Homework 3

Deadline 2024/12/15 23:59

1 KL divergence and Maximum Likelihood

The Kullback-Leibler (KL) divergence is a measure of how much one probability distribution is different from a second one. It is a concept that originated in Information Theory, but has made its way into several other fields, including Statistics, Machine Learning, Information Geometry, and many more. In Machine Learning, the KL divergence plays a crucial role, connecting various concepts that might otherwise seem unrelated.

In this problem, we will introduce KL divergence over discrete distributions, practice some simple manipulations, and see its connection to Maximum Likelihood Estimation.

The KL divergence between two discrete-valued distributions $P(X), Q(X)$ over the outcome space $\mathcal{X}$ is defined as follows:

$$D_{KL}(P\|Q) = \sum_{x\in\mathcal{X}} P(x)\log\frac{P(x)}{Q(x)}$$

For notational convenience, we assume $P(x) > 0, \forall x$. (One other standard thing to do is to adopt the convention that "0 log 0 = 0.") Sometimes, we also write the KL divergence more explicitly as $D_{KL}(P||Q) = D_{KL}(P(X)||Q(X))$.

Background on Information Theory

Before we dive deeper, we give a brief (optional) Information Theoretic background on KL divergence. While this introduction is not necessary to answer the assignment question, it may help you better understand and appreciate why we study KL divergence, and how Information Theory can be relevant to Machine Learning.

We start with the entropy $H(P)$ of a probability distribution $P(X)$, which is defined as

$$H(P) = -\sum_{x\in\mathcal{X}} P(x)\log P(x).$$

Intuitively, entropy measures how dispersed a probability distribution is. For example, a uniform distribution is considered to have very high entropy (i.e. a lot of uncertainty), whereas a distribution that assigns all its mass on a single point is considered to have zero entropy (i.e. no uncertainty). Notably, it can be shown that among continuous distributions over $\mathbb{R}$, the Gaussian distribution $N(\mu, \sigma^2)$ has the highest entropy (highest uncertainty) among all possible distributions that have the given mean μ and variance σ^2.

To further solidify our intuition, we present motivation from communication theory. Suppose we want to communicate from a source to a destination, and our messages are always (a sequence of) discrete symbols over space X (for example, X could be letters $\{a, b, \ldots, z\}$). We want to construct an encoding scheme for our symbols in the form of sequences of binary bits that are transmitted over the channel. Further, suppose that in the long run the frequency of occurrence of symbols follow a probability distribution $P(X)$. This means, in the long run, the fraction of times the symbol x gets transmitted is $P(x)$.

A common desire is to construct an encoding scheme such that the average number of bits per symbol transmitted remains as small as possible. Intuitively, this means we want very frequent symbols to be assigned to a bit pattern having a small number of bits. Likewise, because we are interested in reducing the average number of bits per symbol in the long term, it is tolerable for infrequent words to be assigned to bit patterns having a large number of bits, since their low frequency has little effect on the long term average. The encoding scheme can be as complex as we desire, for example, a single bit could possibly represent a long sequence of multiple symbols (if that specific pattern of symbols is very common). The entropy of a probability distribution $P(X)$ is its optimal bit rate, i.e., the lowest average bits per message that can possibly be achieved if the symbols $x \in \mathcal{X}$ occur according to $P(X)$. It does not specifically tell us how to construct that optimal encoding scheme. It only tells us that no encoding can possibly give us a lower long term bits per message than $H(P)$.

To see a concrete example, suppose our messages have a vocabulary of $K = 32$ symbols, and each symbol has an equal probability of transmission in the long term (i.e, uniform probability distribution). An encoding scheme that would work well for this scenario would be to have $\log_2 K$ bits per symbol, and assign each symbol some unique combination of the $\log_2 K$ bits. In fact, it turns out that this is the most efficient encoding one can come up with for the uniform distribution scenario.

It may have occurred to you by now that the long term average number of bits per message depends only on the frequency of occurrence of symbols. The encoding scheme of scenario A can in theory be reused in scenario B with a different set of symbols (assume equal vocabulary size for simplicity), with the same long term efficiency, as long as the symbols of scenario B follow the same probability distribution as the symbols of scenario A. It might also have occurred to you, that reusing the encoding scheme designed to be optimal for scenario A, for messages in scenario B having a different probability of symbols, will always be suboptimal for scenario B. To be clear, we do not need know what the specific optimal schemes are in either scenarios. As long as we know the distributions of their symbols, we can say that the optimal scheme designed for scenario A will be suboptimal for scenario B if the distributions are different.

Concretely, if we reuse the optimal scheme designed for a scenario having symbol distribution $Q(X)$, into a scenario that has symbol distribution $P(X)$, the long term average number of bits per symbol achieved is called the cross entropy, denoted by $H(P, Q)$:

$$H(P, Q) = -\sum_{x \in \mathcal{X}} P(x) \log Q(x).$$

To recap, the entropy $H(P)$ is the best possible long term average bits per message (optimal) that can be achieved under a symbol distribution $P(X)$ by using an encoding scheme (possibly unknown) specifically designed for $P(X)$. The cross entropy $H(P, Q)$ is the long term average bits per message (suboptimal) that results under a symbol distribution $P(X)$, by reusing an encoding scheme (possibly unknown) designed to be optimal for a scenario with symbol distribution $Q(X)$.

Now, KL divergence is the penalty we pay, as measured in average number of bits, for using the optimal scheme for $Q(X)$, under the scenario where symbols are actually distributed as $P(X)$. It is straightforward to see this

$$\begin{aligned} D_{KL}(P\|Q) &= \sum_{x \in \mathcal{X}} P(x) \log \frac{P(x)}{Q(x)} \\ &= \sum_{x \in \mathcal{X}} P(x) \log P(x) - \sum_{x \in \mathcal{X}} P(x) \log Q(x) \\ &= H(P, Q) - H(P). \quad \text{(difference in average number of bits.)} \end{aligned}$$

If the cross entropy between P and Q is $H(P)$ (and hence $D_{KL}(P||Q) = 0$) then it necessarily means $P = Q$. In Machine Learning, it is a common task to find a distribution Q that is "close" to another distribution P . To achieve this, it is common to use $D_{KL}(Q||P)$ as the loss function to be optimized. As we will see in this question below, Maximum Likelihood Estimation, which is a commonly used optimization objective, turns out to be equivalent to minimizing the KL divergence between the training data (i.e. the empirical distribution over the data) and the model.

Now, we get back to showing some simple properties of KL divergence.

(a) Nonnegativity.

Prove the following:

$$\forall P, Q. \quad D_{KL}(P\|Q) \geq 0$$

and

$$D_{KL}(P\|Q) = 0 \quad \text{if and only if} \quad P = Q.$$

[Hint: You may use the following result, called Jensen's inequality. If f is a convex function, and X is a random variable, then $E[f(X)] \geq f(E[X])$. Moreover, if f is strictly convex (f is convex if its Hessian satisfies $H \geq 0$; it is strictly convex if $H > 0$; for instance $f(x) = -\log x$ is strictly convex), then $E[f(X)] = f(E[X])$ implies that $X = E[X]$ with probability 1; i.e., X is actually a constant.]

(b) Chain rule for KL divergence.

The KL divergence between 2 conditional distributions $P(X|Y)$, $Q(X|Y)$ is defined as follows:

$$D_{KL}(P(X|Y)\|Q(X|Y)) = \sum_y P(y)\left(\sum_x P(x|y)\log\frac{P(x|y)}{Q(x|y)}\right)$$

This can be thought of as the expected KL divergence between the corresponding conditional distributions on x (that is, between $P(X|Y=y)$ and $Q(X|Y=y)$), where the expectation is taken over the random y.

Prove the following chain rule for KL divergence:

$$D_{KL}(P(X,Y)\|Q(X,Y)) = D_{KL}(P(X)\|Q(X)) + D_{KL}(P(Y|X)\|Q(Y|X)).$$

(c) KL and maximum likelihood.

Consider a density estimation problem, and suppose we are given a training set $\{x^{(i)}; i=1,\ldots,n\}$. Let the empirical distribution be $\hat{P}(x) = \frac{1}{n}\sum_{i=1}^{n} 1\{x^{(i)} = x\}$. ($\hat{P}(x)$ is just the uniform distribution over the training set; i.e., sampling from the empirical distribution is the same as picking a random example from the training set.)

Suppose we have some family of distributions P_θ parameterized by θ. (If you like, think of $P_\theta(x)$ as an alternative notation for $P(x;\theta)$.) Prove that finding the maximum likelihood estimate for the parameter θ is equivalent to finding P_θ with minimal KL divergence from $\hat{P}(x)$. I.e. prove:

$$\arg\min_\theta D_{KL}(\hat{P}\|P_\theta) = \arg\max_\theta \sum_{i=1}^{n}\log P_\theta(x^{(i)})$$

Remark. Consider the relationship between parts (b-c) and multi-variate Bernoulli Naive Bayes parameter estimation. In the Naive Bayes model we assumed P_θ is of the following form: $P_\theta(x,y) = p(y)\prod_{i=1}^{d} p(x_i|y)$. By the chain rule for KL divergence, we therefore have:

$$D_{KL}(\hat{P}\|P_\theta) = D_{KL}(\hat{P}(y)\|p(y)) + \sum_{i=1}^{d} D_{KL}(\hat{P}(x_i|y)\|p(x_i|y))$$

This shows that finding the maximum likelihood/minimum KL-divergence estimate of the parameters decomposes into $2n+1$ independent optimization problems: One for the class priors $p(y)$, and one for each of the conditional distributions $p(x_i|y)$ for each feature x_i given each of the two possible labels for y. Specifically, finding the maximum likelihood estimates for each of these problems individually results in also maximizing the likelihood of the joint distribution. (If you know what Bayesian networks are, a similar remark applies to parameter estimation for them.)

2 Semi-supervised EM

Expectation Maximization (EM) is a classical algorithm for unsupervised learning (i.e., learning with hidden or latent variables). In this problem we will explore one of the ways in which EM algorithm can be adapted to the semi-supervised setting, where we have some labeled examples along with unlabeled examples.

In the standard unsupervised setting, we have $n \in \mathbb{N}$ unlabeled examples $\{x^{(1)},\ldots,x^{(n)}\}$. We wish to learn the parameters of $p(x,z;\theta)$ from the data, but $z^{(i)}$'s are not observed. The classical EM algorithm is designed for this very purpose, where we maximize the intractable $p(x;\theta)$ indirectly by iteratively performing the E-step and M-step, each time maximizing a tractable lower bound of $p(x;\theta)$. Our objective can be concretely written as:

$$\begin{aligned}\ell_{\text{unsup}}(\theta) &= \sum_{i=1}^{n}\log p(x^{(i)};\theta)\\ &= \sum_{i=1}^{n}\log\sum_{z^{(i)}} p(x^{(i)},z^{(i)};\theta)\end{aligned}$$

Now, we will attempt to construct an extension of EM to the semi-supervised setting. Let us suppose we have an additional $\tilde{n} \in \mathbb{N}$ labeled examples $\{(\tilde{x}^{(1)}, \tilde{z}^{(1)}), \ldots, (\tilde{x}^{(\tilde{n})}, \tilde{z}^{(\tilde{n})})\}$ where both x and z are observed. We want to simultaneously maximize the marginal likelihood of the parameters using the unlabeled examples, and full likelihood of the parameters using the labeled examples, by optimizing their weighted sum (with some hyperparameter α). More concretely, our semi-supervised objective $\ell_{\text{semi}-\text{sup}}(\theta)$ can be written as:

$$\ell_{\text{sup}}(\theta) = \sum_{i=1}^{\tilde{n}} \log p(\tilde{x}^{(i)}, \tilde{z}^{(i)}; \theta)$$
$$\ell_{\text{semi}-\text{sup}}(\theta) = \ell_{\text{unsup}}(\theta) + \alpha \ell_{\text{sup}}(\theta)$$

We can derive the EM steps for the semi-supervised setting using the same approach and steps as before. You are strongly encouraged to show to yourself (no need to include in the write-up) that we end up with:

E-step (semi-supervised)

For each $i \in \{1, \ldots, n\}$, set

$$Q_i^{(t)}(z^{(i)}) := p(z^{(i)} | x^{(i)}; \theta^{(t)})$$

M-step (semi-supervised)

$$\theta^{(t+1)} := \arg\max_{\theta} \left[\sum_{i=1}^{n} \left(\sum_{z^{(i)}} Q_i^{(t)}(z^{(i)}) \log \frac{p(x^{(i)}, z^{(i)}; \theta)}{Q_i^{(t)}(z^{(i)})} \right) + \alpha \left(\sum_{i=1}^{\tilde{n}} \log p(\tilde{x}^{(i)}, \tilde{z}^{(i)}; \theta) \right) \right]$$

(a) Convergence.

First we will show that this algorithm eventually converges. In order to prove this, it is sufficient to show that our semi-supervised objective $\ell_{\text{semi}-\text{sup}}(\theta)$ monotonically increases with each iteration of E and M step. Specifically, let $\theta(t)$ be the parameters obtained at the end of t EM-steps. Show that $\ell_{\text{semi}-\text{sup}}(\theta^{(t+1)}) \geq \ell_{\text{semi}-\text{sup}}(\theta^{(t)})$.

Semi-supervised GMM

Now we will revisit the Gaussian Mixture Model (GMM), to apply our semi-supervised EM algorithm. Let us consider a scenario where data is generated from $k \in \mathbb{N}$ Gaussian distributions, with unknown means $\mu_j \in \mathbb{R}^d$ and covariances $\Sigma_j \in \mathbb{S}_+^d$ where $j \in \{1, \ldots, k\}$. We have n data points $x^{(i)} \in \mathbb{R}^d, i \in \{1, \ldots, n\}$, and each data point has a corresponding latent (hidden/unknown) variable $z^{(i)} \in \{1, \ldots, k\}$ indicating which distribution $x^{(i)}$ belongs to. Specifically, $z^{(i)} \sim \text{Multinomial}(\phi)$, such that $\sum_{j=1}^{k} \phi_j = 1$ and $\phi_j \geq 0$ for all j, and $x^{(i)} | z^{(i)} \sim \mathcal{N}(\mu_{z^{(i)}}, \Sigma_{z^{(i)}})$ i.i.d. So, μ, Σ and ϕ are model parameters.

We also have additional $\tilde{n}$ data points $\tilde{x}^{(i)} \in \mathbb{R}^d, i \in \{1, \ldots, \tilde{n}\}$, and n associated observed variable $\tilde{z}^{(i)} \in \{1, \ldots, k\}$ indicating the distribution $\tilde{x}^{(i)}$ belongs to. Note that $\tilde{z}^{(i)}$ are known constants (in contrast to $z^{(i)}$ which are unknown random variables). As before, we assume $\tilde{x}^{(i)} | \tilde{z}^{(i)} \sim \mathcal{N}(\mu_{\tilde{z}^{(i)}}, \Sigma_{\tilde{z}^{(i)}})$ i.i.d.

In summary we have $n + \tilde{n}$ examples, of which n are unlabeled data points x's with unobserved z's, and $\tilde{n}$ are labeled data point $\tilde{x}^{(i)}$ with corresponding observed label $\tilde{z}^{(i)}$. he traditional EM algorithm is designed to take only the n unlabeled examples as input, and learn the model parameters μ, Σ and ϕ.

Our task now will be to apply the semi-supervised EM algorithm to GMMs in order to also leverage the additional $\tilde{n}$ labeled examples, and come up with semi-supervised E-step and M-step update rules specific to GMMs. Whenever required, you can cite the lecture notes for derivations and steps.

(b) Semi-supervised E-Step

Clearly state which are all the latent variables that need to be re-estimated in the E-step. Derive the E-step to re-estimate all the stated latent variables. Your final E-step expression must only involve x, z, μ, Σ, ϕ and universal constants. (TA's notes: You need to compute the coefficient $w_j^{(i)}$ of the latent variable data point (i) w.r.t class j.)

(c) Semi-supervised M-Step

Clearly state which are all the parameters that need to be re-estimated in the M-step. Derive the M-step to re-estimate all the stated parameters. Specifically, derive closed form expressions for the parameter update rules for $\mu^{(t+1)}, \Sigma^{(t+1)}$ and $\phi^{(t+1)}$ based on the semi-supervised objective.

3 Store Cluster

TA's reminder: For (a) (b) (c), just give the answer without any process; for (d) (e) (f), write simple process within 10 lines.

Amazing, Inc., is going to open their first k stores in the greater Megacity region. They know that the ith customer (out of n total customers) lives at location $x^{(i)}$. They will place the jth store (for $j = 1, \ldots, k$) at location $\mu^{(j)}$. Amazing, Inc., would like to minimize the squared distance between customer locations and stores, and they decide to use k-means clustering to choose their store locations.

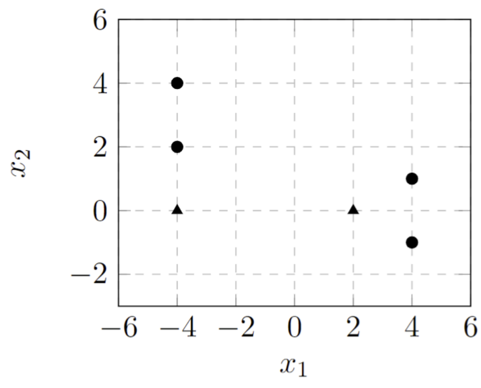
Amazing, Inc., wants to focus on where to place their stores, and decides to formulate the problem as finding the μ that minimizes a loss function

$$L(\mu) = \sum_{i=1}^{n} \min_{j \in 1,\ldots,k} \left\| x^{(i)} - \mu^{(j)} \right\|^2$$

Here $L(\mu)$ is the value of the k-means objective after we have picked the optimal assignments of the data points to cluster means (that's what the $\min_j$ does), for the given set of cluster means μ.

(a)

Amazing, Inc., has four customers in Megacity, located at positions (x_1, x_2) as shown by the circles in the plot below, and plans to build two stores. They have initial guesses for these two store locations, as marked by the triangles below. What is the starting loss $L(\mu)$? (TA's notes: assume two points on the edge have integer coordinations.)



(b)

Amazing, Inc., runs the k-means clustering algorithm starting from the initial guess shown above, until convergence (no further improvements in loss $L(\mu)$ can be made). Calculate coordinations of final locations of the stores and the final loss $L(\mu)$.

(c)

After several years, Amazing Inc., has grown tremendously. Now rather than tracking individual customers, they instead have a database consisting of r records: the ith record, $i = 1, \ldots, r$, provides both $c^{(i)}$ and $x^{(i)}$, where $c^{(i)}$ is the number of customers that are located at position $x^{(i)}$. They still want to put each store (indexed by $j = 1, \ldots, k$) at a position $\mu^{(j)}$ that minimizes the squared distances between customers and store locations, summed up over all customers. They just don't have individual customer data anymore.

Define a new objective $L_C(\mu)$ to account for the fact that now Amazing, Inc., has access only to record data instead of individual customer data. Please define any new expressions that you use within your formulation.

(d)

Amazing, Inc., test-runs their approach in the One-di City, which has a small number of customers. Their data set $\mathcal{D}$ consists of one-dimensional locations x and customer counts c as pairs, $(x, c) : \mathcal{D} = ((-1, 10), (1, 4))$. Amazing, Inc., is only going to build one store: where should it be located? Show your work.

(e)

Amazing, Inc., also has a single distribution center (DC) in Megacity, located at x_{DC} , that supplies all of their stores in Megacity. There is a cost associated with transporting goods from the DC to each of the stores, that grows with both squared distance and with the number of customers served by the store. Specifically, for each store, that cost is equal to the number of customers assigned to cluster j times the square of the distance from $\mu^{(j)}$ to x_{DC} . Amazing, Inc., would like to minimize both its own tranportation costs and the transportation cost of customers visiting their stores, the latter of which is the loss from (c). To that end, they decide to minimize the sum of these two costs.

Define the loss function $L_S(\mu)$ hat expresses the overall loss that Amazing Inc. is seeking to minimize: the sum of the Amazing, Inc., transportation costs and the customer-to- store costs. Please define any new expressions that you use within your formulation. All of your definitions should be in equations that ultimately depend only on quantities defined in this problem, not just definitions in words. You may find it useful to define $y^{(i)} = \arg\min_j \left\|x^{(i)} - \mu^{(j)}\right\|^2$ for record i.

(f)

Amazing, Inc., returns to the One-di City to see where they should put their single store. The records are the same as in (d), but now we also know that their distribution center is located at $x_{DC} = 10$. If we aim to minimize the objective function desired in (e), at what location μ should their store be located?

4 Diffusion Model: An Example of Variational Inference

Given an observed sample $\boldsymbol{x}$ from a distribution of interest, the goal of a generative model is to learn to model its true data distribution $p(\boldsymbol{x})$. Once the model has learned this, we can generate new samples at will based on our approximate model. In addition, in some cases, we can also use the learned model to evaluate the likelihood of observed or sampled data.

However, directly learning the true data distribution $p(\boldsymbol{x})$ is usually a very difficult task. Diffusion models provide a way to map an arbitrary distribution $p(\boldsymbol{x})$ to a known distribution (such as a normal distribution), and then learn the inverse of this mapping to indirectly complete the generation task.

(a) Diffusion Process

Given a data distribution $\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0)$, we add gradually increasing noise to this distribution and construct the following Markov process:

$$\boldsymbol{x}_t = \sqrt{1-\beta_t}\boldsymbol{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}, \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}; \boldsymbol{0}, \boldsymbol{I})$$

Here $t \in \{1, 2, \cdots, T\}$, $\beta_t \in (0, 1)$ increase gradually. Denote $\alpha_t := 1 - \beta_t, \bar{\alpha}_t := \prod_{s=1}^{t} \alpha_s$. Prove:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\boldsymbol{I})$$

Reverse Process

When $T \to \infty, \beta_t \to 1$, the final result of diffusion process $q(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{x}_T; \boldsymbol{0}, \boldsymbol{I})$, we denote the Markov chain of the diffusion process as:

$$q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) = \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$$

By reversing the diffusion process above, we can gradually remove noise from a Gaussian distribution and "restore" the original data:

$$q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)=\frac{q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1},\boldsymbol{x}_0)q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0)}{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}=\mathcal{N}(\boldsymbol{x}_{t-1};\boldsymbol{\mu}_q(\boldsymbol{x}_t,\boldsymbol{x}_0),\boldsymbol{\Sigma}_q(t)\boldsymbol{I})$$

Here: (It is a given conclusion. Don't need to prove it.)

$$\boldsymbol{\mu}_q(\boldsymbol{x}_t,\boldsymbol{x}_0)=\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t+\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\boldsymbol{x}_0}{1-\bar{\alpha}_t}$$

$$\boldsymbol{\Sigma_q(t)}=\sigma_q^2\boldsymbol{I}=\frac{(1-\alpha_t)(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\boldsymbol{I}$$

(b) Variational Lower Bound

We do not know the transition probability $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ of the reverse process. Here we learn a $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ to model it. Under this representation, the joint distribution of the reverse process is:

$$p(\boldsymbol{x}_{0:T})=p(\boldsymbol{x}_T)\prod_{t=1}^{T}p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$$

Using maximum likelihood estimation, we want to maximize: $\log p(\boldsymbol{x}_0)$. Using variational inference, we can prove that the ELBO of the log-likelihood above is:

$$\mathbb{E}_{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}[\log\frac{p(\boldsymbol{x}_{0:T})}{q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)}]$$

Note that you need to express the result of variational inference in the form of ELBO + a KL divergence.

(Hint: For an unknown distribution $p(\boldsymbol{x})$, we can express it as $\frac{p(\boldsymbol{x},\boldsymbol{z})}{p(\boldsymbol{z}|\boldsymbol{x})}$, introducing a hidden variable $\boldsymbol{z}$)

(c) Optimization Goal

Combined with the above derivation, we can further split ELBO into the following three items:

$$\begin{aligned}ELBO=&\underbrace{\mathbb{E}_{q(\boldsymbol{x}_1|\boldsymbol{x}_0)}\left[\log p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_0\mid\boldsymbol{x}_1\right)\right]}_{\text{reconstruction term}}-\underbrace{D_{\mathrm{KL}}\left(q\left(\boldsymbol{x}_T\mid\boldsymbol{x}_0\right)\|p\left(\boldsymbol{x}_T\right)\right)}_{\text{prior matching term}}\\&-\sum_{t=2}^{T}\underbrace{\mathbb{E}_{q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\left[D_{\mathrm{KL}}\left(q\left(\boldsymbol{x}_{t-1}\mid\boldsymbol{x}_t,\boldsymbol{x}_0\right)\|p_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{t-1}\mid\boldsymbol{x}_t\right)\right)\right]}_{\text{denoising matching term}}\end{aligned}$$

The first term is replaced by the approximation of the third term at $t=1$, and the second term is irrelevant to the parameter $\boldsymbol{\theta}$ to be learned, so we ignore it.

Consider the $\boldsymbol{\mu}_q(\boldsymbol{x}_t,\boldsymbol{x}_0)$ we derived in (b) with respect to $\boldsymbol{x}_0$, and we learn a model to predict it $\hat{\boldsymbol{x}}_0=\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t,t)$

$$\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)=\frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})\boldsymbol{x}_t+\sqrt{\bar{\alpha}_{t-1}}(1-\alpha_t)\hat{\boldsymbol{x}}_0(\boldsymbol{x}_t,t)}{1-\bar{\alpha}_t}$$

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)=\mathcal{N}(\boldsymbol{x}_{t-1};\boldsymbol{\mu}_{\boldsymbol{\theta}},\boldsymbol{\Sigma}_q(t))$$

Prove that our final optimization goal is:

$$\arg_{\boldsymbol{\theta}}\max ELBO=\arg_{\boldsymbol{\theta}}\min\mathbb{E}_{t\sim\mathbb{U}[1,T],q(\boldsymbol{x}_t|\boldsymbol{x}_0)}\frac{\bar{\alpha}_{t-1}(1-\alpha_t)^2}{2\sigma_q^2(1-\bar{\alpha}_t)^2}[\|\hat{\boldsymbol{x}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t,t)-\boldsymbol{x}_0\|_2^2]$$

(Hint:

$$\begin{aligned}&D_{\mathrm{KL}}\left(\mathcal{N}\left(\boldsymbol{x};\boldsymbol{\mu}_x,\boldsymbol{\Sigma}_x\right)\|\mathcal{N}\left(\boldsymbol{y};\boldsymbol{\mu}_y,\boldsymbol{\Sigma}_y\right)\right)\\=&\frac{1}{2}\left[\log\frac{|\boldsymbol{\Sigma}_y|}{|\boldsymbol{\Sigma}_x|}-d+\operatorname{tr}\left(\boldsymbol{\Sigma}_y^{-1}\boldsymbol{\Sigma}_x\right)+\left(\boldsymbol{\mu}_y-\boldsymbol{\mu}_x\right)^T\boldsymbol{\Sigma}_y^{-1}\left(\boldsymbol{\mu}_y-\boldsymbol{\mu}_x\right)\right]\end{aligned}$$

, where d represents dimension.)