6.790 Homework 7 **Solutions**

This homework is for study purposes and will not be handed in or graded.

Contents

1 Generative model warmup

1. (a) We are considering a variety of different generative models, trained on some dataset $\mathcal{D} = \{x^{(i)}\}_{x=1}^{n}$.
   For each of the following models, describe what parametric models (including neural networks) are learned and how we use them (and/or the training data) to obtain a value for the density $p(x)$, for a novel input x (or explain that it's difficult/impossible).
   i. kernel density estimator
   ii. auto-regressive model
   iii. Gaussian mixture
   iv. continuous-time probability flow (out of our scope, feel free to skip)
   v. variational auto-encoder
   vi. diffusion model (out of our scope, feel free to skip)

Solution:

i. Kernel density estimator. It is a non-parametric model so no parameters are learned. Given a new datapoint, and choices of kernel K and bandwidth h:

$$p(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x^{(i)}}{h}\right)$$

ii. Auto-regressive model. Assuming x can be decomposed into m orthogonal coordinates x_j, a model is learned to estimate from training data for every $j \in [1, m]$ $p_{\theta,j}(x_j|x_{1:j-1})$.

$$p(x) = p_{\theta,1}(x_1) \prod_{j=2}^{m} p_{\theta,j}(x_j|x_{1:j-1})$$

iii. Gaussian mixture. The parameters of a Gaussian mixture are typically learned through the EM-algorithm from which we can derive the means μ_k, covariance Σ_k and mixture weights π_k.

$$p(x) = \sum_{k=1}^{K} \pi_k \, \mathcal{N}(x|\mu_k, \Sigma_k)$$

iv. Normalizing flow. The parameters of the transformation, typically parameterized with an invertible neural network, are learnedvia gradient descent to maximize the likelihood of the observed data under the transformed distribution. Given a learned normalizing flow f_θ the likelihood of a new point is:

$$p(x) = p_Z(f_\theta(x)) \det \frac{\partial f_\theta}{\partial x}$$

where p_Z is the density under the base distribution.

v. Variational auto-encoder. The parameters of the autoencoder neural network are learned via gradient descent to maximize the Evidence Lower Bound (ELBO) objective. It is intractable to compute the likelihood of a point under a VAE, as it would require integrating over the latent variable, however one can estimate a lower bound through the ELBO:

$$\log p(x) \geqslant \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}(q_\phi(z \mid x) \,\|\, p(z))$$

vi. Diffusion model. The parameters of the neural network describing the score function of the diffusion model are learned through gradient descent with the score matching objective. The likelihood is not tractable for SDE based diffusion models, but using the ODE formulation we can compute it similar to normalizing flows via:

$$\log p(x) = \log p_T(x_T) - \int_0^T \mathrm{tr}\left(\frac{\partial f_\theta(x_t, t)}{\partial x_t}\right) \mathrm{d}t$$

where x_t is the trajectory obtained running the ODE process in reverse and f_θ the learned drift function and the term inside the integral is the trace of the Jacobian of f.

(b) Now, for each of these same models, describe how to draw iid samples from the learned density.

i. kernel density estimator
ii. auto-regressive model
iii. Gaussian mixture
iv. continuous time probability flow
v. variational auto-encoder
vi. diffusion model

Solution:

i. Kernel density estimator. Randomly sample a datapoint from the training data $x^{(i)}$ and then sample a point from the kernel distribution centered at $x^{(i)}$.

ii. Auto-regressive model. Sample x_1 from $p_{\theta,1}$, and then recursively sample x_j from $p_{\theta,j}(\cdot|x_{1:j-1})$.

iii. Gaussian mixture. Sample a mixture k proportionally to π_k, and then sample a point from $\mathcal{N}(\mu_k, \Sigma_k)$.

iv. Continuous-time flow. Sample a point from the base distribution $z \sim p_Z$, follow the vector field forward (see last question).

v. Variational auto-encoder. Sample a point z from the prior distribution. Run the decoder to get the parameters of the final distribution (e.g. $\mu_x = p_\theta(x|z)$, sample from this final distribution.

vi. Diffusion model. Sample a point from the prior distribution, $x_T \sim p_T$, run the reverse diffusion process with the learned score to obtain a sample x at time 0.

2. We can interpret logistic regression as determining a conditional distribution $\hat{p}(Y \mid X)$.

   (a) Explain how to sample from $\hat{p}(Y \mid X = x)$ for a novel value x.

   Solution: Having learned the parameters w and b from the training data, the conditional probability of $\hat{p}(Y = 1|X = x) = \sigma(w^\top x + b)$ where σ is the sigmoid function. Therefore to sample from it, one can sample a random element from $r = U[0, 1]$ and return 1 if $r < \sigma(w^\top x + b)$ and 0 otherwise.

   (b) Let's explore the ability of logistic regression to represent posterior distributions. Imagine a training set $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^n$ where $x^{(i)} = (1)$ for all i, and where $y^{(i)} = +1$ for proportion p of the data (and 0 for proportion $1 - p$).
   Show that in the limit of large n, the learned likelihood $\hat{p}(Y = 1 \mid X = (1))$ converges to p.

   Solution:
   In the limit of large n we can assume that our training data contains a proportion of exactly p positives and $1 - p$ negatives. Let $q = \sigma(w + b) = \hat{p}(Y \mid X = (1))$. The corresponding binary cross-entropy loss to minimize is:

$$\mathcal{L}(w, b) = -\frac{1}{n} \sum_{i=1}^{n} \left[y^{(i)} \log(\sigma(w + b)) + (1 - y^{(i)}) \log(1 - \sigma(w + b)) \right]$$

   therefore:

$$\mathcal{L}(q) = -p \log q - (1 - p) \log(1 - q)$$

   Taking the derivative w.r.t. q and setting that to 0 we obtain that $q = p$, therefore $\hat{p}(Y = 1 \mid X = (1)) = p$.

2 VAE

This question closely follows the development in Kingma, Diederik P., and Max Welling. "An introduction to variational autoencoders." Foundations and Trends in Machine Learning.

3. (a) If our goal is to construct a model of a data distribution $\hat{p}_\theta(x)$, what advantage is there in turning it into an apparently harder problem of modeling $\hat{p}_\theta(x, z)$ for some latent variable z?

   Solution: In the case of a Gaussian mixture, $p(x, z) = p(x \mid z)p(z)$ has a simpler parametric form than $p(x)$.

(b) Consider a distribution over $x \in \mathbb{R}$ such that $p(x \leqslant v) = F(v)$ for some cdf F.

i. How is $F(x)$ distributed if $x \sim p(x)$?

> **Solution:** $\text{Unif}(0, 1)$

ii. If we wanted to make a latent variable model with $p(z) = \text{Unif}(0, 1)$, which of the following choices for $p(x \mid z)$ would guarantee that $p(x) = \int_z p(x \mid z)p(z)dz$?

- √ **A distribution that assigns probabilty 1 to $x = F^{-1}(z)$.**
- ◯ A distribution that assigns probabilty 1 to $x = F^{-1}(p(z))$.
- ◯ A Gaussian $\mathcal{N}(F(z), 1)$.

iii. If we wanted to make a latent variable model with $p(z) = \mathcal{N}(0, 1)(z)$, which of the following choices for $p(x \mid z)$ would guarantee that $p(x) = \int_z p(x \mid z)p(z)dz$?

- ◯ A distribution that assigns probabilty 1 to $x = F^{-1}(z)$.
- ◯ A distribution that assigns probabilty 1 to $x = F^{-1}(p(z))$.
- √ **A distribution that assigns probabilty 1 to $x = F^{-1}(G(z))$, where G is the Gaussian cdf.**
- ◯ A Gaussian $\mathcal{N}(F(z), 1)$.

iv. Now, let's say we want to train a neural network with parameters θ to represent $p(x \mid z)$, by maximizing the log likelihood of some training set $\mathcal{D} = \{x^{(i)}\}_{i=1}^n$. What loss function would we minimize, ignoring (for now) computational intractability?

> **Solution:**
> $$\mathcal{L}(\theta) = -\sum_{i=1}^{n} \log \int_z p_\theta(x^{(i)} \mid z)p(z)dz$$

v. Why is it hard to minimize, especially in high dimensions?

> **Solution:** We don't generally have a closed form for the integral and numerical integration is expensive in high dimensions.

vi. Provide an approximation to the loss function, based on sampling.

> **Solution:**
> $$\mathcal{L}(\theta) = -\sum_{i=1}^{n} \log \frac{1}{m} \sum_{z_j \sim \mathcal{N}(0,1); j=1}^{m} p_\theta(x^{(i)} \mid z^{(j)})$$

vii. What problems might we have with this estimator if we sample $z \sim \mathcal{N}(0, I)$ in high dimensions?

> **Solution:** The "weights" $p_\theta(x^{(i)} \mid z^{(j)})$ would generally be near 0 and it would take a lot of samples to get a useful gradient for optimization.

(c) The strategy in a VAE is to learn a new distribution $q_\phi(z \mid x)$, called the *inference model*

that will hopefully generate samples that will tend to have high values of $p_\theta(x^{(i)} \mid z)$. Let's focus on a single data-point x.

i. We observe that

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

Verify this.

> **Solution:** It follows directly from the identity:
>
> $$p_\theta(x) p_\theta(z \mid x) = p_\theta(x, z)$$
>
> dividing by the conditional probability and taking the logarithm.

ii. If we are going to sample using the inference model, then it's useful to view this as

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

which we can (apparently gratuitously) rewrite as

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

But this lets us divide into two terms that are useful:

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

and they have names!

$$\log p_\theta(x) = \mathrm{ELBO}_{\theta,\phi}(x) + \mathrm{KL}\big(q_\phi(z \mid x) \,\|\, p_\theta(z \mid x)\big)$$

We are going to work on maximizing the ELBO rather than $\log p_\theta(x)$. Why is that more straightforward?

> **Solution:** Because we have learned $p_\theta(x|z)$, from this obtaining $p_\theta(z|x)$ is intractable.

iii. Show that $\mathrm{ELBO}_{\theta,\phi}(x) \leqslant \log p_\theta(x)$. When are they equal?

> **Solution:** Above we showed:
>
> $$\log p_\theta(x) = \mathrm{ELBO}_{\theta,\phi}(x) + \mathrm{KL}\big(q_\phi(z \mid x) \,\|\, p_\theta(z \mid x)\big)$$
>
> the KL is always non-negative, therefore $\mathrm{ELBO}_{\theta,\phi}(x) \leqslant \log p_\theta(x)$. They are equal when the KL is zero, therefore $q_\phi(z \mid x) = p_\theta(z \mid x)$ for all z.

iv. Write an expression for the ELBO in terms of the data likelihood and $\mathrm{KL}\big(q_\phi(z \mid x) \,\|\, p_\theta(z \mid x)\big)$. Assuming that our neural networks have infinite representational capacity and the optimization works perfectly, if we optimize $\mathrm{ELBO}_{\theta,\phi}(x)$, what can we say about $p_\theta(x)$?

Solution:

$$\text{ELBO}_{\theta,\phi}(x) = \log p_\theta(x) - \text{KL}\big(q_\phi(z \mid x) \,\|\, p_\theta(z \mid x)\big)$$

assuming infinite capacity and perfect optimization for any value of θ, ϕ will be chosen to minimize the KL divergence and therefore chosen such that $q_\phi(z \mid x) = p_\theta(z \mid x)$ and the KL is zero. Then the optimization of θ boils down to maximizing the likelihood of the data. With infinite capacity $p_\theta(x)$ will converge to model exactly the training data distribution.

(d) It's time to maximize the ELBO via stochastic gradient descent!

i. First, with respect to θ. If we represent $p_\theta(x, z) = p_\theta(x \mid z)p(z)$ where $p(z)$ is a fixed spherical Gaussian, and $p_\theta(x \mid z) = \mathcal{N}(\text{NN}_\theta(z), \sigma^2)$ where $\text{NN}_\theta(z)$ is a deterministic neural network parameterized by θ and σ is a small fixed standard deviation, write an expression for

$$\nabla_\theta \text{ELBO}_{\theta,\phi}(x)$$

Solution:

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

$$\nabla_\theta \text{ELBO}_{\theta,\phi}(x) = \mathbb{E}_{q_\phi(z|x)}\Big[-\frac{1}{\sigma^2} J_{\text{NN}_\theta(z)}(\theta)^\top (\text{NN}_\theta(z) - x)\Big]$$

ii. Now, with respect to ϕ. This is harder because ϕ appears in the distribution that we're taking the expectation over:

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

So we can't just push the gradient inside the expectation, tempting though it may be. Instead, we need to do the *reparameterization trick!* (See Murphy book 2 section 6.3.5) Instead of taking the expectation with respect to a distribution q over z, we'll define a new random variable $\epsilon \sim \mathcal{N}(0, 1)$ and define $z = g(\epsilon, \phi, x)$. Now,

$$\begin{aligned}
\nabla_\phi \text{ELBO}_{\theta,\phi}(x) &= \nabla_\phi \mathbb{E}_\epsilon\big[\log p_\theta(x, z) - \log q_\phi(z \mid x)\big] \\
&= \mathbb{E}_\epsilon \nabla_\phi\big[\log p_\theta(x, z) - \log q_\phi(z \mid x)\big] \\
&\approx \nabla_\phi\big[\log p_\theta(x, z) - \log q_\phi(g(\epsilon, \phi, x) \mid x)\big]
\end{aligned}$$

Write an expression for this gradient, assuming we represent $q_\phi(z \mid x) = \mathcal{N}(\text{NN}_\phi(x), \sigma^2)$ where $\text{NN}_\phi(x)$ is deterministic a neural network parameterized by ϕ and σ is a small fixed standard deviation. It will depend on g. (We won't go into strategies for choosing g, but the paper describes it nicely.)

Solution:

$$\begin{aligned}
\nabla_\phi \text{ELBO}_{\theta,\phi}(x) &\approx \nabla_\phi \left[\log p_\theta(x,z) - \log q_\phi(g(\epsilon,\phi,x) \mid x)\right] \\
&= \nabla_\phi \log p_\theta(x,z) - \nabla_\phi \log q_\phi(g(\epsilon,\phi,x) \mid x) \\
&= J_{g(\epsilon,\phi,x)}(\phi)^\top (\nabla_z \log p_\theta(x|z) + \nabla_z \log p(z)) \\
&\quad + \frac{1}{2\sigma^2} \nabla_\phi \|\text{NN}_\phi(x) - g(\epsilon,\phi,x)\|^2 \\
&= J_{g(\epsilon,\phi,x)}(\phi)^\top (\nabla_z \text{NN}_\theta(z) - \frac{z-\mu_*}{\sigma_*^2}) \\
&\quad + \frac{1}{\sigma^2} (J_{\text{NN}_\phi(x)}(\phi) - J_{g(\epsilon,\phi,x)}(\phi))^\top (\text{NN}_\phi(x) - g(\epsilon,\phi,x))
\end{aligned}$$

where $p(z) = N(\mu_*, \sigma_*^2 I)$.

3 Mixture models

4. Consider a simple mixture model involving two spherical Gaussians in two dimensions. So $x \in \mathcal{R}^2$ and

$$P(x|\theta) = \sum_{z=1}^{2} P(z|\theta) P(x|z,\theta) = \sum_{z=1}^{2} p_z N(x; \mu_z, \sigma_z^2 I)$$

We will initialize the parameters of this mixture model as follows

$$p_1 = p_2 = 0.5, \quad \mu_1 = \mu_2, \quad \sigma_2^2 = 2\sigma_1^2$$

The initialization is also shown graphically in Figure 1 (top middle). The circles are drawn exactly one standard deviation (e.g., σ_1) away from the corresponding mean (e.g., μ_1). The larger dashed circle corresponds to the second component with larger variance.

Given the initialization above, which one of the figures a-d) of Figure 1 represents the mixture model that we get after one EM-iteration? Briefly justify your answer.

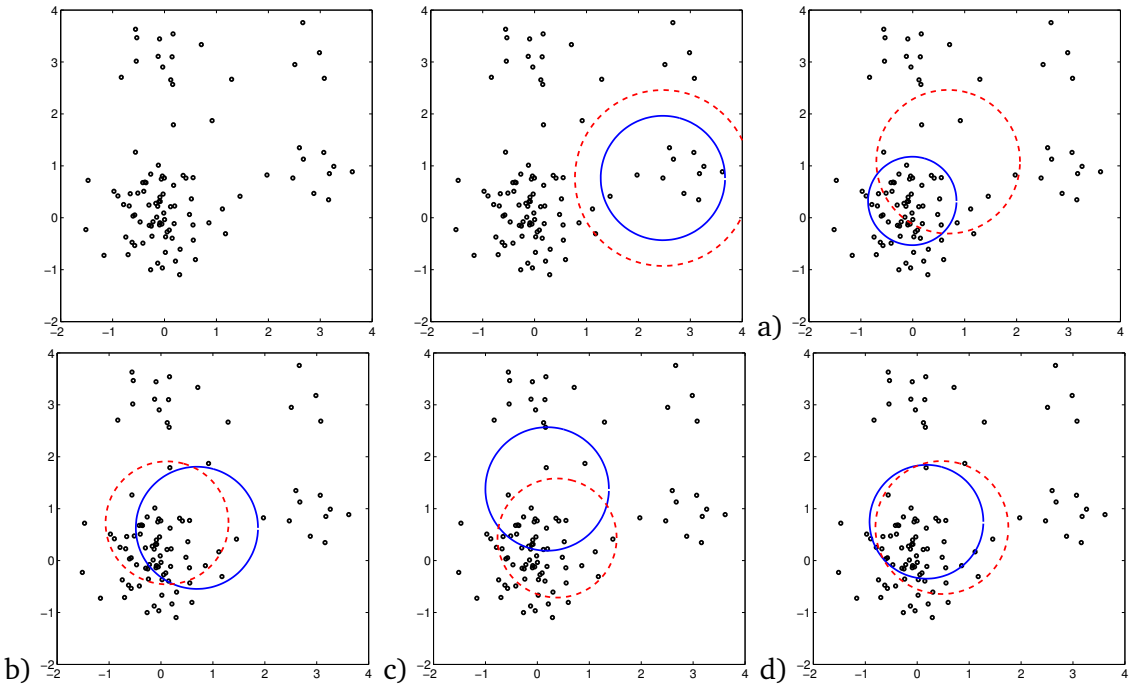

Figure 1: Top left) observed data; top right) initial mixture model; a-d) candidate mixture models resulting from one EM update.

Solution: *Key points:* Full credit given for choosing option b) and providing justification.

The EM algorithm proceeds by first assigning points to Gaussians based on posterior probabilities (E-step), and then re-estimating the Gaussians (and the mixing proportions) based on these weighted assignments. The correct figure is b).

The Gaussian with larger variance (#2) receives higher posterior assignments for points that are further away. It therefore moves closer to the overall mean of the points, overshooting slightly as the points at the center of the initialization are assigned primarily to Gaussian #1 with smaller variance. One can analogously explain why Gaussian #1 moves less.

5. Suppose we have a k-component mixture of spherical Gaussians model. Which of the following initializations of mixing proportions, means, and variances have a chance of recovering the underlying clusters assuming our assumption about the model family is correct? If the initialization is likely to fail, describe how. We use a shorthand $[k] = \{1, \ldots, k\}$.

    1. $p_j = 1/k$, $j \in [k]$; $\mu_j = \mu_0$, $j \in [k]$, for some common μ_0; $\sigma_j = \sigma_0$, $j \in [k]$, for some common σ_0.
    2. $p_j = 1/k$, $j \in [k]$; $\mu_j = \mu_0$, $j \in [k]$, for some common μ_0; σ_j, $j \in [k]$, are set to different values

3. $p_j = 1/k$, $j \in [k]$; μ_j, $j \in [k]$, are set to randomly chosen data points; $\sigma_j = \sigma_0$, $j \in [k]$, for some common σ_0.
4. p_j, $j \in [k]$ are randomized; $\mu_j = \mu_0$, $j \in [k]$, for some common μ_0; $\sigma_j = \sigma_0$, $j \in [k]$, for some common σ_0.

Solution: *Key points:* Full credit given for stating that 1 and 4 are likely to fail due to identical mean and variance updates, while 2 and 3 have a chance at success. Partial credit given for saying that only 1 is likely to fail.

1. The initialization fails. Since the Gaussians are all equal initially, the posterior assignments are also all equal. All the Gaussians in the mixture are updated the same, as if there was only a single Gaussian.
2. The initialization can succeed. Compare to the earlier question. The different variances of the Gaussians result in different posterior assignments allowing them to separate.
3. The initialization can succeed. The posterior assignments in the first E-step differ. This can fail if one or more Gaussians are centered on outlier points. The resulting posterior assignments would progressively focus around those individual points and the Gaussians would peak (variance going to zero) around those individual points.
4. The initialization fails. This is the same case as the first one. The posterior assignments are spread uniformly across the points for each Gaussian. Specifically, each point is assigned to each Gaussian with posterior probability that equals the prior mixing proportion. These probabilities do not vary from point to point. As a result, the weighted mean, variance estimates of the Gaussians in the M-step are the same for all Gaussians, and the problem repeats.

6. In this question and the next we revisit the ELBO introduced in Section 2, and show how it can be also used to view the EM algorithm for estimating a mixture of k spherical Gaussians model.

   Let $D = \{x^i\}_{i=1,\ldots,n}$ be our observed data where $x^i \in \mathbb{R}^d$. Given any choice of distributions $Q(z|x^i)$, provide explicit parameter estimates of the mixture model as a function of these choices (M-step). In other words, solve $\hat{\theta} = \arg\max_\theta \text{ELBO}(Q;\theta)$ where

$$\text{ELBO}(Q;\theta) = \sum_{i=1}^{n} \left\{ \sum_{z=1}^{k} Q(z|x^i) \log \left[p_z N(x^i; \mu_z, \sigma_z^2 I) \right] + H(Q_{z|x^i}) \right\}$$

Solution: *Key points:* Derive maximizing values for each μ_z, σ_z, and p_z, by taking gradients or applying inequalities. In the case of p_z, the solution must take into account the constraint that $\sum_{z=1}^{k} p_z = 1$, e.g. using Lagrange multipliers.

We derive the maximizing p_z, μ_z, σ_z^2, $z \in [k]$, either by finding stationary points of the ELBO, or performing the Lagrange multipliers method on the ELBO whenever one of our maximizing parameters is constrained. Note that, given the choice of posterior $Q(z|x^i)$, the $H(Q_{z|x^i})$ term vanishes in any partial derivative of ELBO with respect to the parameters. In addition, for clarity, let $g_z(x^i)$ denote the probability density function of $N(x^i; \mu_z, \sigma_z^2 I)$ evaluated at x^i.

First we show the derivation of maximizing means μ_z, $z \in [k]$. Each μ_z is independent, and for any given μ_z, the stationary point is given by:

$$\begin{aligned}
\frac{\partial}{\partial \mu_z}\text{ELBO}(Q;\theta) &= \frac{\partial}{\partial \mu_z}\sum_{i=1}^{n}\sum_{z'=1}^{k} Q(z'|x^i)(\log p_{z'} + \log g_{z'}(x^i)) = 0 \\
&= \frac{\partial}{\partial \mu_z}\sum_{i=1}^{n} Q(z|x^i)\left(-\frac{1}{2\sigma_z^2}\|x^i - \mu_z\|^2\right) = 0 \\
&\Rightarrow \sum_{i=1}^{n} Q(z|x^i)\frac{1}{\sigma^2}(x^i - \mu_z) = 0 \\
&\Rightarrow \mu_z \sum_{i=1}^{n} Q(z|x^i) = \sum_{i=1}^{n} Q(z|x^i)x^i
\end{aligned}$$

where in the second line we remove terms that don't depend on μ_z. So our estimate for μ_z is

$$\hat{\mu}_z = \frac{\sum_{i=1}^{n} Q(z|x^i)x^i}{\sum_{i=1}^{n} Q(z|x^i)}.$$

The variances σ_z^2 are similarly unconstrained and maximized at the stationary points of the ELBO. Notice that our process of arg-maximizing the μ_z did not depend on the σ_z^2, so we can use our chosen μ_z as a given in our process for arg-maximizing σ_z^2. We perform a similar calculation as the above μ_z derivation:

$$\begin{aligned}
\frac{\partial}{\partial \sigma_z^2}\text{ELBO}(Q;\theta) &= \frac{\partial}{\partial \sigma_z^2}\sum_{i=1}^{n}\sum_{z'=1}^{k} Q(z'|x^i)(\log p_{z'} + \log g_{z'}(x^i)) = 0 \\
&= \frac{\partial}{\partial \sigma_z^2}\sum_{i=1}^{n} Q(z|x^i)\left(-\frac{d}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma_z^2}\|x^i - \mu_z\|^2\right) = 0 \\
&\Rightarrow \sum_{i=1}^{n} Q(z|x^i)\left(-\frac{d}{2\sigma_z^2} + \frac{1}{2\sigma^4}\|x^i - \mu_z\|^2\right) = 0 \\
&\Rightarrow \sigma^2 d \sum_{i=1}^{n} Q(z|x^i) = \sum_{i=1}^{n} Q(z|x^i)\|x^i - \mu_z\|^2 \\
&\Rightarrow \hat{\sigma}_z^2 = \frac{\sum_{i=1}^{n} Q(z|x^i)\|x^i - \mu_z\|^2}{d\sum_{i=1}^{n} Q(z|x^i)}.
\end{aligned}$$

Our weights p_z are constrained, so we maximize the ELBO subject to $\sum_{z=1}^{k} p_z = 1$. Applying Lagrangian multipliers, we minimize $\text{ELBO}(Q;\theta) - \lambda\left(\sum_{z=1}^{k} p_z - 1\right)$, where λ is the multiplier:

$$\frac{\partial}{\partial p_z}\left(\text{ELBO}(Q;\theta) - \lambda\left(\sum_{z=1}^{k} p_z - 1\right)\right) = \sum_{i=1}^{n} \frac{Q(z|x^i)}{p_z} - \lambda = 0$$

which gives

$$p_z = \frac{1}{\lambda}\sum_{i=1}^{n} Q(z|x^i).$$

In other words, we should have $p_z \propto \sum_{i=1}^{n} Q(z|x^i)$ scaled by a constant to satisfy the constraint $\sum_{z=1}^{k} p_z = 1$. We get

$$\lambda = \sum_{i=1}^{n}\sum_{z=1}^{k} Q(z|x^i) = n$$

so we conclude that the maximizing p_z is

$$\hat{p}_z = \frac{1}{n}\sum_{i=1}^{n} Q(z|x^i).$$

7. Recall that we can equivalently write the ELBO estimation criterion as

$$\text{ELBO}(Q;\theta) = \sum_{i=1}^{n}\left\{\log P(x^i|\theta) - \text{KL}(Q_{z|x^i} \| P_{z|x^i,\theta})\right\}$$

Show that when $Q(z|x^i) = P(z|x^i,\theta_0)$ for all $z \in [k]$ and $i = 1,\ldots,n$, then

$$\nabla_\theta \text{ KL}(Q_{z|x^i} \| P_{z|x^i,\theta})\big|_{\theta=\theta_0} = 0 \text{ (vector)}$$

for all $i = 1,\ldots,n$. This result ensures that $\nabla_\theta \text{ELBO}(Q;\theta)|_{\theta=\theta_0} = \nabla_\theta \sum_{i=1}^{n} \log P(x^i|\theta)|_{\theta=\theta_0}$ after each E-step. In other words, the lower bound criterion not only agrees in value at $\theta = \theta_0$ but it also has the same derivative as the log-likelihood.

Solution: *Key points:* Take the gradient of the KL divergence and show this is zero. Solutions should not evaluate the KL divergence at $\theta = \theta_0$ before taking the gradient.

Because $Q(z|x^i)$ does not depend on θ, the gradient of the KL divergence simplifies as:

$$\begin{aligned}\nabla_\theta \text{ KL}(Q_{z|x^i} \| P_{z|x^i,\theta}) &= \nabla_\theta \left(\sum_{z=1}^{k} Q(z|x^i)\log Q(z|x^i) - Q(z|x^i)\log P(z|x^i,\theta)\right) \\ &= \sum_{z=1}^{k} -\frac{Q(z|x^i)}{P(z|x^i,\theta)}\nabla_\theta P(z|x^i,\theta).\end{aligned}$$

Evaluating the gradient at $\theta = \theta_0$, the coefficients cancel since $Q(z|x^i) = P(z|x^i, \theta_0)$, resulting in

$$\nabla_\theta \, \mathrm{KL}(Q_{z|x^i} \| P_{z|x^i,\theta})\big|_{\theta=\theta_0} = \sum_{z=1}^{k} -\nabla_\theta P(z|x^i, \theta)\big|_{\theta=\theta_0}.$$

Finally we move the gradient outside the sum and apply the constraint $\sum_{z=1}^{k} P(z|x^i, \theta) = 1$ for all $i = 1, \ldots, n$.

$$\nabla_\theta \, \mathrm{KL}(Q_{z|x^i} \| P(z|x^i, \theta))\big|_{\theta=\theta_0} = \nabla_\theta \left(\sum_{z=1}^{k} -P(z|x^i, \theta) \right) \Bigg|_{\theta=\theta_0} = \nabla_\theta \, (-1)\big|_{\theta=\theta_0} = 0.$$

4 Diffusion models

8. Let's consider a simple diffusion model in 2D. In other words, we are generating samples $x \in \mathbb{R}^2$. The dataset available to us consists of only two points, $[1, 0]^T$ and $[0, 1]^T$.

   (a) Let β_t, $t = 1, 2, \ldots, T$ refer to the noise variance we add at step t. In other words, at step t in the forward process we update the example according to $x_t = \sqrt{1 - \beta_t} x_{t-1} + \sqrt{\beta_t} \epsilon_t$, where $\epsilon_t \sim N(0, I)$. Let $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. What is the resulting forward model distribution at step t conditioned on x_0? Hint: you can start by writing x_2 as a linear combination of x_0 and Gaussian noise $\epsilon \sim N(0, I)$ and note I is the identity 2d matrix.

   Solution: We can iterate the noise updates to get

   $$x_2 = \sqrt{1 - \beta_2}(\sqrt{1 - \beta_1} x_0 + \sqrt{\beta_1} \epsilon_1) + \sqrt{\beta_2} \epsilon_2$$

   where $\epsilon_i \sim N(0, I)$ are independent. Simplifying this further, we get $x_2 = \sqrt{\bar{\alpha}_2} x_0 + \sqrt{1 - \bar{\alpha}_2} \epsilon$, $\epsilon \sim N(0, I)$. Note that we only need to match the means and variances of this x_t expression conditioned on x_0 since it is distributed like a Gaussian. Extending this argument for all t, we get

   $$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, \quad \epsilon \sim N(0, I)$$

   In terms of probability distributions, we can write

   $$q_{t|0}(x|x_0) = N(x|\sqrt{\bar{\alpha}_t} x_0, (1 - \bar{\alpha}_t) I)$$

   which we also refer to as "transition kernel".

   (b) Since the forward process is applied the same to each example in our dataset, we can ask what the distribution is over x_t marginally across the examples. Write down an expression for this distribution. You can assume that the examples are selected with equal probability, i.e., $q(x_0) = 1/2$ for $x_0 = [1, 0]^T$ or $x_0 = [0, 1]^T$.

Solution: We get a mixture distribution over the examples.

$$\begin{aligned} q_t(x) &= \sum_{x_0} q(x_0) q_{t|0}(x|x_0) \\ &= \sum_{x_0} q(x_0) N(x|\sqrt{\bar{\alpha}_t} x_0, (1-\bar{\alpha}_t) I) \\ &= \frac{1}{2} N(x|\sqrt{\bar{\alpha}_t}[1,0]^T, (1-\bar{\alpha}_t) I) + \frac{1}{2} N(x|\sqrt{\bar{\alpha}_t}[0,1]^T, (1-\bar{\alpha}_t) I) \end{aligned}$$

(c) Suppose we use a simple estimation criterion for our reverse process, i.e., we find $\epsilon_\theta(x_t, t)$ that minimizes

$$E_{x_0,t,\epsilon}\left\{\|\epsilon - \epsilon_\theta(x_t(x_0,\epsilon), t)\|^2\right\}$$

where $x_t(x_0,\epsilon) = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$ and $x_0 \sim q(x_0)$, $\epsilon \sim N(0, I)$. Consider a fixed $\hat{x}$. What is the resulting optimal estimate for $\epsilon_\theta(\hat{x}, t)$ if our reverse model can be arbitrarily complex? Write down the solution as an expression involving ϵ, $x_t(x_0,\epsilon)$ and $\hat{x}$.

Solution: The squared error is minimized at the conditional mean, i.e., we wish to calculate

$$E\{\epsilon | x_t(x_0,\epsilon) = \hat{x}\}$$

where x_0 is also a random variable in this expression.

(d) To evaluate your answer to the previous question note that you can think of the problem in terms of a graphical model $x_0 \to x_t$, $\epsilon \to x_t$ where we know the marginal distributions over x_0 and ϵ and how they give rise to x_t through $x_t(x_0,\epsilon)$. We observe $x_t = \hat{x}$ and wish to calculate the resulting posterior over ϵ. What is this posterior?

Solution: Given $\hat{x}$, the posterior over ϵ can be calculated directly from the graphical model:

$$P(\epsilon, \hat{x}) = \sum_{x_0} q(x_0) N(\epsilon|0, I) P(x_t = \hat{x}|x_0, \epsilon)$$

where $P(x_t = \hat{x}|x_0,\epsilon)$ enforces that $\sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon = \hat{x}$. Thus ϵ can only take two possible values, one for each value of x_0:

$$\epsilon(x_0, t) = (\hat{x} - \sqrt{\bar{\alpha}_t} x_0)/\sqrt{1-\bar{\alpha}_t}$$

each with probability proportional to $q(x_0) N(\,\epsilon(x_0,t)\,|0, I)$.

(e) Briefly describe how the optimal answer for the reverse process, i.e., our estimate $\epsilon_\theta(\hat{x}, t)$ for a fixed $\hat{x}$, behaves as t becomes very large.

Solution: Both of the answers that ϵ can take converge to $\hat{x}$ as $\bar{\alpha}_t \to 0$. Thus the conditional expectation will also be $\hat{x}$.

5 Flow Matching

9. This question is based on lecture 23; for more background see Yaron Lipman, Ricky T.Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le, "Flow Matching for Generative Modeling."

   Given sample space $\mathbb{R}^d$ we can use a continuous probability "flow" to represent a target distribution $p_1(x)$, by learning an invertible transformation from some simple known distribution $p_0(x)$. We will use this method to model a data distribution $q(x) \sim \text{Unif}(\{x^{(1)}, \ldots, x^{(n)}\})$.

   (a) A *probability flow* is continuous time-indexed function, such that for all $t \in [0,1]$, p_t is a pdf over $\mathbb{R}^d$. We are going to think about the probability flow[1] induced by fixing p_0 and p_1, then defining a distribution of linear paths

   $$x_0 \sim p_0(x)$$
   $$x_1 \sim p_1(x)$$
   $$x_t = (1-t)x_0 + tx_1$$

   We'll start by considering the case of a fixed x_1. If $x_0 \sim \mathcal{N}(0, I)$, what is the distribution $p_t(x_t \mid x_1)$ such that $x_t \sim p_t$?

   Solution: This is a linear combination of two Gaussian random variables. We know (and not too hard to show!) that if $Y = a_1X_1 + a_2X_2$ where $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$ and $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$, then $Y \sim \mathcal{N}(a_1\mu_1 + a_2\mu_2, a_1^2\sigma_1^2 + a_2^2\sigma_2^2)$. So,

   $$x_t \mid x_1 \sim \mathcal{N}(tx_1, (1-t)^2 I)$$

   (b) What is $p_t(x \mid x_1)$ as $t \to 1$?

   Solution: It approaches a delta distribution that assigns probability 1 to x_1.

   (c) Now, more generally, if $x_1 \sim p_1(x)$ what is $p_t(x)$?

   Solution:

   $$p_t(x) = \int_{x_1} p_t(x \mid x_1) p(x_1) dx_1$$

   (d) The whole reason we're trying to find alternative ways of thinking about learning p_1 is that complicated densities are hard to represent and learn directly. An alternative parameterization of the whole probability flow p_t is in terms of a time varying vector field $dx_t/dt = v_t(x_t)$, that intuitively has the property that if we start with p_0, and let the probability "flow" as specified by this vector field, the distributions p_t will match the ones we desire and, in particular, will converge to p_1 as $t \to 1$.

[1] This is called a *probability path* in the paper.

The *continuity equation* from fluid flow tells us the relationship between this velocity field and the probability flow:

$$\frac{d}{dt}p_t(x) = -\nabla_x \cdot (p_t(x)v_t(x))$$

In one dimension, for intuition, this is simply[2]

$$\frac{d}{dt}p_t(x) = -\frac{d}{dx}p_t(x)v_t(x)$$

So now. We know what we want our probability flow to be: p_t. What is the v_t that will result in our desired p_t?

$$v_t(x) = \int_{x_1} \frac{x_1 - x}{1-t} p(x_1 \mid x, t)\, dx_1$$

Explain intuitively why this makes sense.

Solution: If we know x_1 (a "target" x value), and we are currently at x at time t, then we only have time $1-t$ left, and so we had better have velocity $\frac{x_1-x}{1-t}$ in order to make it there in time!

But, if we just know we are at x at time t, we don't know where we should be heading (that is, we don't know x_1). So, we'll have to integrate over possible target points, given where we are now.

(e) What is $p(x_1 \mid x, t)$?

Solution:

$$p(x_1 \mid x, t) = \frac{p_t(x \mid x_1)p_1(x)}{p_t(x)}$$

(f) Show that our definitions of $v_t(x)$ and $p_t(x)$ satisfy the continuity equation, in 1D, and assuming $x_0 \sim \mathcal{N}(0, 1)$.

It's kind of tedious to do by hand (and Mathematica can do it!) so fine to use the fact that for a fixed x_1,

$$\frac{d}{dt}p_t(x \mid x_1) = -\frac{d}{dx}(p_t(x \mid x_1)v_t(x \mid x_1))$$
$$\frac{d}{dt}\mathcal{N}(tx_1, (1-t)^2) = -\frac{d}{dx}\mathcal{N}(tx_1, (1-t)^2)\frac{x_1 - x}{1-t}$$

[2]You may have forgotten the notation but $\nabla_x \cdot$ is the *divergence* operator. Go look it up.

Solution:

$$\frac{d}{dt}p_t(x) = -\nabla_x \cdot (p_t(x)v_t(x))$$

$$\begin{aligned}\frac{d}{dt}\int_{x_1} p_t(x \mid x_1)p_1(x_1)dx_1 &= -\nabla_x \cdot \left(p_t(x)\int_{x_1} v_t(x \mid x_1)p_t(x_1 \mid x)dx_1\right) \\ &= -\nabla_x \cdot \left(p_t(x)\int_{x_1} v_t(x \mid x_1)\frac{p_t(x \mid x_1)p_1(x_1)}{p_t(x)}dx_1\right) \\ &= -\nabla_x \cdot \left(\int_{x_1} v_t(x \mid x_1)p_t(x \mid x_1)p_1(x_1)dx_1\right)\end{aligned}$$

$$\int_{x_1}\left(\frac{d}{dt}p_t(x \mid x_1)\right)p_1(x_1)dx_1 = \int_{x_1}\Big(-\nabla_x \cdot \big(v_t(x \mid x_1)p_t(x \mid x_1)\big)\Big)p_1(x_1)dx_1$$

By the fact in the problem, the expressions inside the parens on both sides are equal, so these are equal.

(g) Whew! The key takeaway here was that we showed letting the velocities be

$$v_t(x) = \int_{x_1} \frac{x_1 - x}{1-t}p_t(x_1 \mid x)dx_1$$

would yield the right probability flow.
Use this insight to describe a stochastic gradient descent training procedure for learning a neural-network approximation $v_\theta(x,t)t$ to v_t from dataset $\mathcal{D} = \{x^{(1)}, \ldots, x^{(n)}\}$.

Solution: Draw

$$\begin{aligned}x_0 &\sim p_0(x) \\ x_1 &\sim \text{Unif}(\mathcal{D}) \\ t &\sim \text{Unif}(0,1) \\ x_t &= (1-t)x_0 + tx_1\end{aligned}$$

Take a gradient step

$$\theta = \theta - \eta\nabla_\theta\big(v_\theta(x_t,t) - \frac{x_1 - x_t}{1-t}\big)^2$$

(h) Finally, once we have trained $v_\theta(x_t,t)$, how do we sample from $\hat{p_1}$?

Solution: Pick a number of iterations m

```
x ~ Normal(0, I)
for i in [0..m-1]:
    t = i / m
```

```
    x = x + v(x, t) / (m-i)
return x
```