Lecture 23:

Iterative Scaling Algorithms

Sam Roweis

March 29, 2004

Learning with non-maximal Clique potentials

- We saw when we studied IPF that learning in undirected models could be hard if the clique potentials were not only over maximal cliques of a triangulated graph, because of the normalizer (partition function) Z.
- A necessary condition achieved at the maximum likelihood parameters is that the model's marginals over all the cliques used in the parameterization must match the empirical marginals.
- IPF solved this problem by fixing the conditional $p(x_{\bar{C}}|x_C)$ and updating the marginal $p(x_C)$ to match the empirical marginal.
- This allowed us to start with an overcomplete parameterization (e.g. from a non-decomposable model) where only some of the marginals matched, and to fix things up as we went.

$$\psi_c^{(t+1)}(x_c) = \psi_c^{(t)}(x_c)\frac{q(x_c)}{p^{(t)}(x_c)}$$

Learning in Fully Observed Models

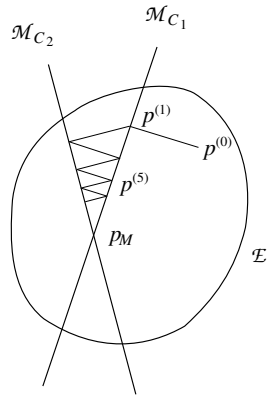
- We now know how to do inference efficiently and exactly in a wide class of directed and undirected models using either belief propagation or junction tree.
- Let us return to learning for a while.
- For fully observed directed models, learning was trivial: we just estimated the parameters separately for micro-models consisting of each node and all of its parents.
- For decomposable (triangulated) models, with potentials only on the maximal cliques, learning is also trivial: set each clique/separator potential to the observed marginal:

$$p_{ML} = \frac{\prod_C \tilde{p}(x_C)}{\prod_S \tilde{p}(x_S)}$$

and then assign each separator to exactly one clique and set $psi(x_C)$ to be the marginal divided by the separators assigned to it.

Geometric Interpretation of IPF

- We can think of IPF as performing a sequence of "projections" in the space of distributions. Each projection replaces the current distribution with the closest new distribution to it in some restricted family.
- For IPF, each step projects onto a manifold which has the correct marginal distribution for *one* of the the cliques. The projection lands on the distribution closest in KL distance to our current distribution but with the correct new marginal. All iterations stay in the exponential family $\mathcal{E}$. The algorithm converges to the single point that is in $\mathcal{E}$ and at the intersection of all the manifolds $\mathcal{M}_C$.



Iterative Scaling

- Iterative Scaling takes the idea of IPF one step further, and provides us with an algorithm for learning parameters in general feature-based (maximum-entropy) models:

$$p(\mathbf{x}|\theta) = \frac{1}{Z(\theta)} \exp\left\{\sum_i \theta_i f_i(\mathbf{x}_{Ci})\right\}$$

- Let us consider the scaled likelihood function

$$\tilde{\ell}(\theta; \mathcal{D}) = \frac{1}{N}\sum_{\mathbf{x}} n(\mathbf{x}) \log p(\mathbf{x}|\theta) = \sum_{\mathbf{x}} \tilde{p}(\mathbf{x}) \log p(\mathbf{x}|\theta)$$

- For now, we place the following constraints on the features:

$$f_i(\mathbf{x}) \geq 0 \qquad \sum_i f_i(\mathbf{x}) = 1 \quad \forall \mathbf{x}$$

Comparison between GIS and IPF

- IPF updates the distribution one clique at a time, while GIS updates all cliques in parallel.
- You can think of IPF as using special features which just measure the marginals over clique subsets of the variables: $f_i(x_C)$=1 if $x_C = x_C^{(i)}$ and 0 otherwise
- Now the sum of the features is equal to the number of cliques.
- For any one clique x_C, only one of the features is active, so within the clique, the features are non-negative and sum to one.
- So one IPF update is like one step of GIS restricted to a single clique.
- However, IPF corresponds to a sequence of GIS iterations in which the set of features updated at each iteration is always changing.

Generalized Iterative Scaling (GIS)

- The GIS algorithm updates based on the ratios of empirical marginals to model marginals (like IPF) but it updates the whole distribution in parallel (unlike IPF):

$$p^{(t+1)}(\mathbf{x}) = p^{(t)}(\mathbf{x}) \prod_i \left(\frac{\sum_{\mathbf{y}} \tilde{p}(\mathbf{y}) f_i(\mathbf{y})}{\sum_{\mathbf{y}} p^{(t)}(\mathbf{y}) f_i(\mathbf{y})}\right)^{f_i(\mathbf{x})}$$

- This is equivalent to the individual updates:

$$\theta_i^{(t+1)} = \theta_i^{(t)} + \log\left(\frac{\sum_{\mathbf{y}} \tilde{p}(\mathbf{y}) f_i(\mathbf{y})}{\sum_{\mathbf{y}} p^{(t)}(\mathbf{y}) f_i(\mathbf{y})}\right)$$

- In general, the intermediate distributions $p^{(t)}$ from GIS are *not* normalized, but it turns out that the updates still work fine, and also that the limiting distribution to which we converge *will* be normalized.

Learning with Latent Variables

- Finally, we return to the most general learning situation, that of learning with latent (hidden) variables.
- In general, we deal with this using the EM algorithm, by lower bounding the likelihood. We then require two steps:
  - The **E-step** needs to perform inference over the hidden variables given the observed data and the current parameters. For this we use Bayes rule, belief propagation (on trees), or a junction tree algorithm.
  - The **M-step** needs to update the parameters to the values which maximize the lower bound (e.g. maximize the expected complete log likelihood). For this we can use exact analytic updates (e.g. mixtures of Gaussians, factor analysis, HMMs) or we can plug in IPF/GIS/... as our "fully observed" M-step update algorithm.

Learning with GIS

- For example, if we wanted to apply GIS to learn the parameters of a complex maximum-entropy feature model with hidden variables x_H, we would need to perform inference to compute the expected sufficient statistics of the features given the observed data x_E and the parameters θ:

$$\bar{f}_i = \sum_{x_H} \tilde{p}(x_E) p(x_H | x_E, \theta) f_i(x_E, x_H)$$

- We would then use these expected statistics in the updates:

$$p^{(t+1)}(x_E, x_H) = p^{(t)}(x_E, x_H) \prod_i \left(\frac{\sum_{y_E} \bar{f}_i(y_E)}{\sum_{y_E} p^{(t)}(y_E, y_H) f_i(y_E, y_H)} \right)^{f_i(x_E,}$$

Derivation of GIS

- The derivation of GIS, which proves that it always increases the (scaled) likelihood is quite complex. It involves bounding the likelihood *twice*, once using an upper bound on the logarithm of the sum over all data which appears in the partition function (thus lower bounding the term $-\log Z$) and a second time to lower bound the exponential of a sum which couples all the parameters together.
- This second bound is what requires all the features to be positive and sum to one, since the feature values play the role of the convex combination coefficients when we apply Jensen to execute the second bound.