Elements of Machine Learning - 2021
Home Assignment 4

Raghavendra Selvan, Jens Petersen
Department of Computer Science
University of Copenhagen
{raghav,phup}@di.ku.dk

1 Introduction

Generative modelling in Machine Learning is formulated as the task of estimating the underlying data-generating distribution, $P_X(x)$, where X is the random variable of interest. X can represent any kind of signal ranging from scalar measurements to high dimensional data such as text, time series data or even videos.

Except in toy examples we never have access to the distribution $P_X(x)$ in analytical form or either via an efficient sampler. We can only *observe* this distribution from a subset of data samples. In this assignment we look at some interesting Machine Learning tasks based on modelling the data generating distribution in the context of *Unsupervised* learning.

1.1 Instructions

1. Read the task descriptions carefully and answer accordingly
2. You are expected to present discussions that reflect your understanding for each task. Solutions without discussions will carry no points.
3. Provide the code for tasks whenever requested
4. Plots must carry axes labels, legends and titles clearly
5. The assignments must be completed individually and written individually including the programming assignments. Group discussions are allowed and encouraged, but in such cases you should list the group members in your handin.
6. Your handin should include your solution to the exercises in a single pdf (not zipped).

2 Monte Carlo Estimation

2.1 Variance of Sample Mean Estimator (9 points)

Consider an unlabelled training set with N i.i.d samples $\{x[0], x[1], \ldots, x[N-1]\}$ observed from $P_X(x)$[1]. We are interested in modelling the data distribution $P_X(x)$ which has two fundamentally illuminating properties: mean μ and variance σ^2. In this task, we only estimate the mean parameter μ.

A reasonable estimator of mean $\mathbb{E}[X]$ from N observed samples is the Monte Carlo sample average [2]:

$$\mathbb{E}[X] \approx \hat{\mu} = \frac{1}{N} \sum_{n=0}^{N-1} x[n]. \tag{1}$$

[1] **Note:**The probability density function, $P_X(x)$ gives the probability that the random variable X lies in an infinitesimal interval about the point X=x, normalized by the length of the interval.[5]

1. Given this mean estimator, find its variance. Assume the sampler from which the samples were obtained has variance σ^2.
2. Examine what happens to the variance of the estimator as $N \to \infty$.
3. Discuss the influence of N in the context of Machine Learning tasks.

2.2 Functions of Random variables (9 points)

In many scenarios we are interested in evaluating a function of the random variable [5]. For instance, expectations of functions of random variables are unavoidable (entropy, KL divergence etc.). This is relatively easy if the probability density function (PDF) is known and is tractable to work with.

However, we rarely have access to, $P_X(x)$, the analytical form of the PDF of the variable of interest. In many cases we end up approximating the densities of interest with some parameterised distribution within the exponential family (most popular being the Gaussian density) as they are easy to work with and can offer reasonable approximations.

In this example, we consider a continuous random variable X with an exponential PDF:

$$P_X(x) = \frac{1}{2}\exp\left(-\frac{x}{2}\right) \quad \forall \quad x \geq 0 \tag{2}$$

1. Compute the analytical expected value of X under the density $P_X(x)$ given by

$$\mathbb{E}_{P_X}[X] = \int x P_X(x) \mathrm{dx} \tag{3}$$

2. Build a Monte Carlo sampler in Python (provide the code in the report) to obtain samples from the exponential PDF $P_X(x)$ in Eq. (2).
3. Report estimates of the following expectations of functions of X using $N = [5, 10, 100]$ samples:
   - $\mathbb{E}_{P_X}[X]$ and compare it with the analytical mean in Task 1.
   - $\mathbb{E}_{P_X}[\log X]$
   - $\mathbb{E}_{P_X}[-\log P_X(x)]$

3 Divergence Measures

To quantify the quality of approximations of probability distributions we also require reasonable measures. Distances between probability density functions are not straightforward and we have to resort to *divergence* measures such as the Kullback-Leibler (KL) divergence [6]. In this task you will work with two forms of KL divergences and relate them to divergence measures used in Variational Inference [1].

3.1 Kullback-Liebler Divergence (16 points)

KL divergence is a widely used measure to compare probability distributions usually denoted $\mathrm{KL}(P_X(x)||Q_X(x))$, given by

$$\mathrm{KL}(P_X(x)||Q_X(x)) = \mathbb{E}_{P_X}\Big[\log \frac{P_X(x)}{Q_X(x)}\Big] \tag{4}$$

Note, however, it is not a metric i.e. $\mathrm{KL}(P_X(x)||Q_X(x)) \neq \mathrm{KL}(Q_X(x)||P_X(x))$ resulting in two KL divergences. $\mathrm{KL}(P_X(x)||Q_X(x))$ is called the *forward* KL divergence and $\mathrm{KL}(Q_X(x)||P_X(x))$ is known as the *reverse* KL divergence given by

$$\mathrm{KL}(Q_X(x)||P_X(x)) = \mathbb{E}_{Q_X}\Big[\log \frac{Q_X(x)}{P_X(x)}\Big] \tag{5}$$

1. Consider two Gaussian PDFs: $P_X(x) = \mathcal{N}(\mu, \sigma^2)$ and $Q_X(x) = \mathcal{N}(0, 1)$. Show that the closed form for the forward KLD:

$$\mathrm{KL}(P_X(x)||Q_X(x)) = -0.5(1 + \log\sigma^2 - \mu^2 - \sigma^2) \tag{6}$$

	1	2	3	4
1	1/8	1/8	0	0
2	1/8	1/8	0	0
3	0	0	1/4	0
4	0	0	0	1/4

Table 1: Joint distribution for P(x,y)

2. Evaluate the Gaussian PDF for $x \in [-10, 10]$ at 100 uniformly spaced points for
   - $Q_X(x) = \mathcal{N}(0, 1)$
   - $P_X(x) = \mathcal{N}(2, 4)$

   and visualise the two PDFs in the same plot. Ensure the two PDFs integrate to 1 which then yields a *truncated* Gaussian as the support for univariate Gaussian is $\mathbb{R}$.
   Note: As we are working with discrete variables to approximate continuous ones, consider approximating integrals here using Riemann sum or similar.
3. Compute KL$(P_X(x)||Q_X(x))$ and KL$(Q_X(x)||P_X(x))$ from the samples in Task 3.1.2 above. Compare the forward KL with the analytical KLD in Eq. (6).
4. (Source: Exercise 21.7 from [7])
   Consider the following joint distribution, $P(x, y)$[2], in Table 1 where the rows represent y and columns x. If we approximate this $P(x, y)$ with $Q(x, y)$ by minimizing the reverse KL divergence KL$(Q||P)$,
   - Show that there are three distinct minima.
   - Identify those minima and evaluate KL$(Q||P)$ at each of them.
   - What is the value of KL$(Q||P)$ if we set $Q_{XY}(x, y) = P_X(x)P_Y(y)$?

   *H*int: Assume a factored approximation i.e $Q_{XY}(x, y) = Q_X(x)Q_Y(y)$

4 Representation Learning & Generative Modelling

In this task, we investigate three classes of Machine Learning tools in the spirit of unsupervised *Representation Learning* of unlabelled data. Learning useful representations of data can be used in downstream tasks (classification, regression, segmentation etc...), and more importantly to understand and model the underlying data distribution $P_X(x)$.

We will be using MNIST [3] – a popular ML benchmarking dataset – in these tasks.

A Jupyter notebook is provided along with this assignment with tools useful to load and visualize the data, and also empty templates to fill in the assignment tasks. We encourage you to adhere with this template when submitting the solutions.

4.1 Getting to know the data (8 points)

1. Download and load MNIST data.
2. Check how many data points are provided in the training and test sets
3. While we will be assuming this is unlabelled data, MNIST provides 10 class labels. We can use these to evaluate the performance of our unsupervised models. Make a histogram of the different classes and see if the data distribution is balanced or not.
4. Visualize a few samples
5. Reshape the 28x28 images into long vectors of dimension 784 as the methods we will be using assume the dataset, $X \in \mathbb{R}^{N \times F}$ where N is the number of data points and F is their feature dimension.
6. (Optional) The objective of this task is to *get to know the data*. Play around with it and report any other useful statistics/analyses/visualizations that can be helpful.

[2] **Note:** The subscripts with the random variables are usually dropped for notational convenience in $P_{XY}(x, y)$. This is also closer to the norm in most of Machine Learning literature.

[3] `https://pytorch.org/docs/stable/torchvision/datasets.html#mnist`

4.2 Principal component analysis (PCA) on MNIST (14 points)

Principal component analysis is one of the fundamental tools in Statistics and Machine Learning. It is used widely to reduce the dimensionality of the data to obtain combinations of features that optimize to retain maximum variance in the data. We use PCA along with k-Means clustering in this task as the first Unsupervised model.

1. Create a smaller dataset comprising a subset of data corresponding to classes $[0, 1, 2, 3, 4]$
2. You are allowed to use the PCA package in Scikit-Learn. However, read the documentation of the package carefully before using it.
3. Perform PCA on the training data with D=200 components
4. Plot the Eigen spectrum (indicating the ratio of explained variance). What is the total variance explained?
5. Vary the number of principal components in the following range:
   $D = [1, 2, 4, 8, 16, 32, 64, 128, 256, 512]$. Make a visualization with the total explained variance and the number of components. Discuss the pros and cons of increasing D.
6. Using $D = 2$ perform PCA on the training data and obtain the low dimensional representation of the data. Now that the data is in 2d, visualize them in a 2D scatter plot. Color code the data points with their class labels. Discuss your observations in relation to unsupervised learning.
7. Transform and visualize the test data based on the principal components obtained from training data.
8. Perform k-Means clustering with $k = 5$ which assigns each data point to one class. Report the training and test accuracy. Here again you are allowed to use Scikit-Learn implementations.

4.3 Autoencoders on MNIST (20 points)

In this task, you will perform dimensionality reduction using an Autoencoder instead of PCA as autoencoders can be interpreted to be non-linear PCAs.

1. **Encoder:** The encoder $f_{enc}(\cdot) : X \rightarrow Z$ reduces the dimensionality of the input $X \in \mathbb{R}^{N \times F}$ to latent representation $Z \in \mathbb{R}^{N \times D}$ in successive steps. Complete the encoder in the Jupyter notebook to be a 3 layered multi-layer perceptron (MLP) with $[F, 32, 16, D]$ hidden units, where D is the latent dimension. Use rectified linear unit activation function after each layer.
2. **Decoder:** The decoder $d_{dec}(\cdot) : Z \rightarrow \hat{X}$ reconstructs the input data $\hat{X} \in \mathbb{R}^{N \times F}$ from the low dimensional latent representation Z. As a convention the architecture is flipped for the decoder. That is, it is an MLP with $[D, 16, 32, F]$ hidden units. Use rectified linear unit activation function for all layers except the last layer where you should use a sigmoid non-linear unit. Comment on why the last layer should use a sigmoid activation function.
3. **Optimization:** We assume each pixel in the image to be modelled as a Bernoulli random variable. This allows us to model the reconstruction task using logistic regression (or the Perceptron algorithm) [1]. Meaning, we can use Binary Cross Entropy [4] loss function $\mathcal{L}(X, \hat{X}) = BCE(X, \hat{X})$ to optimize the autoencoder. Based on this information set up the appropriate loss function to be optimized in your implementation.
4. **Low dimensional data:** Once the model is trained with the training data, you can predict the low dimensional representation from the encoder for all data points.
   Using $D = 2$ perform the same visualizations and clustering as described in 4.2.6, 4.2.7 and 4.2.8 for both training and test data.
5. Compare the low dimensional representations obtained using PCA and Autoencoder using the plots and test set accuracy. Discuss the differences between them. What are the pros and cons of using each of these models for dimensionality reduction.
   Note: To compare classification accuracy when class labels are unknown as it is in case

[4] `https://pytorch.org/docs/stable/nn.html#bceloss`

of unsupervised settings, measures such as the adjusted random index or adjusted mutual information can be used.

6. (Optional) Feel free to try out different models by changing the number of layers and hidden units in both encoder and decoder.

4.4 Variational Autoencoders on MNIST (24 points)

The *latent* space learnt by Autoencoders can be used to perform dimensionality reduction of the data. However, the space itself does not have structure in it to be exploited. Regularizing the latent space allows us to use the latent representations to synthesize new data. One popular strategy to regularize the latent space is the use of Variational Autoencoders (VAE) [4].

Building on the autoencoder from the previous task we introduce VAEs to investigate the regularized latent space, and synthesize new data in this task.

1. **Encoder** Modify the Encoder to output two $D-$ dimensional vectors: one for the mean μ and one for the $\log \sigma^2$ of the latent posterior distribution $Q_Z(z|x)$. Also, think about the implications of using the ReLU non-linearity for the last layer of the encoder and make a suitable choice.
2. **Reparameterisation:** Stochastic elements are non-differentiable and thus the gradients necessary for performing gradient descent cannot be estimated. *Reparameterisation* trick introduced in [3] overcomes this by decomposing the random variable into a determinstic component and a stochastic element.
   Complete the portion for the reparameterisation trick in the code.
3. **Decoder:** The decoder is similar to the one used in the Autoencoder. Modify the decoder (if necessary) and use it to obtain the reconstructed data.
4. **Optimization:** VAEs are optimized by maximizing the evidence lower bound (ELBO) which consists of the reconstruction term which is identical to the reconstruction term in Autoencoders and a regularization term (KL$[Q_Z(z|x)|P_Z(z)]$) which forces the latent posterior distribution to match the prior on the latent distribution.
   Optimize ELBO and train the VAE until convergence. Plot ELBO, regularization term and the reconstruction loss for each epoch.
5. **Visualization:** Visualize the latent space with $D = 2$ and compare it with the latent spaces obtained with PCA and Autoencoder.
6. **Generative Sampling:** The primary advantage of using VAEs is that the latent space can be used to synthesize new data. After training the VAE, obtain 32 samples from the prior distribution $P_Z(z)$ and decode these low dimensional representations. Visualize these synthetic samples.
7. (Optional) Feel free to try out different models by changing the latent dimension, number of layers and hidden units in both encoder and decoder.

References

[1] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[2] Steven M Kay. *Fundamentals of statistical signal processing*. Prentice Hall PTR, 1993.

[3] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[4] Diederik P Kingma, Max Welling, et al. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[5] Scott Miller and Donald Childers. *Probability and random processes: With applications to signal processing and communications*. Academic Press, 2012.

[6] Tom Minka et al. Divergence measures and message passing. Technical report, Technical report, Microsoft Research, 2005.

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