10-708: Probabilistic Graphical Models 10-708, Spring 2018

8 : Learning Fully Observed Undirected Graphical Models

Lecturer: Kayhan Batmanghelich *Scribes: Chenghui Zhou, Cheng Ran (Harvey) Zhang*

When learning fully observed Undirected Graphical Models (UGM), there are two settings in which we consider. The first setting only considers learning parameters when the random variables are discrete and the second setting involving learning parameters when random variables can be both discrete and continuous. Note that throughout this lecture we assume that 1) the graphical model structure is given and 2) every variable appears in the training examples.

1 Setting I: MLE of UGM with Discrete RV

In this section we mostly consider the case in which $\phi_C(x_c) = \theta_{C,x_c}$, e.x. each variables are discrete, where θ_k is the probability of a discrete value being equal to a certain value.

Consider categorical distribution:

$$p(x = t) \propto \prod_k \theta_k \mathtt{I}(x = t)$$

Tabular clique potentials would look like:

$$\phi_c(\mathcal{X}_c^n) = \prod_{\mathcal{Y}_c} \phi_c(\mathcal{Y}_c)^{\mathtt{I}[\mathcal{Y}_c = \mathcal{X}_c^n]}$$

The log likelihood under this distribution is:

$$L(\phi) = \sum_n \sum_c \sum_{\mathcal{Y}_c} \mathtt{I}[\mathcal{Y}_c = \mathcal{X}_c^n] \log \phi_c(\mathcal{Y}_c) - N \log Z(\phi) \tag{1}$$

where $Z(\phi) = \sum_{\mathcal{Y}_c} \prod_c \phi_c(\mathcal{Y}_c)$

Derivative of the log-likelihood is:

$$\frac{\partial}{\partial \phi_c(\mathcal{Y}_c)} L(\theta) = \sum_n \mathtt{I}[\mathcal{Y}_c = \mathcal{X}_c^n] \frac{1}{\phi_c(\mathcal{Y}_c)} - N \frac{p(\mathcal{Y}_c)}{\phi_c(\mathcal{Y}_c)} \tag{2}$$

Condition to sastisfy to get maximum likelihood parameter:

$$p(\mathcal{X}_c) = \epsilon(\mathcal{X}_c) \equiv \frac{1}{N} \sum_{n=1}^{N} \mathtt{I}[\mathcal{X}_c = \mathcal{X}_c^n]$$

The model marignals must be equal to the observed marginals.

1.1 MLE by Inspection (Decomposable Model)

Definiteion: Graph is decomposable if it can be subdivded into sets A, B, and S such S separates A and B.

Recipe for MLE by guessing

Conditions:

1. Graphical model is decomposable
2. Potentials defined on maximal cliques
3. Potentials are parameterized as $\phi_c(x_c) = \theta_{C,x_c}$

1. Set each clique potential to its empirical marginal
2. Divide out every non-empty intersection between cliques exactly once.

NOTE: If the graph is not decomposable or the potentials are not defined on the maximal cliques, we cannot equate empirical marginals to MLE of the clique potentials.

1.2 Iterative Proportional Fitting (IPF)

FIxed point iteration is a general tool for solving a system of equations. It can also iteratively optimize an objective function until convergenec.

General procedure:

- Given Objective function: $J(\theta)$
- Compute derivative, set to zero: $\frac{dJ(\theta)}{d\theta_i} = f(\theta)$
- Rearrange the equation s.t. one of parameters appears on the LHS: $0 = f(\theta) \Rightarrow \theta_i = g(\theta)$
- Initialize the parameters (only applicable in the first step)
- For parameters of i from $(1, \ldots, K)$, update each parameters and increment t: $\theta_i^{t+1} = g(\theta^t)$
- Repeat until convergence

Properties of IPF updates:

- Applies only when potentials are parameterized as $\phi_C(x_c) = \theta_{C,x_c}$
- IPF iterates a set of fixed point equations: $\phi_c^{(t+1)}(\mathcal{Y}_c) \leftarrow \phi_c^{(t)}(\mathcal{Y}_c)\frac{\epsilon(\mathcal{Y}_c)}{p^{(t)}(\mathcal{Y}_c)}$
- It's proven that IPF is a coordiante ascent algorithm. At each step the log likelihood increases and eventually converge to a global maximum.

2 Setting II: MLE of UGM with Continuous RV

2.1 Feature-Based Models

For the case where potential functions in an UGM may be continuous, we resort to a specific type of representation where the cliques can be represented as the inner product of parameters $\boldsymbol{\theta}$ with some feature feature function $f(\mathbf{x}_c)$ which we can design. This is the idea behind **feature-based models** and the potential functions will take the following form:

$$\phi(\mathbf{x}_c) = exp(\boldsymbol{\theta} \cdot f(\mathbf{x}_c))$$

It is important to note that feature-based models are useful even for graphs with discrete potential functions. For large cliques, general potentials are exponentially costly for inference and we must learn exponential number of parameters. In this case, we can choose to change the graphical model to make the cliques smaller; or we can keep the same graphical model but use features to achieve a less general parametrization of the clique potentials with less parameters. An example of this is provided below.

Example - Optical Character Recognition (OCR)
Given images of hand-writings, we want to recognize hand written words and make sure the written words is in the dictionary. Each character c_i of the word is a node in the graph and can be one of 26 letters. Assume we want to model all the dependencies between the nodes and have a fully connected graph, a graph of just 3 nodes will have $26^3 - 1$ parameters. However, we often know that some particular joint settings of the variables in a clique are quite likely or unlikely. For example, words ending with "ed" is likely a past tense while no words have "jkx" together. We can design features over the variables of the graph to reduce the number of parameters in the joint distribution.

Formally, a **feature** is a function which is vacuous over all joint settings except a few particular ones on which it is high or low. For example, we can have $f_{ing(c_1,c_2,c_3)}$ which is 1 if the string is "ing" and 0 otherwise. We can also design features over the input when the input is continuous.

Each feature function can be made into a micro-potential and we can multiply these micro-potentials together to get a clique potential. For example, a clique potential $\phi(c_1, c_2, c_3)$ can be expressed as:

$$\phi(c_1, c_2, c_3) = e^{\theta_{ing} f_{ing}} \times e^{\theta_{ed} f_{ed} \cdots} = exp(\sum_{k=1}^{K} \theta_k f_k(c_1, c_2, c_3))$$

Note that instead of using $26^3 - 1$ parameters to express the potential, we are only using K parameters, where K is the number of features we use. Parameters θ_i can be interpreted as the numerical strength of feature i.

If we write out the Gibb's distribution using the clique potentials with features, we have:

$$p(\mathbf{x}) = \frac{1}{Z(\theta)} exp(\sum_i \theta_i f_i(\mathbf{x}_i))$$

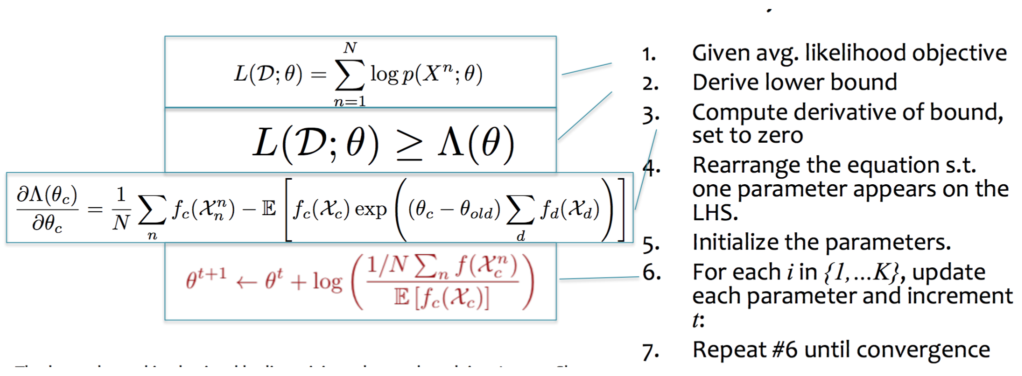
The joint distribution is in the form of exponential family, where $h(x) = 1$ and $T(x) = f(x)$.

2.2 Generalized Iterative Scaling

We can modify the IPF algorithm to solve for clique potentials of feature-based models. The key ideas of Generalized Iterative Scaling (GIS) is to:

1. Define a function that is the lower-bound of the log-likelihood that we are trying to optimize
2. Observe that the bound is tight at current parameters
3. Compute MLE on the lower bound using fixed-point iteration in order to increase log-likelihood.

The algorithm is presented below:



$$L(\mathcal{D};\theta) = \sum_{n=1}^{N} \log p(X^n;\theta)$$

$$L(\mathcal{D};\theta) \geq \Lambda(\theta)$$

$$\frac{\partial \Lambda(\theta_c)}{\partial \theta_c} = \frac{1}{N}\sum_n f_c(\mathcal{X}_n^n) - \mathbb{E}\left[f_c(\mathcal{X}_c)\exp\left((\theta_c - \theta_{old})\sum_d f_d(\mathcal{X}_d)\right)\right]$$

$$\theta^{t+1} \leftarrow \theta^t + \log\left(\frac{1/N\sum_n f(\mathcal{X}_c^n)}{\mathbb{E}\left[f_c(\mathcal{X}_c)\right]}\right)$$

1. Given avg. likelihood objective
2. Derive lower bound
3. Compute derivative of bound, set to zero
4. Rearrange the equation s.t. one parameter appears on the LHS.
5. Initialize the parameters.
6. For each *i* in *{1,…K}*, update each parameter and increment *t:*
7. Repeat #6 until convergence

The lower bound is obtained by linearizing a log and applying Jensen-Shannon in the following form:

$$\frac{1}{N}L(\theta) \geq \sum_c \left\{ \frac{1}{N}\sum_n f_c(\mathcal{X}_c^n)\theta_c - \left\langle p_c \exp\left(\alpha_c \sum_d f_d(\mathcal{X}_c)\right)\right\rangle_{p(\mathcal{X}|\theta^{old})} \right\}.$$

Note that the idea of GIS is very similar to that of Expectation-Maximization (EM) Algorithm.

GIS:

$$\theta^{t+1} \leftarrow \theta^t + \log\left(\frac{1/N\sum_n f(\mathcal{X}_c^n)}{\mathbb{E}\left[f_c(\mathcal{X}_c)\right]}\right)$$

IPF:

$$\phi_c^{(t+1)}(\mathcal{Y}_c) \leftarrow \phi_c^{(t)}(\mathcal{Y}_c)\frac{\epsilon(\mathcal{Y}_c)}{p^{(t)}(\mathcal{Y}_c)}$$

We can contrast IPF and GIS and notice that GIS looks like IPF in the log space. If we take the log of the IPF update equation, we have the following:

$$log(\phi_c^{t+1}) \leftarrow log(\phi_c^t) + log(\frac{\epsilon}{p^t(y)})$$

Both IPF and GIS then have update equation in the form of:

$$\text{old parameter} \leftarrow \text{new parameter} + \log(\frac{\text{emprical}}{\text{marginal}})$$

2.3 Gradient-Based Methods

We know that the joint distribution of feature-based models is in the exponential family and we can also use gradient-based methods to optimize the likelihood function.

Recipe for Gradient-Based Learning

1. Write down the objective function
2. Compute the partial derivative of the objective
3. feed objective function and derivative into black box gradient-based optimizer
4. retrieve optimal parameters from black box

We can use any optimization algorithms such as Newton's Methods, Conjugate Gradient, or Stochastic Gradient Descent. The pseudo-code for Stochasti Gradient Descent is presented below:

- Suppose we have N training examples s.t. $f(x) = \sum_{i=1}^{N} f_i(x)$.
- This implies that $\triangledown f(x) = \sum_{i=1}^{N} \triangledown f_i(x)$.

```
SGD Algorithm:
  1. Choose a starting point $x$.
  2. While not converged:
       ∘ Choose a step size $t$.
       ∘ Choose $i$ so that it sweeps through the training set.
       ∘ Update
                     $\vec{x}^{(k+1)} = \vec{x}^{(k)} + t\triangledown f_i(\vec{x})$
```

To derive the gradient of the likelihood function in a feature based model, we know that for one data point:

$$log(p(x;\theta)) = \sum_c \theta_c^T f(x_c) - log(Z(\theta))$$
$$\nabla_{\theta_c} log(p(x;\theta)) = f(x_c) - \nabla Z((\theta))/Z(\theta)$$

We know that

$$Z(\theta) = e^{A(\theta)}$$
$$\nabla_{\theta_c} Z(\theta) = \frac{\partial A}{\partial \theta_c} e^A(\theta)$$
$$\nabla_{\theta_c} Z(\theta) = E[f_c(x_c)]Z(\theta)$$

So for a single data point:

$$\nabla_{\theta_c} log(p(x;\theta)) = f(x_c) - E[f_c(x_c)]$$

and the gradient of the likelihood over all data points can be written as:

$$\nabla_{\theta_c} L(D;\theta) = \frac{1}{N}\sum_{n=1}^{N} f_c(x_c) - E[f_c(x_c)]$$

Computing $E[f_c(x_c)]$ requires us to do inference to compute the marginals. But the above equation does provide us with a way to compute the gradient and therefore we can use gradient-based optimizers to optimize the likelihood function.

We can interpret optimizing a likelihood as optimizing a "loss function". It's also a good idea to add regularizer to your loss function. In this case, we would be computing the regularized maximum likelihood. The regularization can be L1 or L2 regularization etc. Note that SGD also has effects of regularization as well.