Machine Learning Worksheet 11

Inference in latent variable models

1 Expectation Maximisation

The exercises on EM might be easier after the tutorial.

Problem 1: The *K-Means algorithm* is an intuitive clustering algorithm. Initialize K cluster centres μ_k so that they are distinct. Then iterate the following procedure until convergence:

1. Calculate clusters

$$C_k = \left\{ \mathbf{x}_i \in \mathcal{D} : k = \arg\min_j d(\mathbf{x}_i, \boldsymbol{\mu}_j) \right\},$$

   i.e., C_k gathers all data points for which $\boldsymbol{\mu}_k$ is the closest centre in some metric $d(\cdot, \cdot)$.

2. Recalibrate the cluster mean:

$$\boldsymbol{\mu}_k \leftarrow \frac{1}{|C_k|} \sum_{\mathbf{x} \in C_k} \mathbf{x}$$

Show that this algorithm is an instance of the EM-algorithm for the Euclidean metric $d(\mathbf{x}, \mathbf{y}) = ||\mathbf{x} - \mathbf{y}||_2$.

To do so, consider a special mixture of isotropic Gaussians where every mixture component has identical *fixed* covariance $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I}, \sigma \in \mathbb{R}$, determine the EM algorithm for this, and then take $\sigma \to 0$.

Notice that the second step given here is a special case of the more generic

$$\boldsymbol{\mu}_k \leftarrow \arg\min_{\boldsymbol{\mu}} \sum_{\mathbf{x} \in C_k} d(\mathbf{x}, \boldsymbol{\mu})$$

for the case of the Euclidean metric.

First, we observe that the algorithm already resembles EM. The cluster assignment in step 1 resembles the responsibility assignment in the E-step. The cluster centre update in step 2 looks like the M-step: Given one-hot responsibilities, we do an ML-style update of a Gaussian mean.

Following the hint, we consider a MoG with identical, isotropic variances. In that case, the responsibilities take the following form:

$$\begin{aligned} p(\mathbf{z}=k \mid \mathbf{x},\theta) &= \frac{p(\mathbf{x} \mid \mathbf{z}=k,\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)p(\mathbf{z}=k \mid \pi_k)}{\int p(\mathbf{x} \mid \mathbf{z})p(\mathbf{z})\,\mathrm{d}\mathbf{z}} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\sum_{l=1}^{K}\pi_l \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_l,\boldsymbol{\Sigma}_l)} \\ &= \frac{\pi_k \exp\left(\frac{-||\boldsymbol{x}-\boldsymbol{\mu}_k||^2}{2\sigma^2}\right)}{\sum_l \pi_l \exp\left(\frac{-||\boldsymbol{x}-\boldsymbol{\mu}_l||^2}{2\sigma^2}\right)} \\ &= \frac{1}{\sum_l \frac{\pi_l}{\pi_k}\exp\left(\frac{-||\boldsymbol{x}-\boldsymbol{\mu}_l||^2+||\boldsymbol{x}-\boldsymbol{\mu}_k||^2}{2\sigma^2}\right)} \end{aligned}$$

If $\boldsymbol{\mu}_k$ denotes the centre that is closest to $\boldsymbol{x}$, then

$$\frac{-||\boldsymbol{x}-\boldsymbol{\mu}_l||^2+||\boldsymbol{x}-\boldsymbol{\mu}_k||^2}{2\sigma^2} \leq 0$$

for all l, with equality if and only if $k=l$. For $\sigma \to 0$, the denominator converges to 1: If $k=l$, the argument of $\exp(\cdot)$ is exactly zero, while for $k \neq l$ we are exponentiating increasingly negative arguments.

If $\boldsymbol{\mu}_k$ denotes a centre that is *not* closest to $\mathbf{x}$, there is at least one $l \neq k$ for which

$$0 < \frac{-||\boldsymbol{x}-\boldsymbol{\mu}_l||^2+||\boldsymbol{x}-\boldsymbol{\mu}_k||^2}{2\sigma^2} \longrightarrow \infty, \qquad \text{as } \sigma \to 0.$$

Consequently, the denominator converges to ∞, too.

Overall, the responsibilities degenerate to a hard one-hot assignment of the data point $\mathbf{x}$ to the component closest to $\mathbf{x}$. This coincides with step 1 of the K-Means algorithm.

Inserting one-hot responsibilities into the general MoG M-step immediately yields step 2 in K-means. Notice that we do not learn covariances, they are assumed fixed.

We can conclude that K-Means is a special case of the more general Mixture of Gaussians EM algorithm.

Problem 2: You have two equal-looking coins, one of which is fair and the other one is biased, but you cannot tell them apart. You pick a coin to start at random, and then flip the two coins alternately (in an ABAB... pattern). After a total of ten flips, you obtain a sequence 0100110111, where 1 corresponds to heads and 0 to tails. That is, out of the $N_e = 5$ even flips, $K_e = 4$ showed up heads, and out of the $N_o = 5$ odd flips, $K_o = 2$ showed up heads.

You want to do maximum likelihood optimization in θ, the probability that the biased coin shows up heads. Derive a general EM-algorithm to determine θ in terms of N_o, N_e, K_o, K_e. In particular, do not assume $N_o = N_e$.

Initialize $\theta^{(0)} = 0.5$. Implement your algorithm in a framework of your choice and report the converged result with the given data.

Hint 1: $\theta^{(1)} = 0.6$.

Hint 2: Reduce your workload by reusing computations from previous coin examples in this course.

Let $b \in \{o, e\}$ denote the position of the biased flips in the sequence. b is our latent variable. As usual, $\mathcal{D}$ denotes the data.

Then

$$p(\mathcal{D} \mid b = e, \theta) = \left(\frac{1}{2}\right)^{N_o} \theta^{K_e}(1-\theta)^{N_e - K_e},$$
$$p(\mathcal{D} \mid b = o, \theta) = \left(\frac{1}{2}\right)^{N_e} \theta^{K_o}(1-\theta)^{N_o - K_o}.$$

As we draw the coin fairly and have no other prior information, we have

$$p(b = e) = p(b = o) = \frac{1}{2}$$

For the E-step, we need the posterior distribution, which we obtain once again by Bayes' rule:

$$\begin{aligned} p\left(b \mid \mathcal{D}, \theta^{(t)}\right) &= \frac{p\left(\mathcal{D} \mid b, \theta^{(t)}\right) p(b)}{\sum_{b \in \{o,e\}} p\left(\mathcal{D} \mid b, \theta^{(t)}\right) p(b)} \\ &= \begin{cases} \frac{\left(\frac{1}{2}\right)^{N_o+1} \theta^{K_e}(1-\theta)^{N_e-K_e}}{\left(\frac{1}{2}\right)^{N_o+1} \theta^{K_e}(1-\theta)^{N_e-K_e} + \left(\frac{1}{2}\right)^{N_e+1} \theta^{K_o}(1-\theta)^{N_o-K_o}} & b = e, \\ \frac{\left(\frac{1}{2}\right)^{N_e+1} \theta^{K_o}(1-\theta)^{N_o-K_o}}{\left(\frac{1}{2}\right)^{N_o+1} \theta^{K_e}(1-\theta)^{N_e-K_e} + \left(\frac{1}{2}\right)^{N_e+1} \theta^{K_o}(1-\theta)^{N_o-K_o}}, & b = o, \end{cases} \\ &= \begin{cases} \mu \equiv \mu\left(\theta^{(t)}\right), & b = e, \\ 1 - \mu, & b = o. \end{cases} \end{aligned}$$

Every θ in the above equation is actually a $\theta^{(t)}$. This was left away for notational clarity. Moreover, for brevity, we introduce the short-hand notation μ. We exploit the fact that our latent variable is binary, so that all cases are parametrised by one parameter. It is important to note that μ depends on $\theta^{(t)}$, but not θ, the free variable in the M-step.

For the M-step, we have to perform

$$\theta^{(t+1)} \longleftarrow \arg\max_{\theta} \mathbb{E}_{p\left(b \mid \mathcal{D}, \theta^{(t)}\right)}[\ln p(\mathcal{D}, b \mid \theta)].$$

Inserting our notation and findings from above:

$$\begin{aligned} \mathbb{E}_{p\left(b \mid \mathcal{D}, \theta^{(t)}\right)}[\ln p(\mathcal{D}, b \mid \theta)] =& \mu \left((N_o + 1) \ln \frac{1}{2} + K_e \ln \theta + (N_e - K_e) \ln(1-\theta) \right) \\ &+ (1-\mu) \left((N_e + 1) \ln \frac{1}{2} + K_o \ln \theta + (N_o - K_o) \ln(1-\theta) \right) \end{aligned}$$

Now we can take the derivative w. r. t. θ. In this step, it is crucial that μ does not depend on θ, but only on $\theta^{(t)}$.

$$\begin{aligned}\frac{\partial}{\partial\theta}\mathbb{E}_{p(b|\mathcal{D},\theta^{(t)})}[\ln p(\mathcal{D}, b \mid \theta)] &= \mu\left(\frac{K_e}{\theta} - \frac{N_e - K_e}{1-\theta}\right) + (1-\mu)\left(\frac{K_o}{\theta} - \frac{N_o - K_o}{1-\theta}\right) \\ &= \underbrace{[\mu K_e + (1-\mu)K_o]}_{a}\frac{1}{\theta} - \underbrace{[\mu(N_e - K_e) + (1-\mu)(N_o - K_o)]}_{b}\frac{1}{1-\theta}\end{aligned}$$

At this point, we recognize the exact same functional form of the derivative as in the standard MLE case. The difference is that the expressions for the number of heads and tails have been replaced with more complicated expressions that incorporate the missing latent variable. The latent sequence is weighted with our current posterior belief that the biased coin is responsible for the even or odd part of the data, respectively. Nevertheless, we can reuse our work and jump straight to the optimal solution

$$\theta^{(t+1)} \longleftarrow \frac{a}{a+b} = \frac{\mu K_e + (1-\mu)K_o}{\mu N_e + (1-\mu)N_o}$$

It was not requested, but we can now even join E- and M-step into one iterative update rule by inserting the definition for μ. Luckily, the posterior's denominator cancels out:

$$\begin{aligned}\theta^{(t+1)} \longleftarrow \frac{\mu K_e + (1-\mu)K_o}{\mu N_e + (1-\mu)N_o} &= \frac{\left(\left(\frac{1}{2}\right)^{N_o}\theta^{K_e}(1-\theta)^{N_e-K_e}\right)K_e + \left(\left(\frac{1}{2}\right)^{N_e}\theta^{K_o}(1-\theta)^{N_o-K_o}\right)K_o}{\left(\left(\frac{1}{2}\right)^{N_o}\theta^{K_e}(1-\theta)^{N_e-K_e}\right)N_e + \left(\left(\frac{1}{2}\right)^{N_e}\theta^{K_o}(1-\theta)^{N_o-K_o}\right)N_o} \\ &= \frac{\left(\frac{1}{2}\right)^{N_o-N_e}\theta^{K_e}(1-\theta)^{N_e-K_e}K_e + \theta^{K_o}(1-\theta)^{N_o-K_o}K_o}{\left(\frac{1}{2}\right)^{N_o-N_e}\theta^{K_e}(1-\theta)^{N_e-K_e}N_e + \theta^{K_o}(1-\theta)^{N_o-K_o}N_o}\end{aligned}$$

Again, for notational clarity, the time indices of $\theta^{(t)}$ are not displayed.

When inserting $\theta^{(0)} = 0.5$, all powers cancel out and we get

$$\theta^{(1)} \longleftarrow \frac{K_e + K_o}{N_e + N_o} = \frac{2+4}{5+5} = 0.6,$$

as predicted by the hint.

This can now be implemented. A python implementation is found in the attached Jupyter notebook. After a few iterations, the algorithm converges to $\theta \approx 0.7658$.

2 Variational Auto-Encoders

Problem 3: Compute the KL divergence between two Gaussian distributions $\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ with diagonal covariance matrices.

Hint: If you exploit the facts you know, you can save yourself a lot of work before walking down the straightforward path.

Let $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ denote the respective densities. Assuming the notation

$$\boldsymbol{\mu}_i = (\mu_{i,1}, \ldots, \mu_{i,n}), \boldsymbol{\Sigma}_i = \operatorname{diag}\left(\sigma_{i,1}^2, \ldots, \sigma_{i,n}^2\right),$$

we remember the Multivariate Normal lecture: It holds that for diagonal Gaussians, the components are independent and the pdf decomposes:

$$f(\boldsymbol{x}) = \prod_j f_j(x_j) = \prod_j \mathcal{N}\left(x_j \mid \mu_{1,j}, \sigma_{1,j}^2\right),$$

and similarly for g. Now

$$\mathrm{KL}(f \mid\mid g) = \int f(\boldsymbol{x}) \ln \frac{f(\boldsymbol{x})}{g(\boldsymbol{x})} \,\mathrm{d}\boldsymbol{x}.$$

Since f and g factorize, the logarithm of the fraction turns into a sum of log fractions. Linearity of expectation then gives us that the KL decomposes into a sum of KL divergences of the components:

$$\mathrm{KL}(f \mid\mid g) = \sum_j \mathrm{KL}(f_j \mid\mid g_j)$$

We have reduced the problem to the one-dimensional case, which is less bothersome. In fact, we have already considered this case in tutoring exercise 4.2. Hence we conclude

$$\mathrm{KL}(f \mid\mid g) = \sum_j \mathrm{KL}(f_j \mid\mid g_j) = -\frac{n}{2} + \sum_j \left(\ln \frac{\sigma_{2,j}}{\sigma_{1,j}} + \frac{\sigma_{1,j}^2 + (\mu_{1,j} - \mu_{2,j})^2}{2\sigma_{2,j}^2} \right).$$

Problem 4: What do you expect to happen when leaving away the KL term in the evidence lower bound when training a variational auto-encoder?

If that helps you, assume isotropic prior and approximate posterior for both scenarios and use your insights from the previous exercise.

The learning algorithm widens the information bottleneck in the stochastic sampling layer of the VAE, as this is an obstacle to easy (yet easily overfitting) learning.

The variance of the approximate posterior will be learned to be close to 0, as this lack of stochasticity simplifies learning of the decoding part. We are essentially back to the deterministic auto-encoder—even the error function is the same, least-squares.