Graphical Models

Muchang Bahng

Spring 2025

Contents

The concept of using latent variables to model some process will be used over and over again. We have seen simple examples of latent linear models, but what about nonlinear ones? It turns out that these can be seen as a specific instance of *graphical models*.

When computing high-dimensional distributions, the parameters needed to encode this density scales badly. We can see that a general Gaussian mixture model in $\mathbb{R}^n$ with k clusters requires $O(n^2k)$ parameters. If we wanted to sample from a distribution of portraits, then the dimension n would be the resolution of the image. For a 1024×1024 image, this requires $n = 3 \cdot 2^{20}$ dimensions, and modeling it with a GMM is hopeless. Fortunately, for complex distributions there is usually some dependencies (e.g. between neighboring pixels) that we can take advantage of. This is exactly what graphical models do. They factor complex distributions so that the scaling is much better. While there are graphical models that do not use latent variables, most interesting applications of graphical models require latent variables, and so we will focus on that. Additionally, we will introduce the EM algorithm, which will be used repeatedly and is particularly important in optimizing *variational autoencoders* in deep learning.

1 Bayesian Networks (Directed Graphical Models)

Note that the whole purpose of directed graphical models is to model some sort of *causal* relationship between two random variables. Note that while this is successful in practice, there is really no way to know for sure about any causality.

Definition 1.1 (Bayesian Network)

A **Bayesian network**, also known as a **directed probability model**, is a directed acyclic graph of M nodes representing a joint probability distribution of M scalar random variables. An edge pointing $A \to B$ means that the B is conditionally dependent on A, and that there is a very clear casual relationship coming from A to B. The **parents** of a node x_i is denoted pa_i, and the entire joint distribution can be broken up as such:

$$p(\mathbf{x}) = \prod_{m=1}^{M} p(x_m \mid x_{\text{pa}_m}) \tag{1}$$

which is unique due to it being a DAG. Not only is a Bayesian network easy to parameterize. We can also sample from the joint distribution by sequentially sampling starting from the parents to the final children, and discarding the ones (marginalizing) that we don't wish to sample. This is known as **ancestral sampling**.

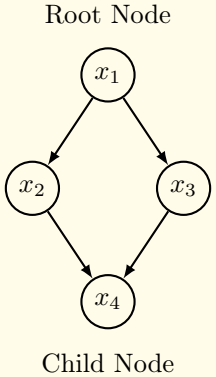


Figure 1

This following example cleared up any confusion when I learned Bayesian networks for the first time.

Example 1.1 (Relay Race)

Consider a 4×100m relay race where the final race time depends on multiple factors. We can model this as a Bayesian network where the total race time T depends on:

- Individual runner capabilities (R_1, R_2, R_3, R_4)
- Handoff success between runners (H_1, H_2, H_3)
- Individual leg performances (P_1, P_2, P_3, P_4)

The joint probability distribution factorizes as:

$$p(T, R_1, R_2, R_3, R_4, H_1, H_2, H_3, P_1, P_2, P_3, P_4) =$$
$$p(T|P_1, P_2, P_3, P_4) \prod_{i=1}^{4} p(R_i) \prod_{i=1}^{3} p(H_i|R_i, R_{i+1}) \prod_{i=1}^{4} p(P_i|R_i, H_{i-1})$$

where H_0 is undefined for P_1, and each runner's performance depends on their capability and the success of the previous handoff (except for the first runner). This network captures both the individual contributions and the critical dependencies between runners during baton exchanges.

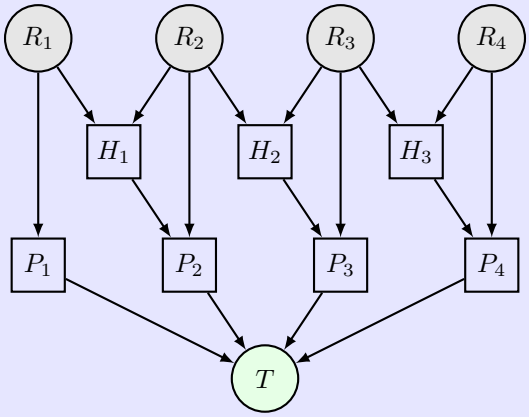


Figure 2: Bayesian Network for a 4x100m Relay Race. The graphical representation is much more compact and intuitive than simply writing out all the products.

Bayesian modelling with hierarchical priors.

Example 1.2 (Multinomial)

We first provide some motivation from a computational complexity perspective. Given a joint distribution of 2 random variables $\mathbf{x}_1, \mathbf{x}_2$, say which are multinomial with K classes, their joint distribution $p(\mathbf{x}_1, \mathbf{x}_2)$ is captured by $K^2 - 1$ parameters. For a general M random variables, then we have to keep a total of $K^M - 1$ parameters, and this increases exponentially. By building a directed graph with say r maximum number of variables appearing on either side of the conditioning bar in a single probability distribution, then the computational complexity scales as $O(K^r)$, which may save a lot of time if $r << M$.

Extending upon this example, we can see that we want to balance two things:

1. Fully conncted graphs have completely general distributions and have $O(K^M - 1)$ number of parameters (too complex).
2. If there are no links, the joint distribution fully factorizes into the product of its marginals and has $M(K-1)$ parameters (too simple) .

Graphs that have an intermediate level of connectivity allow for more general distributions compared to the

fully factorized one, while requiring fewer parameters than the general joint distribution. One model that balances this out is the hidden markov model.

Example 1.3 (Chain Graph)

Consider an M-node Markov chain. The marginal distribution $p(\mathbf{x}_1)$ requires $K-1$ parameters, and the remaining conditional distributions $p(\mathbf{x}_i \mid \mathbf{x}_{i-1})$ requires $K(K-1)$ parameters. Therefore, the total number of parameters is

$$K - 1 + (M-1)(K-1)K \in O(MK^2) \tag{2}$$

which scales relatively well, and we have

$$p(\{\mathbf{x}_m\}) = p(\mathbf{x}_1) \prod_{m=2}^{M} p(\mathbf{x}_m \mid \mathbf{x}_{m-1}) \tag{3}$$

TBD

We can turn this same graph into a Bayesian model by introducing priors for the paramters. Therefore, each node requires an additional parent representing the distribution over parameters (e.g. prior can be Dirichlet)

$$p(\{\mathbf{x}_m, \mu_m\}) = p(\mathbf{x}_1 \mid \mu_1)p(\mu_1) \prod_{m=2}^{M} p(\mathbf{x}_m \mid \mathbf{x}_{m-1}, \mu_m)p(\mu_m) \tag{4}$$

with $p(\mu_m) = \text{Dir}(\mu_m \mid \alpha_m)$ for some predetermined fixed hyperparameter α_m.

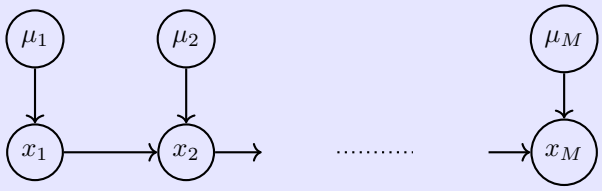


Figure 3

We could also choose to share a common prior over the parameters, trading flexibility for computational feasibility.

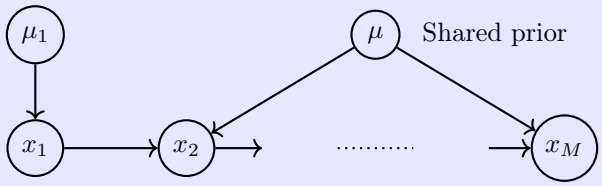


Figure 4

Another way to make more compact representations is through parameterized models. For example, if we have to compute $p(y = 1 \mid \mathbf{x}_1, \dots, \mathbf{x}_M)$, this in general has $O(K^M)$ parameters. However, we can obtain a more parsimonious form by using a logistic function acting on a linear combination of the parent variables

$$p(y = 1 \mid \mathbf{x}_1, \dots, \mathbf{x}_m) = \sigma\left(w_0 + \sum_{i=1}^{M} w_i x_i\right) = \sigma(\mathbf{w}^T \mathbf{x}) \tag{5}$$

We can look at an example how this is applied to sampling from high-dimensional Gaussian with **linear Gaussian models**.

Example 1.4 (Multivariate Gaussian)

Consider an arbitrary acyclic graph over D random variables, in which eachnode represents a single continuous Gaussian distribution with its mean given by a linear function of its parents.

$$p(x_i \mid \mathbf{pa}_i) = N\left(x_i \middle| w_{ij}x_j + b_j, v_i\right)$$

Given a multivariate Gaussian, let us try to decompose it into a directed graph. The log of the joint distribution takes form

$$\ln p(\mathbf{x}) = \sum_{i=1}^{D} \ln p(x_i \mid \text{pa}_i) = -\sum_{i=1}^{D} \frac{1}{2v_i}\left(x_i - \sum_{j \in \text{pa}_i} w_{ij}x_j - b_i\right)^2 + \text{const}$$

To compute the mean, we can see that by construction, every x_i is dependent on its ancestors, so

$$x_i = \sum_{j \in \text{pa}_i} w_{ij}x_j + b_i + \sqrt{v_i}\epsilon_i, \quad \epsilon_i \sim N(0,1)$$

so by linearity of expectation, we have

$$\mathbb{E}[x_i] = \sum_{j \in \text{pa}_i} w_{ij}\mathbb{E}[x_j] + b_i$$

So again, we can start at the top of the graph and compute the expectation. To compute covariance, we can obtain the i,jth element of $\mathbf{\Sigma}$ with a recurrence relation:

$$\begin{aligned}
\Sigma_{ij} &= \mathbb{E}[(x_i - \mathbb{E}[x_i])(x_j - \mathbb{E}[x_j])] \\
&= \mathbb{E}\left[(x_i - \mathbb{E}[x_i])\left(\sum_{k \in \text{pa}_j} w_{jk}(x_k - \mathbb{E}[x_k]) + \sqrt{v_i}\epsilon_j\right)\right] \\
&= \sum_{k \in \text{pa}_j} w_{jk}\Sigma_{ik} + I_{ij}v_j
\end{aligned}$$

If there were no links in the graphs, then the w_{ij}'s are 0, and so $\mathbb{E}[\mathbf{x}] = [b_1, \dots, b_D]$, making the covariance diagonal.If the graph is fully connected, then the total number of parameters is $D + D(D-1)/2$, which corresponds to a general symmetric covariance matrix.

Example 1.5 (Bilinear Gaussian Model)

Consider the following model

$$\begin{aligned}
u &\sim N(0,1) \\
v &\sim N(0,1) \\
r &\sim N(uv, 1)
\end{aligned}$$

where the mean of r is a product of 2 Gaussians. This is also a parameterized model.

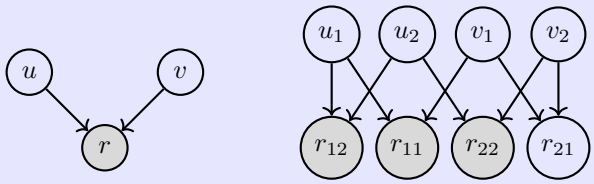


Figure 5

Definition 1.2 (Conditional Independence in Directed Graphs)

We say that a is independent of b given c if

$$p(a \mid b, c) = p(a \mid c)$$

or equivalently,

$$p(a, b \mid c) = p(a \mid b, c)\, p(b \mid c) = p(a \mid c)\, p(b \mid c)$$

Conveniently, we can directly read conditional independence properties of the joint distribution from the graph without any analytical measurements.

Example 1.6 (Conditional Independence on Dataset)

We can demonstrate conditional independence with iid data. Consider the problem of density estimation of some dataset $\mathcal{D} = \{x_i\}$ with some parameterized distribution of μ. Originally, the observations are not independent since they depend on μ.

$$p(\mathcal{D}) = \int_\mu p(\mathcal{D} \mid \mu)\, p(\mu)\, d\mu \tag{6}$$

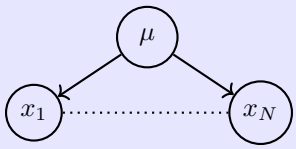


Figure 6

If we condition on μ and considered the joint over the observed variables, the variables are independent.

$$p(\mathcal{D} \mid \mu) = \prod_{n=1}^{N} p(x_n \mid \mu) \tag{7}$$

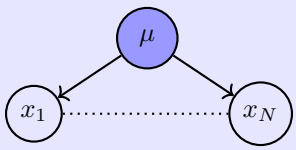


Figure 7

The example above identifies a node (the parent μ) where, if observed, causes the rest of the nodes to become independent. We can extend on this idea by taking an arbitrary x_i and finding a set of nodes such that if

they are observed, then x_i is indepedent from every other node.

Definition 1.3 (Markov Blanket in Directed Graphs)

The **Markov blanket** of a node is the minimal set of nodes that must be observed to make this node independent of all other nodes. It turns out that the parents, children, and coparents are all in the Markov blanket.

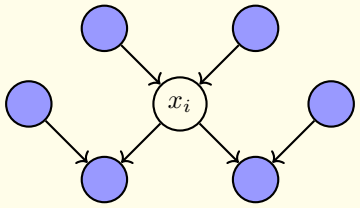


Figure 8

Note that

$$p(x_i \mid x_{j\neq i}) = \frac{p(x_1, \dots, x_M)}{\int p(x_1, \dots, x_M)\, dx} = \frac{\prod_k p(x_k \mid \text{pa}_k)}{\int \prod_k p(x_k \mid \text{pa}_k)\, dx_i} \tag{8}$$

One final interpretation is that we can view directed graphs as **distribution filters**. We take the joint probability distribution, will starts off as fully connected, and the directed graphs “filters" away the edges that are not needed. Therefore, the joint probability distribution $p(\mathbf{x})$ is only allows through the filter if and only if it satisfies the factorization property.

2 Markov Random Field (Undirected Graphical Models)

As the name implies, undirected models use undirected graphs, which are used to model relationships that go both ways rather than just one. Unlike directed graphs, which are useful for expressing casual relationships between random variables, undirected graphs are useful for expressing soft constraints between random variables.

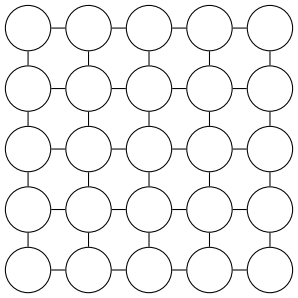


Figure 9: An MRF can be represented with this graph.

Definition 2.1 (Conditional Independence in Undirected Graphs)

Fortunately, conditional independence is easier compared to directed models. We can say A is conditionally independent to B given C if C blocks all paths between any node in A and any node in B.

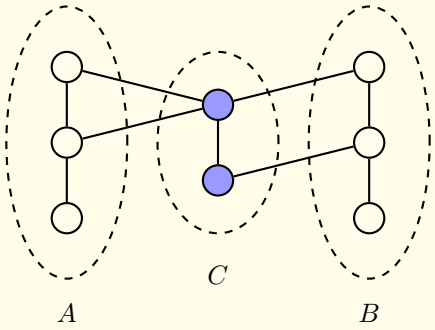


Figure 10: A is conditionally independent given C, denoted $A \perp\!\!\!\perp B|C$.

Definition 2.2 (Markov Blanket in Undirected Graphs)

The Markov blanket of a node, which is the minimal set of nodes that must be observered to make this node independent of the rest of the nodes, is simply the nodes that are directly connected to that node.

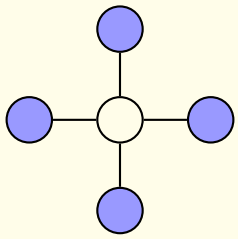


Figure 11: Once the neighbors of a node are realized, the node is independent of the rest of the nodes.

Therefore, the conditional distribution of x_i conditioned on all the variables in the graph is dependent only on the variables in the Markov blanket.

Now, let us talk about how we can actually define a probability distribution with this graph.

Definition 2.3 (Clique)

In an undirected graph, a **clique** is a set of nodes such that there exists a link between all pairs of nodes in that subset. A **maximal clique** is a clique such that it is not possible to include any other nodes in the set without it ceasing it to be a clique.

Given a joint random variable $\mathbf{x}$ represented by an undirected graph, the joint distribution is given by the product of non-negative potential functions over the maximal cliques

$$p(\mathbf{x}) = \frac{1}{Z} \prod_C \phi_C(x_C) \tag{9}$$

where

$$Z = \int p(\mathbf{x})\, d\mathbf{x} \tag{10}$$

is the normalizing constant, called the **partition function**. That is, each x_C is a maximal clique and ϕ_C is the nonnegative potential function of that clique.

This assignment looks pretty arbitrary. How do we know that any arbitrary joint distribution of $\mathbf{x}$, which has a undirected graphical representation, can be represented as the product of a bunch of functions over

the maximum cliques? Fortunately, there is a mathematical result that proves this.

Theorem 2.1 (Hammersley-Clifford)

The joint probability distribution of any undirected graph can be written as the product of potential functions on the maximal cliques of the graph. Furthermore, for any factorization of these potential functions, there exists an undirected graph for which is the joint.

Example 2.1 ()

For example, the joint distribution of the graph below

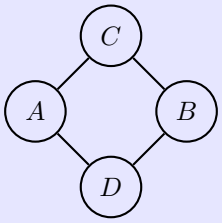


Figure 12

factorizes into

$$p(A,B,C,D) = \frac{1}{Z}\phi(A,C)\,\phi(C,B)\,\phi(B,D)\,\phi(A,D) \tag{11}$$

Note that each potential function ϕ is a mapping from the joint configuration of random variables in a clique to non-negative real numbers. The choice of potential functions is not restricted to having specific probabilistic interpretations, but since they must be nonnegative, we can just represent them as an exponential. The negative sign is not needed, but is a remnant of physics notation.

$$p(\mathbf{x}) = \frac{1}{Z}\prod_C \phi_C(x_C) = \frac{1}{Z}\exp\left\{-\sum_C E(x_C)\right\} = \frac{1}{Z}\underbrace{\exp\{-E(\mathbf{x})\}}_{\substack{\text{Boltzmann}\\ \text{distribution}}} \tag{12}$$

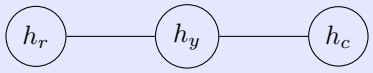
Any distribution that can be represented as the form above is called a **Boltzmann distribution**. So far, all we stated is that the joint probability distribution can be expressed as the product of a bunch of potential functions, but besides the fact that it is nonnegative, there is no probabilistic interpretation of these potentials (or equivalently, the energy functions). While this does give us greater flexibility in choosing potential functions, we must be careful in choosing them (e.g. choosing something like x^2 may cause the integral to diverge, making the joint not well-defined).

Clearly, these potential functions over the cliques should express which configuration of the local variables are preferred to others. It should assign higher values to configurations that are deemed (either by assumption or through training data) to be more probable. That is, each potential is like an "expert" that provides some opinion (the value) on a configuration, and the product of the values of all the potential represents the total opinion of all the experts. Therefore, global configurations with relatively high probabilities are those that find a good balance in satisfying the (possibly conflicting) influences of the clique potentials.

Example 2.2 (Transmission of Colds)

Say that you want to model a distribution over three binary variables: whether you or not you, your coworker, and your roommate is sick (0 represents sick and 1 represents healthy). Then, you can make simplifying assumptions that your roommate and your coworker do not know each other, so it is very unlikely that one of thme will give the other an infection such as a cold directly. Therefore,

we can model the indirect transmission of a cold from your coworker to your roommate by modeling the transmission of the cold from your coworker to you and then you to your roommate. Therefore, we have a model of form



One max clique contains h_y and h_c. The factor for this clique can be defined by a table and might have values resembling these.

	$h_y = 0$	$h_y = 1$
$h_c = 0$	2	1
$h_c = 1$	1	10

Table 1: States and Values of h_y and h_c

This table completely describes the potential function of this clique. Both of you are usually healthy, so the state $(1, 1)$ gets the maximum value of 1. If one of you are sick, then it is likely that the other is sick as well, so we have a value of 2 for $(0, 0)$. Finally, it is most unlikely that one of you is sick and the other healthy, which has a value of 1.

3 Hidden Markov Models

4 Nonlinear Latent Variable Models

Now we will consider ourselves with nonlinear latent variables models, which still defines a simple latent random variable Z with prior $p(z)$, but now a family of nonlinear functions $\{f_\theta(z)\}$ that defines the generative component $f_\theta(x \mid z)$. In factor models, we have taken linear transformations of random variables and therefore the likelihood had been easy to calculate, differentiate, and therefore optimize.

In the general nonlinear case, we usually deal with f_θ not as a transformation of Z to X, but really $f_\theta(z)$ becomes the *parameters* of $X \mid Z = z$. This allows to define the *implicitly parameterized* family of distributions $\{p_\theta\}$. Given that the true distribution of the data is $p^*(x)$, we would like to find a distribution $p_\theta(x)$ that is a good approximation.

$$p^*(x) \approx p_\theta(x) \tag{13}$$

To calculate the likelihood $p_\theta(x)$, we must compute the marginal

$$p_\theta(x) = \int p_\theta(x, z)\, dz = \int p_\theta(x \mid z)\, p(z)\, dz \tag{14}$$

which is known to be computationally intractable due to the integral. At first, it seems like all hope is lost, but statisticians have a few tricks up their sleeves.

1. The first trick is to notice that by Bayes rule, we can compute the likelihood not as an integral, but as

$$p_\theta(x) = \frac{p_\theta(x \mid z)\, p(z)}{p_\theta(z \mid x)} \tag{15}$$

   So it suffices to find a good approximation of $p_\theta(z \mid x)$, which is a probabilistic discriminative model for the latent variable (i.e. we are trying to compute the distribution of z given x as if we were predicting it). We can do MCMC since $p_\theta(z \mid x) \propto p_\theta(x \mid z)\, p(z)$, but often this can be slow to fit.

2. The next trick is called the variational lower bound, which is a lower bound on the log likelihood, and therefore by optimizing it we can hope to optimize the log-likelihood as well. This works well in practice.

3. The next trick is by optimizing the Fisher score, which is the gradient of the log likelihood *with respect to the covariates* (not the parameters!).

4.1 Variational Lower Bounds

We focus on this problem and define a family of distributions $\{q_\phi(z \mid x)\}_\phi$ and use it to approximate $p_\theta(z \mid x)$. Therefore, searching for a good ϕ and therefore a good q_ϕ is basically the problem of **variational Bayesian inference**. Essentially we are trying to construct an encoder and a decoder.

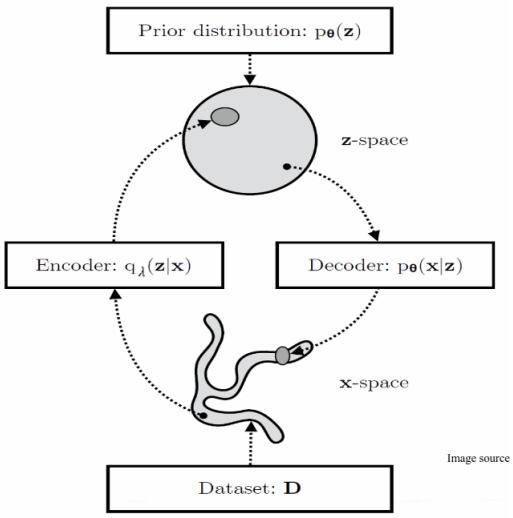


Figure 13: If $q_\phi = p_\theta$, then the diagram commutes, i.e. $p(z)p_\theta(x \mid z) = p(x)p_\theta(z \mid x) = p_\theta(x, z)$.

As we have stated before (and in pretty much all density estimation problems), our job is to maximize the log likelihood of the training set:

$$\sum_i \log p(x^{(i)}) \tag{16}$$

In order to do this for this problem, we need a little fact from information theory.

Theorem 4.1 (Log Likelihood vs Conditional Entropy)

The KL divergence can be decomposed to

$$KL\big(q_\phi(z \mid x) \,||\, p_\theta(z \mid x)\big) = \mathbb{E}_{q_\phi(z|x)}[\log q_\phi(z \mid x)] + \log p_\theta(x) - \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x, z)] \tag{17}$$

and hence

Proof.

Starting with the definition of KL divergence:

$$KL(q_\phi(z \mid x) \,||\, p_\theta(z \mid x)) = \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{q_\phi(z \mid x)}{p_\theta(z \mid x)}\right] \tag{18}$$

$$= \mathbb{E}_{q_\phi(z|x)}[\log q_\phi(z \mid x)] - \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(z \mid x)] \tag{19}$$

By Bayes' rule, we know that

$$p_\theta(z \mid x) = \frac{p_\theta(x, z)}{p_\theta(x)} \tag{20}$$

Substituting this into our equation gives

$$KL(q_\phi(z \mid x) \mid\mid p_\theta(z \mid x)) = \mathbb{E}_{q_\phi(z|x)}[\log q_\phi(z \mid x)] - \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{p_\theta(x)}\right] \tag{21}$$

$$= \mathbb{E}_{q_\phi(z|x)}[\log q_\phi(z \mid x)] - \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x,z)] + \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x)] \tag{22}$$

Since $\log p_\theta(x)$ is constant with respect to z, we can take it out of the expectation.

$$\mathbb{E}_{q_\phi(z|x)}[\log q_\phi(z \mid x)] - \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x,z)] + \log p_\theta(x) \tag{23}$$

Therefore maximizing the log-likelihood is equivalent to minimizing the KL-divergence.

$$\log p_\theta(x) = KL\big(q_\phi(z \mid x) \mid\mid p_\theta(z \mid x)\big) + \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x,z)] - \mathbb{E}_{q_\phi(z|x)}[\log q_\phi(z \mid x)] \tag{24}$$

But again the KL divergence part is intractable due to $p_\theta(z \mid x)$ being intractable. Using the fact that the KL divergence is always greater than or equal to 0, we can drop the term and set a lower bound on the log likelihoods. This lower bound is called the *variational lower bound*.

$$\sum_{i=1}^N \log p_\theta(x^{(i)}) \geq \sum_{i=1}^N \mathbb{E}_{q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}, z)] - \sum_{i=1}^N \mathbb{E}_{q_\phi(z|x^{(i)})}[\log q_\phi(z \mid x^{(i)})] \tag{25}$$

Definition 4.1 (Variational Lower Bound)

The **variational lower bound** of the dataset $\mathcal{D}$ is defined

$$\text{ELBO}(\mathcal{D}, \phi, \theta) = \sum_{i=1}^N \mathbb{E}_{q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}, z)] - \sum_{i=1}^N \mathbb{E}_{q_\phi(z|x^{(i)})}[\log q_\phi(z \mid x^{(i)})] \tag{26}$$

which can be decomposed into the sums of the variational lower bounds of the individual data points.

$$\text{ELBO}(\mathcal{D}, \phi, \theta) = \sum_i \text{ELBO}(x^{(i)}, \phi, \theta) \tag{27}$$

where

$$\text{ELBO}(x^{(i)}, \phi, \theta) = \mathbb{E}_{q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}, z)] - \mathbb{E}_{q_\phi(z|x^{(i)})}[\log q_\phi(z \mid x^{(i)})] \tag{28}$$

Note that we can alternatively define ELBO using Jensen's inequality.

Definition 4.2 (Evidence Lower Bound)

To lower bound it, we can use Jensen's inequality[a] with the concave function $f(x) = \log(x)$ over domain $\mathbb{R}^+$ and the following holds true for all θ and more importantly, for *any arbitrary density function* $q(z)$. Therefore, we have

$$\ell(\theta) = \log p_\theta(x) \tag{29}$$

$$= \log \int p_\theta(x,z)\, dz \tag{30}$$

$$= \log \int q_\phi(z) \frac{p_\theta(x,z)}{q_\phi(z)}\, dz \tag{31}$$

$$\geq \int q_\phi(z \mid x) \log\left(\frac{p_\theta(x,z)}{q_\phi(z)}\right) dz \tag{32}$$

$$= \text{ELBO}(x, q_\phi) \tag{33}$$

The lower bound is called the **evidence lower bound (ELBO)**, and the ELBO of the whole dataset is

$$\text{ELBO}(\mathcal{D}, \phi, \theta) = \sum_{i=1}^{N} \text{ELBO}(x^{(i)}, \phi, \theta) \tag{34}$$

[a]Given convex function $f : \mathbb{R} \to \mathbb{R}$, and random variable X, $\mathbb{E}[f(x)] \geq f(\mathbb{E}[X])$.

Note that this lower bound is with respect to *any* distribution q_ϕ, and it is because of this flexibility that we choose q_ϕ in the first place. Therefore, we can vary ϕ in hopes that the lower bound is maximized, and optimize with respect to this, hence the name *variational*. For more interpretability, look at the corollary.

Corollary 4.1 (Decomposition of ELBO)

The following decomposition of ELBO shows that maximizing the ELBO simultaneously attempts to keep q_ϕ close to p and concentrate $q_\phi(z \mid x)$ on those z that maximizes $\ln p_\theta(x \mid z)$. That is, the approximate posterior q_ϕ balances between staying close to the prior $p(z)$ and moving towards the maximum likelihood $\text{argmax}_z \ln p_\theta(x \mid z)$.

$$\text{ELBO}(x^{(i)}, \phi, \theta) = \underbrace{\mathbb{E}_{q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)} \mid z)]}_{\substack{\text{likelihood term} \\ \text{(reconstruction part)}}} - \underbrace{KL(q_\phi(z \mid x^{(i)}) \mid\mid p(z))}_{\substack{\text{closeness of encoding to } p(z) \\ \text{(typically Gaussian)}}} \tag{35}$$

Note the first expression is the likelihood term, which measures the reconstruction quality of the decoder $p_\theta(x^{(i)} \mid z)$ averaged over encodings sampled from $q_\phi(z \mid x^{(i)})$. The second term is the KL divergence between the encoder distribution $q_\phi(z \mid x^{(i)})$ and the prior $p(z)$, which acts as a regularizer by ensuring the encoded distributions remain close to the chosen prior, typically a standard normal distribution.

Proof.

Starting with the ELBO for a single data point:

$$\text{ELBO}(x^{(i)}, \phi, \theta) = \mathbb{E}_{q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)}, z)] - \mathbb{E}_{q_\phi(z|x^{(i)})}[\log q_\phi(z \mid x^{(i)})]$$

Using the chain rule of probability for the joint distribution:

$$p_\theta(x^{(i)}, z) = p_\theta(x^{(i)} \mid z) p(z)$$

Substituting this into our ELBO:

$$\begin{aligned}
\text{ELBO}(x^{(i)}, \phi, \theta) &= \mathbb{E}_{q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)} \mid z) + \log p(z)] - \mathbb{E}_{q_\phi(z|x^{(i)})}[\log q_\phi(z \mid x^{(i)})] \\
&= \mathbb{E}_{q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)} \mid z)] + \mathbb{E}_{q_\phi(z|x^{(i)})}[\log p(z)] - \mathbb{E}_{q_\phi(z|x^{(i)})}[\log q_\phi(z \mid x^{(i)})] \\
&= \mathbb{E}_{q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)} \mid z)] - \Big(\mathbb{E}_{q_\phi(z|x^{(i)})}[\log q_\phi(z \mid x^{(i)})] - \mathbb{E}_{q_\phi(z|x^{(i)})}[\log p(z)]\Big) \\
&= \underbrace{\mathbb{E}_{q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)} \mid z)]}_{\text{reconstruction term}} - \underbrace{KL(q_\phi(z \mid x^{(i)}) \mid\mid p(z))}_{\text{KL divergence term}}
\end{aligned}$$

Therefore, maximizing the ELBO will simultaneously allow us to obtain an accurate generative model $p_\theta(x \mid z) \approx p^*(x \mid z)$ and an accurate discriminative model $q_\phi(z \mid x) \approx p_\theta(z \mid x)$. The next step is to actually maximize the ELBO with respect to both θ and ϕ. To do this we need to compute the derivatives of ELBO w.r.t. to ϕ and θ.

$$\max_{\phi, \theta} \text{ELBO}(\mathcal{D}, \phi, \theta) \tag{36}$$

It turns out that this itself is a nonconvex optimization problem, and to make it doable we iterate between updating ϕ and θ. Remember that the ELBO is really an expectation, i.e. an integral, and to get a good estimate of its derivative we must try to change it from the derivative of an expectation to the expectation of a derivative. The gradient with respect to θ is very easy since from measure theory, we are deriving and integrating over different variables.

Lemma 4.1 (Gradient of ELBO w.r.t. θ)

For θ, its unbiased gradient is

$$\nabla_\theta \operatorname{ELBO}(x, \theta, \phi) = \mathbb{E}_{q_\phi(z|x)}\big[\nabla_\theta \log p_\theta(x \mid z)\big] \tag{37}$$

and therefore we can approximate the gradient by sampling L points $p^{(1)}, \dots, p^{(L)}$ from $p(z)$ and computing the gradient of the log (since we know the closed form of the conditional distribution given z), and finally averaging them.

$$\nabla_\theta \operatorname{ELBO}(x, \theta, \phi) \approx \frac{1}{L} \sum_{l=1}^{L} \nabla_\theta \log p_\theta(x \mid z^{(l)}) \tag{38}$$

which is guaranteed to converge by the law of large numbers, and furthermore, we can do this for any batch size L.

Proof.

Note that the KL divergence does not depend on θ and neither does the prior, so they can be removed

$$\begin{aligned}
\nabla_\theta \operatorname{ELBO}(x, \theta, \phi) &= \nabla_\theta \big\{ \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x, z)] - \mathbb{E}_{q_\phi(z|x)}[\log q_\phi(z \mid x)] \big\} && (39)\\
&= \nabla_\theta \big\{ \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x, z)] \big\} && (40)\\
&= \mathbb{E}_{q_\phi(z|x)}\big[\nabla_\theta \{ \log p_\theta(x, z) \big] && (41)\\
&= \mathbb{E}_{q_\phi(z|x)}\big[\nabla_\theta \{ \log p_\theta(x \mid z) - \log p_(z) \} \big] && (42)\\
&= \mathbb{E}_{q_\phi(z|x)}\big[\nabla_\theta \log p_\theta(x \mid z) \big] && (43)
\end{aligned}$$

However, taking the gradient w.r.t. ϕ is more complicated since we cannot put the gradient in the expectation, i.e. swap the derivative and integral (since we are deriving and integrating w.r.t. ϕ). Fortunately, we have a well-known mathematical identity often used in policy gradient algorithms in reinforcement learning. [Wil92]

Lemma 4.2 (Log-Derivative Trick)

The following identity holds.

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] = \mathbb{E}_{q_\phi(z)}[f(z) \nabla_\phi \log q_\phi(z)] \tag{44}$$

Proof.

First, let's write out the left-hand side using the definition of expectation:

$$\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] = \nabla_\phi \int f(z) q_\phi(z) dz$$

Under suitable regularity conditions, we can exchange the gradient and integral operators:

$$= \int f(z) \nabla_\phi q_\phi(z) dz$$

Now, we multiply and divide by $q_\phi(z)$ inside the integral:

$$= \int f(z) q_\phi(z) \frac{\nabla_\phi q_\phi(z)}{q_\phi(z)} dz$$

Recognize that $\nabla_\phi \log q_\phi(z) = \frac{\nabla_\phi q_\phi(z)}{q_\phi(z)}$ by the chain rule:

$$= \int f(z) q_\phi(z) \nabla_\phi \log q_\phi(z) dz$$

Finally, we can rewrite this back as an expectation:

$$= \mathbb{E}_{q_\phi(z)}[f(z) \nabla_\phi \log q_\phi(z)]$$

Example 4.1 (Gradient of Expection of $f(x) = x^2$ w.r.t. Gaussian)

Assume we have a normal distribution q that is parameterized by ϕ, specifically $q_\phi(x) = N(\phi, 1)$. We want to solve the below problem

$$\min_\phi \mathbb{E}_q[x^2] \tag{45}$$

This is of course a rather silly problem and the optimal $\phi = 0$ is obvious. One way to calculate $\nabla_\phi \mathbb{E}[x^2]$ is using the log-derivative trick as follows

$$\begin{aligned}
\nabla_\phi \mathbb{E}_q[x^2] &= \nabla_\phi \int q_\phi(x) x^2 dx && (46)\\
&= \int x^2 \nabla_\phi q_\phi(x) \frac{q_\phi(x)}{q_\phi(x)} dx && (47)\\
&= \int q_\phi(x) \nabla_\phi \log q_\phi(x) x^2 dx && (48)\\
&= \mathbb{E}_q[x^2 \nabla_\phi \log q_\phi(x)] && (49)
\end{aligned}$$

For our example where $q_\phi(x) = N(\phi, 1)$, this method gives

$$\nabla_\phi \mathbb{E}[x^2] = \mathbb{E}_q[x^2(x - \phi)] \tag{50}$$

Using this on the gradient of ELBO w.r.t. ϕ gives the following form as the expectation of the gradient.

Lemma 4.3 ()

We can use the score function estimator.

$$\begin{aligned}
\nabla_\phi \text{ELBO}(x, \theta, \phi) &= \nabla_\phi \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x, z) - \log q_\phi(z|x)] && (51)\\
&= \mathbb{E}_{q_\phi(z|x)}\left[\nabla_\phi \left\{ \log q_\phi(z|x)(\log p_\theta(x, z) - \log q_\phi(z|x)) \right\}\right] && (52)
\end{aligned}$$

Proof.

However, REINFORCE is known to have high variance, and so we need large batch sizes L for good convergence. Many methods such as [GBB01, PBJ12] were developed to reduce this. Later it was shown in [KW22] that the *reparamaterization trick* beat everything else, allowing us to efficiently train neural-net-based nonlinear latent variable models, e.g. the variational autoencoder. We will focus on the reparameterization trick in my deep learning notes and omit it here. Now that we have approximate closed form solutions for the

gradients, we can optimize the two using coordinate ascent. Note that we have shown this for a single sample x, and ideally we would do this for a minibatch of samples $x^{(i)}$.

Algorithm 4.1 (Coordinate Ascent Variational Inference)

A common approach to maximize the ELBO is coordinate ascent, where we alternatively optimize with respect to ϕ and θ:

Algorithm 1 Coordinate Ascent Variational Inference (CAVI) with Reparameterization

Require: Initial parameters $\theta^{[0]}$, $\phi^{[0]}$, batch size B, number of samples L

while not converged **do**
  // E-step: optimize variational parameters
  Sample minibatch $\{x^{(1)}, \dots, x^{(B)}\}$ from dataset $\mathcal{D}$
  Sample noise $\{\epsilon^{(1)}, \dots, \epsilon^{(L)}\} \sim p(\epsilon)$ for reparameterization
  Transform noise to latent variables: $z^{(l)} = g_{\phi^{[t]}}(\epsilon^{(l)}, x)$ for $l = 1, \dots, L$
  // Approximate gradient using Monte Carlo samples
  $\hat{g}_\phi \leftarrow \frac{1}{BL} \sum_{i=1}^{B} \sum_{l=1}^{L} [\nabla_\phi \log p_{\theta^{[t]}}(x^{(i)} \mid z^{(l)}) - \nabla_\phi \log q_{\phi^{[t]}}(z^{(l)} \mid x^{(i)}) + \nabla_\phi \log p(z^{(l)})]$
  $\phi^{[t+1]} \leftarrow \phi^{[t]} + \eta_\phi \hat{g}_\phi$ ▷Update with learning rate η_ϕ
  // M-step: optimize model parameters
  $\hat{g}_\theta \leftarrow \frac{1}{BL} \sum_{i=1}^{B} \sum_{l=1}^{L} \nabla_\theta \log p_{\theta^{[t]}}(x^{(i)} \mid z^{(l)})$
  $\theta^{[t+1]} \leftarrow \theta^{[t]} + \eta_\theta \hat{g}_\theta$ ▷Update with learning rate η_θ
end while

Once we are done, we have our optimized encoder and decoders p_θ and q_ϕ.

4.2 EM Algorithm

Let's consider a slightly simpler sub-problem where we have covariates $x^{(i)} \sim X$ coming from distribution $p(x)$. We can again add latent random variables Z but rather than being fixed, the prior $p_\theta(z)$ is also parameterized by θ. Therefore, we would like to find

$$\underset{\theta}{\operatorname{argmax}}\, p_\theta(x) = \underset{\theta}{\operatorname{argmax}} \int p_\theta(x \mid z)\, p_\theta(z)\, dz \tag{53}$$

Even though this integral is not tractable, we will assume that $p_\theta(z \mid x)$ can be computed for a given θ. Let's try to redo our algorithm again with computable posterior assumptions. We have a training set $\mathcal{D} = \{x^{(i)}\}_{i=1}^n \in \mathbb{R}^d$, which we assume are generated by some latent distributions $p_\theta(z)$ followed by the generative component $p_\theta(x \mid z)$. Then, we bound the likelihood of each sample $x^{(i)}$ by an ELBO that varies for all distributions $q^{(i)}$ (we write q rather than q_ϕ since the ϕ will be irrelevant here).

$$\log p_\theta(x^{(i)}) \geq \text{ELBO}(x^{(i)}, q^{(i)}, \theta) = \mathbb{E}_{q^{(i)}(z|x^{(i)})}[\log p_\theta(x^{(i)}, z)] - \mathbb{E}_{q^{(i)}(z|x^{(i)})}[\log q^{(i)}(z \mid x^{(i)})] \tag{54}$$

Summing this all up gives the ELBO of our dataset, which is a lower bound for *all* collections of distributions $q^{(1)}, \dots, q^{(n)}$.

$$\sum_{i=1}^{N} \log p_\theta(x^{(i)}) \geq \text{ELBO}(\mathcal{D}, q^{(1)}, \dots, q^{(n)}, \theta) \tag{55}$$

$$= \sum_{i=1}^{N} \mathbb{E}_{q^{(i)}(z|x^{(i)})}[\log p_\theta(x^{(i)}, z)] - \sum_{i=1}^{N} \mathbb{E}_{q^{(i)}(z|x^{(i)})}[\log q^{(i)}(z \mid x^{(i)})] \tag{56}$$

We maximized the ELBO w.r.t. q and θ by using CAVI, but by invoking our assumption that the posterior $p_\theta(z \mid x)$ can be computed, we can immediately find a maximum.

Theorem 4.2 (Posterior Maximizes ELBO)

When we set $q^(i)(z \mid x) = p(z \mid x^{(i)})$, equality is achieved.

$$\sum_{i=1}^{N} \log p_\theta(x^{(i)}) = \text{ELBO}(\mathcal{D}, q^{(1)}, \dots, q^{(n)}, \theta) \tag{57}$$

$$= \sum_{i=1}^{N} \mathbb{E}_{q^{(i)}(z|x^{(i)})}[\log p_\theta(x^{(i)}, z)] - \sum_{i=1}^{N} \mathbb{E}_{q^{(i)}(z|x^{(i)})}[\log q^{(i)}(z \mid x^{(i)})] \tag{58}$$

Proof.

Let's start by examining the gap between $\log p_\theta(x^{(i)})$ and the ELBO. From our previous derivations, this gap is the KL divergence:

$$\log p_\theta(x^{(i)}) - \text{ELBO}(x^{(i)}, q^{(i)}, \theta) = KL(q^{(i)}(z|x^{(i)}) \| p_\theta(z|x^{(i)})) \tag{59}$$

$$= \mathbb{E}_{q^{(i)}}[\log q^{(i)}(z|x^{(i)}) - \log p_\theta(z|x^{(i)})] \tag{60}$$

When we set $q^{(i)}(z|x^{(i)}) = p_\theta(z|x^{(i)})$:

$$KL(p_\theta(z|x^{(i)}) \| p_\theta(z|x^{(i)})) = \mathbb{E}_{p_\theta}[\log p_\theta(z|x^{(i)}) - \log p_\theta(z|x^{(i)})] \tag{61}$$

$$= \mathbb{E}_{p_\theta}[0] = 0 \tag{62}$$

Therefore, when summing over all samples:

$$\sum_{i=1}^{N} \log p_\theta(x^{(i)}) - \text{ELBO}(\mathcal{D}, q^{(1)}, \dots, q^{(n)}, \theta) = \sum_{i=1}^{N} KL(q^{(i)}(z|x^{(i)}) \| p_\theta(z|x^{(i)})) = 0 \tag{63}$$

Therefore, our CAVI algorithm has been decomposed into the following.

1. E-step. Maximizing ELBO over the variational parameters q_ϕ is really just setting all the $q^{(i)}$ to the posteriors. Note that this is with respect to a fixed θ only.
2. M-step. Maximizing ELBO over the model parameters θ with fixed q is the same by taking the gradient w.r.t. θ which is easy.

This results in the following algorithm.

Algorithm 4.2 (EM Algorithm)

The EM algorithm is described as such:

1. Initialize θ.
2. *E-Step.* Since $\log p_\theta(x)$ is bounded below for all $q^{(1)}, \dots, q^{(n)}$ as

$$\sum_{i=1}^{N} \log p_\theta(x^{(i)}) \geq \sum_{i=1}^{N} \text{ELBO}(x^{(i)}, q^{(i)}, \theta) \tag{64}$$

setting $q^{(i)}(z|x^{(i)}) = p_\theta(z|x^{(i)})$ for all $i = 1, \dots, N$ achieves equality. Note that this equality only holds for the current fixed value of θ.

3. *M-Step.* We maximize with respect to θ whilst fixing $q^{(i)}$.[a]

$$\theta = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{N} \text{ELBO}(x^{(i)}, q^{(i)}, \theta) \tag{65}$$

$$= \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{N} \mathbb{E}_{q^{(i)}(z|x^{(i)})}[\log p_\theta(x^{(i)}, z)] - \sum_{i=1}^{N} \mathbb{E}_{q^{(i)}(z|x^{(i)})}[\log q^{(i)}(z|x^{(i)})] \tag{66}$$

4. Repeat steps 2 and 3 until convergence. Step 2 brings improvements because changing θ creates a new sum of ELBO functions as a new lower bound.

[a]For specific models like GMM as we will see later, this maximization has closed-form solutions, e.g. ϕ = average of responsibilities $\mu_k =:$ weighted average of points, Σ_k = weighted covariance. For other distributions, this maximum must be found analytically or numerically.

The EM algorithm is a specific instance of ELBO optimization! The additional assumption that EM has is that we can calculate the posterior densities.

Corollary 4.2 (Connection to ELBO)

The EM algorithm can be viewed as coordinate ascent on the ELBO where:
- E-step: Sets $q(z) = p_{\theta^{[t]}}(z|x)$, maximizing ELBO over q
- M-step: Maximizes ELBO over θ with fixed q

Note that there is a duality between the true parameters θ and the latent variables z. If θ is known, then the values of z can be found by maximizing the log-likelihood over all possible values of z. Conversely, if we know the value of the latent variables z, then we can find an estimate of the parameters by grouping the data points into each value of z and optimizing $p_\theta(x \mid z)$, e.g. by averaging the values. This suggests an iterative algorithm in the case where both θ and z are unknown. We assume that we know θ and optimize z, then optimize θ, and so on, similar to k-means clustering.

We can formulate the algorithm alternatively yet equivalently.

Algorithm 4.3 (EM Algorithm)

The **Expectation-Maximization algorithm** optimizes the likelihood above with the following steps.

1. First initialize $\theta = \theta^{[0]}$ in some way.[a]
2. *E-Step.* Define

$$Q(\theta \mid \theta^{[t]}) = \mathbb{E}_{p_\theta(z|x)}[\log p_\theta(x, z)] = \int p_{\theta^{[t]}}(z \mid x) \log p_\theta(x, z)\, dz \tag{67}$$

   as the expected value of the log-likelihood with respect to the current conditional distribution of z, given x and $\theta^{[t]}$.
3. *M-Step.* Find the parameters that maximize this quantity.

$$\theta^{[t+1]} = \underset{\theta}{\operatorname{argmax}}\, Q(\theta \mid \theta^{[t]}] \tag{68}$$

[a]Note that within this θ are the parameterizations of the initial multinomial density p_Z, which is our initial "guess" of the distribution of Z.

Theorem 4.3 (EM Monotonicity)

The EM algorithm monotonically increases the observed data log-likelihood:

$$\log p_{\theta^{[t+1]}}(x] \geq \log p_{\theta^{[t]}}(x) \tag{69}$$

Therefore, though there is no guarantee that this will hit the global maximum, it will hit a local maximum.

Proof.

Let's consider the difference in log-likelihoods between iterations:

$$\log p_{\theta^{[t+1]}}(x) - \log p_{\theta^{[t]}}(x) = \Big[Q(\theta^{[t+1]}|\theta^{[t]}) - H(\theta^{[t+1]}|\theta^{[t]})\Big] \tag{70}$$

$$- \Big[Q(\theta^{[t]}|\theta^{[t]}) - H(\theta^{[t]}|\theta^{[t]})\Big] \tag{71}$$

where $H(\theta|\theta^{[t]}) = \mathbb{E}_{z|x,\theta^{[t]}}[\log p_\theta(z|x)]$. By the M-step, we know $Q(\theta^{[t+1]}|\theta^{[t]}) \geq Q(\theta^{[t]}|\theta^{[t]})$. Also, by Jensen's inequality:

$$H(\theta^{[t+1]}|\theta^{[t]}) \leq H(\theta^{[t]}|\theta^{[t]}) \tag{72}$$

Therefore, the difference is non-negative.

For some intuition, we can visualize l as a function of θ. For the sake of visuals, we will assume that $\theta \in \mathbb{R}$ and $l : \mathbb{R} \longrightarrow \mathbb{R}$. On the contrary to what a visual is supposed to do, we want to point out that we cannot just visualize l as a curve in $\mathbb{R} \times \mathbb{R}$. This can be misleading since then it implies that the optimal θ value is easy to find, as shown in the left. Rather, we have no clue what the whole curve of l looks like, but we can get little snippets (right).

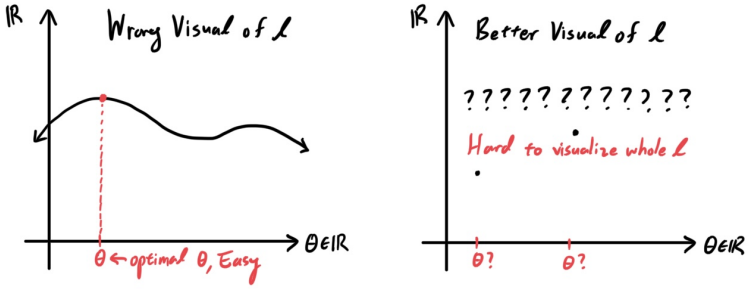


Figure 14

Rather, all we can do is hope to take whatever easier-to-visualize, lower-bound functions and maximize them as much as we can in hopes of converging onto l. Let us walk through the first two iterations of the EM algorithm. We first initialize θ to, say θ_0. This immediately induces the lower-bound ELBO-sum function $\sum_i \text{ELBO}(x^{(i)}; p_Z^{*i}, \theta)$, which takes in multinomial density functions $p_Z^{*i} = p_1, p_2, \ldots$ and outputs different functions of θ that are valid lower bounds. Two of these possible lower-bound functions are shown (in green) for when we input some arbitrary density p_1, p_2. However, there exists a density $p_Z^{(i)}$ that produces not only the maximum possible lower-bound (called max ELBO, shown in red) but is equal to $l(\theta)$ for that density input $p_Z^{(i)}$. We maximize this function with respect to θ to get θ_1 as our next assignment of θ.

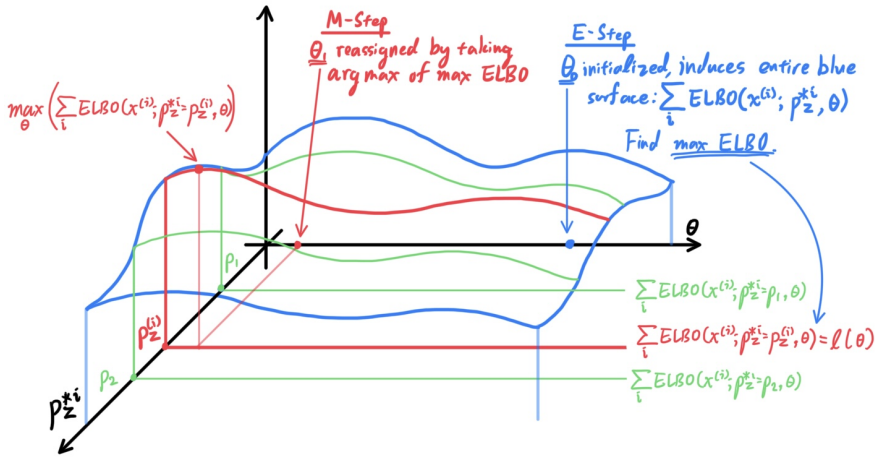


Figure 15

The next step is identical. Now that we have a new value of $\theta = \theta_1$, this induces the lower-bound ELBO-sum function $\sum_i \text{ELBO}(x^{(i)}; p_Z^{*i}, \theta)$ that also takes in multinomial densities p_Z^{*i} and outputs different functions of θ that are valid lower-bounds. Two possible lower bounds are shown (in green), but the maximum lower-bound (in blue) is produced when we input density $p_Z^{(i)}$. Since this max ELBO function is equal to θ for this fixed density input $p_Z^{(i)}$, we maximize this function with respect to θ to get θ_2 as our next assignment of θ.

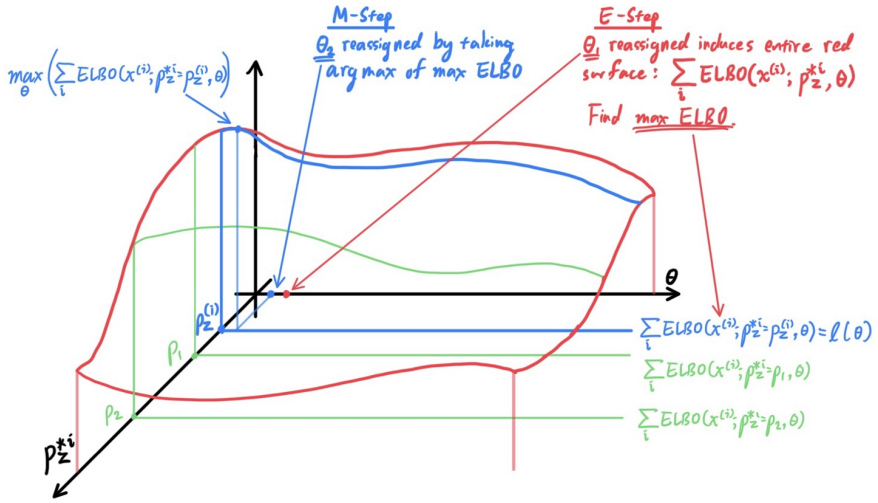


Figure 16

4.3 Gaussian Mixture Models

Given a training set $x^{(i)}{}_{i=1}^{n}$ (without the y-labels and so in the unsupervised setting), there are some cases where it may seem like we can fit multiple Gaussian distributions in the input space $\mathcal{X}$. For example, the points below seem like they can be fitted well with 3 Gaussians.

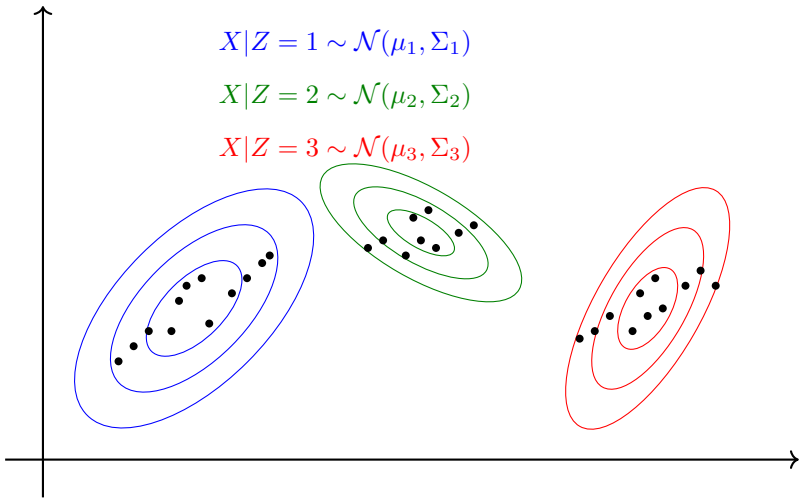


Figure 17: Example of data that can be fitted with 3 Gaussians

Therefore, we can construct a best-fit model as a composition of a multinomial distribution (to decide which one of the Gaussians x should follow) followed by a Gaussian.

Definition 4.3 (Gaussian Mixture Model)

The **Gaussian mixture model (GMM)** assumes that the covariates $x \sim X \in \mathbb{R}^d$ are generated by the following.[a] The parameters are $\theta = \{\lambda, \mu_1, \dots, \mu_k, \Sigma_1, \dots, \Sigma_k\}$.[b]

1. A latent variable $z \sim \text{Multinomial}(\lambda)$, where $\lambda = (\lambda_1, \dots, \lambda_k)$ with PMF defined
$$p_\theta(z) = \lambda_z \tag{73}$$
2. The generative random variable $X \mid Z = z \sim \mathcal{N}(\mu_i, \Sigma_i)$ where $\mu_z \in \mathbb{R}^d, \Sigma_z \in \mathbb{R}^{d \times d}$ and PDF defined
$$p_\theta(x \mid z) = \frac{1}{(2\pi)^{d/2}|\Sigma_z|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_z)^\top \Sigma_z^{-1}(x-\mu_z)\right) \tag{74}$$

[a] Therefore, our model says that each $x^{(i)}$ was generated by randomly choosing $z^{(i)}$ from $1, \dots, k$ according to some multinomial, and then the $x^{(i)}$ was drawn from one of the k Gaussians depending on $z^{(i)}$.

[b] Note that λ really has $k-1$ free parameters and Σ_i's should be symmetric and positive-definite.

We can write down the log-likelihood of the given data $x^{(i)}$'s as a function of all the parameters above as

$$\sum_{i=1}^n \log p_\theta(x^{(i)}) = \sum_{i=1}^n \log\left(\sum_{z=1}^k p_\theta(x^{(i)} \mid z^{(i)}), p_\theta(z^{(i)})\right) \tag{75}$$

Example 4.2 (Dual Nature of Latents and Parameters)

Note that since we only know that the *final* value of the ith sample is $x^{(i)}$ and not anything at all about which value $z^{(i)}$ the ith sample had, there is an extra unknown in this model. If we did know the values of the hidden variables $z^{(i)}$ (i.e. if we knew which of the k Gaussians each $x^{(i)}$ was generated from), then our log likelihood function would be much more simple since now, our givens will be both $x^{(i)}$ *and* $z^{(i)}$. Therefore, we don't have to condition on the $z^{(i)}$ and can directly calculate the log of the probability of us having sample values $(z^{(1)}, x^{(1)}), (z^{(2)}, x^{(2)}), \dots, (z^{(n)}, x^{(n)})$.

$$\sum_{i=1}^{n} \log p(x^{(i)}) = \sum_{i=1}^{n} \log p(x^{(i)}, z^{(i)}) = \sum_{i=1}^{n} \log p(x^{(i)} \mid z^{(i)})\, p(z^{(i)}) \tag{76}$$

This model, with known $z^{(i)}$'s, is basically the GDA model, which is easy to calculate. That is, the maximum values of ϕ, μ, Σ are

$$\begin{aligned}
\phi_j &= \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{z^{(i)}=j} \\
\mu_j &= \frac{\sum_{i=1}^{n} \mathbb{1}_{z^{(i)}=j} x^{(i)}}{\sum_{i=1}^{n} \mathbb{1}_{z^{(i)}=j}} \\
\Sigma_j &= \frac{1}{\sum_{i=1}^{n} \mathbb{1}_{z^{(i)}=j}} \sum_{i=1}^{n} \mathbb{1}_{z^{(i)}} \left(x^{(i)} - \mu_j\right), \left(x^{(i)} - \mu_j\right)^T
\end{aligned}$$

But since we do *not* know the values of $z^{(i)}$, we first try to "guess" the values of the $z^{(i)}$'s and then update the parameters of our model assuming our guesses are correct.

Algorithm 4.4 (EM Algorithm on GMMs)

The EM Algorithm applied to GMMs has the following steps:

1. Randomly initialize $\theta^{[0]} = \{\lambda, \mu_1, \dots, \mu_k, \Sigma_1, \dots, \Sigma_k\}$.[a]
2. **(E Step)** Calculate the posterior density $p(z \mid x)$ by applying Bayes rule to each sample keeping the parameter $\theta^{[t]}$ fixed.

$$p_{\theta^{[t]}}(z \mid x^{(i)}) = \frac{p_{\theta^{[t]}}(x^{(i)} \mid z)\, p_{\theta^{[t]}}(z)}{p(x)} = \frac{p_{\theta^{[t]}}(x^{(i)} \mid z)\, p_{\theta^{[t]}}(z)}{\sum_z p_{\theta^{[t]}}(x^{(i)} \mid z)\, p_{\theta^{[t]}}(z)} \tag{77}$$

   We should have n different multinomial distribution parameters, each representing our best guess of what multinomial density $p(z \mid x^{(i)})$ each $x^{(i)}$ had followed in order to be at the given points. Let's label the updated parameters of the multinomial distribution of the ith sample to be $\lambda^{[t](i)}$ at the tth iteration.
3. **(M Step)** We update θ as such.

$$\lambda^{[t+1]} = \frac{1}{n} \sum_{i=1}^{n} \lambda^{[t](i)} \tag{78}$$

$$\mu_j^{[t+1]} = \frac{\sum_{i=1}^{n} \lambda_j^{[t](i)} x^{(i)}}{\sum_{i=1}^{n} \lambda^{[t](i)}} \tag{79}$$

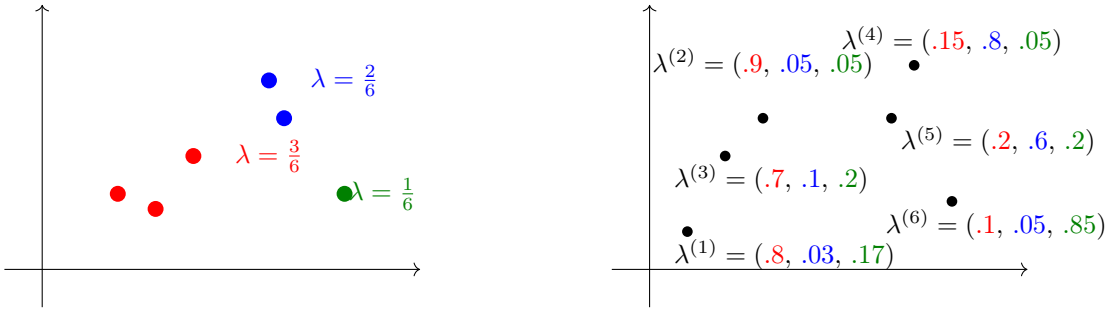
$$\Sigma_j^{[t+1]} = \frac{1}{\sum_{i=1}^{n} \lambda^{[t](i)}} \sum_{i=1}^{n} \lambda_j^{[t](i)} \left(x^{(i)} - \mu_j^{[t+1]}\right] \left(x^{(i)} - \mu_j^{[t+1]}\right]^T \tag{80}$$

4. Repeat steps 2 and 3 until convergence.

[a] This might converge faster using K-means initialization.

Let us elaborate further on the intuition of this step. In the normal GDA with given values of $z^{(i)}$, we have $\lambda = \frac{1}{n} \sum_{i=1}^{n} 1\{z^{(i)} = j\} = \frac{1}{n}$(Number of Samples in jth Gaussian), which is a sum of "hard" guesses, meaning that each $x^{(i)}$ is undoubtedly in cluster j or not, and so to find out our best guess for the true vector λ, all we have to do is find out the proportion of all examples in each of the k groups and we're done (without needing to iterate). However, in our EM model, we do not know the $z^{(i)}$'s, and so the best we can do is give the *probability* $\lambda_j^{(i)}$ that $x^{(i)}$ is in cluster j. So for each point $x^{(i)}$, the model has changed from it

being undoubtedly in group $z^{(i)} = j$ to it having a probability of being in $\lambda_j^{(i)}$ for $j = 1, \dots, k$.



(a) Hard label assignments.

(b) Soft probability assignments.

Figure 18: The superscript $[t]$ is omitted for clarity.

When we update the λ in the M-step, we can interpret the vectors $\lambda^{(i)}$ as tuples where $\lambda_j^{(i)}$ describes the expected "portion" of each sample $x^{(i)}$ to be in group j. So, we are adding up all the "portions" of the points that are expected to be in cluster j to get $\lambda = \sum_{i=1}^{n} \lambda^{(i)}$.

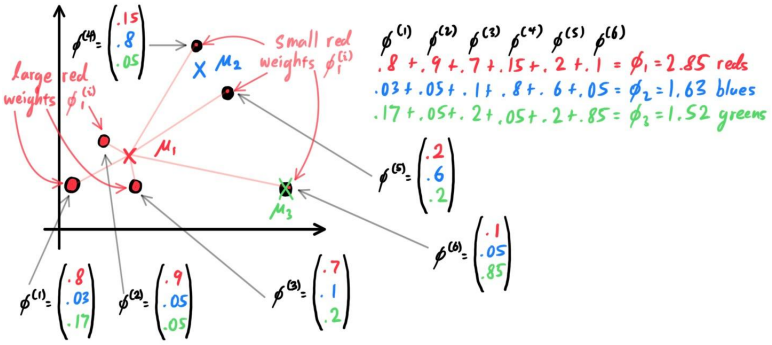


Figure 19

Now, given the jth Gaussian cluster, we would like to compute its mean μ_j. Since each $x^{(i)}$ has probability $\lambda_j^{(i)}$ of being in cluster j, we can weigh each of the n points by $\lambda_j^{(i)}$ (which determines how "relevant" $x^{(i)}$ is to cluster j) and average these (already weighted) points to get our "best-guess" of the mean μ_j. Given the MLE of the means, we can straightforwardly compute the MLE of the covariance matrices.

In summary, this entire algorithm results from modifying the "hard" data of each point $x^{(i)}$ being undoubtedly in one cluster to a model containing points $x^{(i)}$ that have been "smeared" around different clusters, with a probability $\lambda^{(i)}$ being in cluster j.

4.4 Nonlinear ICA

Bibliography

[GBB01] Evan Greensmith, Peter Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.

[KW22] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2022.

[PBJ12] John Paisley, David Blei, and Michael Jordan. Variational bayesian inference with stochastic search, 2012.

[Wil92] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.