KF Book Chapter Summaries

Marco Cusumano-Towner

May 2, 2012

KF CH 10: Exact Inference with Clique Trees

For some query, any given variable elimination ordering generates a 'clique tree', along with messages that are passed to the root of the tree (which includes the query variables). We realize that we can compute the messages needed for *all* possible queries by sending messages in both directions, when they are ready. This gives the 'sum-product belief' propagation algorithm. Let $\psi_i(C_i)$ be the initial potential (product of original potentials ϕ) for a clique C_j. The message from a clique C_i to a clique C_j, where the 'sepset' is $S_{i,j} = C_i \cap C_j$, is

$$\delta_{i\to j}(S_{i,j}) := \sum_{C_i \setminus S_{i,j}} \psi_i(C_i) \prod_{k \in \text{Nbr}_i \setminus \{j\}} \delta_{k\to i}(S_{i,k})$$

After all messages are sent, the 'beliefs' for each clique are defined as

$$\beta_i(C_i) = \psi_i(C_i) \prod_{j \in \text{Nbr}_i} \delta_{j\to i}(S_{i,j})$$

The beliefs for neighboring cliques agree on their shared variables:

$$\mu_{i,j}(S_{i,j}) := \sum_{C_i \setminus S_{i,j}} \beta_i(C_i) = \sum_{C_j \setminus S_{i,j}} \beta_j(C_j)$$

The $\mu_{i,j}(S_{i,j})$ are called the 'sep-set beliefs'. The beliefs are un-normalized versions of the marginals :

$$\beta_i(C_i) = \tilde{P}_\Phi(C_i) \text{ and } \mu_{i,j}(S_{i,j}) = \tilde{P}_\Phi(S_{i,j})$$

(normalizing $\beta_i(C_i)$ gives $P(C_i)$). A 'calibrated clique tree' or a set of 'calibrated beliefs' is one in which all neighboring beliefs agree (the sum-product algorithm always results in such calibrated beliefs). In particular, the neighboring cliques agree on the marginals of their shared variables. A set of calibrated beliefs β, μ gives a different parameterization of $\tilde{P}_\Phi$ than the original $\tilde{P}_\Phi = \prod \phi$:

$$\tilde{P}_\Phi(X) = \frac{\prod_i \beta_i(C_i)}{\prod_{i,j} \mu_{i,j}(S_{i,j})}$$

There is a slightly different view of the above sum-product algorithm. When sending a message $\delta_{i\to j}(S_{i,j})$, we deliberately do not factor in the message $\delta_{j\to i}(S_{i,j})$ to avoid 'double-counting' information. Instead, we could equivalently allow ourselves to factor the message $\delta_{j\to i}$ in, but divide by the message before sending $\delta_{i\to j}$. This allows us to view the algorithm as incrementally updating the beliefs β_i, instead of only computing them at the end:

$$\delta_{i\to k} := \frac{\sum_{C_i \setminus S_{i,j}} \beta_i(C_i)}{\delta_{j\to i}(S_{i,j})}$$

Now, we don't have to remember the initial potentials ψ_i, and we view the algorithm as incrementally updating the β_i. This is called 'belief-updating':

1. Initialize beliefs to the initial potentials: $\beta_i \leftarrow \psi_i$
2. Initialize sep-set beliefs to 1: $\mu_{i,j} \leftarrow 1$.
3. Send messages: $\sigma_{i\to j} \leftarrow \sum_{C_i \setminus S_{i,j}} \beta_i(C_i)$, $\beta_j(C_j) \leftarrow \beta_j(C_j)\frac{\sigma_{i\to j}}{\mu_{i,j}}$, $\mu_{i,j} \leftarrow \sigma_{i\to j}$

Now, we don't have to be strict about the order we send messages. The algorithm will converge exactly when the beliefs are calibrated (more updates won't change any beliefs). It can be shown that the 'clique tree invariant'

$$\tilde{P}_\Phi(X) = \frac{\prod_i \beta_i(C_i)}{\prod_{i,j} \mu_{i,j}(S_{i,j})}$$

which we showed holds for calibrated beliefs at the end of the earlier sum-product algorithm, actually holds throughout the whole belief-updating procedure (even though the beliefs are not necessarily calibrated yet). Initially, the β_i are the initial potentials and the $\mu_{i,j} = 1$, and by the end at convergence, the beliefs are the un-normalized marginals:

$$\tilde{P}_\Phi(X) = \frac{\prod_i \psi_i(C_i)}{1} = \cdots = \frac{\prod_i \beta_i(C_i)}{\prod_{i,j} \mu_{i,j}(S_{i,j})}$$

Thus, we are gradually changing the parameterization into the one we want (the one that trivially provides the marginals).

Other queries

Out-of-clique queries

Constructing a clique tree

We can get a clique tree either from variable elimination applied with a certain elimination ordering, or by finding the maximal cliques in the triangulated chordal graph. Finding the optimal triangulation is NP-hard. Finding the maximal in a chordal graph is easy.

KF CH 11: Inference as optimization

We want to do inference in P_Φ. We construct an approximation to P_Φ, denoted Q, for which inference is easy. For each method, we will define a target class of easy distributions, then select the Q from that class that best approximates P_Φ. Queries are then answered using inference on Q. Finding the best Q is a constrained optimization problem. We see that in setting the derivatives of the Lagrangian to zero, we obtain fixed point equations. Interestingly, optimizing by iteratively applying these fixed point equations can be interpreted as message passing.

The distance measure that we want to optimize is the KL divergence between P_Φ and Q. There are two versions, $\mathbb{KL}(P_\Phi||Q)$ and $\mathbb{KL}(Q||P_\Phi)$. Generally, $\mathbb{KL}(P_\Phi||Q)$ requires us to compute marginals in P_Φ, which is intractable, so we use $\mathbb{KL}(Q||P_\Phi)$ instead.

Now, the $\mathbb{KL}(||)$ has a sum over all instantiations x. We note that we can decompose this KL divergence into:

$$\mathbb{KL}(Q||P_\Phi) = \log Z - F[\tilde{P}_\Phi, Q]$$

$$F[\tilde{P}_\Phi, Q] = E_Q[\log \tilde{P}_\Phi] + H(Q) = \sum_{\phi \in \Phi} E_Q[\log \phi] + H(Q)$$

Since $\log Z$ is a property of $\tilde{P}_\Phi$ and doesn't depend on Q, minimizing $\mathbb{KL}(Q||P_\Phi)$ is equivalent to maximizing $F[\tilde{P}_\Phi, Q]$. Furthermore, the new quantity $F[\tilde{P}_\Phi, Q]$ breaks down into expecations involving smaller numbers of variables, due to the factorization structure of P_Φ. F is called the 'free energy' or 'energy functional'.

Exact inference as optimization problem

We know that a set of beliefs $\beta_i, \mu_{i,j}$ over a clique tree defines a distribution Q:

$$Q(X) \propto \frac{\prod_i \beta_i(C_i)}{\prod_{i,j}(S_{i,j})}$$

If, given a clique tree that satisfies the 'family preservation property' for P_Φ (meaning all the original potentials ϕ are fully contained in a clique in the tree), then we maximize $F[\tilde{P}_\Phi, Q]$ over $\beta_i, \mu_{i,j}$ subject to these beliefs being calibrated (and summing to one—the book is confusing on the normalization issue here). Because of the family preservation, the energy functional simplifies to local expectations:

$$F[\tilde{P}_\Phi, Q] = \sum_i E_{\beta_i}[\log \psi_i] + \sum_i H(\beta_i) - \sum_{i,j} H(\mu_{i,j})$$

Forming the Lagrangian for this constrained optimization problem and setting derivatives to zero gives fixed point equations which are identical to the message passing for sum-product.

Loopy belief propagation

We generalize the belief propagation for clique trees to more general 'cluster graphs'. These still have to satisfy a running intersection property, but we no longer require that $S_{i,j} = C_i \cap C_j$. Beliefs β_i in a cluster graph can also be calibrated, meaning that neighboring cliques agree on the marginals of the $S_{i,j}$. If a calibrated cluster graph satisfies the new running intersection property, then all clusters containing a variable X agree on its marginal. We use the same procedure as in the 'belief propagation' version of sum-product above.

Loopy belief propagation still satisfies the representation invariant at all iterations:

$$\tilde{P}_\Phi = \frac{\prod_i \beta_i(C_i)}{\prod_{i,j} \mu_{i,j}(S_{i,j})}$$

If we (locally) calibrate a set of beliefs β_i over a general cluster graph, the beliefs are not necessarily the marginals of P_Φ.

Unification of Message Passing Schemes

Every message passing scheme represents the distribution Q by a set of local beliefs (denoted β_i or τ) over some smaller subsets of the variables. The set of the local beliefs we allow comprises the constraints in the optimization problem. For some cluster graph G, denote the set of possible cluster marginals realizable by some distribution over the graph (the set of marginals that are all globally consistent), by $\mathbb{M}(G)$, called the 'marginal polytope'. If we optimize over beliefs $\tau \in \mathbb{M}(G)$, we are exploring the full set of all distributions Q that factorize over the graph G.

For the objective, we want to minimize $\mathbb{KL}(Q||P_\Phi)$, which is equivalent to maximizing $F[\tilde{P}_\Phi, Q] = \sum_{\phi \in \Phi} E_Q[\log \phi] + H(Q)$. Assuming our potentials ϕ fit into our cliques, the first term is linear in the local τ_i for each clique, and is tractable (meaning the local beliefs can easily see how to best match their local potentials). The second term (the total entropy of Q), however, is global and involves all the τ_i.

Minimize $\sum_{\phi \in \Phi} E_{Q_\mu}[\log \phi] + H(Q_\mu)$ subject to $\mu \in \mathbb{M}(G)$

In fact, we don't know how to express the total entropy as a simple function of the individual marginals τ_i (we can't go from the *marginal* to the *global* representations easily).

Characterizing $\mathbb{M}(G)$ for a general graph G with loops is intractable, because there are exponentially many constraints that define this set.

For example, in trees (or clique trees), the set of beliefs β_i that satisfy the local consistency contraints is identical to the set of all possible marginals realizable by some distribution over the tree.

1 my own simple EP explanation

Note that loopy belief propagation can in principle be used with continuous variables as well as discrete variables. As an example, consider a Bayesian network of univariate-

Gaussians, with linear conditionals on their parents. Then, the messages are continuous functions, which end up being expressed as the natural (precision, precision-weighted mean) parameters of Gaussians. BP in a factor-graph formalism can be expressed as:
Messages from variables to factors:

$$m_{i\to f}(x_i) = \frac{\prod_{g\to i} m_{g\to i}(x_i)}{m_{f\to i}(x_i)}$$

and messages from factors to variables:

$$m_{f\to i}(x_i) = \sum_{x_{-i}} f(x) \prod_{j\to f} m_{j\to f}(x_j)$$

Note that at any time the current beliefs (marginals) of a variable is given by the product of all incoming messages from factors:

$$\beta_i(x_i) \propto \prod_{f\to i} m_{f\to i}(x_i)$$

If our factors are all of a nice form (e.g. Gaussians), then the message functions sent around will be able to be combined (generalize this to the whole exponential family). However, if some factors are not of a nice form (e.g. step function in trueSkill), we need to send an approximate message that is of a nice form. A naive way of doing this would be to approximate the bad factors immediately with a Gaussian for example.

This will not work well if the original functions are not similar to the approximating family F. But we can do better! We realize that the approximation we send out to the rest of the graph will eventually get multiplied in with all the other information elsewhere in the graph, if we can make a really good approximation in the part of the state space that the whole graph gives high probability, the other parts of the state space don't matter very much. Also, we already have information locally about the beliefs of the rest of the graph—the messages that just came in give us a sense of where the probability mass lies in the state space. Therefore, we do:

$$\tilde{m}_{i\to f}(x_i) = \frac{\text{proj}[\prod_{g\to i} m_{g\to i}(x_i)]}{m_{f\to i}}$$

The intuition here is that ... 'even if a node is non-Gaussian internally, it will both receive and send messages in this format'.