DD2434/FDD3434 Machine Learning, Advanced Course Exercises Solutions (DGM, HMM, EM & VI)

Negar Safinianaini, Hazal Koptagel, Antonio Matosevic. Revised by Harald Melin and Semih Kur

November 2022

Abstract

This document consists of the theory and exercises for DD2434/FDD3434 Machine Learning Advanced Course, covering DGM, HMM, EM, and VI. Parts of the exercises are from year 2018 by Borja Rodriguez Galvez and Elin Samuelsson.

*KTH Royal Institute of Technology, Stockholm, Sweden

Contents

1 Directed Graphical Models (DGM) – Theory

1.1 D-separation rules

If two nodes are d-separated, they are independent. In Figure 1, S and R are independent from each other.

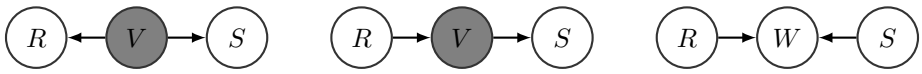


Figure 1: S and R are d-separated. Note that W is not observed and no descendants of W is observed either. Structures left to right: fork, chain, v-structure.

[Pearl 1988] example:
d-separated: dead battery → car won't start ← no gas
not d-separated: dead battery → **car won't start** ← no gas

A charged battery does not give information about the gas, but, a charge battery become informative (dependency) after observing that the car won't start (because then we know that gas was empty from knowing the battery was full and car didn't start)

A note regarding how to read conditional independence queries: If no conditioning is mentioned in the query of independence/dependence, then assume that all the shaded variables are given or conditioned on. But, if there is a specific conditioning in the query, only consider that, and, not all the shaded variables in the graphical model.

Two methods to determine if variables are independent or not:

1.1.1 Method 1

This method is efficient when examining the dependence over the whole graph. We put a block symbol on the graph wherever we detect the d-separation using the rules in Figure 1. Then we can answer whether variables are d-separated or not by following the path between them i.e. if the path contains a block they are obviously d-separated. **Example** The blocks are illustrated as red point over the graph in Figure 2; the table shows the answers to d-separation which is gained by following the path between each pair of i and j.

i, j	d-separated given C
A, D	yes
A, B	no
D, G	yes
D, H	no
D, E	yes
D, F	no
B, H	yes

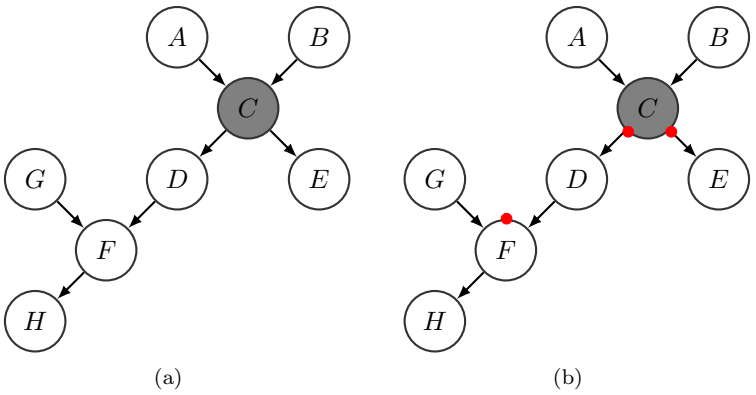


Figure 2: Blocks are shown in red on the graph in (b).

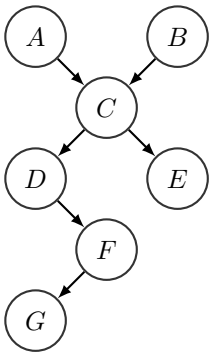
1.1.2 Method 2

This method is efficient when the dependencies are being examined among certain variables and not all the variables in the graph. We follow this procedure [3]:

- 1. Construct the "ancestral graph" of all variables mentioned in the probability expression. This is a reduced version of the original net, consisting only of the variables **mentioned** and all of their **ancestors** (parents, parents' parents, etc.)
- 2. "Moralize" the ancestral graph by "marrying" the parents. For each pair of variables with a common child, draw an undirected edge (line) between them. (If a variable has more than two parents, draw lines between every pair of parents.)
- 3. "Disorient" the graph by replacing the directed edges (arrows) with undirected edges (lines).
- 4. Delete the givens **(not the actual variables that we intend to judge their dependency - in some scenario the variables which are to be judged about their dependency, are also observed, and, them being observed does not mean that they will be deleted.)** and their edges. If the independence question had any given variables, erase those variables from the graph and erase all of their connections, too.
- 5. Read the answer off the graph.
  - If the variables are disconnected in this graph, they are guaranteed to be independent.
  - If the variables are connected in this graph, they are not guaranteed to be independent.* Note that "are connected" means "have a path between them," so if we have a path X-Y-Z, X and Z are considered to be connected, even if there's no edge between them.
  - If one or both of the variables are missing (because they were givens, and were therefore deleted), they are independent.

* We can say "the variables are dependent, as far as the Bayes net is concerned" or "the Bayes net does not require the variables to be independent," but we cannot guarantee dependency using d-separation alone, because the variables can still be numerically independent (e.g. if $P(A|B)$ and $P(A)$ happen to be equal for all values of A and B).

Example Q: Are A and B conditionally independent, given D and F?



A: They are not required to be conditionally independent, using method 2 as below:

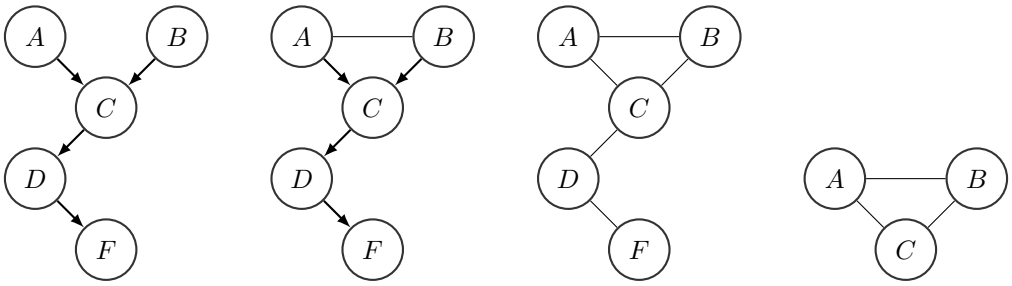
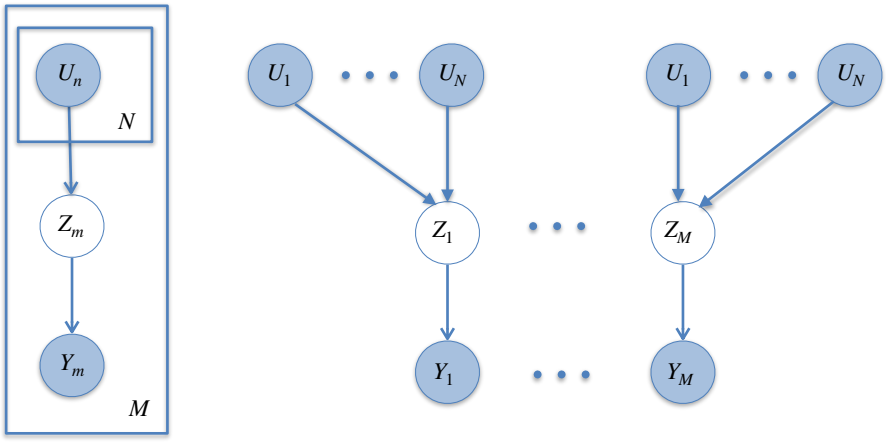


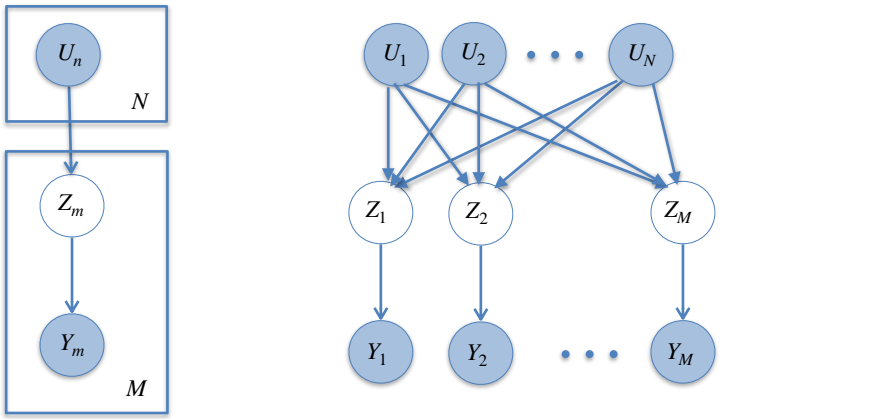
Figure 3: From left to right: 1. Draw ancestral graph 2. Moralize 3. Disorient 4. Delete givens

1.2 Plate notation

In plate notation, the node/graph is replicated N times (when N is written in the plate); moreover, any edge that crosses a plate boundary is replicated once for each subgraph repetition. The plate notation is used to illustrate a compact representation of the graphical model. Using such a simplified notation allows to have an overview of model parameters and variables. However, it is sometimes useful to unfold the plate, e.g., when evaluating the conditional dependencies. In Fig. 4, two examples of unfolding a plate graph are illustrated.



The plate graph The corresponding unfolded graph



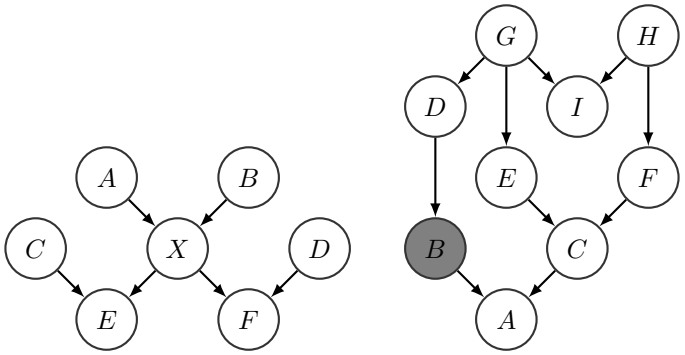
The plate graph The corresponding unfolded graph

Figure 4: Two examples of unfolding a plate DGM.

2 Directed Graphical Models (DGM) – Exercises

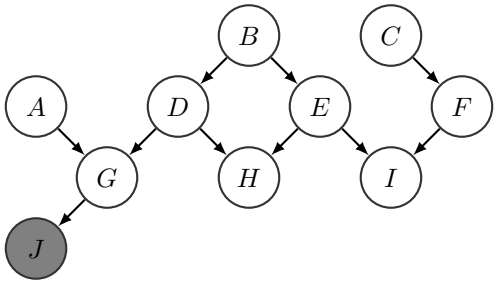
2.1 Bayes Ball

Question: List all variables that are independent of A given evidence on the shaded node for each of the DGMs a), b) and c) below.



(a)

(b)



(c)

Figure 5: Some DGMs.

2.2 Bayes nets for a rainy day (Exercise 10.5 from Murphy [4])

Question: (Source: Nando de Freitas) In this question you must model a problem with 4 binary variables: G = "$gray$", V = "$Vancouver$", R = "$rain$" and S = "sad". Consider the directed graphical model describing the relationship between these variables shown in Figure 6 (and the probability tables shown in Table 1).

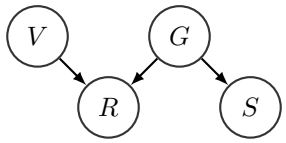


Figure 6: Bayesian net for a rainy day

Table 1: Probability tables of Bayes net for a rainy day

$V=0$	$V=1$
δ	$1-\delta$

$G=0$	$G=1$
α	$1-\alpha$

	$S=0$	$S=1$
$G=0$	γ	$1-\gamma$
$G=1$	β	$1-\beta$

	$R=0$	$R=1$
$VG=00$	0.6	0.4
$VG=01$	0.3	0.7
$VG=10$	0.2	0.8
$VG=11$	0.1	0.9

a. Write down an expression for $P(S=1|V=1)$ in terms of $\alpha, \beta, \gamma, \delta$.

b. Write down an expression for $P(S=1|V=0)$. Is this the same or different to $P(S=1|V=1)$? Explain why.

c. Find maximum likelihood estimates of α, β, γ using the following data set, where each row is a training case. (You may state your answers without proof.)

V	G	R	S
1	1	1	1
1	1	0	1
1	0	0	0

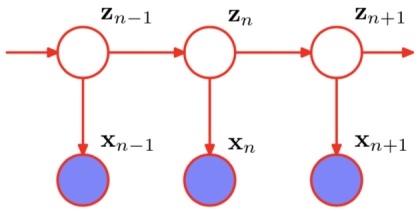


Figure 7: An HMM unfolded across the sequence (or time sometimes), i.e., trellis diagram.
from Bishop [1]

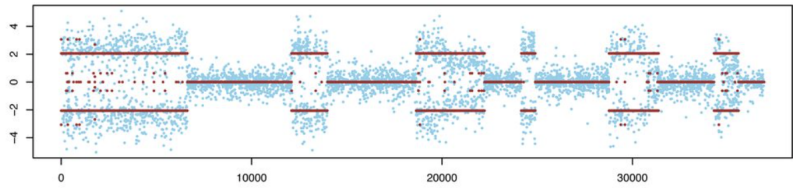


Figure 8: An example of HMM with continuous data (observable R.V.)
accessed at https://www.biorxiv.org/content/10.1101/410845v1.full

3 Hidden Markov Models (HMM) – Theory

A Hidden Markov Model (HMM), illustrated in Fig. 3, is a simple and rich probabilistic graphical model, widely used for sequential or temporal real world data analysis, see example of biological application in Fig. 8. In an HMM, a sequence of symbols/observable R.V.'s (observation sequence colored in blue in the figure) is emitted/generated from a sequence of states (state sequence). The state sequence comprises of hidden discrete variables and the sequence follows a Markov structure, i.e., the next state in the state sequence only depends on the current state. For a finite HMM, we assume that the input sequences have length N. The observation sequence is denoted as $X = \{x_1, \ldots, x_N\}$ and the state sequence is denoted as $Z = \{z_1, \ldots, z_N\}$. Each state in Z takes a value j for j = 1,...,J. The value of each element in X can be discrete, continuous, or multi dimensional. An HMM is parameterized by initial probabilities, $p(z_1)$, transition probabilities, $p(z_n|z_{n-1})$, and emission probabilities, $p(x_n|z_n)$. We term these as θ. These are short notations, and, the proper one for transition probabilities is for instance: $p(z_n = j|z_{n-1} = i)$, i.e., the probability of moving from state i to j where i and j take values in $\{1, ..., J\}$. The emission probability is either pdf or pmf of the observable variable, depending if it is continuous or discrete. The transition probabilities form a matrix, called the transition matrix with shape $J \times J$; each row on this matrix forms a probability distribution, hence the probabilities sum to one.
The joint distribution of all of the random variables in the HMM (X's and Z's) factors as the following; the key point in HMMs is the factorization property which makes inference tractable and easy.

$$p(x_{1:N}, z_{1:N}) = p(z_1)p(x_1|z_1) \prod_{n=2}^{N} p(z_n|z_{n-1})p(x_n|z_n)$$

3.1 Marginal distribution of the hidden variables of HMM

For learning and inference in HMMs (e.g. EM), we need to calculate marginal distribution of the hidden variables, such as $p(z_n|x_{1:N}, \theta)$. So we assume to know the model parameters and we have observed the data. For readability, we write $p(z_n|x_{1:N})$. We could calculate this by performing marginalization over all the hidden variables, but, that is costly (we would have to calculate N nested sum with each iterating over J possible values of the hidden variable i.e. $O(J^N)$). Instead we use an approach which takes $O(NJ^2)$ number of calculations. We write the probability as $\frac{p(x_{1:N}|z_n)p(z_n)}{p(x_{1:N})}$ due to Bayes, and then, because of conditional independence, we can write $p(x_{1:N}|z_n)$ as $p(x_{1:n}|z_n)p(x_{n+1:N}|z_n)$. We have now split the problem into two problems which are solved by recursion (this whole process is referred to as dynamic programming). We can rewrite the problem, finally, as: $\frac{p(x_{1:n}, z_n)p(x_{n+1:N}|z_n)}{p(x_{1:N})}$ or $\frac{\alpha(z_n)\beta(z_n)}{\sum_{z_N}\alpha(z_N)}$. In the next two sections we calculate the probabilities $\alpha(z_n)$ and $\beta(z_n)$, forward and backward, respectively.
For calculation of the marginal of the two consecutive hidden variables, see [1].

3.1.1 Forward pass

$$
\begin{aligned}
\alpha(z_n) &= p(x_{1:n}, z_n) = p(x_{1:n}|z_n)p(z_n) = p(x_n|z_n)p(x_{1:n-1}|z_n)p(z_n) \\
&= p(x_n|z_n)p(x_{1:n-1}, z_n) \\
&= p(x_n|z_n)\sum_{z_{n-1}} p(x_{1:n-1}, z_n, z_{n-1}) \\
&= p(x_n|z_n)\sum_{z_{n-1}} p(x_{1:n-1}, z_n|z_{n-1})p(z_{n-1}) \\
&= p(x_n|z_n)\sum_{z_{n-1}} p(x_{1:n-1}|z_{n-1})p(z_{n-1})p(z_n|z_{n-1}) \\
&= p(x_n|z_n)\sum_{z_{n-1}} p(x_{1:n-1}, z_{n-1})p(z_n|z_{n-1}) = p(x_n|z_n)\sum_{z_{n-1}} \alpha(z_{n-1})p(z_n|z_{n-1}) \\
&, where \quad \alpha(z_1) = p(x_1|z_1)p(z_1)
\end{aligned}
$$

Note that we have $\alpha(z_1)$ from the initializations (we initialize the initial and emission probabilities, thus we have $\alpha(z_1)$).

3.1.2 Backward pass

$$
\begin{aligned}
\beta(z_n) &= p(x_{n+1:N}|z_n) \\
&= \sum_{z_{n+1}} p(x_{n+1:N}, z_{n+1}|z_n) \\
&= \sum_{z_{n+1}} p(x_{n+1:N}|z_{n+1}, z_n)p(z_{n+1}|z_n) \\
&= \sum_{z_{n+1}} p(x_{n+1:N}|z_{n+1})p(z_{n+1}|z_n) \\
&= \sum_{z_{n+1}} p(x_{n+1}|z_{n+1})p(x_{n+2:N}|z_{n+1})p(z_{n+1}|z_n) \\
&= \sum_{z_{n+1}} p(x_{n+1}|z_{n+1})\beta(z_{n+1})p(z_{n+1}|z_n) \\
&, where \quad \beta(z_N) = 1
\end{aligned}
$$

Example of an input-output hidden Markov model. In this case, both the emission probabilities and the transition probabilities depend on the values of a sequence of observations $\mathbf{u}_1, \ldots, \mathbf{u}_N$.

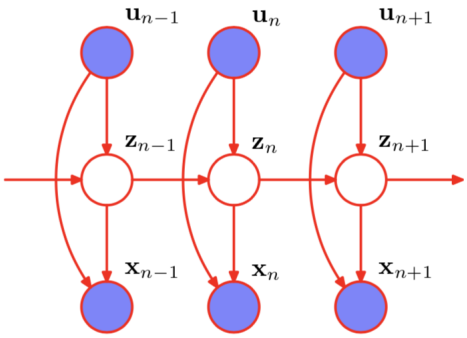


Figure 9: IOHMM
from Bishop [1]

4 Hidden Markov Models (HMM) – Exercises

4.1 Marginal distribution of the hidden variable of IOHMM

Find the marginal distribution of one hidden variable, given the observed input and output (shaded in Fig. 9.), i.e., $p(z_n|x_{1:N}, u_{1:N})$.

5 Expectation-Maximization – Theory

General setup Let $\mathbf{x} = \{x_1, \ldots, x_N\}$ be a set of observations (of corresponding random variables (RVs) $X_1, \ldots, X_N$) and $\mathbf{Z} = \{Z_1, \ldots, Z_K\}$ a set of latent RVs. Oftentimes one constructs a model where $\mathbf{x}$ and $\mathbf{Z}$ are directly connected and parameterises it (i.e. distributions involved) by a set of parameters θ. The goal is then to infer θ given data $\mathbf{x}$.

Maximum likelihood estimation A common tool for estimating distribution parameters is the maximum likelihood estimation (MLE). In its essence it seeks to find the point estimate θ^* of θ so that the probability density of observing exactly this sample $\mathbf{x}$ of $X_1, \ldots, X_n$ is maximized. This probability density is referred to as data likelihood and denoted as $p(\mathbf{x}; \theta)$, which is just uncluttered way of writing $p(X_1 = x_1, \ldots, X_N = x_n; \theta)$, henceforth used for all distributions.

MLE for latent variables models In the context of latent variable models, data likelihood can be written using the law of total probability as

$$p(\mathbf{x}; \theta) = \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}; \theta), \tag{1}$$

for **all** possible combinations $\mathbf{z} = \{z_1, \ldots, z_K\}$ that $Z_1, \ldots, Z_K$ can attain. It is obvious that this sum[1] is computationally exponential, often referred to as intractable[2]. Another issue using MLE when having latent variables is, usually, facing unsolvable equations after setting the derivatives of the likelihood function with respect to all the unknown values to zero (a typical approach in finding the optimum). In other words, to solve the resulting equations requires the values of the latent variables; if we substitute one set of equations into the other, we still have too many unknown to solve the equations.

Expectation-maximization algorithm To alleviate the aforementioned problems, one can optimize something close to likelihood, but tractable (this approximation is targeted to result in solvable equations). Let us present one way to do this by maximizing its lower bound instead. Returning to Eq. 1 we can, thanks to monotonicity, equivalently maximize data likelihood in the log-space i.e. we have that

$$\log p(\mathbf{x}; \theta) = \log \sum_{\mathbf{z}} p(\mathbf{x}, \mathbf{z}; \theta). \tag{2}$$

Now, there is a result called Jensen's inequality stating that for a convex function $f(\cdot)$ of a RV Ψ it holds that $f(\mathbb{E}[\Psi]) \geq \mathbb{E}[f(\Psi)]$. Since the logarithm conveniently appearing in Eq. 2 is convex, we can perhaps use this inequality to somehow simplify the initial problem. It only remains to rewrite Eq. 2 into an expectation so that the inequality can be applied. The trick is a common mathematical manipulation of multiplying and dividing by the same thing, essentially not changing the expression, but making it possible to manipulate the expression

[1] Or integral in case of continuous latent RVs.

[2] This terms also applies to integrals without closed form in the continuous case.

in a favourable way. Following this, we can write

$$\log p(\mathbf{x};\theta) = \log \sum_{\mathbf{z}} p(\mathbf{x},\mathbf{z};\theta) \tag{3}$$

$$= \log \sum_{\mathbf{z}} p(\mathbf{x},\mathbf{z};\theta)\frac{q(\mathbf{z})}{q(\mathbf{z})} \tag{4}$$

$$= \log \mathbb{E}_{\mathbf{z}}\left[\frac{p(\mathbf{x},\mathbf{z};\theta)}{q(\mathbf{z})}\right] \tag{5}$$

$$\geq \mathbb{E}_{\mathbf{z}}\left[\log \frac{p(\mathbf{x},\mathbf{z};\theta)}{q(\mathbf{z})}\right] \tag{6}$$

$$= \sum_{\mathbf{z}} q(\mathbf{z}) \log \frac{p(\mathbf{x},\mathbf{z};\theta)}{q(\mathbf{z})}, \tag{7}$$

for any $q(\mathbf{z})$. Now that we have a lower bound to the true log-likelihood, it remains to choose $q(\mathbf{z})$ wisely so that the lower bound is as tight as possible. By rewriting Eq. 7 as

$$\mathcal{L}(q,\theta) = \sum_{\mathbf{z}} q(\mathbf{z}) \log \frac{p(\mathbf{x},\mathbf{z};\theta)}{q(\mathbf{z})} = -\mathbb{KL}(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x};\theta)) + \log p(\mathbf{x};\theta) \tag{8}$$

one can notice (ignoring the second term since it does not depend on $q(\mathbf{z})$) that the inequality becomes equality for $q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x};\theta)$. However, θ parameterising this $q(\mathbf{z})$ is exactly what we wanted to estimate from the very beginning and now we need it to compute the lower bound, which we need to estimate θ, which we need to compute lower bound etc. This cyclic reasoning implies an iterative algorithm that will at one stage (E-step) re-estimate

$$q^t = \arg\max_q \mathcal{L}(q,\theta^t),$$

where θ^t is fixed and comes from the previous iteration[3], and at the next stage (M-step) re-estimate

$$\theta^{t+1} = \arg\max_\theta \mathcal{L}(q^t,\theta),$$

where now q^t is fixed and comes from the current iteration. This algorithm is better known as expectation-maximization (EM), although it is here actually presented as maximization-maximization. *Note:* Now that we have introduced an iterative scheme, we simply have that $q^t = p(\mathbf{z}|\mathbf{x};\theta^t)$, which can somewhat simplify the expression for $\mathcal{L}(q,\theta)$ at the M-step. By plugging q^t in $\mathcal{L}(q,\theta)$, one can easily show that

$$\mathcal{L}(q^t,\theta) = Q(\theta,\theta^t) + c, \tag{9}$$

where $Q(\theta,\theta^t) = \mathbb{E}_{\mathbf{z}|\mathbf{x},\theta^t}[\log p(\mathbf{x},\mathbf{z};\theta)]$, also called expected complete log-likelihood[4], and c is the entropy of q^t which is constant w.r.t. θ[5]. Hence, it is actually enough to maximize only $Q(\theta,\theta^t)$ with respect to θ at the M-step instead of the whole $\mathcal{L}(q^t,\theta)$.

[3] At the first iteration we initialize θ^t.

[4] It is this expectation that motivates the name E-step.

[5] Here it is crucial to distinguish between fixed θ^t from the previous iteration that q^t depends on and θ as a variable parameter to be maximized w.r.t.

The EM algorithm involves alternately computing a lower bound on the log likelihood for the current parameter values and then maximizing this bound to obtain the new parameter values. See the text for a full discussion.

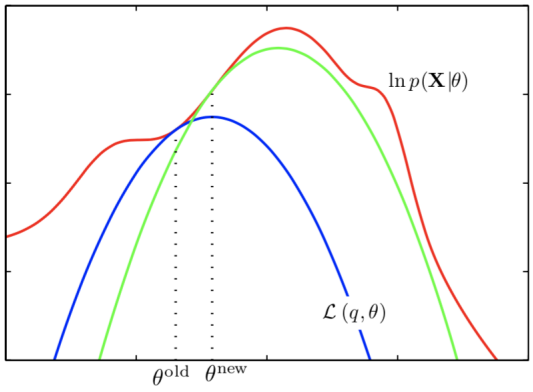


Figure 10: Maximizing the lower bound of likelihood by using EM
from Bishop [1]

The pseudo code of the EM algorithm is shown in Alg. 1. And, Fig. 10 shows how EM is used as maximizing the lower bound of likelihood, instead of maximizing the likelihood. Note the following:
Convergence condition: not increasing the likelihood or reaching the max number of iterations.
Convergence: local maximum.
Initialization is important: can start from a poor point (initialization) and reach a poor local maximum.

Algorithm 1 EM

procedure LEARN(X):
Initialise θ
repeat
E-step: calculate lower bound: $Q(\theta, \theta^{old}) = E_{Z|X,\theta^{old}}\big[\log P(X, Z|\theta)\big]$
M-step: update θ by maximizing the lower bound: $\arg\max_\theta Q(\theta, \theta^{old})$
until convergence
return θ

6 Expectation-Maximization – Exercises

6.1 Mixture of scale mixtures

Usual Gaussian mixture model (GMM) is represented as a weighted sum of k components, each being a Gaussian distribution with some mean and variance. The goal in such model is to estimate the mixture weights, as well as all means and variances. One can extend this model to be more flexible by making each component a mixture on its own. Namely, instead of components being Gaussian distributions, we could instead model them as mixtures of Gaussians with the same mean, but different variances (so-called scale mixtures). Graphical model in plate notation for this setup is given in Figure 11.

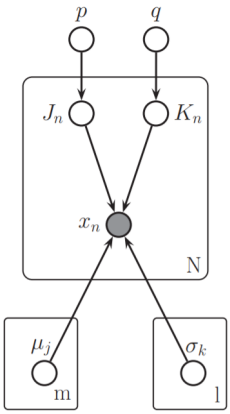


Figure 11: Mixture of scale mixtures

We can interpret J_n as the latent variable specifying which mean to use, and analogously K_n which variance to use. With this notation we can formally define the scale mixture as

$$p(x_n|J_n = j) = \sum_{k=1}^{l} q_k \mathcal{N}(\mu_j, \sigma_k^2), \tag{10}$$

where $q_j \coloneqq p(K_n = k)$, and the mixture of scale mixtures as

$$p(x_n) = \sum_{j=1}^{m} p_j \left[\sum_{k=1}^{l} q_k \mathcal{N}(\mu_j, \sigma_k^2) \right], \tag{11}$$

where $p_j \coloneqq p(J_n = j)$, for all n.[6]

Question. *Given the observations* $\mathbf{x} = \{x_n\}_{n=1}^N$ *estimate using EM algorithm parameters* $\theta = \{p_1, \ldots, p_m, \mu_1, \ldots, \mu_m, q_1, \ldots, q_l, \sigma_1^2, \ldots, \sigma_l^2\}$.

[6]Note that Eq. 10 and 11 are simple applications of marginalization and conditioning, e.g. for 10 $p(x_n|J_n = j) = \sum_{k=1}^{l} p(x_n, K_n = k|J_n = j) = \sum_{k=1}^{l} p(x_n|J_n = j, K_n = k)p(K_n = k)$.

7 Variational Inference – Theory

A deterministic approximate inference algorithm.

7.1 Motivation

We can't compute the posterior for many interesting models. For example for the Bayesian mixture of Gaussian, we draw $z_i \sim Mult(\pi)$ and $x_i \sim N(\mu_{zi}, \sigma^2)$ resulting in the following posterior [2]:

$$p(\mu_{1:K}, z_{1:n}|x_{1:n}) = \frac{\prod_k p(\mu_k) \prod_i p(z_i)p(x_i|z_i, \mu_{1:K})}{\int_{\mu_{1:K}} \sum_{z_{1:n}} \prod_k p(\mu_k) \prod_i p(z_i)p(x_i|z_i, \mu_{1:K})} \tag{12}$$

The denominator is the problem; even if we take the summation outside, this is intractable when n is reasonably large.

7.2 Main idea

So far we used EM and did point estimate for the model parameters; now we want to find posterior distribution for the **unknown** model parameters and hidden variables. For a DGM with observations X, hidden variables Z and model parameters Θ, want to pick an approximation $q(Z, \Theta)$ to the distribution from some tractable family, and then to try to make this approximation as close as possible to the true posterior.

$$p(Z, \theta|X) \approx q(Z, \theta) \tag{13}$$

This reduces inference to an optimization problem [4]. We measure the closeness of the two distributions q and p with Kullback-Leibler (KL) divergence.

$$\mathbb{KL}(q||p) = \sum_{Z,\Theta} q(Z, \Theta) log \frac{q(Z, \Theta)}{p(Z, \Theta|X)} \tag{14}$$

If we call the set of latent variables and parameters, Ψ, we can rewrite Eq. 14 as:

$$\sum_{\Psi} q(\Psi) log \frac{q(\Psi)}{p(\Psi|X)} = -E_\Psi\big[\log p(X, \Psi)\big] + E_\Psi\big[\log q(\Psi)\big] + \log p(X) \tag{15}$$

We cannot actually minimize KL divergence in Eq. 14 but since we have Eq. 15, we maximise lower bound of log likelihood, Evidence Lower Bound (ELBO) [2], which is (same as equation 7):

$$\text{ELBO}(q) = E_\Psi\big[\log P(X, \Psi)\big] - E_\Psi\big[\log q(\Psi)\big]. \tag{16}$$

Note that we can use ELBO for convergence test at each iteration i.e. the difference of the ELBO of current value and the previous one should be smaller than some small epsilon.

In mean field Variational Inference, we assume that the variational family factorizes,

$$q(Z_1, ..., Z_n, \Theta_1, ..., \Theta_K) = \prod_i q(Z_i) \prod_k q(\Theta_k) \tag{17}$$

Now for each update equation of $q(z_i)$ we perform the expectation, over the log of the joint distribution, w.r.t. all the hidden variables except the one we are deriving the approximate posterior for. For example, we obtain the update equation for z_j, $q(z_j)$, by calculating $E_{-z_j}\big(\log P(X, Z, \Theta)\big)$. Here we explain [2] that this is the result of the maximization of ELBO. We rewrite the ELBO and then take the partial derivative and set it to zero.

If we use the chain rule, we can write $p(X,\Psi) = p(X)\prod_j p(\Psi_j|\Psi_{1:j-1},X)$. Also because of the independence we have: $\log q(\Psi) = \log\big(q(\Psi_1)...q(\Psi_l)\big) = \sum_j \log q(\Psi_j)$. Now the ELBO becomes the following. Note that in the last expectation term in Eq. 21, the expectation w.r.t. Ψ can be reduced to expectation w.r.t. Ψ_j since there is only one variable, i.e. Ψ_j, in the brackets, i.e. $\log q(\Psi_j)$.

$$\text{ELBO}(q) = \log P(X) + \sum_j \Big(E_\Psi\big[\log P(\Psi_j|\Psi_{1:j-1},X)\big] - E_{\Psi_j}\big[\log q(\Psi_j)\big]\Big) \tag{18}$$

Now if you write the ELBO as a function of the last variable in the chain, say Ψ_k, you get:

$$\mathcal{L}(q(\Psi_k)) = const + E_\Psi\big[\log P(\Psi_k|\Psi_{-k},X)\big] - E_{\Psi_k}\big[\log q(\Psi_k)\big] = \tag{19}$$

$$\int q(\Psi_k)E_{-\Psi_k}[\log P(\Psi_k|\Psi_{-k},X)]d\Psi_k - \int q(\Psi_k)\log q(\Psi_k)d\Psi_k \tag{20}$$

Taking the partial derivative of the integrals in 20 and setting it to zero (and using lagrange multiplier) you get the update equation for each posterior during the coordinate ascent algorithm:

$$\log q^*(\Psi_k) \propto E_{-\Psi_k}[\log P(\Psi_k|\Psi_{-k},X)] = E_{-\Psi_k}[\log \underbrace{\frac{P(\Psi_k,\Psi_{-k},X)}{P(\Psi_{-k},X)}}_{\text{see }**}] \propto E_{-\Psi_k}[\log P(\Psi,X)] \tag{21}$$

** The term $P(\Psi_{-k},X)$ is constant w.r.t. Ψ_k.

Optional part: We now prove the result of the partial derivative set to zero.

To maximize the objective function $\mathcal{L}$ with constraint, as shown below, we need to find the *stationary function* accounting for the lagrangian term. That is solved by solving the Euler–Lagrange equation which includes the constraint, i.e., $\frac{\partial\mathcal{L}-\lambda G}{\partial q(\Psi_k)} - \frac{d}{d\Psi_k}\Big[\frac{\partial\mathcal{L}-\lambda G}{\partial q'(\Psi_k)}\Big] = 0$; G is the constraint.

$$\begin{aligned}&\mathcal{L}(q(\Psi_k)) = \\ &\int q(\Psi_k)E_{-\Psi_k}[\log P(\Psi_k|\Psi_{-k},X)]d\Psi_k - \int q(\Psi_k)\log q(\Psi_k)d\Psi_k \\ &s.t. \quad \int q(\Psi_k)d\Psi_k = 1 \quad \forall k\end{aligned} \tag{22}$$

Opening up the Euler–Lagrange equation with constraint, we achieve:

$$\frac{\partial\mathcal{L}}{\partial q(\Psi_k)} - \frac{d}{d\Psi_k}\cancelto{0}{\frac{\partial\mathcal{L}}{\partial q'(\Psi_k)}} - \lambda_k\Big[\cancelto{1}{\frac{\partial\int q(\Psi_k)d\Psi_k}{\partial q(\Psi_k)}} - \frac{d}{d\Psi_k}\Big(\cancelto{0}{\frac{\partial\int q(\Psi_k)d\Psi_k}{\partial q'(\Psi_k)}}\Big)\Big] \tag{23}$$

That is, to calculate $\frac{\partial\mathcal{L}}{\partial q(\Psi_k)} - \lambda_k = 0$. The zero cancellations above are due to $q'(\Psi_k)$ not appearing explicitly in the objective.

$$
\begin{aligned}
&1. \quad \frac{\partial \mathcal{L}}{\partial q(\Psi_k)} - \lambda_k = 0 \implies \\
&E_{-\Psi_k}[\log P(\Psi_k|\Psi_{-k}, X)] - \Big(\log q(\Psi_k) + \frac{q(\Psi_k)}{q(\Psi_k)}\Big) - \lambda_k = 0 \\
&\implies \log q(\Psi_k) = \Big[E_{-\Psi_k}[\log P(\Psi_k|\Psi_{-k}, X)] - 1 - \lambda_k\Big] \\
&\implies \lambda_k = E_{-\Psi_k}[\log P(\Psi_k|\Psi_{-k}, X)] - 1 - \log q(\Psi_k) \\
&2. \quad \partial\mathcal{L}/\partial\lambda_k = 0 \implies \int q(\Psi_k) d\Psi_k = 1 \\
&\downarrow replace \quad q(\Psi_k) \\
&\int e^{\Big[E_{-\Psi_k}[\log P(\Psi_k|\Psi_{-k}, X)] - 1 - \lambda_k\Big]} d\Psi_k = 1 \\
&\frac{1}{e^{\lambda_k}} \int e^{\Big[E_{-\Psi_k}[\log P(\Psi_k|\Psi_{-k}, X)] - 1\Big]} d\Psi_k = 1 \\
&e^{\lambda_k + 1} = \int e^{\Big[E_{-\Psi_k}[\log P(\Psi_k|\Psi_{-k}, X)]\Big]} d\Psi_k \\
&\downarrow from \quad 1 \quad and \quad 2 \\
&q(\Psi_k) = \frac{e^{E_{-\Psi_k}[\log P(\Psi_k|\Psi_{-k}, X)]}}{e^{1+\lambda_k}} = \frac{e^{E_{-\Psi_k}[\log P(\Psi_k|\Psi_{-k}, X)]}}{\int e^{\Big[E_{-\Psi_k}[\log P(\Psi_k|\Psi_{-k}, X)]\Big]} d\Psi_k}
\end{aligned}
\tag{24}
$$

We can see that the denominator is the normalizing constant for the distribution $q(\Psi_k)$. Therefore, we can formulate the log distribution as $\log q(\Psi_k) \propto E_{-\Psi_k}[\log P(\Psi_k|\Psi_{-k}, X)]$.

Algorithm 2, summarizes the VI considering mean field (factorization of the latent variables). Similar to EM, it is crucial to initialize the algorithm, i.e., the better initialization, the better local optima.

Convergence condition: not increasing the ELBO i.e. Eq. 16, or, reaching the max number of iterations. Note that to guarantee that the ELBO increases even after each update equation a proper coordinate ascend should be implemented. That is, the updates should be *sequentially*. For example, if ψ_1 is used in the calculation of ψ_2 then in the update of $q(\psi_2)$ the latest ψ_1 (hopefully from the current iteration) should be used. If the mean field VI is implemented by a *parallel* version of coordinate ascent it means that each $q(\psi_i)$ uses the variables values from the previous iteration.

Algorithm 2 Mean Field VI

procedure APPROXIMATE POSTERIOR(X)
Initialise $q(\Psi_i)$ for $i = 1, \ldots, l$
  repeat
    for $i = 1, \ldots, l$ **do**
      Update: $\log q^*(\Psi_i) \propto E_{-\Psi_i}[\log P(\Psi, X)]$
  until convergence
  Approximate posterior: $q(\Psi) = \prod_i q^*(\Psi_i)$
return $q(\Psi)$

8 Variational Inference – Exercises

The Cartesian Matrix Model (CMM) is defined as follows. There are R row distributions $\{N(\mu_r, \lambda_r^{-1}) : 1 \leq r \leq R\}$, each variance λ_r^{-1} is known and each μ_r has prior distribution $N(\mu, \lambda^{-1})$. There are also C column distributions $\{N(\xi_r, \tau_c^{-1}) : 1 \leq c \leq C\}$, each variance τ_c^{-1} is known and each ξ_c has prior distribution $N(\xi, \tau^{-1})$. All hyper-parameters are known. A matrix S is generated by, for each row $1 \leq r \leq R$ and each column $1 \leq c \leq C$, setting $S_{rc} = X_r + Y_c$ where X_r is sampled from $N(\mu_r, \lambda_r^{-1})$ and Y_c from $N(\xi_r, \tau_c^{-1})$. Use Variational Inference in order to obtained a variational distribution

$$q(\mu_1, \ldots, \mu_R, \xi_1, \ldots, \xi_C) = \prod_r q(\mu_r) \prod_c q(\xi_c)$$

that approximates $p(\mu_1, \ldots, \mu_R, \xi_1, \ldots, \xi_C | S)$. Tip: what distribution do you get from the sum of two Gaussian random variables? What is the relation between the means?

> **Question 15:** *Present the algorithm written down in a formal manner (using both text and mathematical notation, but not pseudo code).*

Figure 12: From Assignment 2, 2017

Question: See the question 15 from Figure 12.

A TL;DR

We encourage you to know the theory of each method and algorithm. We also prepared a small cheat sheet to help.

A.1 Conditional Independence in Graphical Models

(See Section 1 for details)
Use D-separation or "Method 2" to check conditional independence in graphical models. Don't forget to pay attention to the plate notation.

A.2 Hidden Markov Model

(See Section 3 for details)

- The summing, e.g. over z_{n-1}, implies $\sum_{z_{n-1}=1}^{J}$.
- The known probabilities in the inference (forward and backward) are: initial, transition, and emission probabilities.
- Equations 13.24 until 13.31 in [1] are useful to perform the forward and backward passes.
- In $\alpha(z_n)$, we mean that the z_n is observed.
- More advanced stuff: we assume a stationary distribution for HMM and that the HMM is finite in sequence length. For the definitions, see [4] section 17.2.3 Stationary distribution of a Markov chain.
- An HMM can be formulated as a GMM and vice versa. To understand this better, you need to understand mixture models, see [1, 4].
- To perform EM for HMM, you need to calculate the marginal of the two consecutive hidden variables too. See [1] for the calculation w.r.t. HMM. For IOHMM, you should derive it yourself.
- For solving IOHMM in an easier way, see factor graph representation in [1] section 13.2.3.

A.3 Expectation-Maximization (EM) Algorithm

(See Section 5 for details)
Goal: Find `point estimation` of parameters ($\theta^* = \{\theta_1^*, \ldots\}$) with an iterative process.
$X = \{X_1, \ldots, X_N\}$ are observed variables, $Z = \{Z_1, \ldots\}$ are latent variables, $\theta = \{\theta_1, \ldots\}$ are the parameters.
At each iteration

- **E-Step:**
  - Write complete-data log-likelihood ($logP(X, Z|\theta)$)
  - Write Q function ($Q(\theta, \theta^{old}) = E_{Z|X,\theta^{old}}[logP(X, Z|\theta)]$) and move E symbol inside (pay attention to the subscript of expectation, $Z|X, \theta^{old}$, it helps you to identify which terms' expectation you should calculate)

- Calculate γ value, based on previous iteration's parameters θ^{old} (i.e, $\gamma_{n,k} = E_{Z|X,\theta^{old}}[I(\ldots)]$) and put γ in the Q function

- **M-Step:**
  - Update parameters, which maximizes the Q function (i.e $\theta_1^* = argmax_{\theta_1} Q(\theta, \theta^{old})$) by simply taking partial derivative and setting it to zero (pay attention to the parameters with constraints, i.e $\sum_j \theta_{1,j} = 1$)

A.4 Variational Inference (VI)

(See Section 7 for details)
Goal: Instead of computing the exact posterior distribution of the parameters and latent variables ($P(Z, \theta|X)$) you obtain new, simpler `distributions` for parameters ($q(Z_1, \ldots, \theta_1, \ldots) = \prod_i q(Z_i) \prod_j q(\theta_j)$) with an iterative process.
$X = \{X_1, \ldots, X_N\}$ are observed variables, $Z = \{Z_1, \ldots\}$ are latent variables, $\theta = \{\theta_1, \ldots\}$ are the parameters, $\xi = \{\xi_1, \ldots\}$ are the hyperparameters (that you know). We represent the terms we are interested in as $\Psi = \{Z, \theta\}$.
At each iteration

- Write log probability of joint distribution ($logP(X, Z, \theta|\xi)$)

- Find the new distribution of the parameters/latent variables one-by-one (i.e $q(\theta_j)$ or $q(Z_i)$) by
  - Take the expectation of log joint ($E_{\Psi \neg \theta_j}[logP(X, Z, \theta|\xi)]$) w.r.t the rest of the parameters
  - Find a probability distribution which matches the pattern of a well-known distribution (Gaussian, Gamma etc) and identify the new hyperparameters (i.e $q(\theta_j) = Ga(\theta_j|\alpha^*, \beta^*)$)
  - If the new hyperparameter includes moment [7] of another parameter (i.e $q(\theta_k) = N(\theta_k|0.6E[\theta_j^2], \sigma^*)$), check the distribution of θ_j (i.e $Ga(\theta_j|\alpha^*, \beta^*)$) find the moment's value ($E[\theta_j^2] = \frac{\alpha^*(\alpha^*+1)}{(\beta^*)^2}$) and place it to appropriate place ($q(\theta_k) = N(\theta_k|0.6E[\theta_j^2], \sigma^*) = N(\theta_k|0.6\frac{\alpha^*(\alpha^*+1)}{(\beta^*)^2}, \sigma^*)$))

A.5 Useful Properties

- $E[X + Y] = E[X] + E[Y]$ [5]
- $E[aX] = aE[X]$ [5]
- $E[\mathbb{1}_{\{X=x\}}(x)] = \sum_x \mathbb{1}_{\{X=x\}}(x) P(X = x) = P(X = x)$ [6]

[7] For instance the first three moments of a Normal distribution $N(x|\mu, \sigma^2)$ are $E[X] = \mu, E[X^2] = \mu^2 + \sigma^2, E[X^3] = \mu^3 + 3\mu\sigma^2$, the first two moments of a Gamma distribution $Ga(x|a, b)$ are $E[X] = \frac{a}{b}, E[X^2] = \frac{a(a+1)}{b^2}$.

References

[1] C. Bishop. *Pattern Recognition and Machine learning*. Springer, 2006.

[2] David M. Blei. *Variational Inference*. https://www.cs.princeton.edu/courses/archive/fall11/cos597C/lectures/variational-inference-i.pdf. [Online; accessed 20-November-2019]. 2011.

[3] *d-separation*. http://web.mit.edu/jmn/www/6.034/d-separation.pdf. [Online; accessed 12-December-2019].

[4] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[5] Wikipedia. *Expected value*. https://en.wikipedia.org/wiki/Expected_value. [Online; accessed 12-December-2019].

[6] Wikipedia. *Indicator function*. https://en.wikipedia.org/wiki/Indicator_function. [Online; accessed 12-December-2019].