Exploiting Semi-Supervised Generative Model in Active Learning

Tim, Zhenzhong Xiao[1]

MRes Computational Statistics and Machine Learning

Prof. David Barber

Submission date: September 2020

Abstract

Active learning tries to solve a practical problem for machine learning, which is to create a good model with only limited labelling budget. In this project, we exploit useful properties from semi-supervised generative model and use them in active learning. Our experiments in the half-moon and MNIST dataset show that by using semi-supervised generative model with simple acquisition function such as predictive entropy, we are able to improve the performance of active learning. Further experiments on our proposed acquisition functions expose interesting challenges in using data density provided by the model, which can be a valuable pointer for future active learning research.

Acknowledgements

I would like to express my greatest gratitude towards my supervisor Prof. David Barber for his patient and guidance throughout the year. I am very fortunate to have been given the opportunity to work with David. I have gained invaluable experience and knowledge from seeing how David approaches interesting research questions. In particular, David's high standards on quality research encourages me to think more scientifically and use less hacky methods. I hope I will not let David down in my future research career.

I would also like to thank my friends and collaborators Mingtian Zhang, Peter Hayes, Hippolyt Ritter and Pau Ching Yap for various inspiring discussions.

Last but not least, I want to thank my parents and my girlfriend Keyue Chen for their continuous support.

Contents

1 Introduction **5**

1.1 Active Learning 5

1.1.1 Acquisition Function and Uncertainty in Prediction 5

1.2 Motivations 7

1.2.1 Utilise Unlabelled Data During Training 7

1.2.2 Generalise Across Tasks and Models 7

1.2.3 Trade-off Between Informative and Representative Data 9

1.3 Contributions 9

1.4 Thesis Structure 10

2 Background **11**

2.1 Deep Generative Model 11

2.1.1 Probabilistic Model 11

2.1.2 Learning 12

2.1.3 Latent Variables 14

2.1.4 Learning with Latent Variables 14

2.1.5 Variational Autoencoder 17

2.2 Semi-Supervised Learning 18

2.3 Semi-Supervised Generative Model 18

2.3.1 Modelling Assumptions 19

2.3.2 Learning and Inference 19

3 Active Learning with Semi-Supervised Generative Model **21**

3.1 Correctness of the Model 21

3.1.1 Proof of Objective $\mathcal{J}$ 21

3.1.2 Proof of Objective $\mathcal{J}^{\alpha}$ 22

3.2 Integrating the Model with Active Learning 22

3.3 Acquisition 23

3.3.1 With Entropy 23

3.3.2 With Linear Combination of Entropy and Density 24

3.3.3 With Ordered Filtering of Entropy and Density 24

4 Experiments **26**

4.1 Specification 27

4.2 Acquisition with $\mathcal{A}_{\mathrm{H}}(\cdot)$ 28

4.3 Acquisition with $\mathcal{A}_{\gamma}(\cdot)$ 30

4.4 Acquisition with $\mathcal{A}_{1}(\cdot)$ then $\mathcal{A}_{0}(\cdot)$ 34

5 Discussion and Conclusion **35**

5.1 Problems with Density 35

5.2 Sampling Bias 36

5.3 Summary 36

References **38**

Chapter 1

Introduction

In recent years, a lot of progress has been made in the field of machine learning. One major challenge in applying many of these cutting edge research to real life problems is obtaining labelled data. Unlike toy problems such as MNIST [LeCun et al., 2010] hand written digits recognition, accurate labelling for tasks like speech recognition, machine translations, medical imaging is time consuming and requires help from domain experts. Therefore, it can be very expensive. The follow up question is: How can we make the most out of a fixed budget and a limited labelling capacity? In this project, we would like to explore whether the use of semi-supervised generated model in active learning can give a good answer to this question.

1.1 Active Learning

Active Learning (AL), also known as *optimal experimental design* in statistics literature [MacKay, 1992, Fedorov et al., 1972], provides a framework to approach the practical question: which data should we acquire next for labelling so that we can maximise the performance of our model? For a common supervised learning task, the goal is to learn the conditional distribution $p(y|\mathbf{x})$ from a set of labelled training data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $(x_i, y_i) \in \mathcal{L}$. For most real life tasks, the number of labelled data is usually very limited. Comparatively, unlabelled data are much easier to obtain, and are continuously growing in many cases. In this project, we consider the common pool-based scenario of active learning [Lewis and Gale, 1994, Settles, 2009]: an active learner, who wants to model the $p(y|\mathbf{x})$, starts with a small labelled training set $\mathcal{L}^{(0)}$, but has the ability to *actively* select a subset among the unlabelled data $\mathcal{U}^{(0)}$ and acquire the labels for them through an oracle (e.g. human experts) [MacKay, 1992, Cohn et al., 1995]. This is a repeated process, after the acquisition of new labelled data, the learner will update the model $p(y|\mathbf{x})$ using the incremented labelled set $\mathcal{L}^{(1)}$. Then, the learner will seek to acquire a new batch of labelled data from the decremented unlabelled set $\mathcal{U}^{(1)}$.

1.1.1 Acquisition Function and Uncertainty in Prediction

An acquisition function $\mathcal{A}(\mathbf{x}), \mathbf{x} \in \mathcal{U}$ is used to determine which unlabelled data to be labelled next. A naive acquisition function can be one that randomly samples data from

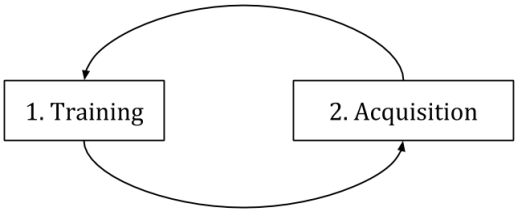


Figure 1.1: A diagram illustrating the active learning loop.

$\mathcal{U}$ for labelling. However, if we make use of the knowledge about the data or the current model, we are able to create a much better acquisition function [Schröder and Niekler, 2020]. Therefore, the main challenge in active learning community is how to design a good acquisition function. Ideally, a good acquisition function will prioritise the selection of the most *informative* data in $\mathcal{U}$ such that the model can have a large improvement on its performance.

Informativeness is an abstract idea and hard to evaluate. An intuitive and more tangible interpretation of informativeness is *model uncertainty*. The assumption is that by learning something a model is most uncertain of, it should gain the most knowledge (i.e. information about the world, thus most informative). Uncertainty for a model in supervised learning tasks can be revealed from its confidence for predicting the label y of an input $\mathbf{x}$. Such uncertainty in prediction is called *predictive uncertainty*, which captures the model's uncertainty in its output [Gal, 2016].

REMARK 1 (Model Uncertainty) *If Bayesian principle is used to derive the posterior distribution of model parameters, the predictive uncertainty will also take into account the uncertainty inherits from model parameters, which usually results in a better estimation for model uncertainty.*

There are a few methods to measure and quantify predictive uncertainty [Lewis and Gale, 1994, Scheffer et al., 2001, Gal, 2016, Kendall et al., 2017]. *Entropy* [Shannon, 1948] is one of most popular uncertainty measures, which originates from information theory and is used to measure the average amount of information in a distribution. Assume our model is parameterised by θ, then given any input $\mathbf{x}$, the entropy for the predictive distribution $p_\theta(y|\mathbf{x})$ is

$$\mathrm{H}[p_\theta(y|\mathbf{x})] = \int -p_\theta(y|\mathbf{x}) \log p_\theta(y|\mathbf{x}) \, \mathrm{d}y = \langle -\log p_\theta(y|\mathbf{x}) \rangle_{p_\theta(y|\mathbf{x})} , \tag{1.1}$$

which calculates a weighted average of $\log p_\theta(y|\mathbf{x})$ over all possible y. Large entropy implies high predictive uncertainty. In contrast, small entropy implies the model is certain about its prediction.

We can use a binary classification task (e.g. $y \in \{a, b\}$) as a simple example to understand how entropy can measure the predictive uncertainty. Given $\mathbf{x}$, if a model $p_{\theta_1}(y|\mathbf{x})$ is extremely *uncertain* about its prediction, it will predict y to be a with the same probability as y to be b, which implies that the predictive distribution $p_{\theta_1}(y|\mathbf{x})$ is a uniform distribution and $p_{\theta_1}(y = a|\mathbf{x}) = p_{\theta_1}(y = b|\mathbf{x}) = 0.5, \mathrm{H}[p_{\theta_1}(y|\mathbf{x})] = -\sum_{y\in\{a,b\}} \frac{1}{2} \log_2 \frac{1}{2} = 1$[1]. In comparison, for a *confident* model $p_{\theta_2}(y|\mathbf{x})$, it predicts y to be a with probability 1, then the predictive distribution is $p_{\theta_2}(y = a|\mathbf{x}) = 1, p_{\theta_2}(y = b|\mathbf{x}) = 0$ and $\mathrm{H}[p_{\theta_2}(y|\mathbf{x})] = 0$.

When entropy is used to measure predictive uncertainty, it is called predictive entropy.

[1]Here we use base 2 for log, but the choice can vary between different applications.

Predictive entropy is used in one of the most common acquisition functions, i.e. $\mathcal{A}_{\mathrm{H}}(\mathbf{x}) = \mathrm{H}[p_\theta(y|\mathbf{x})]$. Given the current model $p_\theta(y|\mathbf{x})$ with parameter θ, based on predictive entropy the most informative data $\mathbf{x}^* \in \mathcal{U}$ to be labelled next is

$$\mathbf{x}^* = \arg\max_{\mathbf{x}\in\mathcal{U}} \mathcal{A}_{\mathrm{H}}(\mathbf{x}) = \arg\max_{\mathbf{x}\in\mathcal{U}} \mathrm{H}[p_\theta(y|\mathbf{x})] \tag{1.2}$$

1.2 Motivations

In previous sections, we go through the procedure of active learning and an acquisition function that based on predictive uncertainty. There are two attributes in above descriptions that we might be able to exploit and further improve the performance of active learning, which will in turn help us to save the labelling cost.

One is that during each iteration of active learning, the learner only make use of labelled data for training the model. The other is that when we use predictive uncertainty for acquisition, the data we acquire are highly dependent on $p_\theta(y|\mathbf{x})$. In other words, the acquisition is model specific (i.e. for the current model $\mathcal{M}$ s.t. $\theta \sim p(\theta|\mathcal{M})$) and task specific (i.e. to learn the current underlining condition distribution $p(y|\mathbf{x})$).

These two attributes, especially the latter, have not been well addressed with powerful deep learning toolkits. We believe the latest development in semi-supervised generative model can be used to exploit these two attributes and improve the performance of active learning.

1.2.1 Utilise Unlabelled Data During Training

Most of the active learning methods only utilise the unlabelled data during acquisition [Tong and Koller, 2001, Cohn et al., 1995, Freund et al., 1997, Gal et al., 2017], but they leave a large amount of valuable information unused in the training step at each iteration. Similar to active learning, *Semi-Supervised Learning* also offers a paradigm for taking advantages from both labelled data and unlabelled data. However, semi-supervised learning does not consider an acquisition step and the labelling oracle. Instead, it uses the unlabelled data together with the labelled data during training.

Therefore, it is sensible to combine semi-supervised learning with active learning. In this case, unlabelled data will be fully utilised in both the training step and the acquisition step. In fact, many people have already tried the combination of the two schemes and it does show improvement in the performance of active learning [McCallum and Nigam, 1998, Zhu et al., 2003, Muslea et al., 2002, Guo and Schuurmans, 2008].

1.2.2 Generalise Across Tasks and Models

Less Bias Towards A Task

In the usual uncertainty based active learning setting, the unlabelled data are selected through acquisition functions based on our model's estimation of $p(y|\mathbf{x})$ (e.g. predictive entropy), where $p(y|\mathbf{x})$ is specified by the mapping $\mathbf{X} \to Y$ with $\mathbf{x} \in \mathbf{X}, y \in Y$. Such acquisitions might be helpful for the current task but not for the others. For example, given

a face image dataset $\mathcal{X}$ (e.g. CelebA [Liu et al., 2015]) and the data labelled through active learning based on the task $\mathcal{X} \rightarrow Smiling, Smiling \in \{True, False\}$ might not be a good training set for the task $\mathcal{X} \rightarrow Eyeglasses, Eyeglasses \in \{True, False\}$. Therefore, such task dependent acquisition functions might not have a good generalisability if we would like to reuse the acquired data for various tasks.

Our original goal is to make the most out of a limited labelling budget. If the acquired data from one task can be partially or fully reused for other tasks, then we can save the cost of acquiring and relabelling the same data set $\mathbf{X}$. Thus, it will be nice to have an active learning scheme that take the *representativeness* of a data alongside its informativeness into consideration.

Less Bias Towards A Model

Moreover, when we try to measure the model uncertainty using $p_\theta(y|\mathbf{x})$, we have to bear in mind that the current parameters θ of the model depends on its architecture, its hypeparameters etc. (i.e. $\theta \sim p(\theta|\mathcal{M})$, where $\mathcal{M}$ denotes a specific model architecture and its hypeparameters). Therefore, the current uncertainty based active learning can only acquire data with respect to a specific model structure $\mathcal{M}$. However, in a fast growing field like machine learning, the current state-of-the-art model structures could be replaced by the better and more powerful ones every few months. If the data we acquired through active learning are based on, for example, the predictive entropy of a past model $\mathcal{M}_1$, such labelled data might not be informative for the newer model $\mathcal{M}_2$ as they are biased towards $\mathcal{M}_1$.

As always, we would like to cut the cost in data labelling and avoid acquiring and relabelling the same data set $\mathbf{X}$ when we move on to a newer model. Hence, the acquired data should be representative and not bias towards a specific model.

Representiveness and Data Density

Uncertainty based acquisition can also suffer from selecting the outliners as they have high uncertainty [McCallum and Nigam, 1998] but not being representative. Therefore, including the two scenarios mentioned before, we would like to put representiveness into consideration during the acquisition of data from $\mathcal{U}$. Similar to informativeness, we need to define a tangible measure for representiveness. Data density $p(\mathbf{x})$ is used widely in active learning literature to interpret the meaning of representiveness. Given two data points $\mathbf{x}_1, \mathbf{x}_2$, it is reasonable to say that $\mathbf{x}_1$ is more representative than $\mathbf{x}_2$ if $p(\mathbf{x}_1) > p(\mathbf{x}_2)$.

There are several attempts to extract density information through clustering [Nguyen and Smeulders, 2004, Xu et al., 2003, Donmez et al., 2007]. Some tried to estimate the density $p(\mathbf{x})$ directly with naive Bayes [McCallum and Nigam, 1998], but naive Bayes is not an accurate estimator because of its strong independence assumptions between the features. Ideally, in order to accurately measure the representiveness of a data sample, we want to create a parametric distribution $p_\theta(\mathbf{x})$ and make it as close to the underlining $p(\mathbf{x})$ as possible. This can now be better achieved through *deep generative models*. However, to the best of our knowledge, deep generative models have not been exploited in active learning for density estimation, which motivates us to incorporate deep generative models into active learning.

	$\mathcal{L}$ at Training	$\mathcal{U}$ at Training	$\mathcal{U}$ at Acquisition	Model $p(y\|\mathbf{x})$	Model $p(\mathbf{x})$
Supervised Model (SM)	✓	-	-	✓	-
Semi-Supervised Model (SSM)	✓	✓	-	✓	-
Semi-Supervised Generative Model (SSGM)	✓	✓	-	✓	✓
AL with SM	✓	-	✓	✓	-
AL with SSM	✓	✓	✓	✓	-
AL with SSGM	✓	✓	✓	✓	✓

Table 1.1: Properties of different types of models and their conjunctions with Active Learning (AL). We use $\mathcal{L}$ to denote labelled data and $\mathcal{U}$ to denote unlabelled data.

1.2.3 Trade-off Between Informative and Representative Data

From the two motivations above, it is easy to see that semi-supervised generative models are good candidates to be explored in active learning (see Table 1.1 for comparisons). By using semi-supervised generative models in active learning, we can utilise unlabelled data during the training step and take advantage of density information from $p_\theta(\mathbf{x})$.

However, there is clearly a trade-off between acquiring informative data and acquiring representative data. The high uncertainty data of $p_\theta(y|\mathbf{x})$ are very often located close to the decision boundary [Huang et al., 2010], which are less representative in terms of the density $p(\mathbf{x})$. In contrast, the high density data are located around the modes of the distribution $p(\mathbf{x})$, which might not be informative if we want to closely estimate the decision boundary.

There are lots of heuristics around this topic [Huang et al., 2010, Donmez et al., 2007, Xu et al., 2003]. Nevertheless, semi-supervised generative models can give us a different perspective to approach this trade-off, because their learning objectives naturally model $p(y|\mathbf{x})$ and $p(\mathbf{x})$, which might provide us a straightforward trade-off between informativeness and representiveness during acquisition.

1.3 Contributions

In this project, our main contributions are:

- We create a new active learning scheme using semi-supervised generative model. We explore its performance through various experiments and show that the new scheme outperforms our baselines when using simple acquisition functions like predictive entropy.[2]
- With our new active learning scheme, we further create new acquisition functions aim for the trade-off between informative and representative data. We test these new acquisition functions and identify valuable challenges in acquiring representative data.

[2]Our proposed methods and the results in this project will be submitted to the 24th International Conference on Artificial Intelligence and Statistics (AISTATS 2021).

1.4 Thesis Structure

In this chapter, we give an introduction into the popular uncertainty based active learning scheme and discuss the limitations in this scheme which motivates our idea of using semi-supervised generative model in active learning. At the end of the chapter, we summarise our main contributions in this project. In chapter §2, we go through the concepts of deep generative model, semi-supervised learning and semi-supervised generative model, which further clarify the intuition behind our project. In chapter §3, we specify the details of how to combine semi-supervised generative model with active learning and propose three acquisition functions. Next, we test the performance of our methods. The results of the experiments are plotted and analyse in chapter §4. For some of our unsatisfying experiment results, we identify their causes and document them in chapter §5 for future investigation. In the end of chapter §5, we also provide a summary for the project.

Chapter 2

Background

The semi-supervised generative model used in the project touches two different concepts, which are semi-supervised learning and deep generative model. In this chapter, we walk through the necessary details behind these ideas and provide a background for our methods in the next chapter.

2.1 Deep Generative Model

Generative model is a concept from probabilistic modelling, which tries to model the generating process of some random data. A simple example will be creating a model $p_\theta(\mathbf{x})$ through samples $\{\mathbf{x}_i\}_{i=1}^n$ generated from $p(\mathbf{x})$ to approximate $p(\mathbf{x})$. A more complicated and useful generative model assumes the existence of some latent structures behind $\mathbf{x}$, e.g. $\mathbf{x}$ depends on the latent variable $\mathbf{z}$. In this section, we introduce the idea behind probabilistic model, latent variable model and how to do learning and inference on these models. In the end, we use an example of deep generative model (i.e. VAE) to connect these concepts together.

2.1.1 Probabilistic Model

In probabilistic modelling, we assume data are random samples from an underlying distribution that encodes the unknown data generating process. In an unsupervised learning problem, we are given a set of data $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ independently and identically sampled (i.e. i.i.d. data[1]) from $p(\mathbf{x})$, our goal is to use a model $p_\theta(\mathbf{x})$ with parameters θ to approximate $p(\mathbf{x})$ as accurately as possible so that for any $\mathbf{x}'$ we have

$$p_\theta(\mathbf{x}') \approx p(\mathbf{x}'). \tag{2.1}$$

Similarly, for supervised learning problems such as classification, we are given i.i.d. labelled data set $(\mathbf{X}, Y) = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and would like to model the underlying conditional distribution $p(y|\mathbf{x})$ with $p_\theta(y|\mathbf{x})$, where θ denotes the model parameters. Likewise, we want $p_\theta(y|\mathbf{x})$

[1] i.i.d. stands for independently and identically distributed

to be as close to $p(y|\mathbf{x})$ as possible, which means for any pair $(\mathbf{x}', y')$ we have

$$p_\theta(y'|\mathbf{x}') \approx p(y'|\mathbf{x}'). \tag{2.2}$$

Once we have the optimal θ^*, we can use $p_{\theta^*}(y|\mathbf{x})$ to predict the class y^* of a new input $\mathbf{x}^*$:

$$y^* = \arg\max_y \ p_{\theta^*}(y|\mathbf{x}^*). \tag{2.3}$$

2.1.2 Learning

After choosing a model, say $p_\theta(\mathbf{x})$, for our tasks, we need to find the θ^* that gives us the best approximation of $p(\mathbf{x})$. This process is called *learning*. To better explain the process of learning, let us rewrite $p_\theta(\mathbf{x})$ to $p(\mathbf{x}|\theta)$, which preserves the same meaning (i.e. given θ, the probability of seeing $\mathbf{x}$). With our training data $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$, we are interested in the *posterior* distribution of θ, which can be derived through Bayes' theorem:

$$p(\theta|\mathbf{X}) = \frac{p(\mathbf{X}|\theta) \cdot p(\theta)}{p(\mathbf{X})}. \tag{2.4}$$

$p(\mathbf{X}|\theta)$ is known as the *likelihood* for $\mathbf{X}$, $p(\theta)$ is the *prior* belief of θ and $p(\mathbf{X})$ is the *evidence*. Following are two popular perspectives for the same learning objective.

Maximum Likelihood Learning

In many cases, since we do not have prior knowledge about θ, we will use an 'uniform' prior for θ, which means $p(\theta)$ is a constant C. Therefore, our learning objective is

$$\theta^* = \arg\max_\theta \ p(\theta|\mathbf{X}) = \arg\max_\theta \ \frac{p(\mathbf{X}|\theta) \cdot C}{p(\mathbf{X})} = \arg\max_\theta \ p(\mathbf{X}|\theta). \tag{2.5}$$

In other words, the θ that maximises the posterior $p(\theta|\mathbf{X})$ is the same one that maximises the likelihood $p(\mathbf{X}|\theta)$. Hence, we call it *maximum likelihood learning*. We can further simplify Eq.(2.5) and represent the learning objective only with our model $p_\theta(\mathbf{x})$:

$$\begin{aligned}
\theta^* &= \arg\max_\theta \ p(\mathbf{X}|\theta) && (2.6)\\
&= \arg\max_\theta \ \log p(\{\mathbf{x}_i\}_{i=1}^n|\theta) && (2.7)\\
&= \arg\max_\theta \ \log \prod_{i=1}^n p(\mathbf{x}_i|\theta) && (2.8)\\
&= \arg\max_\theta \ \sum_{i=1}^n \log p_\theta(\mathbf{x}_i) && (2.9)\\
&= \arg\min_\theta \ -\sum_{i=1}^n \log p_\theta(\mathbf{x}_i). && (2.10)
\end{aligned}$$

The final term $-\sum_{i=1}^n \log p_\theta(\mathbf{x}_i)$ is known as *negative log-likelihood*. The original maximisation objective is now written as a minimisation task. It is common to represent a learning problem as a minimisation task, as we can make use of the existing optimisation tools such as *gradient descent* to solve it. By now, we have transformed our learning problem into a

simple optimisation task.

Kullback-Leibler Divergence

Our initial goal is to find a close estimate for an unknown distribution $p(\mathbf{x})$ with a chosen model $p_\theta(\mathbf{x})$. A straight forward idea to solve this problem is to find the θ that minimise the 'distance' between this two distributions. *Kullback-Leibler*(KL) divergence [Kullback and Leibler, 1951] is a measure of the difference between two distributions

$$\mathrm{KL}[p||q] = \int p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} \, \mathrm{d}\mathbf{x} = \left\langle \log \frac{p(\mathbf{x})}{q(\mathbf{x})} \right\rangle_{p(\mathbf{x})} = \langle \log p(\mathbf{x}) - \log q(\mathbf{x}) \rangle_{p(\mathbf{x})}. \tag{2.11}$$

KL divergence has the property that $\mathrm{KL}[p||q] \geq 0$ for any pair of distributions, and $\mathrm{KL}[p||q] = 0$ if and only if $p(\mathbf{x}) = q(\mathbf{x})$. In the literature, it is common to denote an unknown distribution as p and denote our modelling distribution as q. Another property is that KL divergence is not symmetric, i.e. $\mathrm{KL}[p||q] \neq \mathrm{KL}[q||p]$. We call $\mathrm{KL}[p||q]$ the forward KL, minimising it will prioritise finding the q that match its mean with the mean of p. $\mathrm{KL}[q||p]$ is referred as the reverse KL, which tends to select a q that matches a mode of p [Goodfellow et al., 2016]. Depends on our priority, we can choose the one that fits our need.

Following our example, we can try to find the θ that minimise the forward KL between $p(\mathbf{x})$ and $p_\theta(\mathbf{x})$:

$$\begin{aligned}
\theta^* &= \arg\min_\theta \ \mathrm{KL}[p(\mathbf{x})||p_\theta(\mathbf{x})] && (2.12)\\
&= \arg\min_\theta \ \langle \log p(\mathbf{x}) - \log p_\theta(\mathbf{x}) \rangle_{p(\mathbf{x})} && (2.13)\\
&= \arg\min_\theta \ \langle \log p(\mathbf{x}) \rangle_{p(\mathbf{x})} - \langle \log p_\theta(\mathbf{x}) \rangle_{p(\mathbf{x})} && (2.14)\\
&= \arg\min_\theta \ -\langle \log p_\theta(\mathbf{x}) \rangle_{p(\mathbf{x})}. && (2.15)
\end{aligned}$$

However, as we do not know the distribution $p(\mathbf{x})$, we can only use the limited samples in our training set $\mathbf{X}$ to approximate it:

$$\arg\min_\theta \ -\langle \log p_\theta(\mathbf{x}) \rangle_{p(\mathbf{x})} \approx \arg\min_\theta \ -\frac{1}{n} \sum_{i=1}^{n} \log p_\theta(\mathbf{x}_i). \tag{2.16}$$

Therefore, the θ^* that minimises $\mathrm{KL}[p(\mathbf{x})||p_\theta(\mathbf{x})]$ is:

$$\begin{aligned}
\theta^* &= \arg\min_\theta \ -\frac{1}{n} \sum_{i=1}^{n} \log p_\theta(\mathbf{x}_i) && (2.17)\\
&= \arg\min_\theta \ -\sum_{i=1}^{n} \log p_\theta(\mathbf{x}_i), && (2.18)
\end{aligned}$$

which is exactly the θ^* that minimises the negative log-likelihood in Eq.(2.10). Hence, minimising the forward KL is the same as maximum likelihood learning.

2.1.3 Latent Variables

Our model $p_\theta(\mathbf{x})$ from above usually has a simple closed-form expressions (e.g. $p_\theta(\mathbf{x}) = \mathcal{N}(\mu, \sigma^2)$ a Gaussian distribution with $\theta = \{\mu, \sigma\}$). Such model is likely to suffer from the lack of expressiveness for many practical learning problems. To increase the expressiveness of our model and to capture the underlying generative process more accurately, we can introduce latent (hidden) variables into the model. Now, we can update our model to be $p_\theta(\mathbf{x}, \mathbf{z})$, where $\mathbf{z}$ denotes the latent variable. It is called latent variable because unlike $\mathbf{x}$, we cannot observe $\mathbf{z}$ from our data set. The new model $p_\theta(\mathbf{x}, \mathbf{z})$ can capture the dependence relation between $\mathbf{x}$ and $\mathbf{z}$. For example, the factorisation $p_\theta(\mathbf{x}, \mathbf{z}) = p_{\theta_x}(\mathbf{x}|\mathbf{z}) \cdot p_{\theta_z}(\mathbf{z})$ implies a latent structure that $\mathbf{x}$ depends on $\mathbf{z}$, where $\theta = \{\theta_x, \theta_z\}$. In addition, we also need to define the distribution $p_{\theta_z}(\mathbf{z})$ for $\mathbf{z}$ based on our belief of the underlying latent process. With the latent variable, our the model for $p(\mathbf{x})$ becomes

$$p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z}) \, \mathrm{d}\mathbf{z}. \tag{2.19}$$

Similarly, we would like to use maximum likelihood learning to find the optimal θ^*.

2.1.4 Learning with Latent Variables

Following Eq.(2.10), we have:

$$\theta^* = \arg\max_\theta \sum_{i=1}^{n} \log p_\theta(\mathbf{x}_i) \tag{2.20}$$

$$= \arg\max_{\theta_x, \theta_z} \sum_{i=1}^{n} \log \int p_\theta(\mathbf{x}_i, \mathbf{z}) \, \mathrm{d}\mathbf{z}. \tag{2.21}$$

There is a challenges for doing maximum likelihood learning here. Since $\mathbf{z}$ is a latent variable and we do not have observable samples of $\mathbf{z}$, maximisation is not straightforward.

Free Energy

One idea is to optimise over an alternative objective. Let us rewrite the log likelihood from Eq.(2.21) with a single observation $\mathbf{X} = \{\mathbf{x}\}$ for simplicity:

$$\log p_\theta(\mathbf{x}) = \log \int p_\theta(\mathbf{x}, \mathbf{z}) \, \mathrm{d}\mathbf{z}. \tag{2.22}$$

Jensen's inequality [Jensen et al., 1906] states that for a probability measure α and a concave function $f(\cdot)$, we have

$$f\left(\langle x \rangle_\alpha\right) \geq \langle f\left(x\right) \rangle_\alpha . \tag{2.23}$$

Since $\log(\cdot)$ is a concave function, for any distribution $q(\mathbf{z})$ over the latent variable $\mathbf{z}$, there is a lower bound for the log likelihood:

$$\log p_\theta(\mathbf{x}) = \log \int p_\theta(\mathbf{x}, \mathbf{z}) \cdot \frac{q(\mathbf{z})}{q(\mathbf{z})} \, \mathrm{d}\mathbf{z} \tag{2.24}$$

$$= \log \left\langle \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})} \right\rangle_{q(\mathbf{z})} \tag{2.25}$$

$$\geq \left\langle \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})} \right\rangle_{q(\mathbf{z})} \tag{2.26}$$

$$= \langle \log p_\theta(\mathbf{x}, \mathbf{z}) \rangle_{q(\mathbf{z})} + \mathrm{H}[q(\mathbf{z})] \tag{2.27}$$

$$\stackrel{\text{def}}{=} \mathcal{F}(q, \theta). \tag{2.28}$$

We call the lower bound $\mathcal{F}(q, \theta)$ *free energy*, which is also known as *Evidence Lower Bound (ELBO)* [Blei et al., 2017] in literature. It can also be rewritten as

$$\mathcal{F}(q, \theta) = \langle \log p_\theta(\mathbf{x}, \mathbf{z}) \rangle_{q(\mathbf{z})} + \mathrm{H}[q(\mathbf{z})] \tag{2.29}$$

$$= \langle \log p_\theta(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}) \rangle_{q(\mathbf{z})} \tag{2.30}$$

$$= \left\langle \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q(\mathbf{z})} \right\rangle_{q(\mathbf{z})} \tag{2.31}$$

$$= \left\langle \log \frac{p_\theta(\mathbf{z}|\mathbf{x}) \cdot p_\theta(\mathbf{x})}{q(\mathbf{z})} \right\rangle_{q(\mathbf{z})} \tag{2.32}$$

$$= \left\langle \log \frac{p_\theta(\mathbf{x})}{q(\mathbf{z})} \right\rangle_{q(\mathbf{z})} + \left\langle \log \frac{p_\theta(\mathbf{z}|\mathbf{x})}{q(\mathbf{z})} \right\rangle_{q(\mathbf{z})} \tag{2.33}$$

$$= \log p_\theta(\mathbf{x}) - \mathrm{KL}\left[q(\mathbf{z}) || p_\theta(\mathbf{z}|\mathbf{x})\right]. \tag{2.34}$$

From the properties of KL divergence, we know that when $q(\mathbf{z}) = p_\theta(\mathbf{z}|\mathbf{x})$, the value of $\mathrm{KL}\left[q(\mathbf{z}) || p_\theta(\mathbf{z}|\mathbf{x})\right] = 0$, which also leads to $\mathcal{F}(q, \theta) = \log p_\theta(\mathbf{x})$.

Expectation Maximisation

We can now use free energy for our optimisation. Instead of trying to find the θ that maximise the log likelihood $\log p_\theta(\mathbf{x})$, the new objective is to find θ and $q(\mathbf{z})$ that maximise the lower bound $\mathcal{F}(q, \theta)$. Expectation Maximisation (EM) algorithm suggests an iterative approach [Dempster et al., 1977]. It starts with arbitrary values of parameters $\theta = \{\theta_x, \theta_z\}$, then iterates two steps till $\mathcal{F}$ converged, which is guaranteed [Wu, 1983]. At k-th iteration, we have:

- **E-step:** Fix parameters θ, maximise $\mathcal{F}(q, \theta)$ w.r.t. $q(\mathbf{z})$. From Eq.(2.34), we know this maximum is achieved when $q(\mathbf{z}) = p_\theta(\mathbf{z}|\mathbf{x})$, thus

$$q^{(k)}(\mathbf{z}) \coloneqq \arg\max_{q(\mathbf{z})} \mathcal{F}(q, \theta^{(k-1)}) = p_{\theta^{(k-1)}}(\mathbf{z}|\mathbf{x})$$

- **M-step:** Fix $q(\mathbf{z})$, maximise $\mathcal{F}(q, \theta)$ w.r.t. parameters θ. From Eq.(2.29), we know the entropy term $\mathrm{H}[q(\mathbf{z})]$ does not depend on the parameters θ, thus

$$\theta^{(k)} \coloneqq \arg\max_{\theta} \mathcal{F}(q^{(k-1)}, \theta) = \langle \log p_\theta(\mathbf{x}, \mathbf{z}) \rangle_{q^{(k-1)}(\mathbf{z})}.$$

With EM, we do not need to worry about the latent $\mathbf{z}$ being not observable. For simple model that has a closed form posterior $p_\theta(\mathbf{z}|\mathbf{x})$, it is easy to perform EM. However, it can be intractable if we cannot infer the posterior $p_\theta(\mathbf{z}|\mathbf{x})$ in the E-step or if we cannot calculate the expectation in the M-step.

Variational Inference

Following Bayes' theorem, to infer the posterior $p_\theta(\mathbf{z}|\mathbf{x})$, we have

$$p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p_\theta(\mathbf{x}|\mathbf{z}) \cdot p_\theta(\mathbf{z})}{p_\theta(\mathbf{x})}, \tag{2.35}$$

which has the integral $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z})\ \mathrm{d}\mathbf{z}$. However, for many interesting problems, the integral may not have a closed form solution or we cannot differentiate it. Thus, it is often intractable to do exact inference in latent variable models. Instead, we can use variational method to do approximate inference.

The idea for variational inference is that rather then simply using the posterior $p_\theta(\mathbf{z}|\mathbf{x})$ as the distribution $q(\mathbf{z})$, we choose $q(\mathbf{z})$ from a constraint variational family $\mathcal{Q}$. If $\mathcal{Q}$ is a set of tractable distributions, then the problem becomes tractable. In other words, we can use tractable distributions from $\mathcal{Q}$ to approximate the intractable posterior $p_\theta(\mathbf{z}|\mathbf{x})$. The cost for such approximation is that we may not converge to the maximum log likelihood $\log p_\theta(\mathbf{x})$, because from Eq.(2.34) we know that if $p_\theta(\mathbf{z}|\mathbf{x}) \notin \mathcal{Q}$ the $\mathrm{KL}\left[q(\mathbf{z})||p_\theta(\mathbf{z}|\mathbf{x})\right] \neq 0$. To minimise the approximation error, we often choose $q(\cdot)$ to has the same conditional form as the posterior, i.e. $q(\mathbf{z}|\mathbf{x})$. The optimisation objective can be updated to

- **E-step:**
$$q^{(k)}(\mathbf{z}|\mathbf{x}) \coloneqq \arg\max_{q(\mathbf{z}|\mathbf{x})\in\mathcal{Q}}\ \mathcal{F}(q, \theta^{(k-1)})$$
- **M-step:** (Unchanged)
$$\theta^{(k)} \coloneqq \arg\max_{\theta}\ \mathcal{F}(q^{(k-1)}, \theta) = \left\langle \log p_\theta(\mathbf{x}, \mathbf{z}) \right\rangle_{q^{(k-1)}(\mathbf{z}|\mathbf{x})}$$

There are several methods to pick the variational family $\mathcal{Q}$. One is to define $\mathcal{Q}$ as a parametric family $q(\mathbf{z}; f_\phi(\mathbf{x}))$ such that $q_\phi(\mathbf{z}|\mathbf{x}) \in q(\mathbf{z}; f_\phi(\mathbf{x}))$. For example, let $\mathcal{Q}$ be a Gaussian distribution parameterised with neural networks (i.e. $q(\mathbf{z}; f_\phi(\mathbf{x})) = \mathcal{N}(f_{\phi_\mu}(\mathbf{x}), f_{\phi_\sigma}(\mathbf{x}))$), where $f_{\phi_\mu}(\cdot)$ and $f_{\phi_\sigma}(\cdot)$ are neural networks with weights $\phi = \{\phi_\mu, \phi_\sigma\}$. These mapping function f are also known as *recognition models*. Thus, with parametric variational methods, choosing the optimal $q \in \mathcal{Q}$ w.r.t. $\mathcal{F}$ is the same as choosing the optimal ϕ for $q(\mathbf{z}; f_\phi(\mathbf{x}))$ w.r.t. $\mathcal{F}$. And we can rewrite $\mathcal{F}(q, \theta)$ as $\mathcal{F}(\phi, \theta)$. The E-step becomes

$$\phi^{(k)} \coloneqq \arg\max_{\phi}\ \mathcal{F}(\phi, \theta^{(k-1)}).$$

Our goal is now to maximise

$$\begin{aligned}
\mathcal{F}(\phi, \theta) &= \left\langle \log p_\theta(\mathbf{x}, \mathbf{z}) \right\rangle_{q_\phi(\mathbf{z}|\mathbf{x})} + \mathrm{H}[q_\phi(\mathbf{z}|\mathbf{x})] && (2.36)\\
&= \left\langle \log p_\theta(\mathbf{x}|\mathbf{z}) \right\rangle_{q_\phi(\mathbf{z}|\mathbf{x})} + \left\langle \log p_\theta(\mathbf{z}) \right\rangle_{q_\phi(\mathbf{z}|\mathbf{x})} - \left\langle \log q_\phi(\mathbf{z}|\mathbf{x}) \right\rangle_{q_\phi(\mathbf{z}|\mathbf{x})} && (2.37)\\
&= \left\langle \log p_\theta(\mathbf{x}|\mathbf{z}) \right\rangle_{q_\phi(\mathbf{z}|\mathbf{x})} - \mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})\right] && (2.38)
\end{aligned}$$

w.r.t. the variational parameters ϕ and generative parameters θ. Gradient based optimisation is the common alternative when we do not have a closed form solution. The term $\mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})\right]$ has an analytical form representation, which is easy to differentiate. However, taking the gradient of $\mathcal{F}(\phi,\theta)$ w.r.t. ϕ requires us to differentiate an expectation $\langle h(\mathbf{z})\rangle_{q_\phi(\mathbf{z}|\mathbf{x})}$[2], which is a challenge.

Reparameterisation Trick

To solve the problem, we can reparameterise our random variable $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$ as a deterministic variable given a random noise $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$, i.e. $\mathbf{z} = g_\phi(\mathbf{x}, \boldsymbol{\epsilon})$, where $g_\phi(\cdot,\cdot)$ is some deterministic mapping parameterised by ϕ. Therefore, the original expectation can be rewritten as

$$\langle h(\mathbf{z})\rangle_{q_\phi(\mathbf{z}|\mathbf{x})} = \langle h(g_\phi(\mathbf{x},\boldsymbol{\epsilon}))\rangle_{p(\boldsymbol{\epsilon})}, \tag{2.39}$$

which is differentiable w.r.t. ϕ and can be unbiasedly evaluated through Monte Carlo estimate

$$\langle h(g_\phi(\mathbf{x},\boldsymbol{\epsilon}))\rangle_{p(\boldsymbol{\epsilon})} \approx \frac{1}{N}\sum_{n=1}^{N} h(g_\phi(\mathbf{x},\boldsymbol{\epsilon}^{(n)})). \tag{2.40}$$

Another advantage for reparameterising the variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$ is that we can now combine the E-step and M-step together. So we can optimise $\mathcal{F}(\phi,\theta)$ w.r.t. ϕ and θ at the same time by gradient descent.

2.1.5 Variational Autoencoder

Variational Autoencoder (VAE) provides us an example of how to do learning and inference in deep latent variable models [Kingma and Welling, 2014], which combines all the concepts above. Again, we would like to model $p(\mathbf{x})$ through training set $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$ and assume $\mathbf{x}$ is generated from latent $\mathbf{z}$ through conditional distribution $p(\mathbf{x}|\mathbf{z})$. The prior for our latent $\mathbf{z}$ is set to be a multivariate Gaussian $p(\mathbf{z}) = \mathcal{N}(\mathbf{0},\mathbf{I})$, where $\mathbf{0}$ denotes a vector of 0 and $\mathbf{I}$ denotes an identity matrix.

Decode: If $\mathbf{x}$ is in the form of binary vector, we can use Bernoulli distributions for our model $p_\theta(\mathbf{x}|\mathbf{z})$. It means that given a latent $\mathbf{z}'$, the resulting Bernoulli parameters are $\mathbf{p}' = \mathit{decoder}_\theta(\mathbf{z}') = p_\theta(\mathbf{x}|\mathbf{z}')$, where $\mathit{decoder}_\theta(\cdot)$ is a neural network with weights θ. A sample can be generated by drawing $\mathbf{x}'$ from Bernoulli distribution $\mathit{Bernoulli}(\mathbf{p}')$. We call the model for $p_\theta(\mathbf{x}|\mathbf{z})$ as *decoder* because we can generate (decode) $\mathbf{x}$ from the hidden (encoded) $\mathbf{z}$.

Encode: Since the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$ is intractable, we use a variational distribution $q_\phi(\mathbf{z}|\mathbf{x})$ to approximate it. We assume the posterior can be approximated using diagonal Gaussian as the variational family $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\sigma}^2_\mathbf{x}\mathbf{I})$. $\boldsymbol{\mu}_\mathbf{x}$ and $\boldsymbol{\sigma}^2_\mathbf{x}$ are the mean and diagonal variance of the distribution, which are output of the recognition function $(\boldsymbol{\mu}_\mathbf{x}, \boldsymbol{\sigma}_\mathbf{x}) = \mathit{encoder}_\phi(\mathbf{x})$. $\mathit{encoder}_\phi(\cdot)$ is a neural network with parameters ϕ. Given an input $\mathbf{x}'$, we can sample $\mathbf{z}$ from the posterior $q_\phi(\mathbf{z}|\mathbf{x}')$ by: First, pass $\mathbf{x}'$ into the encoder $(\boldsymbol{\mu}_{\mathbf{x}'}, \boldsymbol{\sigma}_{\mathbf{x}'}) = \mathit{encoder}_\phi(\mathbf{x})$; Then, with the reparameterisation trick, we have $\mathbf{z}' = \boldsymbol{\mu}_{\mathbf{x}'} + \boldsymbol{\sigma}_{\mathbf{x}'} \odot \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ and $\odot$ denotes element-wise product. We call the model for $q_\phi(\mathbf{z}|\mathbf{x})$ as *encoder*, since it encodes $\mathbf{x}$ into its latent form.

[2]For simplicity, here we use $h(\mathbf{z})$ to denote any function depends on $\mathbf{z}$, such as $\log p_\theta(\mathbf{x}|\mathbf{z})$.

With $p(\mathbf{z})$, $p_\theta(\mathbf{x}|\mathbf{z})$ and $q_\phi(\mathbf{z}|\mathbf{x})$ defined, we can now maximise the free energy

$$\mathcal{F}(\phi, \theta) = \langle \log p_\theta(\mathbf{x}|\mathbf{z}) \rangle_{q_\phi(\mathbf{z}|\mathbf{x})} - \mathrm{KL}\left[q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})\right]$$

using stochastic gradient descent[3] (SGD) by taking its gradient w.r.t. ϕ and θ.

2.2 Semi-Supervised Learning

As mentioned briefly before, semi-supervised learning focuses on learning problems with both labelled data $\mathcal{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and unlabelled data $\mathcal{U} = \{(\mathbf{x}_j)\}_{j=1}^m$, where y_i is the label of $\mathbf{x}_i$. Instead of using only the labelled data, semi-supervised learning aims to utilise both the labelled and unlabelled data to create models with better performance. In order to improve the model performance using unlabelled data, there is a necessary condition in semi-supervised learning needs to be satisfied, which is that the underlying marginal distribution $p(\mathbf{x})$ has to carry information that is useful to infer $p(y|\mathbf{x})$ [Chapelle et al., 2010, Van Engelen and Hoos, 2020]. By meeting this condition, we can learn information about $p(\mathbf{x})$ from the unlabelled data, then help with modelling $p(y|\mathbf{x})$. If this condition is not met, the use of unlabelled data will not improve our model of $p(y|\mathbf{x})$ through semi-supervised learning.

A commonly used approach for semi-supervised learning is *self-training* [Zhu, 2005]. Its idea is simply using the initial labelled data to train a classifier, then the trained classifier is used to assign labels to the unlabelled data. The most confident set of these self-labelled data will be added into the training set and retrain the model from the beginning. It is a repeated procedure. Self-training has been used in various natural language processing and computer vision tasks [Yarowsky, 1995, Riloff et al., 2003, Rosenberg et al., 2005]. More complicated and powerful methods have been studied for the past two decades, which are well categorised in semi-supervised learning survey papers [Zhu, 2005, Van Engelen and Hoos, 2020].

2.3 Semi-Supervised Generative Model

In classification and regression tasks, generative models are used to estimate the joint distribution $p(y, \mathbf{x})$. For classification tasks, it is easy to see the connection of modelling the labelled data and unlabelled data from the marginalisation $p(\mathbf{x}) = \sum_{y \in Y} p(y, \mathbf{x})$. Therefore, generative model can be a natural choice for semi-supervised learning. Previous research has been focused on tractable models such as Gaussian mixture models [Zhu, 2005] and non-parametric models like Gaussian Process [Adams and Ghahramani, 2009]. Recently, a more general, flexible and scalable approach has been proposed, which is based on the ideas from VAE [Kingma et al., 2014]. In their paper, this deep generative model is denoted as M2. To be consistent with their paper, we will use the name M2 in this report as well.

In the semi-supervised generative setting of M2, we are given both the labelled data $\mathcal{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and unlabelled data $\mathcal{U} = \{(\mathbf{x}_j)\}_{j=1}^m$, where each observation $\mathbf{x}_i \in \mathbb{R}^D$ and the class label $y_i \in \{1, \ldots, L\}$. The empirical distributions over our data set $\mathcal{L}$ and $\mathcal{U}$ are referred as $p_L(\mathbf{x}, y)$ and $p_U(\mathbf{x})$ in order to differentiate it from the underlying unknown distributions $p(\mathbf{x}, y)$ and $p(\mathbf{x})$.

[3] Gradient ascent in this expression. Or gradient descent with negation of $\mathcal{F}$.

2.3.1 Modelling Assumptions

In M2, we assume that the data $\mathbf{x}$ is generated from a discrete latent class y and a continuous latent representation $\mathbf{z}$. Let the prior of y to be a categorical distribution with initial parameter $\boldsymbol{\pi}$ and the prior of $\mathbf{z}$ to be a centred isotropic multivariate Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$, where the mean $\mathbf{0}$ is a vector of 0 and the covariance matrix $\mathbf{I}$ is an identity matrix. Given y and $\mathbf{z}$, $\mathbf{x}$ is generated from $p_\theta(\mathbf{x}|y, \mathbf{z})$, which can be set to a suitable distribution such as a Gaussian or Bernoulli with its parameters output from decoder networks with weights θ. When we are given unlabelled data, y will be treated as a latent variable. Based on our assumptions, y and $\mathbf{z}$ are marginally independent and $\mathbf{x}$ depends on both of them, the latent variable $\mathbf{z}$ is expected to capture the class invariant information and y is expected to capture the class specific information. For example, when we are modelling hand written digits, y will capture the digit class and $\mathbf{z}$ will capture the writing style. At the moment, we have made assumptions for

$$p(y) = Cat(\boldsymbol{\pi}), \quad p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I}), \quad p_\theta(\mathbf{x}|y, \mathbf{z}).$$

From previous sections, we know that to do learning with such latent variable model requires the inference of posterior distribution. In addition, for classification tasks, the posterior distribution $p(y|\mathbf{x})$ can be used for predicting the missing labels. However, just like VAE, the exact inference for the posterior is intractable.

2.3.2 Learning and Inference

Instead of exact inference, M2 makes use of variational inference to approximate the posterior. The variational distribution $q_\phi(y, \mathbf{z}|\mathbf{x})$ is created for approximating the posterior $p(y, \mathbf{z}|\mathbf{x})$, which can be factorised as

$$q_\phi(y, \mathbf{z}|\mathbf{x}) = q_\phi(\mathbf{z}|\mathbf{x}, y) \cdot q_\phi(y|\mathbf{x}).$$

The inference model for $q_\phi(\mathbf{z}|\mathbf{x}, y)$ is specified as a diagonal Gaussian distribution, and the model for $q_\phi(y|\mathbf{x})$ is specified as a Categorical distribution:

$$\begin{aligned} q_\phi(\mathbf{z}|\mathbf{x}, y) &= \mathcal{N}(\boldsymbol{\mu}_{\mathbf{x},y}, \boldsymbol{\sigma}^2_{\mathbf{x},y}\mathbf{I}), \quad (\boldsymbol{\mu}_{\mathbf{x},y}, \boldsymbol{\sigma}_{\mathbf{x},y}) = encoder_{\phi_\mathbf{z}}(\mathbf{x}, y), \\ q_\phi(y|\mathbf{x}) &= Cat(\boldsymbol{\pi}_\mathbf{x}), \qquad\qquad \boldsymbol{\pi}_\mathbf{x} = encoder_{\phi_y}(\mathbf{x}), \end{aligned}$$

where $encoder_{\phi_\mathbf{z}}(\cdot)$ and $encoder_{\phi_y}(\cdot)$ are neural networks parameterised by $\phi_\mathbf{z}$ and ϕ_y that output the corresponding distribution parameters. With all the necessary distributions defined, we can write down the free energy as our learning objective.

For Labelled Data

When labelled data are given, both $\mathbf{x}$ and y are observed, the only latent variable is $\mathbf{z}$. Then, the free energy is the lower bound for the join distribution

$$\begin{aligned} \log p_\theta(\mathbf{x}, y) &\geq \langle \log p_\theta(\mathbf{x}, y, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}, y) \rangle_{q_\phi(\mathbf{z}|\mathbf{x},y)} && (2.41) \\ &= \langle \log p_\theta(\mathbf{x}|y, \mathbf{z}) + \log p(y) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}, y) \rangle_{q_\phi(\mathbf{z}|\mathbf{x},y)} && (2.42) \\ &\stackrel{\text{def}}{=} -\mathbf{L}(\mathbf{x}, y) && (2.43) \end{aligned}$$

For Unlabelled Data

When unlabelled data are given, only $\mathbf{x}$ is observed, both y and $\mathbf{z}$ are latent variables. The resulting lower bound is

$$\log p_\theta(\mathbf{x}) \geq \langle \log p_\theta(\mathbf{x}, y, \mathbf{z}) - \log q_\phi(y, \mathbf{z}|\mathbf{x}) \rangle_{q_\phi(y,\mathbf{z}|\mathbf{x})} \tag{2.44}$$

$$= \langle \log p_\theta(\mathbf{x}|y, \mathbf{z}) + \log p(y) + \log p(\mathbf{z}) - \log q_\phi(y, \mathbf{z}|\mathbf{x}) \rangle_{q_\phi(y,\mathbf{z}|\mathbf{x})} \tag{2.45}$$

$$= \left\langle \langle \log p_\theta(\mathbf{x}|y, \mathbf{z}) + \log p(y) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}, y) - \log q_\phi(y|\mathbf{x}) \rangle_{q_\phi(\mathbf{z}|\mathbf{x},y)} \right\rangle_{q_\phi(y|\mathbf{x})}$$

$$= \left\langle \langle \log p_\theta(\mathbf{x}|y, \mathbf{z}) + \log p(y) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}, y) \rangle_{q_\phi(\mathbf{z}|\mathbf{x},y)} \right\rangle_{q_\phi(y|\mathbf{x})} + \mathrm{H}(q_\phi(y|\mathbf{x}))$$

$$= \langle -\mathbf{L}(\mathbf{x}, y) \rangle_{q_\phi(y|\mathbf{x})} + \mathrm{H}(q_\phi(y|\mathbf{x})) \tag{2.46}$$

$$= \sum_y q_\phi(y|\mathbf{x}) \cdot (-\mathbf{L}(\mathbf{x}, y)) + \mathrm{H}(q_\phi(y|\mathbf{x})) \tag{2.47}$$

$$\stackrel{\text{def}}{=} -\mathbf{U}(\mathbf{x}) \tag{2.48}$$

Therefore, the optimisation objective for the entire training set including both labelled and unlabelled data is to minimise the following:

$$\mathcal{J} = \sum_{(\mathbf{x},y)\sim p_L} \mathbf{L}(\mathbf{x}, y) + \sum_{\mathbf{x}\sim p_U} \mathbf{U}(\mathbf{x}), \tag{2.49}$$

where p_L is the empirical distribution of labelled data, p_U is the empirical distribution of unlabelled data. This objective directly indicates how labelled and unlabelled data are used during the training of the generative model, which is what makes it a semi-supervised generative model.

The inference model $q_\phi(y|\mathbf{x})$ from Eq.(2.47) can be used as a discriminative classifier. However, it only gets triggered from the *unlabelled* data in the training objective Eq.(2.49), which means labelled data are not used for training $q_\phi(y|\mathbf{x})$. It is obviously not desirable if we would like to use $q_\phi(y|\mathbf{x})$ as a classifier. To make sure $q_\phi(y|\mathbf{x})$ also learns from the labelled data, the authors extend the learning objective from Eq.(2.49) with an additional classification loss [Kingma et al., 2014]:

$$\mathcal{J}^\alpha = \mathcal{J} + \alpha \cdot \langle -\log q_\phi(y|\mathbf{x}) \rangle_{p_L(\mathbf{x},y)}, \tag{2.50}$$

where α is a hyperparameter for weighting the discriminative learning and the generative learning. In our experiments, we use the same $\alpha = 0.1 \cdot n$ as in the paper, where n represents the number of labelled training data.

With the objective Eq.(2.50), we can simply take the gradient w.r.t. θ and ϕ with reparameterisation trick, and use stochastic gradient descent to train the model.

Chapter 3

Active Learning with Semi-Supervised Generative Model

In previous chapters, we introduce the concept of active learning as well as semi-supervised generative model. In this chapter, we look at how semi-supervised generative models, the M2 model in particular, can be integrated with active learning procedure. We also propose new acquisition functions that take into account the unique properties of the semi-supervised generative model.

3.1 Correctness of the Model

Before using the M2 model in active learning, we would like to make sure the model itself is asymptotically correct, which was not proved in its original paper [Kingma et al., 2014]. In other words, if the variable $\mathbf{x}$ is indeed generated from our model $p_\theta(\mathbf{x}|y,\mathbf{z})$ with the real parameters θ^*, then as the number of training data approaches infinity, minimising the training objective in Eq.(2.49) and Eq.(2.50) should give us the real $\theta = \theta^*$.

3.1.1 Proof of Objective $\mathcal{J}$

We first start with the objective in Eq.(2.49)

$$\mathcal{J} = \sum_{(\mathbf{x},y)\sim p_L} \mathbf{L}(\mathbf{x},y) + \sum_{\mathbf{x}\sim p_U} \mathbf{U}(\mathbf{x}).$$

[Proof 1] Assume the real θ for the likelihood $p_\theta(\mathbf{x}|y,\mathbf{z})$ is θ^*, then the training data are generated by $p_{\theta^*}(\mathbf{x},y)$. Note that from Eq.(2.43) and Eq.(2.48), we have $\log p_\theta(\mathbf{x},y) \geq -\mathbf{L}(\mathbf{x},y)$ and $\log p_\theta(\mathbf{x}) \geq -\mathbf{U}(\mathbf{x})$. Therefore, if the number of samples in our training set

approaches infinity, the objective Eq.(2.49) can be rewritten as:

$$\begin{aligned}
\mathcal{J}^* &= \langle \mathbf{L}(\mathbf{x}, y) + \mathbf{U}(\mathbf{x}) \rangle_{p_{\theta^*}(\mathbf{x},y)} && (3.1)\\
&\geq - \langle \log p_\theta(\mathbf{x}, y) + \log p_\theta(\mathbf{x}) \rangle_{p_{\theta^*}(\mathbf{x},y)} && (3.2)\\
&= - \langle \log p_\theta(\mathbf{x}, y) \rangle_{p_{\theta^*}(\mathbf{x},y)} - \langle \log p_\theta(\mathbf{x}) \rangle_{p_{\theta^*}(\mathbf{x},y)} && (3.3)\\
&= - \langle \log p_\theta(\mathbf{x}, y) \rangle_{p_{\theta^*}(\mathbf{x},y)} - \langle \log p_\theta(\mathbf{x}) \rangle_{p_{\theta^*}(\mathbf{x})} && (3.4)\\
&= \mathrm{KL}\left[p_{\theta^*}(\mathbf{x}, y) || p_\theta(\mathbf{x}, y)\right] + \mathrm{KL}\left[p_{\theta^*}(\mathbf{x}) || p_\theta(\mathbf{x})\right] + \text{const.} && (3.5)
\end{aligned}$$

Hence, minimising $\mathcal{J}^*$ will minimise

$$\mathrm{KL}\left[p_{\theta^*}(\mathbf{x}, y) || p_\theta(\mathbf{x}, y)\right] + \mathrm{KL}\left[p_{\theta^*}(\mathbf{x}) || p_\theta(\mathbf{x})\right] + \text{const.},$$

which achieves its minimum value when $\theta = \theta^*$.

3.1.2 Proof of Objective $\mathcal{J}^\alpha$

Then we can prove the objective in Eq.(2.50)

$$\mathcal{J}^\alpha = \mathcal{J} + \alpha \cdot \langle -\log q_\phi(y|\mathbf{x}) \rangle_{p_L(\mathbf{x},y)}.$$

[Proof 2] Similarly for Eq.(2.50), since the real $p_{\theta^*}(\mathbf{x}, y)$ can be factorised as $p_{\theta^*}(\mathbf{x}) \cdot p_{\theta^*}(y|\mathbf{x})$, we have:

$$\begin{aligned}
&\mathcal{J}^{\alpha *} && (3.6)\\
\geq &\mathcal{J}^* + \alpha \cdot \langle -\log q_\phi(y|\mathbf{x}) \rangle_{p_{\theta^*}(\mathbf{x},y)} && (3.7)\\
= &\mathcal{J}^* + \alpha \cdot \left\langle \langle -\log q_\phi(y|\mathbf{x}) \rangle_{p_{\theta^*}(y|\mathbf{x})} \right\rangle_{p_{\theta^*}(\mathbf{x})} && (3.8)\\
= &\mathcal{J}^* + \alpha \cdot \left\langle \langle -\log q_\phi(y|\mathbf{x}) + \log p_{\theta^*}(y|\mathbf{x}) \rangle_{p_{\theta^*}(y|\mathbf{x})} \right\rangle_{p_{\theta^*}(\mathbf{x})} + \alpha \cdot \langle \mathrm{H}\left[\log p_{\theta^*}(y|\mathbf{x})\right] \rangle_{p_{\theta^*}(\mathbf{x})} && (3.9)\\
= &\mathcal{J}^* + \alpha \cdot \langle \mathrm{KL}[p_{\theta^*}(y|\mathbf{x}) || q_\phi(y|\mathbf{x})] \rangle_{p_{\theta^*}(\mathbf{x})} + \alpha \cdot \langle \mathrm{H}\left[\log p_{\theta^*}(y|\mathbf{x})\right] \rangle_{p_{\theta^*}(\mathbf{x})} && (3.10)\\
\geq &\mathrm{KL}[p_{\theta^*}(\mathbf{x}, y) || p_\theta(\mathbf{x}, y)] + \mathrm{KL}[p_{\theta^*}(\mathbf{x}) || p_\theta(\mathbf{x})] + \alpha \cdot \langle \mathrm{KL}[p_{\theta^*}(y|\mathbf{x}) || q_\phi(y|\mathbf{x})] \rangle_{p_{\theta^*}(\mathbf{x})} \\
&+ \text{const.} && (3.11)
\end{aligned}$$

Thus, the minimum of $\mathcal{J}^{\alpha *}$ is achieved when $\theta = \theta^*$ and $q_\phi(y|\mathbf{x}) = p_{\theta^*}(y|\mathbf{x})$. With the two proofs, we know that the training objective for the semi-supervised generative model is indeed asymptotically correct.

3.2 Integrating the Model with Active Learning

The implementation is relatively straightforward. In the k-th training step of our active learning loop, we use the M2 model and train it with both labelled $\mathcal{L}_k$ and unlabelled data $\mathcal{U}_k$ on the objective function Eq.(2.50) till convergence. Then, in the acquisition step, each unlabelled data $\mathbf{x} \in \mathcal{U}_k$ will be passed into an acquisition function $\mathcal{A}(\mathbf{x})$ for scoring. Based on the acquisition scores, a subset of unlabelled data will be assigned with their real labels

$\mathcal{O}(\mathbf{x})$ and moved into the labelled data set. Next, the updated data sets $\mathcal{L}_{k+1}$ and $\mathcal{U}_{k+1}$ will be used to retrain the M2 model, and the process repeats for K iterations, where K is a hyperparameter represents the maximum number of active learning loop to be run. Note that in our experiments, we only acquire a single unlabelled data in each iteration. In addition, since labelling oracle does not exist, we simulate such process by dividing our original labelled training set $\mathcal{D}$ into $\mathcal{L}$ and $\mathcal{U}$, where labels in $\mathcal{U}$ are hidden from the model and it is only revealed when a data is acquired. This new active learning procedure is summarised in Algorithm 1.

Algorithm 1: Active Learning Procedure with Semi-Supervised Generative Model

Given $\mathcal{L}_1$, $\mathcal{U}_1$, K and $\mathcal{A}(\cdot)$
Define $p(\mathbf{z}), p(y), p_\theta(\mathbf{x}|y,\mathbf{z}), q_\phi(y|\mathbf{x}), q_\phi(\mathbf{z}|y,\mathbf{x})$
`/* Active Learning Loop */`
for $k \in \{1, \dots, K\}$ **do**
  `/* 1.Training */`
  Initialise ϕ, θ
  while $\mathcal{J}^\alpha$ *has not converged* **do**
    $\forall \mathbf{x}_i \in \mathcal{U}_k$ and $(\mathbf{x}_i, y_i) \in \mathcal{L}_k$
    $y_i \sim q_\phi(y_i|\mathbf{x}_i)$ if $\mathbf{x}_i \in \mathcal{U}_k$
    $\mathbf{z}_i \sim q_\phi(\mathbf{z}_i|y_i, \mathbf{x}_i)$
    $\mathcal{J}^\alpha \leftarrow$ Eq.(2.50)
    $(\theta, \phi) \leftarrow (\theta, \phi) + (\frac{\partial \mathcal{J}^\alpha}{\partial \theta}, \frac{\partial \mathcal{J}^\alpha}{\partial \phi})$
  `/* 2.Acquisition */`
  $\mathbf{x}' = \arg\max_{\mathbf{x} \in \mathcal{U}_k} \mathcal{A}(\mathbf{x}, p_\theta(\mathbf{x}|y,\mathbf{z}), q_\phi(y|\mathbf{x}), q_\phi(\mathbf{z}|y,\mathbf{x}))$
  $\mathcal{U}_{k+1} \leftarrow \mathcal{U}_k \setminus \{\mathbf{x}'\}$
  $\mathcal{L}_{k+1} \leftarrow \mathcal{L}_k \cup \{(\mathbf{x}', \mathcal{O}(\mathbf{x}'))\}$

3.3 Acquisition

By replacing the usual discriminative model with semi-supervised generative model, we are given the ability to estimate the density of any given data $\log p_\theta(\mathbf{x})$. Hence, we can design new acquisition functions that utilise the density information.

3.3.1 With Entropy

The inference model $q_\phi(y|\mathbf{x})$ has the form of a discriminative classifier, and it is used for predicting the labels of unseen data. Therefore, the simplest idea is to use the predictive entropy of $q_\phi(y|\mathbf{x})$ as an acquisition function

$$\mathcal{A}_{\mathrm{H}}(\mathbf{x}) = \mathrm{H}[q_\phi(y|\mathbf{x})]. \tag{3.12}$$

Even though using predictive entropy for acquisition is not a new idea, the predictive distributions are modelled differently. In our case, the predictive distribution comes from the inference model $q_\phi(y|\mathbf{x})$, which is trained with both the unlabelled data in the generative objective Eq.(2.47) and the labelled data in the additional discriminative objective Eq.(2.50). In the usual active learning settings, the predictive distribution is only trained with labelled data and classification loss. By taking into account the generative process and the density

$p_\theta(\mathbf{x})$, $q_\phi(y|\mathbf{x})$ should be a better estimate for the distribution $p(y|\mathbf{x})$. Hence, it should acquire more informative data.

3.3.2 With Linear Combination of Entropy and Density

Since the new model can estimate the data density $p_\theta(\mathbf{x})$ as well as the entropy, we can create an acquisition function that combines both information. Ideally, by considering both entropy and density, we are able to acquire data that are both informative and representative. Following Eq.(2.43), i.e.:

$$\log p_\theta(\mathbf{x}) \geq \langle -\mathcal{L}(\mathbf{x}, y)\rangle_{q_\phi(y|\mathbf{x})} + \mathrm{H}(q_\phi(y|\mathbf{x})).$$

We can write our acquisition function as:

$$\mathcal{A}_\gamma(\mathbf{x}) = \gamma \cdot \langle -\mathcal{L}(\mathbf{x}, y)\rangle_{q_\phi(y|\mathbf{x})} + \mathrm{H}(q_\phi(y|\mathbf{x})). \tag{3.13}$$

When $\gamma = 0$, we have the predictive entropy, when $\gamma = 1$, we have the lower bound for $\log p_\theta(\mathbf{x})$. Therefore, γ might be viewed as a trade-off parameter s.t. $\gamma \in [0, 1]$. During acquisition, we select the unlabelled data which has the highest acquisition score $\mathcal{A}_\gamma(\mathbf{x})$.

Using only the predictive entropy (i.e. $\gamma = 0$), the acquisition function will select the unlabelled data $\mathbf{x}'$ with the highest predictive entropy, in other words, the model is most uncertain about the label y' of $\mathbf{x}'$. However, such data point might be an outlier in the dataset (i.e. $p_\theta(\mathbf{x}')$ is low). In contrast, with only the data density $p_\theta(\mathbf{x})$ (i.e. $\gamma = 1$), the acquisition function will pick the most likely data with $\arg\max_{\mathbf{x}} p_\theta(\mathbf{x})$ regardless of the predictive distribution $q_\phi(y|\mathbf{x})$. Therefore, the representative data of the unlabelled set with consideration of the current task $p(y|\mathbf{x})$ is likely to be acquired when $\gamma \in (0, 1)$, which means we can acquire data that has both high predictive entropy and high data density.

The above acquisition function Eq.(3.13) can also be derived from the goal that we would like to acquire data with both high predictive entropy $\mathrm{H}(q_\phi(y|\mathbf{x}))$ and high density $p_\theta(\mathbf{x})$. By using a trade-off parameter $\beta \in [0, \infty)$ between predictive entropy and the density, we have:

$$\begin{aligned}
\mathcal{A}_\beta(\mathbf{x}) &= \beta \cdot \log p_\theta(\mathbf{x}) + \mathrm{H}(q_\phi(y|\mathbf{x})) && (3.14)\\
&\geq \beta \cdot \langle -\mathcal{L}(\mathbf{x}, y)\rangle_{q_\phi(y|\mathbf{x})} + \beta \cdot \mathrm{H}(q_\phi(y|\mathbf{x})) + \mathrm{H}(q_\phi(y|\mathbf{x})) && (3.15)\\
&= \beta \cdot \langle -\mathcal{L}(\mathbf{x}, y)\rangle_{q_\phi(y|\mathbf{x})} + (\beta + 1) \cdot \mathrm{H}(q_\phi(y|\mathbf{x})) && (3.16)\\
&\propto \frac{\beta}{\beta + 1} \cdot \langle -\mathcal{L}(\mathbf{x}, y)\rangle_{q_\phi(y|\mathbf{x})} + \mathrm{H}(q_\phi(y|\mathbf{x})) && (3.17)\\
&= \gamma \cdot \langle -\mathcal{L}(\mathbf{x}, y)\rangle_{q_\phi(y|\mathbf{x})} + \mathrm{H}(q_\phi(y|\mathbf{x})) && (3.18)\\
&= \mathcal{A}_\gamma(\mathbf{x}) && (3.19)
\end{aligned}$$

where $\gamma = \frac{\beta}{\beta+1}$. Since $\beta \in [0, \infty)$, we have $\gamma \in [0, 1]$, therefore $\mathcal{A}_\beta(\mathbf{x})$ is the same as $\mathcal{A}_\gamma(\mathbf{x})$.

3.3.3 With Ordered Filtering of Entropy and Density

Another idea for acquisition is to use the predictive entropy and the density sequentially as separate selection criteria. For example, during the acquisition step, we can first select

a subset of M unlabelled data among $\mathcal{U}$, which have the M highest density values. Then, within the subset, we choose the data with highest predictive entropy. From the previous section, we know that when $\gamma = 1$, $\mathcal{A}_1(\mathbf{x})$ represents the lower bound of the density $\log p_\theta(\mathbf{x})$, and when $\gamma = 0$, $\mathcal{A}_0(\mathbf{x}) = \mathcal{A}_{\mathrm{H}}(\mathbf{x})$. Thus, the acquisition process can be written as

1. Filtering out the low density data

$$\{\mathbf{x}_m^*\}_{m=1}^M = \underset{\{\mathbf{x}_m\}_{m=1}^M \in \mathcal{U}}{\arg\max} \sum_{m=1}^M \log p_\theta(\mathbf{x}_m) = \underset{\{\mathbf{x}_m\}_{m=1}^M \in \mathcal{U}}{\arg\max} \sum_{m=1}^M \mathcal{A}_1(\mathbf{x}_m).$$

2. Selecting data with the highest predictive entropy

$$\mathbf{x}' = \underset{\mathbf{x} \in \{\mathbf{x}_m^*\}_{m=1}^M}{\arg\max} \mathcal{A}_0(\mathbf{x}).$$

Ideally, by first selecting a subset of data based on high density, we can filter out the outliers and avoid acquiring the data with high uncertain but low density.

Chapter 4

Experiments

To verify whether our new active learning settings can improve labels efficiency, experiments have been conducted for the proposed acquisition functions. In this chapter, we will first go through the setups of our experiments, then we will look at the performance and analysis of the three proposed acquisition functions.

Our active learning algorithm is given in section §3.2. Additionally, in order to evaluate the algorithm, we need to test the classification accuracy of the model trained at each active learning loop. For example, if we would like to label 100 data and we only acquire a single unlabelled data at each iteration of the active learning loop, then we will train 100 models during a single run of the algorithm. At the end of each iteration, we use the same test data to test the corresponding model for its accuracy. The model trained with more labelled data should result in higher test accuracy. An example to illustrate the idea is showed in Figure 4.1.

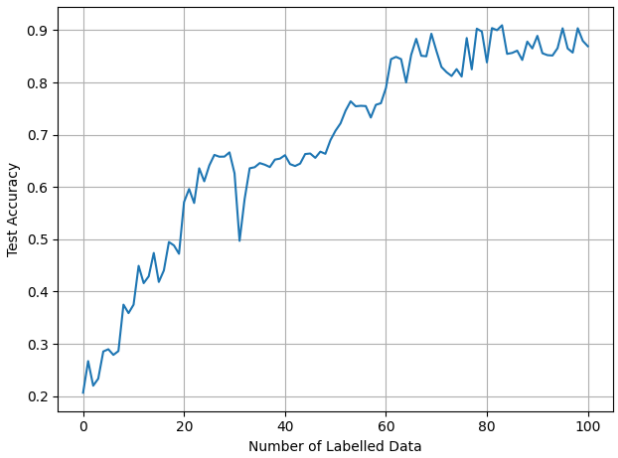


Figure 4.1: An example illustrating how active learning can be evaluated. The x-axis represents the amount of labelled data used for training. The y-axis represents the test accuracy for different models.

When comparing two active learning algorithms, we will expect the one with a better performance to plot its accuracy higher. It means that with the same amount of labelled data, one model outperforms the other. If the two models have the similar *expressiveness*, then

such difference in performance is likely caused by the difference in the labelled data, which is the result of the distinct acquisition functions and the different active learning settings.

REMARK 2 (Model Expressiveness) *Expressiveness is a concept from learning theory, which describes the ability of a model to capture complex functions. A more expressive model has the ability to learn a more complex function. In the case of neural networks, the expressiveness depends on the choice of architecture (including depth, width, layer types etc.) [Raghu et al., 2017]. If two models have similar architectures, then they should be as expressive as each other. The study of model expressiveness explains why we need 'deep' in deep neural networks [Eldan and Shamir, 2016].*

4.1 Specification

The implementations for our active learning algorithms and the experiments are in **PyTorch 1.5.1** and **Python 3.7**. The source code for the project is hosted on GitHub for public access[1]. Experiments were run on CPU as well as GPU.

We use two popular datasets for our experiments. One is the **MNIST** hand written digits[LeCun et al., 2010], which has 10 classes from digit 0 to 9. Each data sample is a grayscale image of size 28×28, where the pixel values have the range 0 to 255. In our experiments, we standardise and threshold the grayscale images into binary images so that each pixel either has the value 0 or 1.

The other dataset is the **half-moon** data, which is a 2D synthetic dataset consisting of two interleaving half circles with optional Gaussian noise. The half-moon dataset provides us the ability to visualise the behaviour of our models on a 2D plane. At the same time, it is more challenging than linear classification tasks. To simulate the nosiness in the real world dataset, we set the Gaussian noise to be large enough so that our half-moon dataset is not linearly separable (see Figure 4.2). Moreover, we fixed the random seed for the Gaussian noise so that the same dataset was used across all half-moon experiments.

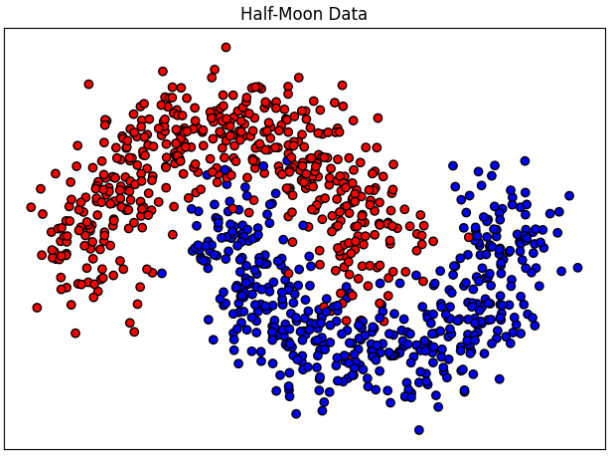


Figure 4.2: The half-moon dataset used in our experiments. There are 1000 data points in the set. The two colours denotes the two classes.

Since we are modelling the dataset using a generative model, we need to specify the genera-

[1] https://github.com/timxzz/AL-SSGM

tive distribution $p_\theta(\mathbf{x}|y,\mathbf{z})$ for the data. The MNIST data are in the form of binary image. Hence, we model them as a multivariate Bernoulli distribution

$$p_\theta(\mathbf{x}|y,\mathbf{z}) = Bernoulli(decoder_\theta(y,\mathbf{z})),$$

where $decoder_\theta(\cdot)$ is a neural network with weights θ. The half-moon data lie in a continuous $\mathbb{R}^2$ plane. Therefore, we model them as a diagonal Gaussian distribution

$$p_\theta(\mathbf{x}|y,\mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}_{y,\mathbf{z}}, \boldsymbol{\sigma}^2_{y,\mathbf{z}}\mathbf{I}), \quad (\boldsymbol{\mu}_{y,\mathbf{z}}, \boldsymbol{\sigma}_{y,\mathbf{z}}) = decoder_\theta(y,\mathbf{z}),$$

where $decoder_\theta(\cdot)$ is also a neural network with weights θ.

To show the possible improvements from our proposed active learning algorithms, we create a **baseline** model for each dataset. Since we are focusing on classification tasks, we use a simple classifier $p_\psi(y|\mathbf{x})$ that has the same architecture as our inference model $q_\phi(y|\mathbf{x})$ in M2 as the baseline model. The baseline model is trained with maximum likelihood learning under the usual active learning procedure with its predictive entropy as the acquisition function. Note that we make sure the baseline model $p_\psi(y|\mathbf{x})$ has the same architecture(including depth, width, layer types etc.) as $q_\phi(y|\mathbf{x})$, which means two models have similar expressiveness. In addition, random seeds for weights initialisation and training/test set split are fixed and reset for each experiment and each active learning loop. Therefore the difference in the results should mainly come from the difference in active learning algorithms.

4.2 Acquisition with $\mathcal{A}_{\mathrm{H}}(\cdot)$

We start with our active learning experiments using semi-supervised generative model M2 and the acquisition function $\mathcal{A}_{\mathrm{H}}(\mathbf{x})$. For each dataset, we compare the performance of two models (M2 and baseline). Additionally, we also compare data acquired through our $\mathcal{A}_{\mathrm{H}}(\mathbf{x})$ with data randomly acquired from $\mathcal{U}$. The acquisition function using randomise strategy is a good baseline as it resembles the situation that no strategy is used while selecting data for labelling. If the results of any designed acquisition are worse than the random acquisition, it means it is better to not use active learning paradigm at all.

In the case of half-moon dataset, our experiments were done with 30 acquisitions. In addition, each experiment has 10 runs with different random seeds. The results are showed in Figure 4.3. From the figure, we can see that acquisition with predictive entropy does outperform random acquisition for both M2 and the baseline model as expected. Moreover, when using $\mathcal{A}_{\mathrm{H}}(\mathbf{x})$, the semi-supervised generative model M2 (red line) has significantly better performance than the baseline (blue line) on the average of 10 runs. Another interesting observation is that the baseline model has a much larger variance across different runs.

In order to find out why the semi-supervised generative model (M2) has a larger variance than the baseline model, we have plotted the acquisitions and the resulting decision boundaries for both models in 4 random seeds using $\mathcal{A}_{\mathrm{H}}(\mathbf{x})$. The results are showed in Figure 4.4. From the figure, it seems that the decision boundaries for the baseline model vary across the 4 different seeds, but the decision boundaries for M2 stay relatively the same. A reasonable explanation is that our inference model $q_\phi(y|\mathbf{x})$ is a better approximation to the underlying conditional distribution $p(y|\mathbf{x})$. Therefore, its predictive uncertainty tightly concentrates on the decision boundary, whereas the baseline model $p_\psi(y|\mathbf{x})$ scatter its uncertainty across a large area (e.g. Figure 4.4(a)). The consequence is that the baseline model 'explores' a larger area, hence results in larger variance.

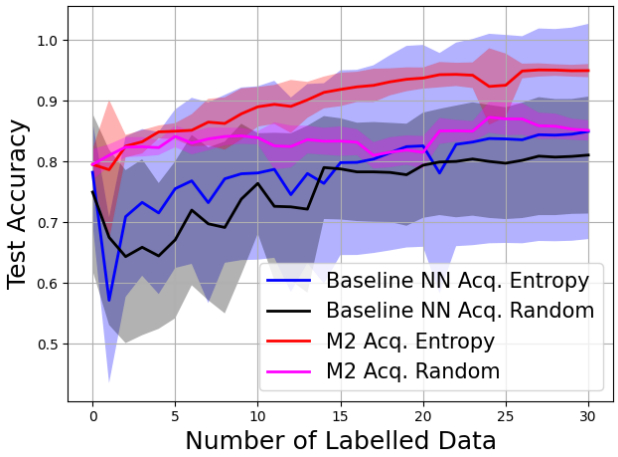
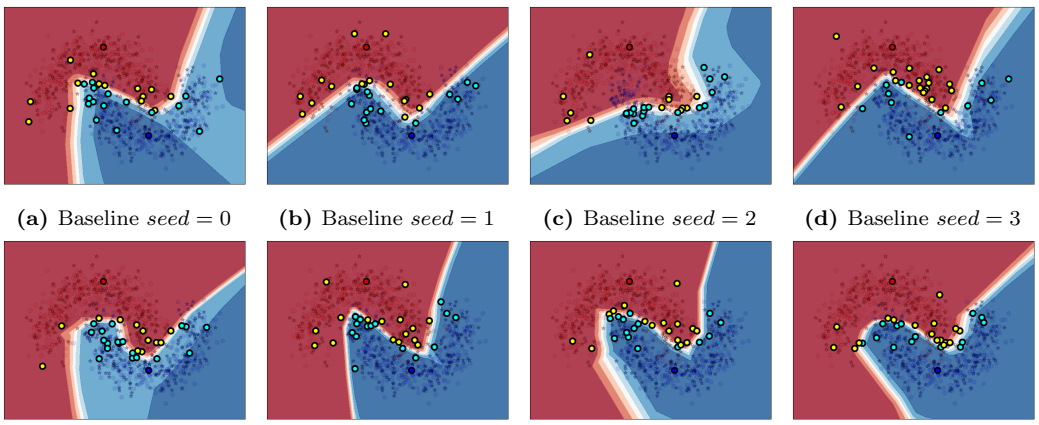


Figure 4.3: Experiments on the half-moon dataset with 30 acquisitions. We did 10 runs for each experiment with different random seeds. The solid lines are the means of the 10 runs, and the shaded areas represent the standard error.



(a) Baseline $seed = 0$ **(b)** Baseline $seed = 1$ **(c)** Baseline $seed = 2$ **(d)** Baseline $seed = 3$

(e) M2 $seed = 0$ **(f)** M2 $seed = 1$ **(g)** M2 $seed = 2$ **(h)** M2 $seed = 3$

Figure 4.4: The acquisitions and decision boundaries for the baseline model and the M2 model over 4 different random seeds. 30 acquisitions by predictive entropy $\mathcal{A}_{\mathrm{H}}(\mathbf{x})$ are plotted for each run. The acquired data points that are labelled as the *red* class are plotted in yellow circles. The acquired data points that are labelled as the *blue* class are plotted in cyan circles. The two dark red and dark blue circles are the initial labelled data, the light red and blue circles are the unlabelled data, the light red and blue stars are the test data.

After the positive results from half-moon, we need to test our methods on higher dimensional dataset such MNIST. Due to the computational complexity, our experiments for MNIST only focus on binary classification. The original MNIST dataset has 10 digits. By using 2 classes of digits, we can save the time on each pass of the dataset[2]. With two digits, each epoch takes 3 seconds for our M2 model on a GPU[3]. To train the model for MNIST binary classification to convergence requires 30 epoch on average. Since we need to train 200 models for a single run of active learning, it takes about 5 hours for each run. Hence, we only use 3 runs with different random seeds for one experiment.

The results for the MNIST binary classification experiments are showed in Figure 4.5. We choose digit 7 and 9 for the task because these two are hard to distinguish comparing to

[2]A single pass of dataset is known as an epoch.
[3]The GPU we used is a single Nvidia GTX1080.

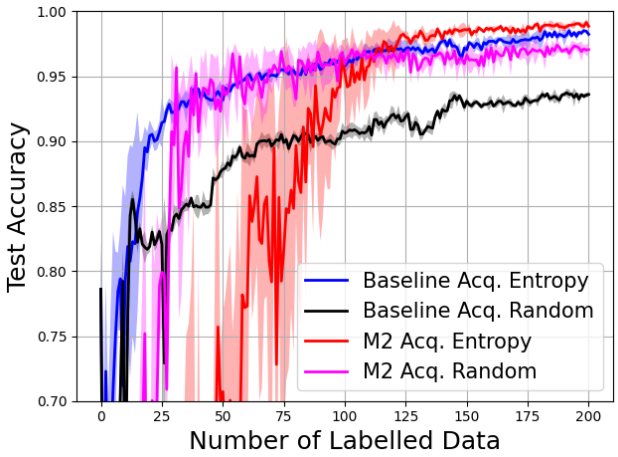


Figure 4.5: Experiments on MNIST binary classification on digit 7 and 9. The experiments were done over 200 acquisitions. We did 3 runs for each experiment with different random seeds. The solid lines are the means of the 3 runs, and the shaded areas represent the standard error.

other pairs. In the figure, we can see that the baseline model with $\mathcal{A}_{\mathrm{H}}(\mathbf{x})$ has a better performance when the number of labelled data is less then 110. When the number of acquired data grows above 110, the M2 model $\mathcal{A}_{\mathrm{H}}(\mathbf{x})$ outperform the rest. It might seem that the M2 model with entropy acquisition takes a long time to overtake the baseline, but in the active learning literature similar MNIST binary classification experiments normally use 500 to 1000 acquisitions [Gal et al., 2017]. Therefore, our method is still better comparing to the baselines.

4.3 Acquisition with $\mathcal{A}_\gamma(\cdot)$

In this section, we will test our active learning algorithm with the acquisition function from Eq.(3.13)

$$\mathcal{A}_\gamma(\mathbf{x}) = \gamma \cdot \langle -\mathcal{L}(\mathbf{x}, y) \rangle_{q_\phi(y|\mathbf{x})} + \mathrm{H}(q_\phi(y|\mathbf{x})).$$

Recall that the idea behind this acquisition function is to acquire data that are both informative and representative. In section 3.2, we argued that by changing γ from 0 to 1, the data we acquired will locate closer to a mode of the density $p_\theta(\mathbf{x})$ and further from the high entropy region (e.g. decision boundary of $p_\theta(y|\mathbf{x})$). Hence, we can first design an experiment to test the assumption.

To visualise how γ affect the acquisition, we use the half-moon dataset. In the experiment, the semi-supervised generative model is given 5 labelled data for each class and 590 unlabelled data $\mathcal{U}$. After the training step has converged, we use the acquisition function $\mathcal{A}_\gamma(\mathbf{x})$ to score every unlabelled data in $\mathcal{U}$ for 7 different $\gamma \in \{0.0, 0.167, 0.333, 0.5, 0.667, 0.833, 1.0\}$. Then, we plot the unlabelled data with the highest acquisition score for each γ. The results are showed in Figure 4.6. From the figure, we can verify that when $\gamma = 0$, the acquired data (yellow circle) is on the decision boundary; and when $\gamma = 1$, the acquired data is a the centre of $p_\theta(\mathbf{x})$'s modes. Other acquired data are indeed lying between the decision boundary and the two centres of $p_\theta(\mathbf{x})$'s modes. If we look at the right column of the figure, we can see how the contour lines of the acquisition score slowly transition as γ approaching 1. It seems like the acquisition function $\mathcal{A}_\gamma(\mathbf{x})$ is working as we expected.

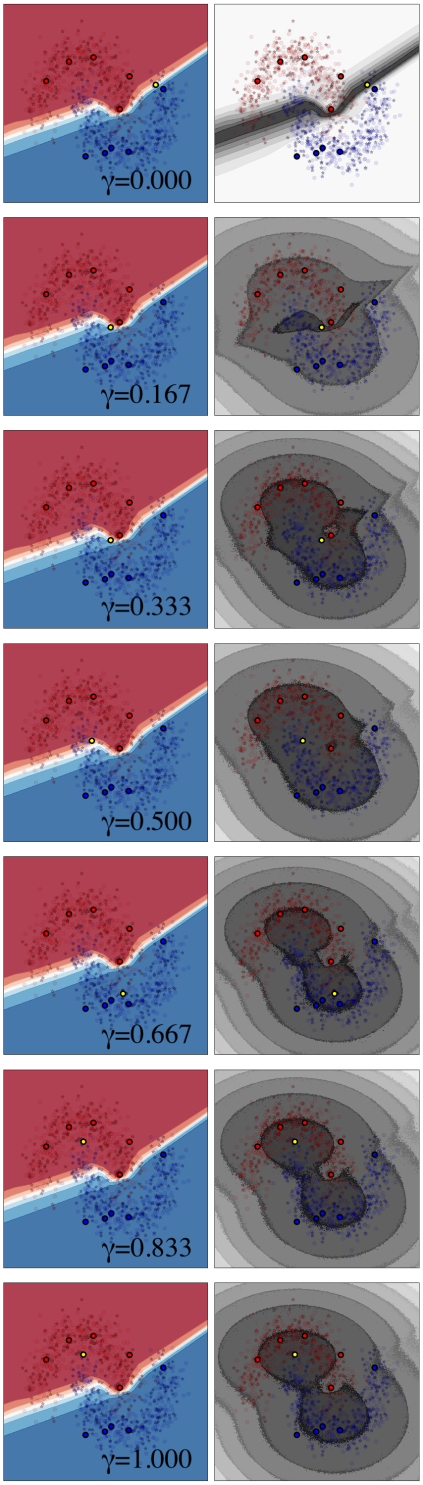


Figure 4.6: Single data point acquired (yellow circle) by our proposed acquisition function $\mathcal{A}_\gamma$ with $\gamma \in \{0.0, 0.167, 0.333, 0.5, 0.667, 0.833, 1.0\}$ from top to bottom. The contour lines in the left column represent the predictive density $q_\phi(y|\mathbf{x})$, and the contour lines in the right column represent the values of our acquisition function (darker colour corresponds to larger $\mathcal{A}_\gamma(\mathbf{x})$). The dark red and blue circles are the labelled data, the light red and blue circles are the unlabelled data, the light red and blue stars are the test data.

Next, we test the acquisition function $\mathcal{A}_\gamma(\mathbf{x})$ in our active learning setting. We use the half-moon dataset with 7 different $\gamma \in \{0.0, 0.167, 0.333, 0.5, 0.667, 0.833, 1.0\}$ and acquire 10 data for each γ. Figure 4.7 shows the results of the experiments. From the figure, it is obvious that $\gamma = 0$ has the best performance. Note that when $\gamma = 0$, the acquisition function $\mathcal{A}_0(\mathbf{x})$ is equivalent to $\mathcal{A}_\mathrm{H}(\mathbf{x})$. More importantly, the figure shows that when $\gamma > 0$, the acquisition function $\mathcal{A}_\gamma(\mathbf{x})$ does not help with the performance at all.

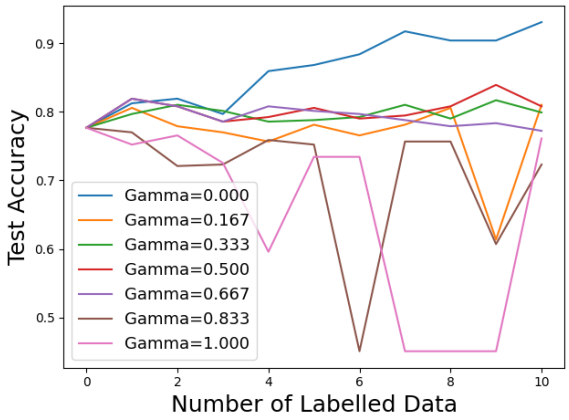


Figure 4.7: Experiments on half-moon dataset using M2 model and acquisition function $\mathcal{A}_\gamma(\mathbf{x})$. The plot shows the test accuracy across 10 acquisitions in a single run for 7 different γ values.

To analyse the reason behind Figure 4.7, we plot the acquisitions and see where they cluster. The results of 20 acquisitions a with 7 different γ are showed in Figure 4.8. In the figure, we can see that when $\gamma = 0$ (i.e. $\mathcal{A}_\mathrm{H}(\mathbf{x})$), the acquisitions scatter around the decision boundary, which meets our expectation for acquisition using predictive entropy. However, when $\gamma \geq 0.167$, all the acquisitions concentrate in a small region. If we compare the centres of these regions with the contour lines of $\mathcal{A}_\gamma(\mathbf{x})$ in Figure 4.6, we will see that they match with the high value area of $\mathcal{A}_\gamma(\mathbf{x})$. This implies that the value of $\mathcal{A}_\gamma(\mathbf{x})$ hardly changes with the acquisitions of new labelled data when $\gamma \geq 0.167$.

Further analysis into the acquisition function Eq.(3.13)

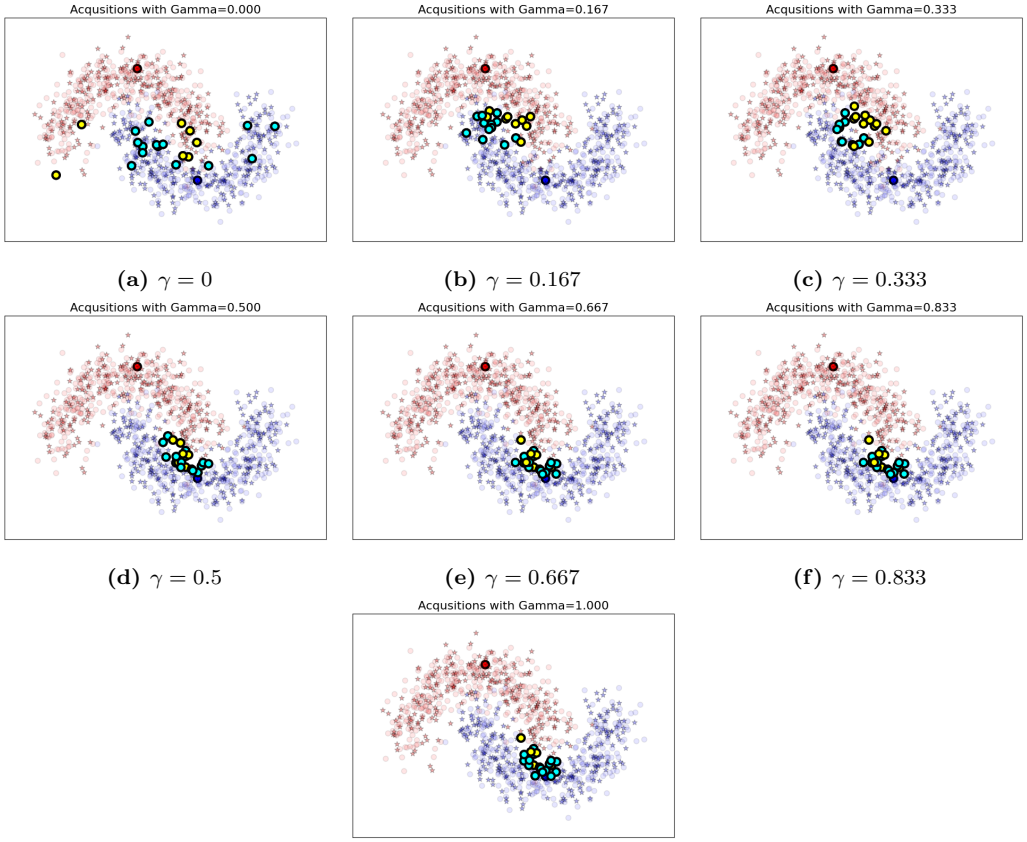
$$\mathcal{A}_\gamma(\mathbf{x}) = \gamma \cdot \langle -\mathcal{L}(\mathbf{x}, y) \rangle_{q_\phi(y|\mathbf{x})} + \mathrm{H}(q_\phi(y|\mathbf{x}))$$

and its components, we found out that the term $\langle -\mathcal{L}(\mathbf{x}, y) \rangle_{q_\phi(y|\mathbf{x})}$ in $\mathcal{A}_\gamma(\mathbf{x})$ dominates the entire function. Since the value of $\langle -\mathcal{L}(\mathbf{x}, y) \rangle_{q_\phi(y|\mathbf{x})}$ is much larger than $\mathrm{H}(q_\phi(y|\mathbf{x}))$, linear variation of γ and small changes in $\mathrm{H}(q_\phi(y|\mathbf{x}))$ does not have a huge impact on its final value. Therefore, when $\gamma \geq 0.167$, the acquisition function is density $p_\theta(\mathbf{x})$ dominated, which gives rise to such results.

To reduce the dominance of $\langle -\mathcal{L}(\mathbf{x}, y) \rangle_{q_\phi(y|\mathbf{x})}$, we also tried

$$\mathcal{A}'_\gamma(\mathbf{x}) = \gamma^k \cdot \langle -\mathcal{L}(\mathbf{x}, y) \rangle_{q_\phi(y|\mathbf{x})} + \mathrm{H}(q_\phi(y|\mathbf{x})), \tag{4.1}$$

where k is a hyperparameter with value $k \geq 1$. If $k > 1$, the linear change of γ from 0 to 1 will result in much slower increase of $\langle -\mathcal{L}(\mathbf{x}, y) \rangle_{q_\phi(y|\mathbf{x})}$ when γ is close to 0. However, after trying various $k \in \{1, 2, 3, 4\}$, the results are still not satisfying. Figure 4.9 shows the results of experiments when using the above $\mathcal{A}'_\gamma(\mathbf{x})$ with $k = 4$. It seems that the use of γ^k does help, but still when $\gamma \geq 0.167$, the performance is worse then $\gamma = 0$. The negative results in the half-moon experiments of using acquisition function $\mathcal{A}_\gamma(\mathbf{x})$ did not encourage us to move on to the similar experiments on MNIST dataset.



(a) $\gamma = 0$ **(b)** $\gamma = 0.167$ **(c)** $\gamma = 0.333$

(d) $\gamma = 0.5$ **(e)** $\gamma = 0.667$ **(f)** $\gamma = 0.833$

(g) $\gamma = 1$

Figure 4.8: 20 acquisitions for 7 different γ in active learning experiments on the half-moon dataset using semi-supervised generative model (M2). The acquired data points that are labelled as the *red* class are plotted in yellow circles. The acquired data points that are labelled as the *blue* class are plotted in cyan circles. The two dark red and dark blue circles are the initial labelled data, the light red and blue circles are the unlabelled data, the light red and blue stars are the test data.

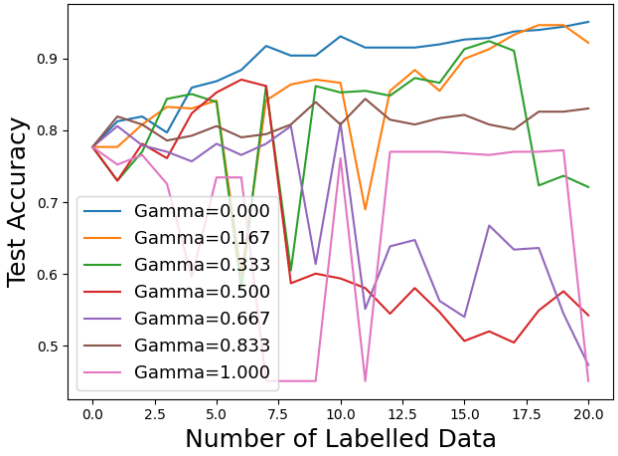


Figure 4.9: Experiments on half-moon dataset using M2 model and acquisition function $\mathcal{A}'_\gamma(\mathbf{x})$ from Eq.(4.1), where $k = 4$. The plot shows the test accuracy across 20 acquisitions in a single run for 7 different γ values.

4.4 Acquisition with $\mathcal{A}_1(\cdot)$ then $\mathcal{A}_0(\cdot)$

From previous section we found out a drawback in the acquisition function $\mathcal{A}_\gamma(\mathbf{x})$, which is the difficulty in balancing the linear combination between predictive entropy and the density $p_\theta(\mathbf{x})$. In this section, we look at whether using the predictive entropy $\mathcal{A}_0(\mathbf{x})$ and the density $\mathcal{A}_1(\mathbf{x})$ separately will help with the active learning performance. Recall from the section 3.3.3, our idea is to first use the acquisition function $\mathcal{A}_1(\mathbf{x})$ to filter out non-representative data (i.e. likely outliers), then we use the acquisition function $\mathcal{A}_0(\mathbf{x})$ to choose the most informative data within the filtered set.

We have done experiments with such acquisition method using 10 different proportions for filtering. During each acquisition step, we first keep the top $X\%$ (such that $X \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$) of the unlabelled data that have the highest values w.r.t. $\mathcal{A}_1(\mathbf{x})$, then select the one with maximum $\mathcal{A}_0(\mathbf{x})$. Results are showed in Figure 4.10. From the figure, we can see that by keeping only a small portion of the highest density data, we are not able to improve the active learning performance. As more and more data are kept, the performance increases. Note that when $X = 100$, we are keeping the entire unlabelled set for $\mathcal{A}_0(\mathbf{x})$, which is the same as the active learning setting using only $\mathcal{A}_\mathrm{H}(\mathbf{x})$ in previous sections. Another finding is that filtering out around 20% of the lowest density data using $\mathcal{A}_1(\mathbf{x})$ does not have a negative impact for the performance. However, filtering using density did not show improvements in our experiments either, we suspect it is because the half-moon dataset is too simple and the impact of outliers is not significant. It is worth to explore whether using $\mathcal{A}_1(\mathbf{x})$ to filter out a small portion of the non-representative data will improve the active learning performance in a more complex dataset.

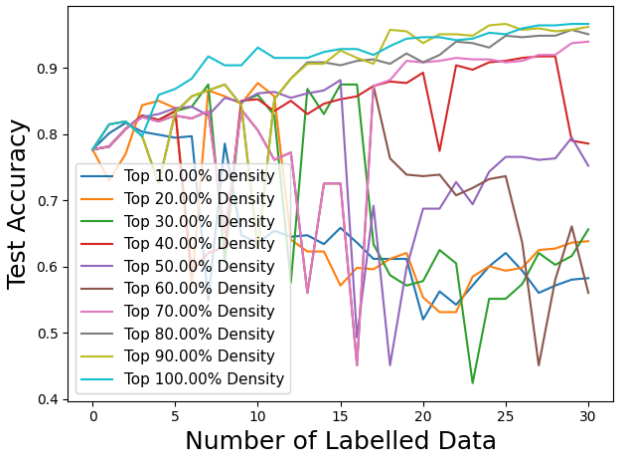


Figure 4.10: Experiments on half-moon dataset using M2 model with 30 acquisitions. The data acquired are first filtered with their density value by $\mathcal{A}_1(\mathbf{x})$, then among the top $X\%$ density data we choose a single point with highest entropy $\mathcal{A}_0(\mathbf{x})$. $X \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$.

Chapter 5

Discussion and Conclusion

In previous chapters, we first provide the motivations for using semi-supervised generative models in active learning. Then, we go through the background knowledge for semi-supervised generative models and introduce our algorithm that integrates the model with active learning as well as our proposed acquisition functions. The results of various experiments on our methods are documented in the last chapter. In this chapter, we will discuss some interesting problems discovered during our experiments and summarise the project in the end.

5.1 Problems with Density

Previously, we argued that data with density value $p(\mathbf{x})$ are representative. Thus, we incorporate the density approximation $p_\theta(\mathbf{x})$ directly into our acquisition function $\mathcal{A}_\gamma(\mathbf{x})$ Eq.(3.13). However, from our experiments, we notice that since the approximated density $p_\theta(\mathbf{x})$ does not depend on labelled data, it remains largely unchanged during the active learning process. It means that if we want prioritise the acquisition of unlabelled data with high density values, we will acquire all the data around the modes of $p_\theta(\mathbf{x})$ first. Such policy results in acquisitions clustering in a small region (see Figure 4.8), which is not desirable and will negatively impact the active learning performance. To solve this problem with generative models in active learning, some proposed to weight the density $p_\theta(\mathbf{x})$ with classification disagreement [McCallum and Nigam, 1998] for acquisition function. We did try it in our active learning setting by weighting our density estimate $\mathcal{A}_1(\mathbf{x})$ with the predictive entropy $\mathcal{A}_0(\mathbf{x})$ (i.e. use $\mathcal{A}_1(\mathbf{x}) \times \mathcal{A}_0(\mathbf{x})$ as the acquisition function), but we did not get the desired result. Therefore, how to use the density estimate $p_\theta(\mathbf{x})$ from semi-supervised generative model properly in active learning is left as an open problem for future investigation.

Another challenge is related to semi-supervised generative models. When we mention the our density model $p_\theta(\mathbf{x})$, we seem to be assuming our tractable variation lower bound from Eq.(2.43), i.e.:

$$\log p_\theta(\mathbf{x}) \geq \langle -\mathcal{L}(\mathbf{x}, y) \rangle_{q_\phi(y|\mathbf{x})} + \mathrm{H}(q_\phi(y|\mathbf{x}))$$

gives us a close estimate to the actual density $p(\mathbf{x})$. Sadly, it is very unlikely to be the case. The latent variable model we choose for the underlying generating process as well as the variational family we use to approximate the posterior are almost certainly not how the real data are generated. We can use one of our experiments as an example to illustrate this. If

we look at the bottom right contour graph in Figure 4.6 where $\gamma = 1$, we can see that the contour lines of the lower bound (i.e. $\mathcal{A}_1(\mathbf{x})$) have a shape of a mixture of two Gaussian distributions. Such approximation is reasonable, but does not capture the outer tails of the two half-moons as showed in the graph. Many researchers have observed similar modelling problems in semi-supervised generative models [Cozman et al., 2003, Zhu, 2005]. They also emphasise that without an accurate model, unlabelled data might even hurt the accuracy. Hence, we need to be careful when constructing semi-supervised generative model for active learning, and do not take the lower bound value as an exact reflection of the density during acquisition.

5.2 Sampling Bias

During the project, we also encountered an interesting problem related to active learning, which is known as *sampling bias* [Dasgupta and Hsu, 2008, Beygelzimer et al., 2009, Dasgupta, 2011]. We discovered it when we tried to prove model correctness in section 3.1. In the section, we prove the semi-supervised generative model (M2) is able to reach the optimal θ^* with its training objective when the number of training data approaches infinity, but under the assumption that the training data are uniformly sampled from the real distribution (i.e. $(\mathbf{x}, y) \sim p_{\theta^*}(\mathbf{x}, y)$). This is usually the case in a normal machine learning setting because we assume the training data are uniformly sampled from their distribution if there is no evidence that they are altered. However, it is not the case in active learning. The training data used by the models in active learning are carefully selected through acquisition function. It means that the training data might not come directly from the distribution $p_{\theta^*}(\mathbf{x}, y)$, thus our proofs might not hold anymore. The consequence is that if we have infinite budget for labelling data, the model produced by active learning algorithm with sampling bias might not converge to an optimal θ^*. Many papers have looked into this problem, some suggested to use importance weights to correct sampling bias [Sugiyama, 2006, Bach, 2007, Beygelzimer et al., 2009].

Although, many existing active learning algorithms forget to put sampling bias into consideration, it is still a desirable property to have. The question that whether our proposed active learning algorithms have sampling bias is left for future investigation.

5.3 Summary

In this project, we try to exploit the features from semi-supervised generative models in the active learning setting, which is a novel idea. Our motivation is that semi-supervised generative models are able to utilise unlabelled data and provide explainable estimations for the density distribution ($p_\theta(\mathbf{x}) \approx p(\mathbf{x})$) and the conditional distribution ($q_\phi(y|\mathbf{x}) \approx p(y|\mathbf{x})$). These properties can be used to improve the performance of active learning. We proposed and tested three kinds of acquisition functions that make use of $p_\theta(\mathbf{x})$ and $q_\phi(y|\mathbf{x})$, which are modelled by the semi-supervised generative model M2.

Our results show that in the half-moon and MNIST dataset, using semi-supervised generative model with predictive entropy as the acquisition function (i.e. $\mathcal{A}_\mathrm{H}(\mathbf{x})$) will improve the performance of active learning. In addition, even though our proposed methods for using density information in acquisition function (i.e. $\mathcal{A}_\gamma(\mathbf{x})$) failed to meet our expectation, we investigated the problem thoroughly and clearly identified the cause.

Overall, our work suggest and demonstrate how semi-supervised generative model can be

used in active learning to improve its performance. In particular, our novel use of predictive entropy with semi-supervised generative model is showed to provide improvements for active learning. Moreover, through our experiments and analysis, we document the challenges we faced in the project, which point to valuable directions for future work.

References

R. P. Adams and Z. Ghahramani. Archipelago: nonparametric bayesian semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, volume 382 of *ACM International Conference Proceeding Series*, pages 1–8. ACM, 2009.

F. R. Bach. Active learning for misspecified generalized linear models. In *Advances in Neural Information Processing Systems*, pages 65–72, 2007.

A. Beygelzimer, S. Dasgupta, and J. Langford. Importance weighted active learning. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, volume 382 of *ACM International Conference Proceeding Series*, pages 49–56. ACM, 2009.

D. M. Blei, A. Kucukelbir, and J. D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

O. Chapelle, B. Schlkopf, and A. Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010. ISBN 0262514125.

D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. In *Advances in Neural Information Processing Systems*, pages 705–712, 1995.

F. G. Cozman, I. Cohen, and M. C. Cirelo. Semi-supervised learning of mixture models. In *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, pages 99–106. AAAI Press, 2003.

S. Dasgupta. Two faces of active learning. *Theoretical computer science*, 412(19):1767–1781, 2011.

S. Dasgupta and D. J. Hsu. Hierarchical sampling for active learning. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, volume 307 of *ACM International Conference Proceeding Series*, pages 208–215. ACM, 2008.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

P. Donmez, J. G. Carbonell, and P. N. Bennett. Dual strategy active learning. In *European Conference on Machine Learning*, pages 116–127. Springer, 2007.

R. Eldan and O. Shamir. The power of depth for feedforward neural networks. In *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, volume 49 of *JMLR Workshop and Conference Proceedings*, pages 907–940. JMLR.org, 2016.

V. Fedorov, W. Studden, E. Klimko, and A. P. (Londyn). *Theory of Optimal Experiments*. Cellular Neurobiology. Academic Press, 1972. ISBN 9780122507502.

Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine learning*, 28(2-3):133–168, 1997.

Y. Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

Y. Gal, R. Islam, and Z. Ghahramani. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1183–1192. PMLR, 2017.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

Y. Guo and D. Schuurmans. Discriminative batch mode active learning. In *Advances in Neural Information Processing Systems*, pages 593–600, 2008.

S.-J. Huang, R. Jin, and Z.-H. Zhou. Active learning by querying informative and representative examples. In *Advances in Neural Information Processing Systems*, pages 892–900, 2010.

J. L. W. V. Jensen et al. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta mathematica*, 30:175–193, 1906.

A. Kendall, V. Badrinarayanan, and R. Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. In *British Machine Vision Conference 2017, BMVC 2017*, 2017.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589, 2014.

S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

Y. LeCun, C. Cortes, and C. Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *SIGIR'94*, pages 3–12. Springer, 1994.

Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

D. J. MacKay. Information-based objective functions for active data selection. *Neural computation*, 4(4):590–604, 1992.

A. McCallum and K. Nigam. Employing EM and pool-based active learning for text classification. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconsin, USA, July 24-27, 1998*, pages 350–358. Morgan Kaufmann, 1998.

I. Muslea, S. Minton, and C. A. Knoblock. Active + semi-supervised learning = robust multi-view learning. In *Machine Learning, Proceedings of the Nineteenth International Conference (ICML 2002), University of New South Wales, Sydney, Australia, July 8-12, 2002*, pages 435–442. Morgan Kaufmann, 2002.

H. T. Nguyen and A. W. M. Smeulders. Active learning using pre-clustering. In *Machine Learning, Proceedings of the Twenty-first International Conference (ICML 2004), Banff, Alberta, Canada, July 4-8, 2004*, volume 69 of *ACM International Conference Proceeding Series*. ACM, 2004.

M. Raghu, B. Poole, J. M. Kleinberg, S. Ganguli, and J. Sohl-Dickstein. On the expressive power of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 2847–2854. PMLR, 2017.

E. Riloff, J. Wiebe, and T. Wilson. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 25–32. ACL, 2003.

C. Rosenberg, M. Hebert, and H. Schneiderman. Semi-supervised self-training of object detection models. In *Proceedings of the Seventh IEEE Workshops on Application of Computer Vision (WACV/MOTION'05)-Volume 1-Volume 01*, pages 29–36, 2005.

T. Scheffer, C. Decomain, and S. Wrobel. Active hidden markov models for information extraction. In *International Symposium on Intelligent Data Analysis*, pages 309–318. Springer, 2001.

C. Schröder and A. Niekler. A survey of active learning for text classification using deep neural networks. *arXiv:2008.07267*, 2020.

B. Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

C. E. Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

M. Sugiyama. Active learning for misspecified models. In *Advances in Neural Information Processing Systems*, pages 1305–1312, 2006.

S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2(Nov):45–66, 2001.

J. E. Van Engelen and H. H. Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2020.

C. J. Wu. On the convergence properties of the em algorithm. *The Annals of statistics*, pages 95–103, 1983.

Z. Xu, K. Yu, V. Tresp, X. Xu, and J. Wang. Representative sampling for text classification using support vector machines. In *European Conference on Information Retrieval*, pages 393–407. Springer, 2003.

D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd Annual Meeting of the Association for Computational Linguistics, 26-30 June 1995, MIT, Cambridge, Massachusetts, USA, Proceedings*, pages 189–196. Morgan Kaufmann Publishers / ACL, 1995.

X. Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2005.

X. Zhu, J. Lafferty, and Z. Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003 workshop on the continuum from labeled to unlabeled data in machine learning and data mining*, volume 3, 2003.