draft

Labelling with Loopy Belief Revision

Max Welling

University of Toronto
Department of Computer Science
welling@cs.toronto.edu

Keywords: labelling, belief revision

Abstract

This is a short note to show that we may use the "loopy" max-product algorithm to find good approximate solutions to a class of labelling problems where the model contains loops.

1 Model

Consider a setting where we have a number of different detectors, e.g. left/right eye detectors, nose detector and left/right mouth corner detectors. When presented with a particular image, these detectors may produce many "hits" on false background locations and may be also fail to detect the particular feature due to noise or occlusion. Assume also that we have a joint model for these features, e.g. a joint Gaussian model with means and co-variances as free parameters. Thus, given that we know which detections on an image correspond to which features we can compute a probability. Let $X = \{x_i\},\ i = 1..N$ denote the set of locations of our feature detectors. A subset of these corresponds to the true features. It is our task to find which detections correspond to which features. This is called the labelling problem. Denote with $L = \{l_a\},\ a = 1..M$ the set of labels over all features in the model, e.g. $l_a = j$ denotes that detection j is being assigned to feature a. The probability of the detections given some labelling is given by,

$$P(X|L) = \frac{\sqrt{\det(W)}}{(2\pi)^{DM/2}} \exp\left[-\frac{1}{2}\sum_{a,b}(x_{l_a} - \mu_a)^T W_{ab}(x_{l_b} - \mu_b)\right] P_{BG}(\{X_{BG}\}) \tag{1}$$

where D is the dimensionality of the feature space, P_{BG} is some noise model for the detections which do not correspond any feature and W_{ab} represent precision matrices (inverse covariances). If we write W we have stacked W_{ab} in one big $DM \times DM$ dimensional matrix. The labelling problem can be formulated as finding the MAP (maximum a posteriori) assignment of

$$P(L|X) \propto P(X|L)\ P(L) \tag{2}$$

where P(L) captures our prior beliefs about the possible labelling assignments. I will now use an essential property of Gaussian distributions, namely that all the interactions are at most quadratic, i.e. between two features. Moreover, we will also express our prior beliefs over possible labellings by writing them as products of prior beliefs between pairs of nodes. For instance, we would like to enforce that *a particular detection cannot be assigned to two or more features*, while all other possible labellings are equally likely. Given the above, we can thus write the posterior probability over labellings of the foreground model as follows,

$$P(L|X) \propto \prod_{(ab)} \Psi_{ab}(l_a, l_b) \prod_{a} \Psi_a(l_a) \tag{3}$$

where (ab) denote all interacting features (which will be called neighbors in the following), and where we have defined,

$$\Psi_a(l_a) = \exp\left[-\frac{1}{2}(x_{l_a} - \mu_a)^T W_{aa} (x_{l_a} - \mu_a)\right] \tag{4}$$

$$\Psi_{ab}(l_a, l_b) = \exp\left[-\frac{1}{2}(x_{l_a} - \mu_a)^T W_{ab} (x_{l_b} - \mu_b)\right] M_{ab}(l_a, l_b) \tag{5}$$

The matrix $M_{ab}(l_a, l_b)$ captures the prior belief that a single detection cannot be assigned to both feature a and feature b, i.e. $l_a \neq l_b$.

$$M_{ab}(l_a, l_b) = \mathbf{1}_N \mathbf{1}_N^T - \mathbf{I}_N \tag{6}$$

i.e. the all ones matrix with zeros on the diagonal and N is the total number of detections. We have ignored the background model for the moment since it will not influence the solution to our labelling problem.

We can think of the above model as a graph with links between all nodes which have interactions. Because of the matrix M_{ab}, all nodes are connected, even if we would assume that some nodes have vanishing weight matrix W_{ab}. Every node represents a feature a, and each feature can be in N possible states, corresponding to the assignment of a detection to that feature. We could also add $N - M$ background nodes which are disconnected from each other and the foreground model since they are independent. Our labelling problem can therefore be viewed as inference on a fully connected graphical model with M nodes and N states per node. It is well known that the complexity of this problem grows exponentially with the number of nodes in the graph.

1.1 Missing Features

Although the above problem formulation is quite general it does not deal with the problem of missing features, i.e. the possibility that *no detection corresponds to a particular feature*. The general strategy should be to integrate that feature out of our model and add a node to the background model. Fortunately, integrating out features from a Gaussian model is very simple, the rule is that the weight matrices between nodes which were neighbors of the node which is integrated out are to be changed as follows,

$$W_{ab} \to W_{ab} - W_{ac} W_{cc}^{-1} W_{cb} \tag{7}$$

where c is the node which is integrated out. In particular this implies that two uncorrelated neighboring nodes can become correlated through the process of integrating out a common neighbor.

2 Belief Revision

To compute the MAP assignment of our labelling problem on a tree like graph we can use dynamic programming. This is equivalent to running the belief revision (BR) updates, which are guaranteed to converge within a number of iteration equal to the longest path in the graph. First define messages $U_{ab}(l_b)$ from node a to node b. These messages are initialized at $\mathbf{1}$. Then run the following message passing updates in arbitrary order,

$$U^*_{ab}(l_b) = \max_{l_a} \Psi_{ab}(l_a, l_b) \Psi_a(l_a) \prod_{c \in N(a) \backslash b} U_{ca}(l_a) \tag{8}$$

i.e. to compute a new message from a to b, first compute the product over all incoming messages to node a, except the one coming from node b, and multiply with local and pairwise potentials. Finally, take the maximum of that expression over l_a. When converged, we compute the MAP assignment as follows,

$$l_a^{MAP} = \arg\max_{l_a} \Psi_a(l_a) \prod_{c \in N(a)} U_{ca}(l_a) \tag{9}$$

i.e. we compute the product over all incoming messages and multiply by the local potential at that node. The final assignment is found by the argument that maximizes this expression.

When applied to graphs with loops, this set of fixed point equations is not guaranteed to converge. However, if it converges, Weiss and Freeman have shown that the solution is a local maximum of the objective. Moreover the solution is guaranteed to be worse if one changes the assignments of nodes on an arbitrary disconnected set of loops, trees or loops with trees attached (this is a very strong result!).

To improve convergence one may apply damping as follows,

$$U_{ab}^{NEW}(l_b) = \left(U_{ab}^{OLD}(l_b)\right)^{\alpha} \left(U_{ab}^{*}(l_b)\right)^{(1-\alpha)} \qquad \alpha \in [0,1) \tag{10}$$

and U_{ab}^{*} is given by eqn. 8.

I propose to apply loopy belief revision (LBR) to the labelling problem defined in the previous section.

3 The Algorithm plus Extensions

Let's assume we have a model corresponding to a loopy graph. It is quite inconvenient that BR may fail to converge. But this need not be a problem if we first solve a simpler model, corresponding to a tree structured subgraph of the original model. This could for instance be done automatically by assigning weights to the links of the loopy graph and determine the maximal spanning tree (there is a very simple greedy algorithm to compute that). One possibility for these weights is the trace of the precision between two nodes,

$$w_{ab} = \mathbf{tr}(W_{ab}) \tag{11}$$

This would capture the idea that the most important links in the graph are the ones which impose the strongest constraints on its neighboring nodes, i.e. the nodes which are most correlated must be connected. Next, we run the BR on this tree-like subgraph which is guaranteed to converge within a number of iterations equal to the longest path in the tree. This is our first order solution. Remember the last messages that were send over the links at convergence. Now add the other links to the graph, which makes the graph loopy. Initialize these new messages at 1 and continue running LBR. If LBR does not converge, keep the first solution. If it does, check whether its solution id better than the first solutions and accept it accordingly. In this way we are guaranteed to get an answer at least as good as the best we got on the tree. This idea is related to Martin Wainwrights "tree-based reparameterization" algorithm, where a subset of all messages is kept fixed such that the ones which are updates live on a tree. One cycles through trees until convergence. This also boils down to a particular message passing scheduling. He informed me that he is now averaging all the messages computed on different trees before actually updating them which improves ones changes of convergence. However, I am sceptic about the result LBR provides if we cannot get it to converge with damping. In that case constraints imposed on the different links are conflicting and the energy landscape is "frustrated" i.e. has many local maxima. No method can miraculously recover the global minimum, since it is NP hard.

The problem of dealing with missing features seems very difficult and smart approximations may be made to incorporate this possibility directly into the iterative process. For now I don't see how.

One can improve performance considerably by clustering nodes into supernodes which will have many more states. For instance using (overlapping) triangles. The analogon of the above would be some sort of loopy junction tree algorithm. One can also use the "generalized belief propagation" algorithm or a loopy version of it which treats larger clusters exactly. I did not look into these possibilities and do not know if there are any obstacles in the way.

4 Experiments

I implemented two models. One is exactly the model described above, and the other one has "springs" between two connected nodes with a parameter for its rest-length and a parameter for the strength of the spring. There is also a Gaussian prior on each node equivalent to the model described in the previous sections,

$$\Psi_a(l_a) = \exp\left[-\frac{1}{2}(x_{l_a} - \mu_a)^T W_{aa}(x_{l_a} - \mu_a)\right] \tag{12}$$

$$\Psi_{ab}(l_a, l_b) = \exp\left[-\frac{1}{2}C_{ab}(r_{ab} - A_{ab})^2\right] M_{ab}(l_a, l_b) \tag{13}$$

where C_{ab} is the strength parameter, A_{ab} is the resting length of the spring and r_{ab} is the distance between nodes a and b.

The simulation shows that the LBR algorithm works well in both cases if the interactions represented by Ψ_{ab} are not too strong when compared to the prior information on the location of the features represented by the Ψ_a terms. When the interactions are strong the system gets highly frustrated with many conflicting demands imposed by the interactions, i.e. the energy surface has many peaks and it is hard to find the maximum (basically by any method). The maximum weights spanning tree method found in many cases an almost optimal solution and is much faster. However, it did not seem to improve on problems for which LBR does not converge. Real data are needed to see if LBR is of any use for labelling.