Graphical Models

Jinlong Wu & Tiejun Li

Nov 2008

1 Introduction

Jordan [3] presentes a very concise introduction to *Graphical Models (GMs)*:

> Graphical models are a marriage between probability theory and graph theory. They provide a natural tool for dealing with two problems that occur throughout applied mathematics and engineering — uncertainty and complexity — and in particular they are playing an increasingly important role in the design and analysis of machine learning algorithms. Fundamental to the idea of a graphical model is the notion of modularity — a complex system is built by combining simpler parts. Probability theory provides the glue whereby the parts are combined, ensuring that the system as a whole is consistent, and providing ways to interface models to data. The graph theoretic side of graphical models provides both an intuitively appealing interface by which humans can model highly-interacting sets of variables as well as a data structure that lends itself naturally to the design of efficient general-purpose algorithms.
>
> Many of the classical multivariate probabilistic systems studied in fields such as statistics, systems engineering, information theory, pattern recognition and statistical mechanics are special cases of the general graphical model formalism — examples include mixture models, factor analysis, hidden Markov models, Kalman filters and Ising models. The graphical model framework provides a way to view all of these systems as instances of a common underlying formalism. This view has many advantages — in particular, specialized techniques that have been developed in one field can be transferred between research communities and exploited more widely. Moreover, the graphical model formalism provides a natural framework for the design of new systems.

GMs are usually divided into two types — *undirected* and *directed*. Undirected GMs are also called *Markov Networks* or *Markov Random Fields (MRFs)*, and directed GMs are also known as *Bayesian Networks (BNs)*, *belief networks*, *generative models* or *causal models*.

1.1 Directed GMs (Bayesian Networks) [4]

In Bayesian Networks (BNs) each vertex represents a random variable, and an arc from vertex X to vertex Y (we also said that X is one of the parents of Y) means X is one of the reasons why Y happens, i.e., X causes Y. Hence BNs are acyclic. BNs assume that a variable is independent of its ancestors given its parents, and use the notation "$\perp\!\!\!\perp$" to show the conditional independence.

A simple example is shown in Figure 1. We can write $R \perp\!\!\!\perp S|C$ and $W \perp\!\!\!\perp C|S,R$ according to the conditional independence. The tables around the BNs are the *conditional probability tables (CPTs)* of the corresponding variables. According to the above assumption, the joint probability of all the variables is

$$P(C,S,R,W) = P(C)P(S|C)P(R|C)P(W|S,R) \quad .$$

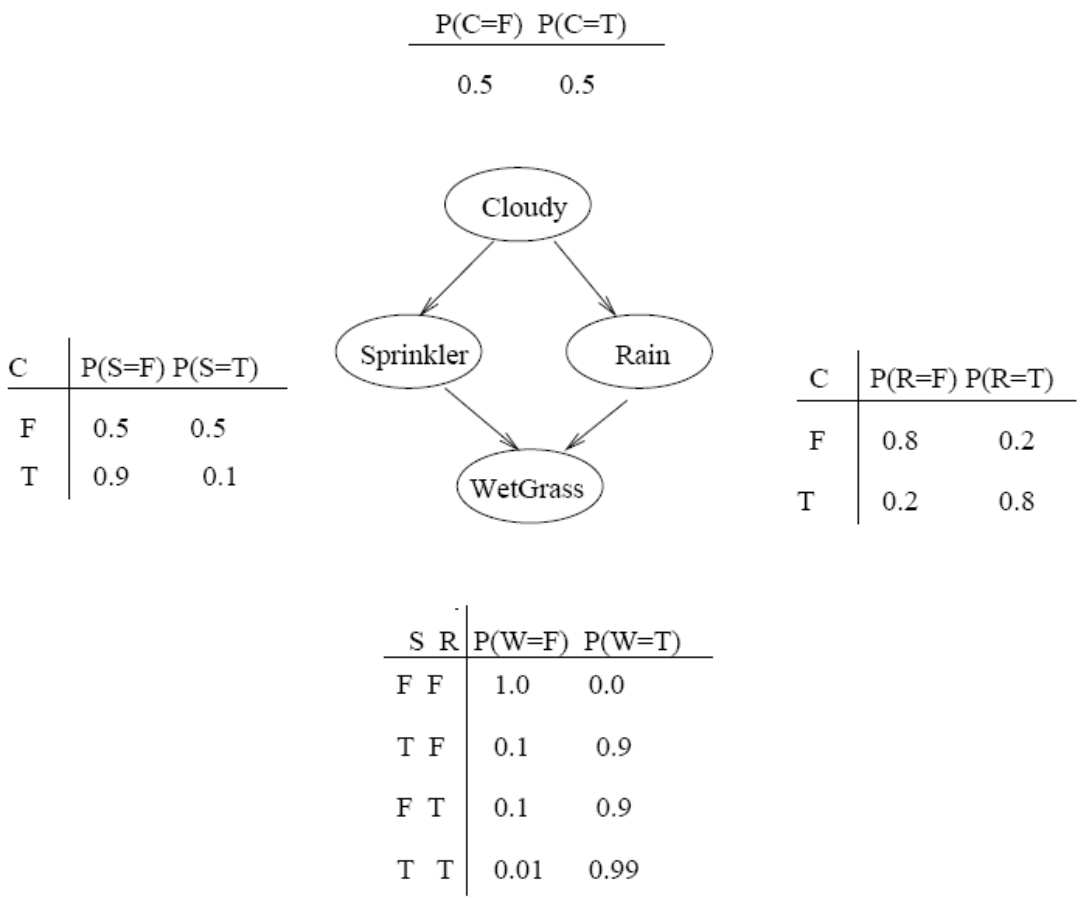

Figure 1: A simple Bayesian network from [5] and the related CPTs.

1.2 Undirected Graphical Models (Markov Random Fields) [4]

The joint distribution of a Markov Random Field (MRF) is defined by

$$P(X) = \frac{1}{\Xi} \prod_{C \in \mathcal{C}} \psi_C(X_C) \tag{1}$$

where C is the set of maximal cliques in the graph, $\psi_C(X_C)$ is a potential function (a positive, but otherwise arbitrary, real-valued function) on the clique X_C, and Ξ is the normalization factor

$$\Xi = \sum_X \prod_{C \in \mathcal{C}} \psi_C(X_C) \quad . \tag{2}$$

Consider the example in Figure 2. In this case, the joint distribution is

$$P(X, Y) \propto \Psi(X_1, X_2)\Psi(X_1, X_3)\Psi(X_2, X_4)\Psi(X_3, X_4) \prod_{i=1}^{4} \Psi(X_i, Y_i) \quad . \tag{3}$$

In low-level vision problems, the X_i are usually hidden, and each X_i node has its own "private" observation node Y_i, as in Figure 2. The potential $\Psi(X_i, Y_i) = P(Y_i|X_i)$ encodes the local likelihood; this is often a conditional Gaussian, where Y_i is the image intensity of pixel i, and X_i is the underlying (discrete) scene "label".

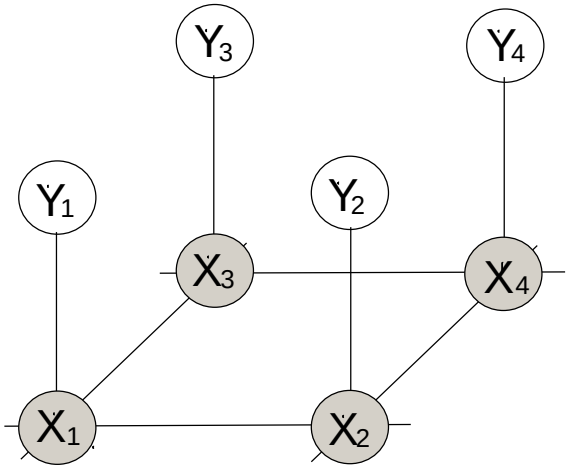

Figure 2: A Markov Random Field for low level vision. X_i is the hidden state of the world at grid position i, and Y_i is the corresponding observed state.

2 Inference and Learning

The main goal of inference is to estimate the values of hidden nodes, given the values of the observed nodes. It is usually based on Bayes' rule. Learning might refer to the structure (topology) of the model, or the parameters, or both [4].

Many algorithms for exact inference have been developed, such as *dynamic programming*, *forwards-backwards* for HMMs and so forth. However, these exact algorithms are time-consuming exponentially in the size of the largest cluster (also called the *induced width* of the graph) if all hidden nodes are discrete. Approximate inference is often much more popular for GMs. Some approximate inference methods are listed as follows. See more details in [4] and its references.

- Sampling (Monte Carlo) methods, such as *Importance Sampling*, *Markov Chain Monte Carlo (MCMC)*.
- Variational methods, such as the *mean-field approximation*.
- Loopy belief propagation.

It is beyond the scope of this chapter to discuss how to learn the structure of models. In the following we only discuss some methods to learn the parameters. Some common methods to learn the parameters is to find the parameters to maximize or minimize some objective function, such as

- to find the maximum likelihood estimates (MLEs) of the parameters;
- to find the maximum a posterori (MAP) estimates of the parameters;
- to find the parameters to minimize a type of error functions (usually MSE or MAE for regression, and cross-entropy for classification).

After determining the objective function, we can use *Gradient Descent (Ascent)*, *Expectation Maximization*, *Variational Methods* or other methods to achieve good estimates of the parameters.

3 Variational Methods for ML/MAP Learning

Consider a generative model with hidden variables Z and observed variables X. The observed data set $x = \{x_1, \ldots, x_N\}$ is assumed to be sampled from the model independently. Hence the log-likelihood as a function of the parameters Θ

is

$$\mathcal{L}(\Theta) = \log P(x|\Theta) = \sum_{n=1}^{N} \log P(x_n|\Theta) = \sum_{n=1}^{N} \log \int P(x_n, z_n|\Theta) dz_n \quad . \tag{4}$$

Maximum Likelihood (ML) tries to find the parameters Θ_{ML} to maximize the likelihood, or equivalently, to maximize the log-likelihood $\mathcal{L}(\Theta)$. That is,

$$\Theta_{ML} = \arg \max_{\Theta} \mathcal{L}(\Theta) \quad . \tag{5}$$

In Bayesian networks without hidden variables, $\mathcal{L}(\Theta)$ usually can decompose into local terms on each data point x_n, and so all parameters in $\mathcal{L}(\Theta)$ are decoupled, which makes the maximum problem easy. Unfortunately, if there exist some hidden variables, all parameters typically couple together, which makes the problem hard to solve.

The maximum problem usually can be simplified by introducing an auxiliary distribution $q_z(z)$ over the hidden variables (called *q-distribution*). In fact, for each data point x_n we use a distinct distribution $q_{z_n}(z_n)$ over the hidden variables, which gives rise to a lower bound on $\mathcal{L}(\Theta)$

$$\begin{aligned}
\mathcal{L}(\Theta) &= \sum_{n=1}^{N} \log \int P(x_n, z_n|\Theta) dz_n && (6)\\
&= \sum_{n=1}^{N} \log \int q_{z_n}(z_n) \frac{P(x_n, z_n|\Theta)}{q_{z_n}(z_n)} dz_n && (7)\\
&\geq \sum_{n=1}^{N} \int q_{z_n}(z_n) \log \frac{P(x_n, z_n|\Theta)}{q_{z_n}(z_n)} dz_n && (8)\\
&= \sum_{n=1}^{N} \Big[\int q_{z_n}(z_n) \log P(x_n, z_n|\Theta) dz_n - \int q_{z_n}(z_n) \log q_{z_n}(z_n) dz_n \Big] && (9)\\
&\equiv \mathcal{F}(q_{z_1}(z_1), \ldots, q_{z_n}(z_n); \Theta) \quad . && (10)
\end{aligned}$$

$\mathcal{F}(q_{z_1}(z_1), \ldots, q_{z_n}(z_n); \Theta)$ is called the *free energy*.

Since $\mathcal{F}$ is always a lower bound on $\mathcal{L}$, and maximizing $\mathcal{L}$ is difficult, we maximize $\mathcal{F}$ over $q_z(z)$ and Θ instead of $\mathcal{L}$.

3.1 Variational Methods for Unconstrained Distribution over Hidden Variables — EM

The Expectation Maximization (EM) algorithm alternates between E-Step and M-Step. The E-Step updates the posterior distribution over hidden variables given the current parameter values, and the M-Step updates the parameters given the posterior distribution. In this subsection, we will derive the EM algorithm by maximizing $\mathcal{F}(q_{z_1}(z_1), \ldots, q_{z_n}(z_n); \Theta)$ when q-distribution can be any distribution.

To maximize $\mathcal{F}$, we update the q-distribution and the parameters Θ iteratively. The update steps can be achieved by differentiating $\mathcal{F}$ with respect $q_{z_n}(z_n)$ and Θ, and setting the derivatives to zero respectively. The details are shown in the following.

When updating $q_{z_n}(z_n)$, we should notice the normalization constraints

$$\int q_{z_n}(z_n) dz_n = 1 \ , \quad \forall n = 1, \ldots, N \tag{11}$$

since $q_{z_n}(z_n)$ are density functions. The constraints can be incorporated into the objective function by Lagrange multipliers method perfectly to achieve a new objective function

$$\hat{\mathcal{F}}(q_z(z); \Theta) = \mathcal{F}(q_z(z); \Theta) + \sum_{n=1}^{N} \lambda_n \Big[\int q_{z_n}(z_n) dz_n - 1 \Big] \quad . \tag{12}$$

Setting the derivative to zero,

$$\frac{\partial}{\partial q_{z_n}(z_n)}\hat{\mathcal{F}}(q_z(z);\Theta^{(t)}) = \log P(x_n, z_n|\Theta^{(t)}) - \log q_{z_n}(z_n) - 1 + \lambda_n \quad (13)$$

$$= 0 \quad , \quad (14)$$

we get

$$q_{z_n}^{(t+1)}(z_n) = P(z_n|x_n, \Theta^{(t)}) \, , \quad \forall n = 1, \ldots, N \quad (15)$$

where the superscript (t) shows it is the value after the t-th iteration. The previous equation makes the inequality in (8) become an equality [1]. The update step (15) is identical to the E-Step in the EM algorithm.

In the same way, we update the parameters Θ by setting the derivative of $\mathcal{F}$ over Θ to zero, which is equivalent to

$$\Theta^{(t+1)} = \arg\ \max_\Theta \sum_{n=1}^{N} \int q_{z_n}^{(t+1)}(z_n) \log P(x_n, z_n|\Theta) dz_n \quad (16)$$

$$= \arg\ \max_\Theta \sum_{n=1}^{N} \int P(z_n|x_n, \Theta^{(t)}) \log P(x_n, z_n|\Theta) dz_n \quad (17)$$

since the second term in Equation (9) is independent of Θ. The update of Θ (17) is exactly the same as the M-Step in the EM algorithm.

3.2 Variational Methods for Constrained Distribution over Hidden Variables — VEM

In the previous subsection we show that variational methods are equivalent to the EM algorithms when q-distribution has no constraint conditions. However, in many models there exist multiple interacting hidden variables, which can produce intractable posterior distribution $P(z|x, \Theta)$.

Variational methods simplify the above problem by constraining the q-distribution to be of a particular tractable form, such as a particular parameterized family $q_{z_n}(z_n|\lambda_n)$, or a family factorized over the variable $Z_n = \{Z_{ni}\}_{i=1}^{Q}$. With the constraints, we can update the q-distribution similarly but always keep it in some fixed family. We call the new update algorithm the *Variational EM (VEM)*. Usually the updates do not result in the equality in (8), except that the exact posterior distribution $P(z_n|x_n, \Theta^{(t)})$ lies in the constrained family.

The updates of Θ now is based on the current variational posterior over hidden variables, that is,

$$\Theta^{(t+1)} = \arg\ \max_\Theta \sum_{n=1}^{N} \int q_{z_n}^{(t+1)}(z_n) \log P(x_n, z_n|\Theta) dz_n \quad . \quad (18)$$

We take the *mean field approximation* as an example.

[1] We can rewrite the free energy $\mathcal{F}$

$$\begin{aligned}
\mathcal{F}(q_{z_1}(z_1), \ldots, q_{z_n}(z_n); \Theta) &= \sum_{n=1}^{N} \int q_{z_n}(z_n) \log \frac{P(x_n, z_n|\Theta)}{q_{z_n}(z_n)} dz_n \\
&= \sum_{n=1}^{N} \int q_{z_n}(z_n) \log P(x_n|\Theta) dz_n + \sum_{n=1}^{N} \int q_{z_n}(z_n) \log \frac{P(z_n|x_n, \Theta)}{q_{z_n}(z_n)} dz_n \\
&= \sum_{n=1}^{N} \log P(x_n|\Theta) - \sum_{n=1}^{N} \int q_{z_n}(z_n) \log \frac{q_{z_n}(z_n)}{P(z_n|x_n, \Theta)} dz_n \\
&\equiv \sum_{n=1}^{N} \log P(x_n|\Theta) - \sum_{n=1}^{N} \mathrm{KL}[q_{z_n}(z_n) \parallel P(z_n|x_n, \Theta)] \quad ,
\end{aligned}$$

where $\mathrm{KL}[q_{z_n}(z_n) \parallel P(z_n|x_n, \Theta)] \equiv \int q_{z_n}(z_n) \log \frac{q_{z_n}(z_n)}{P(z_n|x_n, \Theta)} dz_n \geq 0$ is the Kullback-Leibler divergence between $q_{z_n}(z_n)$ and $P(z_n|x_n, \Theta)$. $\mathrm{KL}[q_{z_n}(z_n) \parallel P(z_n|x_n, \Theta)] = 0$ if and only if $q_{z_n}(z_n) = P(z_n|x_n, \Theta)$.

The mean field approximation assumes that each $q_{z_n}(z_n)$ is fully factorized over the hidden variables:

$$q_{z_n}(z_n) = \prod_{i=1}^{Q} q_{z_{ni}}(z_{ni}) \quad . \tag{19}$$

In this case $\mathcal{F}(q_z(z), \Theta)$ can reexpressed as follows:

$$\begin{aligned} \mathcal{F}(q_z(z), \Theta) &= \sum_{n=1}^{N} \int \Big[\prod_{i=1}^{Q} q_{z_{ni}}(z_{ni}) \log P(x_n, z_n|\Theta) - \prod_{i=1}^{Q} q_{z_{ni}}(z_{ni}) \log \prod_{i=1}^{Q} q_{z_{ni}}(z_{ni}) \Big] dz_n & (20) \\ &= \sum_{n=1}^{N} \int \Big[\prod_{i=1}^{Q} q_{z_{ni}}(z_{ni}) \log P(x_n, z_n|\Theta) - \sum_{i=1}^{Q} q_{z_{ni}}(z_{ni}) \log q_{z_{ni}}(z_{ni}) \Big] dz_n \quad . & (21) \end{aligned}$$

We set the derivative with respect to $q_{z_{ni}}(z_{ni})$ to zero, and obtain

$$q_{z_{ni}}(z_{ni}) = C_{ni} \exp\Big[\int \prod_{i'/i}^{Q} q_{z_{ni'}}(z_{ni'}) \log P(x_n, z_n|\Theta) dz_{n/i} \Big] \quad , \tag{22}$$

for each data point $n \in \{1, \ldots, N\}$ and each variational factorized component $i \in \{1, \ldots, Q\}$, where C_{ni} is the normalization constant, $dz_{n/i}$ shows the element of integration for all items in z_n except z_{ni}, and $\prod_{i'/i}$ denotes a product of all terms excluding i. These fixed point equations are called *mean-field equations* by analogy to such methods in statistical physics.

MAP learning by variational methods is similar. See more details in [1].

4 Examples

4.1 Neural Networks (NNs) [2]

A neural network is a two-stage regression or classification model, usually represented by a network diagram as shown in Figure 3. The middle layer is usually called the *hidden layer* and Z_m are called the *hidden units* because the values of Z_m are unknown.

For regression, typically $K = 1$ and the value of Y_1 is the final predictive value. For K-class classification, each Y_k, being coded 0 or 1, is the label variable of class k.

NNs can be modeled with the following formulas:

$$\begin{aligned} Z_m &= \sigma(\alpha_{0m} + \alpha_m^T X) \,, \quad m = 1, \ldots, M \,, & (23) \\ T_k &= \beta_{0k} + \beta_k^T Z \,, \quad k = 1, \ldots, K \,, & (24) \\ Y_k &= f_k(X) = g_k(T) \,, \quad k = 1, \ldots, K \,, & (25) \end{aligned}$$

where $Z = (Z_1, \ldots, Z_M)$, $\Theta \equiv \{\alpha_{0m}, \alpha_m, m = 1, \ldots, M; \beta_{0k}, \beta_k, k = 1, \ldots, K\}$ is the set of the model parameters which should be learnt, and $\sigma(v)$ is the *activation function*.

$\sigma(v)$ is usually chosen to be the *sigmoid* function $1/(1 + e^{-v})$ or *Gaussian radial basis* function $\exp\{-\lambda(v - \xi)^2\}$. Typically $g_k(T)$ is chosen to be the *identity* function $g_k(T) = T$ for regression, and *softmax* function $g_k(T) = e^{T_k} / \sum_{l=1}^{K} e^{T_l}$ for classification.

Sum-of-squared errors are usually used as the measure for regression to fit the NN

$$R(\Theta) = \sum_{k=1}^{K} \sum_{i=1}^{N} (y_{ik} - f_k(x_i))^2 \quad , \tag{26}$$

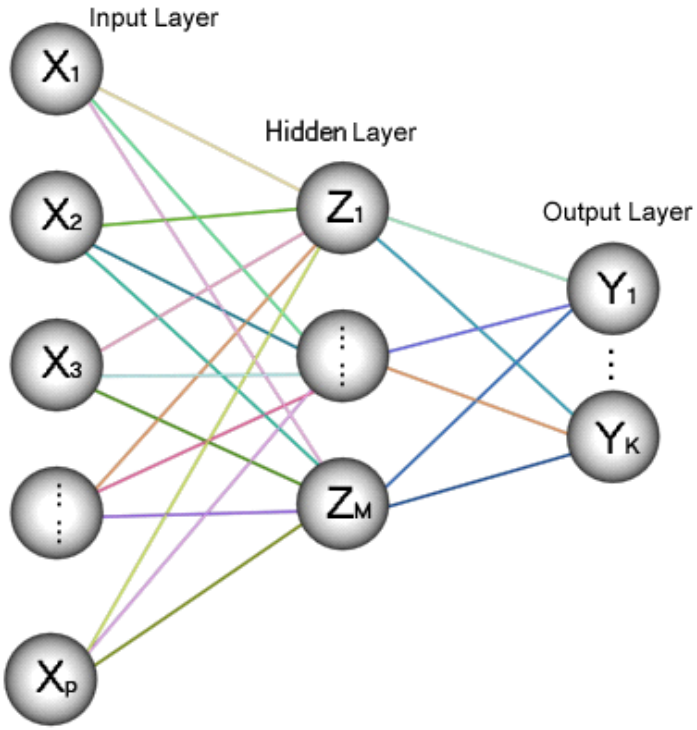

Figure 3: Schematic of a single hidden layer, feed-forward neural network.

where x_i and y_i are the i-th data point and the real value of the output in the training data respectively. For classification, we usually use either squared error or *cross-entropy (deviance)* as the measure

$$R(\Theta) = -\sum_{k=1}^{K}\sum_{i=1}^{N} y_{ik} \log f_k(x_i) \quad , \tag{27}$$

and the corresponding classifier is $G(x) = \arg\max_k f_k(x)$.

The generic approach to minimizing $R(\Theta)$ is gradient descent, which is also called *back-propagation* in this setting. Here we take the sum-of-squared error as an example and show the algorithm detail.

$$\frac{\partial R}{\partial \beta_{km}} = -2\sum_{i=1}^{N}(y_{ik} - f_k(x_i))g_k'(\beta_k^T z_i)z_{mi} \quad , \tag{28}$$

$$\frac{\partial R}{\partial \alpha_{ml}} = -2\sum_{i=1}^{N}\sum_{k=1}^{K}(y_{ik} - f_k(x_i))g_k'(\beta_k^T z_i)\beta_{km}\sigma'(\alpha_m^T x_i)x_{il} \quad , \tag{29}$$

where $z_{mi} = \sigma(\alpha_{0m} + \alpha_m^T x_i)$ and $z_i = (z_{1i}, \ldots, z_{Mi})$.

Given these derivatives, a gradient descent update at the $(t+1)$-th iteration can be done by

$$\beta_{km}^{(t+1)} = \beta_{km}^{(t)} - \eta_t \frac{\partial R}{\partial \beta_{km}^{(t)}} \quad , \tag{30}$$

$$\alpha_{ml}^{(t+1)} = \alpha_{ml}^{(t)} - \eta_t \frac{\partial R}{\partial \alpha_{ml}^{(t)}} \quad , \tag{31}$$

where η_t is the *learning rate* at the $(t+1)$-th iteration. Usually η_t decreases towards zero when the iteration $t \to \infty$ [2].

4.2 Gaussian Mixture

Gaussian Mixture models assume that the data points come from K different clusters, with probability π_k in the k-th cluster. The points in the k-th cluster are drawn from a Gaussian distribution with mean μ_k and deviation σ_k. Its graphical expression is shown in Figure 4, where variable Z_n is the class label of the n-th data point variable X_n which is a P-dimensional vector.

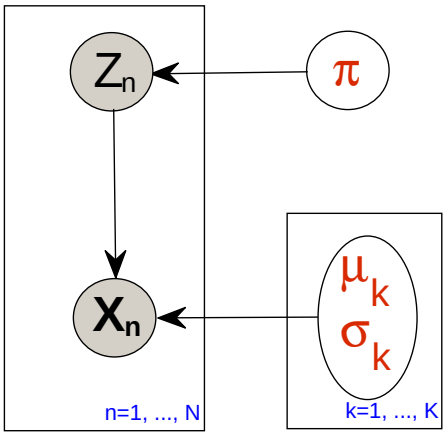

Figure 4: Schematic of a Gaussian Mixture Model.

We use the maximum log-likelihood estimate

$$R(\Theta) = \sum_{n=1}^{N} \log P(x_n|\Theta) = \sum_{n=1}^{N} \log \left[\sum_{k=1}^{K} \pi_k \phi_k(x_n) \right] \tag{32}$$

to learn the model parameters $\Theta = \{\pi_k, \mu_k, \sigma_k; k = 1, \ldots, K\}$, where $\phi_k(x)$ is the Gaussian density function with mean μ_k and deviation σ_k.

The Expectation Maximization (EM) method can be used to solve this problem perfectly as the label variable Z is considered as the latent variable.

At the E-Step, we calculate the posterior probability of the class label Z_n of the n-th point

$$r_{nk} \equiv P(Z_n = k|x_n, \Theta) = \frac{\pi_k \phi_k(x_n)}{\sum_{l=1}^{K} \pi_l \phi_l(x_n)} \quad , \tag{33}$$

and at the M-Step, we update the parameters

$$\mu_k = \frac{\sum_{n=1}^{N} r_{nk} x_n}{\sum_{n=1}^{N} r_{nk}} \quad , \tag{34}$$

$$\sigma_k^2 = \frac{\sum_{n=1}^{N} r_{nk} (x_n - \mu_k)^T (x_n - \mu_k)}{P \sum_{n=1}^{N} r_{nk}} \quad , \tag{35}$$

$$\pi_k = \frac{\sum_{n=1}^{N} r_{nk}}{N} \quad , \tag{36}$$

where P is the dimension of data points.

We update the parameters using equation (33)-(36) until the algorithm converges.

4.3 Probabilistic Matrix Factorization (PMF)

*PMF*s [6] are very powerful for Collaborative Filtering problems, which try to recommend new goods or service to users according to the historical ratings of the users for goods or service.

PMF supposes the rating value $\boldsymbol{R}_{u,m}$ of user u for movie m is a normal distribution with mean $\boldsymbol{V}_u^T \boldsymbol{W}_m$ and variance η^2 given user u's profile $\boldsymbol{V}_u$ and movie m's profile $\boldsymbol{W}_m$. $\boldsymbol{V}_u$ and $\boldsymbol{W}_m$ are K-dimensional variables. They have independent normal priors respectively. That is,

$$P(\boldsymbol{R}_{u,m}|\boldsymbol{V}_u, \boldsymbol{W}_m, \eta^2) = \mathcal{N}(\boldsymbol{R}_{u,m}|\boldsymbol{V}_u^T \boldsymbol{W}_m, \eta^2) \quad , \tag{37}$$

$$P(\boldsymbol{V}_u|\mu, \sigma^2) = \mathcal{N}(\boldsymbol{V}_u|\mu, \sigma^2 \mathrm{I}) \quad , \tag{38}$$

$$P(\boldsymbol{W}_m|\theta, \gamma^2) = \mathcal{N}(\boldsymbol{W}_m|\theta, \gamma^2 \mathrm{I}) \quad , \tag{39}$$

where $\Theta \equiv \{\eta^2, \mu, \sigma^2, \theta, \gamma^2\}$ are the model parameters. PMF also assumes that each of the three distributions is independent for different u and (or) m. Hence the whole probabilistic model can be expressed as follows and its graphical representation is shown in Figure 5. $\boldsymbol{V}$ and $\boldsymbol{W}$ are the *latent variables*.

$$P(\boldsymbol{R}|\boldsymbol{V}, \boldsymbol{W}) = \prod_{(u,m)\in\mathcal{P}} P(\boldsymbol{R}_{u,m}|\boldsymbol{V}_u, \boldsymbol{W}_m, \eta^2) = \prod_{(u,m)\in\mathcal{P}} \mathcal{N}(\boldsymbol{R}_{u,m}|\boldsymbol{V}_u^T \boldsymbol{W}_m, \eta^2) \quad , \tag{40}$$

$$P(\boldsymbol{V}|\mu, \sigma^2) = \prod_{u=1}^{U} P(\boldsymbol{V}_u|\mu, \sigma^2) = \prod_{u=1}^{U} \mathcal{N}(\boldsymbol{V}_u|\mu, \sigma^2 \mathrm{I}) \quad , \tag{41}$$

$$P(\boldsymbol{W}|\theta, \gamma^2) = \prod_{m=1}^{M} P(\boldsymbol{W}_m|\theta, \gamma^2) = \prod_{m=1}^{M} \mathcal{N}(\boldsymbol{W}_m|\theta, \gamma^2 \mathrm{I}) \quad , \tag{42}$$

where U is the number of users and M is the number of goods.

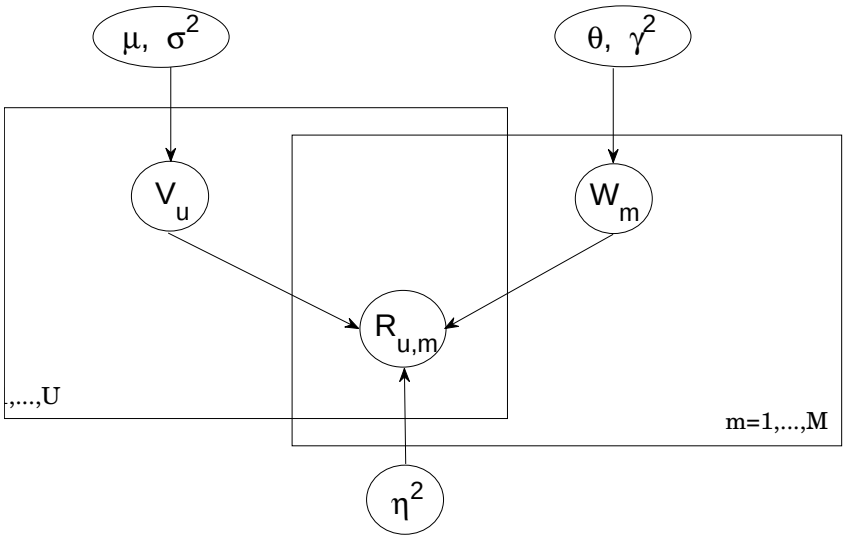

Figure 5: Schematic of a Probabilistic Matrix Factorization (PMF) model

The complete data likelihood is

$$\begin{aligned} P(R, V, W|\Theta) &= P(R|V, W)P(V|\mu, \sigma^2)P(W|\theta, \gamma^2) \\ &= \prod_{(u,m)\in\mathcal{P}} P(r_{u,m}|V_u, W_m, \eta^2) \prod_{u=1}^{U} P(V_u|\mu, \sigma^2) \prod_{m=1}^{M} P(W_m|\theta, \gamma^2) \quad . \end{aligned} \tag{43}$$

The original EM algorithm is not suitable here because $P(V, W|R, \Theta)$ is computationally difficult since V and W are not conditionally independent given R and Θ. The substitution is the *Variational EM (VEM)*. VEM assumes that $P(V, W|R, \Theta)$ is approximated by fully factorized q-distribution, that is,

$$P(V, W|R, \Theta) \simeq Q(V, W) = Q(V)Q(W) = \prod_{u=1}^{U} Q(V_u) \prod_{m=1}^{M} Q(W_m) \quad . \tag{44}$$

The total variational free energy of PMF with respect to the fully factorized q-distribution is

$$\begin{aligned}
&\mathcal{F}(Q(V), Q(W); \Theta) = \mathbb{E}_{V,W}[\log P(R, V, W|\Theta) - \log Q(V, W)] \\
&= -\frac{|\mathcal{P}|}{2}\log(2\pi\eta^2) - \mathbb{E}_{V,W}\Big[\sum_{(u,m)\in\mathcal{P}} \frac{(r_{u,m} - V_u^T W_m)^2}{2\eta^2}\Big] \\
&\quad - \frac{KU}{2}\log(2\pi\sigma^2) - \mathbb{E}_V\Big[\sum_{u=1}^{U} \frac{(V_u - \mu)^T(V_u - \mu)}{2\sigma^2}\Big] \\
&\quad - \frac{KM}{2}\log(2\pi\gamma^2) - \mathbb{E}_W\Big[\sum_{m=1}^{M} \frac{(W_m - \theta)^T(W_m - \theta)}{2\gamma^2}\Big] \\
&\quad - \mathbb{E}_V \log Q(V) - \mathbb{E}_W \log Q(W) \quad ,
\end{aligned} \tag{45}$$

where K is the dimension of V_u and $\mathbb{E}_V[f(V)]$ is the expectation of $f(V)$ with respect to the distribution $Q(V)$. The notation of $\mathbb{E}_W[f(W)]$ is defined similarly.

An iterative update procedure can be derived for optimizing the q-distribution and the parameters easily. We will show the detail step by step as follows.

We first work out the update for $Q(V)$. Differentiating $\mathcal{F}(Q(V), Q(W), \Theta)$ with respect to $Q(V)$ with others fixed, we then set the derivative to 0 and consider the distribution constraint $\int Q(V)dV = 1$ by Lagrange multipliers method.

$$\begin{aligned}
\frac{\partial \mathcal{F}}{\partial Q(V)} &= -\mathbb{E}_W\Big[\sum_{(u,m)\in\mathcal{P}} \frac{(r_{u,m} - V_u^T W_m)^2}{2\eta^2}\Big] - \sum_{u=1}^{U} \frac{(V_u - \mu)^T(V_u - \mu)}{2\sigma^2} - \log Q(V) - 1 - \lambda \\
&= \quad 0
\end{aligned}$$

where λ is the Lagrange multiplier. Hence,

$$Q(V) \propto \prod_{u=1}^{U} \exp\left(-\frac{1}{2}(V_u - \bar{V}_u)^T \Phi_u^{-1}(V_u - \bar{V}_u)\right) \tag{46}$$

$$\Phi_u = \left(\frac{1}{\eta^2}\sum_{m\in\mathcal{P}_u}(\Psi_m + \bar{W}_m\bar{W}_m^T) + \frac{1}{\sigma^2}\cdot \mathrm{I}\right)^{-1} \tag{47}$$

$$\bar{V}_u = \Phi_u\left(\frac{1}{\eta^2}\sum_{m\in\mathcal{P}_u} r_{u,m}\bar{W}_m + \frac{1}{\sigma^2}\mu\right) \quad . \tag{48}$$

Similarly we can derive the update step for $Q(W)$,

$$Q(W) \propto \prod_{m=1}^{M} \exp\left(-\frac{1}{2}(W_m - \bar{W}_m)^T \Psi_m^{-1}(W_m - \bar{W}_m)\right) \tag{49}$$

$$\Psi_m = \left(\frac{1}{\eta^2}\sum_{u\in\mathcal{P}^m}(\Phi_u + \bar{V}_u\bar{V}_u^T) + \frac{1}{\gamma^2}\cdot \mathrm{I}\right)^{-1} \tag{50}$$

$$\bar{W}_m = \Psi_m\left(\frac{1}{\eta^2}\sum_{u\in\mathcal{P}^m} r_{u,m}\bar{V}_u + \frac{1}{\gamma^2}\theta\right) \quad . \tag{51}$$

In the following we derive the update for the parameters.

$$
\begin{aligned}
\frac{\partial \mathcal{F}}{\partial \mu} &= \frac{\partial}{\partial \mu}\left(-\mathbb{E}_V\Big[\sum_{u=1}^{U}\frac{(V_u-\mu)^T(V_u-\mu)}{2\sigma^2}\Big]\right) \\
&= \frac{\partial}{\partial \mu}\left(-\sum_{u=1}^{U}\mathbb{E}_V\Big[\frac{[(V_u-\bar{V}_u)+(\bar{V}_u-\mu)]^T[(V_u-\bar{V}_u)+(\bar{V}_u-\mu)]}{2\sigma^2}\Big]\right) \\
&= \frac{\partial}{\partial \mu}\left(-\sum_{u=1}^{U}\frac{(\bar{V}_u-\mu)^T(\bar{V}_u-\mu)+\mathrm{trace}(\Phi_u)}{2\sigma^2}\right) \\
&= \frac{\partial}{\partial \mu}\left(-\sum_{u=1}^{U}\frac{(\bar{V}_u-\mu)^T(\bar{V}_u-\mu)}{2\sigma^2}\right) \\
&= -\frac{1}{2\sigma^2}\sum_{u=1}^{U}(2\mu-2\bar{V}_u) \\
&= 0 \quad .
\end{aligned}
$$

So,

$$
\mu = \frac{\sum_{u=1}^{U}\bar{V}_u}{U} \quad . \tag{52}
$$

Similarly,

$$
\theta = \frac{\sum_{m=1}^{M}\bar{W}_m}{M} \quad . \tag{53}
$$

$$
\begin{aligned}
\frac{\partial \mathcal{F}}{\partial(\sigma^2)} &= -\frac{KU}{2}\frac{1}{\sigma^2}-\frac{\partial}{\partial\sigma^2}\left(\sum_{u=1}^{U}\mathbb{E}_V\frac{[(V_u-\bar{V}_u)+(\bar{V}_u-\mu)]^T[(V_u-\bar{V}_u)+(\bar{V}_u-\mu)]}{2\sigma^2}\right) \\
&= -\frac{KU}{2}\frac{1}{\sigma^2}-\frac{\partial}{\partial\sigma^2}\left(\sum_{u=1}^{U}\frac{(\bar{V}_u-\mu)^T(\bar{V}_u-\mu)+\mathrm{trace}(\Phi_u)}{2\sigma^2}\right) \\
&= -\frac{KU}{2}\frac{1}{\sigma^2}-\frac{\partial}{\partial\sigma^2}\left(\sum_{u=1}^{U}\frac{(\bar{V}_u-\mu)^T(\bar{V}_u-\mu)+\mathrm{trace}(\Phi_u)}{2\sigma^2}\right) \\
&= -\frac{KU}{2}\frac{1}{\sigma^2}+\frac{1}{2\sigma^4}\sum_{u=1}^{U}[(\bar{V}_u-\mu)^T(\bar{V}_u-\mu)+\mathrm{trace}(\Phi_u)] \\
&= 0 \quad .
\end{aligned}
$$

So,

$$
\sigma^2 = \frac{\sum_{u=1}^{U}[(\bar{V}_u-\mu)^T(\bar{V}_u-\mu)+\mathrm{trace}(\Phi_u)]}{KU} \quad . \tag{54}
$$

Similarly,

$$
\gamma^2 = \frac{\sum_{m=1}^{M}[(\bar{W}_m-\theta)^T(\bar{W}_m-\theta)+\mathrm{trace}(\Psi_m)]}{KM} \quad . \tag{55}
$$

$$\begin{aligned}\frac{\partial \mathcal{F}}{\partial \eta^2} &= -\frac{|\mathcal{P}|}{2\eta^2} + \frac{1}{2\eta^4}\mathbb{E}_{V,W}\Big[\sum_{(u,m)\in\mathcal{P}}(r_{u,m} - V_u^T W_m)^2\Big] \\ &= -\frac{|\mathcal{P}|}{2\eta^2} + \frac{1}{2\eta^4}\sum_{(u,m)\in\mathcal{P}}[r_{u,m}^2 - 2r_{u,m}\bar{V}_u^T\bar{W}_m + \text{trace}((\Phi_u + \bar{V}_u\bar{V}_u^T)(\Psi_m + \bar{W}_m\bar{W}_m^T))] \\ &= 0 \quad .\end{aligned}$$

So,

$$\eta^2 = \frac{1}{|\mathcal{P}|}\sum_{(u,m)\in\mathcal{P}}[r_{u,m}^2 - 2r_{u,m}\bar{V}_u^T\bar{W}_m + \text{trace}((\Phi_u + \bar{V}_u\bar{V}_u^T)(\Psi_m + \bar{W}_m\bar{W}_m^T))] \quad . \tag{56}$$

We update the q-distribution and the parameters iteratively using Equation (46)-(56) until the algorithm converges.

References

[1] M. Beal. *Variational Algorithms for Approximate Bayesian Inference*. Doctoral thesis, 2003.

[2] T. Hastie, R. Tibshirani and J. Friedman. The elements of statistical learning. *Springer-Verlag*, New York, Berlin and Heidelberg, 2001.

[3] M. I. Jordan, editor. *Learning in Graphical Models*. MIT Press, 1999.

[4] Kevin P. Murphy. *An introduction to graphical models*.

[5] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, NJ, 1995.

[6] R. Salakhutdinov, and A. Mnih. Probabilistic Matrix Factorization. *Advances in Neural Information Processing Systems*, 2007.