the metric labeling task, which is the MAP problem with spatially homogeneous MRF parameters; thus, $\theta_{s,j;t,k} = w_{st}\, d(j,k)$, where w_{st} is a non-negative edge weight and d is a metric that is the same for all the edges. Kleinberg and Tardos (1999) propose related linear relaxations for specific metrics.

The above LP relaxation was also proposed for the general pairwise graphical model setting by Wainwright and Jordan (2003). Letting θ and $\phi(x)$ denote the vectors of parameters and potential functions, respectively, and letting $\langle\theta, \phi(x)\rangle$ denote the inner product

$$\langle\theta, \phi(x)\rangle = \sum_{s;j} \theta_{s;j}\mathcal{I}_j(x_s) + \sum_{(s,t)\in E;\, j,k} \theta_{s,j;t,k}\mathcal{I}_{j,k}(x_s, x_t)$$

the MAP problem is then given by

$$x^* = \operatorname*{argmax}_x \langle\theta, \phi(x)\rangle = \sup_{\mu\in\mathcal{M}} \langle\theta, \mu\rangle$$

where $\mathcal{M}$ is the set of moment parameters

$$\mathcal{M} = \left\{\mu \,:\, \sum_x p(x)\phi(x) = \mu \text{ for some distribution } p\right\}. \tag{100}$$

The polytope $\mathcal{M}$ can be seen to be upper-bounded by the set LOCAL(G) of all single and pairwise vectors μ_1 and μ_2 that satisfy the local consistency constraints

$$\textstyle\sum_k \mu_2(s, j; t, k) \;=\; \mu_1(s; j) \tag{101}$$

$$\textstyle\sum_j \mu_1(s, j) \;=\; 1 \tag{102}$$

$$0 \le \mu_1(s; j) \le 1 \tag{103}$$

$$0 \le \mu_2(s, j; t, k) \le 1. \tag{104}$$

Wainwright and Jordan (2003) thus propose the upper-bounding relaxation of using $LOCAL(G)$ as an outer bound for the polytope $\mathcal{M}$,

$$\mu^* = \sup_{\mu\in LOCAL(G)} \langle\theta, \mu\rangle\,, \tag{105}$$

which is the same LP formulation as in equation (99).

10.2. Quadratic Relaxations

In the linear relaxation of equation (99), the variables $\mu_2(s, j; t, k)$ are relaxations of the indicator variables $\mathcal{I}_{j,k}(x_s, x_t)$, with a value of one indicating that for edge $(s, t) \in E$, variable x_s is labeled j and variable x_t is labeled k. These pairwise variables are constrained by demanding that they be consistent with the corresponding "marginal" variables $\mu_1(s, j)$. Note, however, that the binary indicator variables satisfy the additional "independence" constraint

$$\mathcal{I}_{j,k}(x_s, x_t) = \mathcal{I}_j(x_s)\,\mathcal{I}_k(x_t). \tag{106}$$

This then suggests that constraining the relaxation variables in a similar manner, $\mu_2(s, j; t, k) = \mu_1(s; j)\,\mu(t; k)$, might yield a tighter relaxation. This leads to the following quadratic program (Ravikumar and Lafferty, 2006).

$$\begin{aligned} \max \quad & \sum_{s;j} \theta_{s;j}\mu(s; j) + \sum_{s,t;j,k} \theta_{s,j;t,k}\,\mu(s; j)\,\mu(t; k) \\ \text{subject to} \quad & \sum_j \mu(s; j) = 1 \\ & 0 \le \mu(s; j) \le 1 \end{aligned} \tag{107}$$

11. Tree-Reweighted Belief Propagation

Given that the tractable distributions are trees (or tree-like), it makes sense to try to approximate complicated graphical models by trees or mixtures of trees.

Consider the problem of maximizing the energy of the model, which is the MAP estimation problem. Assume the model is of the form

$$\begin{aligned} p_\theta(x) &\propto \exp\left(\langle \theta, \phi(x)\rangle\right) && (108)\\ &= \exp\left(\sum_{s\in V}\sum_j \theta_{s,j} + \sum_{(s,t)\in E}\sum_{j,k}\theta_{st;jk}\right). && (109)\end{aligned}$$

The MAP problem is to carry out the maximization

$$\mathcal{E}_{\text{max}}(\theta) = \max_x \langle \theta, \phi(x)\rangle\,. \tag{110}$$

A key property of $\mathcal{E}_{\text{max}}(\theta)$ that we'll exploit is its convexity in θ, as a maximum of linear functions. Let $\mathcal{T}$ be a collection of spanning trees for the graph $G = (V, E)$, with a weight $\rho(T)$ for each $T \in \mathcal{T}$. Suppose we have a set of parameters $\theta(T)$ restricted to each tree, so that their convex combination is our model parameter vector θ:

$$\sum_{T\in\mathcal{T}} \rho(T)\theta(T) = \theta \tag{111}$$

By convexity,

$$\mathcal{E}_{\text{max}}(\theta) \leq \sum_{T\in\mathcal{T}} \rho(T)\,\mathcal{E}_{\text{max}}(\theta(T)). \tag{112}$$

Since each $\theta(T)$ is a parameter-vector for a tree-structured graph, we can use standard algorithms to calculate $\mathcal{E}_{\text{max}}(\theta(T))$. For instance, the Viterbi algorithm, a form of dynamic programming, can be used, which is equivalent to running the max-product algorithm over the tree.

If we treat $\theta(T)$ as variational parameters, for fixed ρ, then we are led to the following convex optimization problem:

$$\begin{aligned} \text{minimize}_{\{\theta(T)\}} \quad & \sum_{T\in\mathcal{T}} \rho(T)\,\max_x\langle \phi(x), \theta(T)\rangle && (113)\\ \text{subject to} \quad & \sum_{T\in\mathcal{T}} \rho(T)\theta(T) = \theta. && (114)\end{aligned}$$

The inputs to the problem are the graph $G = (V, E)$, the parameters θ of the graphical model, and the distribution $\rho(T)$ over the set of spanning trees in the graph. Strong duality holds for this problem, and we have the following simple characterization of the dual.

Theorem 11.4. *Suppose that the tree weights $\rho(T)$ are such that for each edge $e \in E$ in the graph, the marginal probability ρ_e is strictly positive. Let LOCAL(G) be the set of all mean parameters that satisfy the local consistency constraints,*

$$\textit{LOCAL}(G) = \left\{\mu \;\middle|\; \mu_{s,j} = \textstyle\sum_k \mu_{st;jk} \text{ and } \sum_j \mu_{s,j} = \sum_{j,k}\mu_{st;jk} = 1\right\}. \tag{115}$$

Then the dual to the optimization program

$$\min_{\{\theta(T)\}} \qquad \sum_{T\in\mathcal{T}} \rho(T)\, \max_x \langle \phi(x), \theta(T)\rangle \tag{116}$$

$$\text{subject to} \qquad \sum_{T\in\mathcal{T}} \rho(T)\theta(T) = \theta \tag{117}$$

is the linear program

$$\max_\mu \qquad \sum_{s\in V}\sum_j \mu_{s,j}\theta_{s,j} + \sum_{(s,t)\in E}\sum_{j,k} \mu_{st;jk}\theta_{st;jk} \tag{118}$$

$$\text{such that} \qquad \mu \in \textit{LOCAL}(G). \tag{119}$$

The constraint $\mu_{s,j} = \sum_k \mu_{st;jk}$ is simply a consistency constraint on the mean parameters. The result says that as long as every edge has some probability in the weighted spanning trees, the tightest tree-weighted upper bound on the energy is the value of the natural linear program relaxation of the MAP problem. This can be viewed as an "outer" convex approximation to the marginal polytope, since $\text{LOCAL}(G)$ is convex and $\mathcal{M}(G) \subset \text{LOCAL}(G)$. Note that the dual problem does not depend on the tree weights $\rho(T)$.

The dual linear program is simpler than the primal problem. In particular, while the primal problem involves maximizing a function over all spanning trees of the graph, which can be a very large set, the dual involves only local constraints.

While this reduces the approximation to a standard optimization problem, linear programming, the programs can get large. In particular, if there are m edges in the graph and $|\mathcal{X}|$ possible values for each random variable, then the number of variables in the linear program is $O(n|\mathcal{X} + m|\mathcal{X}|^2)$. In many computer vision applications, $|\mathcal{X}|$ may be rather large, for example $|\mathcal{X}| \approx 256$ for a natural quantization of greyscale images.

As an alternative to solving this linear program using standard techniques such as the simplex method or interior point methods, it is attractive to have a method that takes advantage of the graphical structure of the problem. This is the motivation behind the tree-reweighted max-product algorithm.

When it converges, this algorithm computes the solution to the dual linear program, and therefore computes the tightest upper bound to the energy in (**??**). The algorithm applies with any distribution $\{\rho(T)\}$ over the set of spanning trees. For example, if the graph is a two-dimensional grid graph, one could assign non-zero probability over only two spanning trees, with $\rho(T_{\text{horiz}}) = \frac{1}{2}$ for a spanning tree T_{horiz} over the horizontal rows of the grid, and $\rho(T_{\text{vert}}) = \frac{1}{2}$ for a spanning tree T_{vert} over the vertical columns of the grid. Here $\rho_{st} = \frac{1}{2}$ for most edges, but $\rho_{st} = 1$ for those few edges in common with both the horizontal and vertical spanning trees.

The tree-reweighted belief propagation algorithm also yields an upper bound on the log-normalizing constant, and therefore a lower bound on the likelihood of data with respect to the model. In particular, when the algorithm converges to μ, the following upper bound holds:

Tree-reweighted max-product algorithm

Given a distribution $\rho(T)$ over spanning trees in the graph $G = (V, E)$, let $\rho_{st} = \rho_{ts}$ denote the marginal distribution over edge $(s,t) \in E$. After initializing the messages m_{ts} to random starting values, the message that node t sends to node s is updated as

$$\begin{aligned} m_{ts}(X_s) &\leftarrow& c_{ts} \max_{x_t} \exp\left(\frac{\theta_{st}(x_s, x_t)}{\rho_{ts}} + \theta_t(x_t)\right) \frac{\prod_{v\in\mathcal{N}(t)\backslash s} m_{vt}(x_t)^{\rho_{vt}}}{m_{st}(x_t)^{1-\rho_{st}}} \\ &=& c_{ts} \max_{x_t} \psi_t(x_t)\, \psi_{st}(x_s, x_t)^{1/\rho_{ts}} \frac{\prod_{v\in\mathcal{N}(t)\backslash s} m_{vt}(x_t)^{\rho_{vt}}}{m_{st}(x_t)^{1-\rho_{st}}} \end{aligned} \tag{120}$$

where c_{ts} is a scaling constant. When all of the edge weights are one, this reduces to the standard max-product algorithm. After convergence, the beliefs assigned to vertices and edges, which are estimates of the marginal probabilities, are given by

$$\begin{aligned} \mu_s(x_s) &\propto& \psi_s(x_s) \prod_{t\in\mathcal{N}(s)} m_{ts}(x_s)^{\rho_{ts}} \qquad (121) \\ \mu_{st}(x_s, x_t) &\propto& \psi_s(x_s)\psi_t(x_t)\psi_{st}(x_s, x_t)^{1/\rho_{ts}} \frac{\prod_{u\in\mathcal{N}(s)\backslash t} m_{us}(x_s)^{\rho_{us}}}{m_{ts}(x_s)^{1-\rho_{ts}}} \frac{\prod_{u\in\mathcal{N}(t)\backslash s} m_{ut}(x_t)^{\rho_{ut}}}{m_{st}(x_t)^{1-\rho_{st}}}. \end{aligned}$$

The algorithm reduces to standard max-product belief propagation when $\rho_{st} \equiv 1$. Replacing the $\max$ with a sum gives the *tree-reweighted sum-product* algorithm.

For a set of tree weights ρ, define the *Bethe-Wainwright-Jaakkola-Willsky entropy* or *Bethe tree entropy* H_{BWJW} by

$$\begin{aligned} H_{\text{BWJW}}(\mu, \rho) &=& -\sum_s \sum_{x_s} \mu_s(x_s) \log \mu_s(x_s) - \sum_{s,t} \rho_{st} \sum_{x_s, x_t} \mu_{st}(x_s, x_t) \log \frac{\mu_{st}(x_s, x_t)}{\mu_s(x_s)\mu_t(x_t)} \\ &=& \sum_s H(\mu_s) - \sum_{st} \rho_{st} I(\mu_{st}) \end{aligned} \tag{122}$$

where H is the entropy, I is the mutual information. Then we define

$$\begin{aligned} \mathcal{E}_{BWJW}(\mu, \rho, \theta) &=& -\theta^T \mu - H_{\text{BWWJ}}(\mu, \rho) \qquad (123) \\ &=& -\sum_s \sum_{x_s} \mu_s(x_s)\theta_{s,x_s} - \sum_{s,t} \sum_{x_s, x_t} \mu_{st}(x_s, x_t)\theta_{s,x_s;t,x_t} - H_{\text{BWJW}}(\mu, \rho) \end{aligned}$$

and call it the *Bethe-Wainwright-Jaakkola-Willsky free energy*, or the *Bethe tree energy*. Note that this reduces to the previously defined Bethe free energy when $\rho_{st} = 1$.

The Bethe tree energy is convex in μ, and the log normalizing constant satisfies

$$\Psi(\theta) \leq -\min_{\mu\in\text{LOCAL}(G)} \mathcal{E}_{\text{BWJW}}(\mu, \rho, \theta) \tag{124}$$

for any ρ and θ.

Note that the BWJW free energy agrees with the Bethe free energy in the case where $\rho_{st} \equiv 1$; however, this is a valid choice of ρ only if the graph is a tree. The tree-reweighted sum-product algorithm carries out the minimization on the righthand side, and therefore computes the tightest upper bound on $\Psi(\theta)$ in this variational family of bounds. If the algorithm converges to the marginals μ^*, then we have the probability lower bounds

$$p_\theta(x) = \exp\left(\theta^T\phi(x) - \Psi(\theta)\right) \quad \geq \quad \exp\left(\theta^T\phi(x) + \mathcal{E}_{\text{BWJW}}(\mu^*, \rho, \theta)\right). \tag{125}$$

This motivates the use of approximate maximum likelihood inference where $-\mathcal{E}_{\text{BWJW}}(\mu^*, \rho, \theta)$ is substituted for the true log normalizing constant in all computations. The tree weights ρ can also be optimized using a gradient descent procedure that involves iteratively solving a maximum weight spanning tree problem; see Wainwright et al. (2005b) for details.

Putting these various results together leads to a strategy for approximate inference in conditional graphical models, where the conditional distribution $p_\theta(y \mid x) \propto \exp(\theta^T\phi(x,y))$ is a random field:

1. Form an estimate $\widehat{\theta}$ of the parameters by carrying out approximate maximum likelihood, running gradient ascent on the lower bound

$$\sum_{i=1}^n \log p_\theta(y_i \mid x_i) \geq \sum_{i=1}^n \theta^T\phi(x_i, y_i) + \mathcal{E}_{\text{BWJW}}(\mu^*(x_i), \rho, \theta) \tag{126}$$

   where the righthand side is computed using the tree-reweighted sum-product algorithm.
2. Approximate the most likely labeling $y^*(x) = \operatorname{argmax}_y p_{\widehat{\theta}}(y \mid x)$ using tree-weighted max-product.

This strategy has been effectively used in several applications.

References

Bertsimas, D. and Tsitsiklis, J. (1997). *Introduction to Linear Optimization*. Athena Scientific.

Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society, Series B*.

Bishop, C., Spiegelhalter, D., and Winn, J. (2003). VIBES: A variational inference engine for Bayesian networks. In Becker, S., Thrun, S., and Obermayer, K., editors, *Advances in Neural Information Processing Systems 15*, pages 777–784. MIT Press, Cambridge, MA.

Blei, D., Ng, A., and Jordan, M. (2003). Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11).

Chekuri, C., Khanna, S., Naor, J., and Zosin, L. (2001). Approximation algorithms for the metric labeling problem via a new linear programming formulation. In *Symposium on Discrete Algorithms*.

Chekuri, C., Khanna, S., Naor, J., and Zosin, L. (2005). A linear programming formulation and approximation algorithms for the metric labeling problem. *SIAM Journal on Discrete Mathematics*, 18(3):608–625.

Kleinberg, J. and Tardos, E. (1999). Approximation algorithms for classification problems with pairwise relationships: Metric partitioning and Markov random fields. *IEEE Symposium on the Foundations of Computer Science*.

Kolmogorov, V. (2005). Convergent tree-reweighted message passing for energy minimization. *AISTATS*.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann Publishers Inc.

Pritchard, J., Stephens, M., and Donnelly, P. (2000). Inference of population structure using multilocus genotype data. *Genetics*, 155:945–959.

Ravikumar, P. and Lafferty, J. (2006). Quadratic programming relaxations for metric labeling and Markov random field MAP estimation. In *Uncertainty in Artificial Intelligence (UAI)*.

Teh, Y. W., Newman, D., and Welling, M. (2007). A collapsed variational Bayesian inference algorithm for latent Dirichlet allocation. In *Advances in Neural Information Processing Systems*, volume 19.

Wainwright, M., Jaakkola, T., and Willsky, A. (2005a). Map estimation via agreement on (hyper)trees: Message-passing and linear-programming approaches. *IEEE Transactions on Information Theory*, 51(11):3697–3717.

Wainwright, M., Jaakkola, T. S., and Willsky, A. S. (2005b). A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory*, 51(7):2313–2335.

Wainwright, M. J. and Jordan, M. I. (2003). Variational inference in graphical models: The view from the marginal polytope. *Allerton Conference on Communication, Control, and Computing*.

Weiss, Y. and Freeman, W. T. (2001). On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *IEEE Transactions on Information Theory*, 47.

Xing, E., Jordan, M., and Russell, S. (2003). A generalized mean field algorithm for variational inference in exponential families. In *Proceedings of UAI*.

Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2001). Understanding belief propagation and its generalizations. *IJCAI 2001 Distinguished Lecture track*.