Author:
Chen, Yu

Title:
Efficient Continual Learning

Approaches and Measures

Efficient Continual Learning

Approaches and Measures

By

YU CHEN

Department of Computer Science
UNIVERSITY OF BRISTOL

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of DOCTOR OF PHILOSOPHY in the Faculty of Engineering.

NOVEMBER 2021

Word count: thirty seven thousand and two hundred

Abstract

In the real world, we often encounter situations where data distributions are changing over time, and we would like to timely update our models by the new data, with bounded growth in system size and computational cost. *Continual learning* is a research topic for dealing with such scenarios. The main challenge of continual learning is *catastrophic forgetting*: when training a model by sequential tasks, the model tends to forget previously learned tasks after learning new ones. Here different tasks mean different training and testing sets.

In Bayesian continual learning, we would prefer the posteriors of a new task being as close as possible to the previous ones. We propose Gaussian Natural Gradients for Bayesian continual learning, which uses natural gradients instead of conventional gradients as natural gradients prefer the smallest change in terms of distributions rather than parameters. We also propose Stein Gradient-based Episodic Memories that can construct compact episodic memories using the information in posteriors in Bayesian continual learning. In addition, we propose Discriminative Representation Loss for general continual learning, which decreases the diversity of gradients between new and old tasks through optimizing representations instead of re-projecting the gradients. It effectively improves performance with low computational cost compared with related work.

Moreover, we propose β^3-Item Response Theory model for evaluating classifiers in continual learning. The *ability* inferred by β^3-IRT is weighted by *difficulty* of individual samples, which can provide more sensible evaluations than typically used average accuracy. We furthermore propose Continual Density Ratio Estimation for evaluating generative models in continual learning without storing any data from previous tasks. To the best of our knowledge, this is the first measure for generative models that satisfies the restriction of continual learning.

In summary, we propose several efficient approaches and measures for continual learning in this thesis.

Dedication and acknowledgements

I dedicate this thesis to my mother. Although she cannot recognize me anymore or be aware of what I have achieved today due to Alzheimer's disease, I believe this still will be a comfort to her and she will be proud of me. I hope I could help people like her by my research in the future. I also dedicate this thesis to my grandparents who were the closest family to me besides my mother and took care of me when I was little. I will miss you always, grandpa and grandma.

I could not accomplish this work without the help and support from a number of people. I would like to first acknowledge my supervisor Prof. Peter Flach. I made the success application of the university's scholarship because of his support and hence got the chance to become a Ph.D. student with him. He provided me great guidance when I was stressed or disappointed by my research progress and also gave me the freedom to do what I was interested. I have learned a lot from him for doing research and being an academic. I would also like to express appreciation to my second supervisor Dr. Tom Diethe. Tom helped me a lot with my research as well. He is the one who gave me the initial idea for my first publication in IJCAI 2016. He also inspired my interests in the research topics that have finally become projects composing this thesis. I enjoyed our discussions as he can often broaden my readings and always has a sense of humour. I was lucky to have Peter and Tom as my supervisors.

I would like to acknowledge my collaborators, Telmo Silva Filho, Ricardo B.C. Prudêncio, and Song Liu, as well. They helped me a lot with finalizing ideas, conducting experiments, writing and reviewing papers, which form an indispensable part of this work. I want to thank Song Liu in particular as we had a lot of discussions and he has always been patient to me when I made mistakes or had misunderstandings. My colleagues, Miquel Perello Nieto, Kacper Sokol, Hao Song, Benjamin Arana Sanchez, Bohong Bo, Mengwei Xu, Zijian Shi, were always warmhearted to me when I needed some help during my time in Bristol. It is a pleasure to meet them and I would like to thank them sincerely.

I would also like to thank my family and friends who have supported me when I made the decision to return to the university and study in the UK, especially my brother and sister in law, my uncle and aunt, my very old friends Mei Xiang, Yunhan Luo, Qing Ge, Li Zhang, Tao Wu, Xingyu Zhang. They are the reason that I can concentrate on my research without worrying too much about my mother and without feeling lonely while I'm away from home.

I'm glad I have had this journey in my life, thank God.

Author's declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: .... DATE: ....

TABLE OF CONTENTS

List of Tables

List of Figures

Chapter 1

Introduction

In this thesis, we discuss continual learning which is a research area tackling the essential problem *'catastrophic forgetting'* of sequential training in machine learning. We propose several approaches for efficiently alleviating forgetting as well as novel measures for evaluating classifiers and generative models in continual learning. As a start of this thesis, we first introduce the problem setting of continual learning in Sec. 1.1. We then summarize our contributions and the organization of the thesis in Sec. 1.2. In Sec. 1.3, we provide a brief introduction of research work that I have done but not included in this thesis because they are not related to continual learning.

1.1 Problem Setting

Machine learning has gained significant attention in recent years because of the success of Deep Neural Networks (DNNs) in many applications, such as computer vision, speech recognition, robotics, machine translation, etc. Despite the notable research progress of DNNs, there are still many open problems and continual learning is one among them.

In a conventional scenario of machine learning, a model is usually trained on a specific training set for solving a specific problem (such as classification) where the model is assumed to be static after training. We call such learning scenario as static learning in this thesis, and call a combination of the specific training set and the corresponding problem as a *task*. Since a few decades, neural networks have been found performing poorly when learning different tasks sequentially due to a problem called *catastrophic forgetting* (McCloskey & Cohen, 1989; Ans & Rousset, 1997, 2000), which refers to the phenomenon of a model forgetting previously learned tasks while learning a new one. In recent years, increasing interest has emerged in continual learning which aims to tackle this problem, and a line of work has been proposed for alleviating forgetting in neural networks (Kirkpatrick et al., 2017; Zenke et al., 2017; Schwarz et al., 2018;

Nguyen et al., 2018; Lopez-Paz & Ranzato, 2017). The reason is that it has become a critical problem in more practical scenarios. Nowadays the ubiquitous usage of the internet has resulted in the emergence of numerous online applications which need to deal with amounts of data that are generated sequentially. Consequently, models trained for these applications are required to learn from online data without degradation of performance. Meanwhile, DNNs have shown prominent advantages when training on large scale datasets but the computational expense is costly in terms of time and memory. When new data have been received, retraining such a model on all seen data would be unrealistic. However, conventional online learning algorithms often make strong assumptions about the data, for example, that the data samples are from the same or similar distributions (Shalev-Shwartz et al., 2011). When the data distribution changes significantly, for instance, new categories of data have joined at different time, such algorithms will not be able to resist catastrophic forgetting as well. For example, a recommender system trained for a large retail platform (such as Amazon) often encounters scenarios that new goods and customers are joined. It would be costly to retrain the recommender system on old and new data jointly. A solution that is capable of online training and resistant to forgetting is desirable in such a scenario. Eliminating forgetting may be impossible as we will see in a theoretical analysis in Sec. 3.5. Nonetheless, gradual forgetting in a long period is still attractive for many applications, such as forgetting on obsolete data.

Here we formalize the problem as follows. Suppose a model $f(\cdot;\boldsymbol{\theta})$ is a function mapping an input space to an output space and $\boldsymbol{\theta}$ represents the model parameters. It receives training datasets of a series of tasks sequentially. Let $\mathscr{D}_t = \{X_t, Y_t\}$ denote the training data of the t-th task that can be the input of the model $f(\cdot;\boldsymbol{\theta})$. The loss function of the t-th task can be expressed as $\mathscr{L}(\mathscr{D}_\tau;\boldsymbol{\theta}_t) = \mathscr{L}(f(X_\tau;\boldsymbol{\theta}_t), Y_\tau)$, where $\mathscr{D}_\tau$ can be any data from all learned tasks, it computes a predefined loss by the output of the model function that is trained at the t-th task. The ultimate goal of sequential training is to learn the optimal model parameters that satisfies:

$$\boldsymbol{\theta}_t^* = \operatorname*{arg\,min}_{\boldsymbol{\theta}_t} \mathscr{L}(\mathscr{D}_\tau;\boldsymbol{\theta}_t), \quad \forall \tau \leq t \tag{1.1}$$

Here $\boldsymbol{\theta}_t$ is the parameters of the model learned at the t-th task. It is an ideal case that $\boldsymbol{\theta}_t$ is optimal to all learned tasks. In practice, the performance of the model trained on all tasks under static learning can be viewed as the upper bound of its performance in continual learning. In static learning all training sets are presented to the model at once without the cause of forgetting:

$$\tilde{\boldsymbol{\theta}}_t^* = \mathscr{L}(\mathscr{D}_{1:t};\boldsymbol{\theta}_t), \quad \mathscr{D}_{1:t} = \{\mathscr{D}_1, \mathscr{D}_2, \ldots, \mathscr{D}_t\} \tag{1.2}$$

In continual learning, the model has no or very limited access to data from previously learned tasks. Instead, it may have the access to an episodic memory (a small memory where a few samples of previous tasks have been stored) and/or the model parameters of the previous task. In general, the optimization objective of continual learning methods can be written as:

$$\tilde{\boldsymbol{\theta}}_t = \operatorname*{arg\,min}_{\boldsymbol{\theta}_t} \mathscr{L}(\{\mathscr{D}_t, \mathscr{M}_t\};\{\boldsymbol{\theta}_t, \tilde{\boldsymbol{\theta}}_{t-1}\}) \tag{1.3}$$

where $\mathscr{M}_t$ represents the episodic memory at the t-th task (which could be empty for methods not storing previous samples).

Let $s(\cdot)$ represent a measurement of the performance, the test set of task t is $\mathscr{D}'_t = \{X'_t, Y'_t\}$, similarly, $\mathscr{D}'_{1:t} = \{\mathscr{D}'_1, \mathscr{D}'_2, \dots, \mathscr{D}'_t\}$, then we can evaluate the model performance at the t-th task by $s(\mathscr{D}'_t, \boldsymbol{\theta}_t) = s(f(X'_t; \boldsymbol{\theta}_t), Y'_t)$. Approaches to continual learning try to fill the gap between $s(\mathscr{D}'_{1:t}; \tilde{\boldsymbol{\theta}}_t))$ and $s(\mathscr{D}'_{1:t}; \tilde{\boldsymbol{\theta}}^*_t)$ by making a better usage of $\mathscr{M}_t$ and $\tilde{\boldsymbol{\theta}}_{t-1}$. The difficulty stems from the limited resource that is available for preserving information of previous tasks. We want to do better than to store all training data and retrain the model every time when receiving a new task or train separate models for each task.

Table 1.1: Comparison between related learning paradigms

	Continual learning	Multi-task learning	Transfer learning	Meta learning
Objective	$\mathscr{L}(\{\mathscr{D}_t, \mathscr{M}_t\}; \{\boldsymbol{\theta}_t, \tilde{\boldsymbol{\theta}}_{t-1}\})$	$\mathscr{L}(\mathscr{D}_{1:t}; \boldsymbol{\theta}_t)$	$\mathscr{L}(\mathscr{D}_t; \{\boldsymbol{\theta}_t, \tilde{\boldsymbol{\theta}}_{t-1}\})$	$\mathscr{L}(\mathscr{D}_{1:t}; \boldsymbol{\theta}_t)$
Measure	$s(\mathscr{D}'_{1:t}; \boldsymbol{\theta}_t)$	$s(\mathscr{D}'_{1:t}; \boldsymbol{\theta}_t)$	$s(\mathscr{D}'_t; \boldsymbol{\theta}_t)$	$s(\mathscr{D}_{t+1:t+k}; \boldsymbol{\theta}_t)$

Eq. (1.2) describes a general form of the objective in Multi-task learning, which learns multiple tasks in parallel to obtain better generalization through information shared across those tasks. Transfer learning deals with tasks in a sequential manner like continual learning. The difference is that it focuses on transferring knowledge to a target task (which is the latest task in the learning sequence). Meta learning aims to find a general solution space for a task distribution and then a model can be adapted to a target task by very cheap cost (i.e. a few steps of gradient update). During training phase, meta learning has access to a set of tasks that are sampled from the task distribution. In testing time, the model is tested on another set of tasks from the same distribution. Except continual learning, these learning paradigms do not care about preserving performance on previously learned tasks, i.e. they do not tend to solve the problem of forgetting.

The practical aim of continual learning is to find an optimal trade-off between the resource cost and the performance over all tasks. Moreover, there is another trade-off in terms of the performance on old and new tasks, which we call the trade-off between stability and plasticity. Stability means how stable the performance can be on old tasks; the plasticity indicates how much the model can be changed for learning a new task, which decides how good the performance can be on a new task. Under the same restrictions of resources, a model might lose plasticity for accommodating new tasks when it has been consolidated on old ones because its growth is restricted. This would limit forgetting but would not be satisfactory to new tasks. To this end, our research objective is to provide efficient approaches that can generally perform well with relatively low memory cost and limited computational overhead. In addition, we also aim to provide novel measurements that are effective for model selection in continual learning regarding incapable scenarios of existing measurements due to the trade-off residing in the performance and the resource restriction. For instance, the average accuracy for evaluating classifiers may not be able to tell a difference between two models because one model may have better performance on forgetting and the other is better in terms of intransigence.

1.2 Summary of Contributions

In this thesis, we propose novel approaches and measures for continual learning which focus on efficiently improving model performance and model evaluation under the restrictive settings of continual learning. The main contributions of the work can be summarised as follows.

- We propose efficient methods for Bayesian continual learning (Chapter 4) which are based on the framework of Variational Continual Learning (VCL) and have shown performance improvements with relatively low computational overhead. One is a regularization-based method called Gaussian Natural Gradient (GNG) that utilizes the natural gradients in Bayesian neural networks with Gaussian mean-field priors. GNG can preserve important parameters of previous tasks in a more efficient way compared with the conventional gradient updates. The other is a replay-based method called Stein Gradient-based Episodic Memories (SGEM) that utilizes the Stein Variational Gradient Descent (SVGD) for composing more representative episodic memories. It updates the samples in the memory buffer according to the joint distribution of data and model parameters, which implicitly includes the information of parameter posteriors in the episodic memory.

  This work (Chen et al., 2018) was published in Continual Learning Workshop of 32nd Conference on Neural Information Processing Systems (NeurIPS 2018) as *'Facilitating Bayesian continual learning by natural gradients and Stein gradients'*, coauthored by Tom Diethe, and Neil Lawrence. The main ideas were initiated by me after reading papers about natural gradient (Szegedy et al., 2016) and SVGD (Liu & Wang, 2016). I wrote most of the paper and carried out the experiments including implementation.

- We reveal the relation between diversity of gradients and discriminativeness of representations which connects Deep Metric Learning (DML) to gradient-based methods in continual learning. Following these findings, we propose an efficient replay-based method called Discriminative Representation Loss (DRL) (Chapter 5) for continual learning that have shown better performance with much lower computational cost than gradient-based methods in the most restrictive setting (online training and single-headed model). In addition, we provide comprehensive experimental results of an ablation study on DRL which have demonstrated the effectiveness of DRL from different angles.

  This work (Chen et al., 2020) was published on arXiv preprint as *'Discriminative representation loss (DRL): Connecting Deep Metric Learning to Continual Learning'*, coauthored by Tom Diethe, and Peter Flach. The main idea came from demo experiments I have conducted to analyze the results of a related work (Aljundi et al., 2019b). I wrote most of the paper and implemented the experiments. The proposed method is also a winning solution of the CLVision Challenge [1] of the Continual Learning workshop in CVPR 2020 and a part of the publication *'CVPR 2020 Continual Learning in Computer Vision Competition: Approaches, Results, Current Challenges and Future Directions'* (LOM, 2022) on Artificial Intelligence.

- We propose the β^3-IRT (Chapter 6) for evaluating classifiers in both static and continual learning, which is a new Item Response Theory (IRT) model with richer Item Characteristic Curve (ICC) shapes and hence is more flexible for fitting different datasets. We demonstrate that the β^3-IRT is robust to noisy data and can be applied to detect noisy test samples. The ability of classifiers inferred by the β^3-IRT provides an instance-wise basis metric regarding the difficulty of each data instance, which is capable of evaluating models in continual learning in a more informative way than conventional average accuracy.

  This work (Chen et al., 2019) was published in the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS 2019) as *'β^3-IRT: A new item response model and its applications'*, coauthored by Telmo Silva Filho, Ricardo B. Prudêncio, Tom Diethe, and Peter Flach. The initial idea was from a discussion with Peter and I developed it towards applications in machine learning. I wrote the sections regarding technical details of the method and experiments with classifiers of the paper, as well as implemented those experiments.

- We propose a new framework Continual Density Ratio Estimation (CDRE) for estimating density ratios in an online setting, which does not require storing historical samples in the data stream and can be more accurate than standard DRE when the two distributions are less similar (Chapter 7). CDRE is capable of evaluating generative models without storing samples from the original distribution which is feasible under the restrictions of continual learning, and to the best of our knowledge, there is no prior work that can evaluate generative models in such a setting. Moreover, we provide an instantiation of CDRE by using Kullback–Leibler Importance Estimation Procedure (KLIEP) (Sugiyama et al., 2008) as a building block, and provide theoretical analysis of its asymptotic behaviour. Besides evaluating generative models in continual learning, we also demonstrate the efficacy of CDRE in several online applications, including backward covariate shift, tracing distribution drift, monitoring real stock data for a regression model.

  This work (Chen et al., 2021) was published on arXiv preprint as *'Continual Density Ratio Estimation in an Online Setting'*, coauthored by Song Liu, Tom Diethe, and Peter Flach. It also has been accepted at the Distribution Shift workshop in NeurIPS 2021. The main idea is initiated by a discussion with Song regarding density ratio estimation in continual learning. I have done most of the writing and implemented the experiments of the paper.

[1] `https://sites.google.com/view/clvision2020/challenge?authuser=0`

As the chapters 4 to 7 of the thesis are introduced above, the rest of the thesis is organized as follows:

- Chapter 2 introduces preliminary knowledge that is served as an important basis in some related work and proposed methods, including several general methods of Stochastic Variational Inference (SVI) that can be widely applied in deep Bayesian models, and several Bayesian models using SVI for training, such as hierarchical probabilistic models, Variational Auto Encoder (VAE), Bayesian Neural Network (BNN);
- In Chapter 3, we first introduce background knowledge of continual learning, including different settings in application scenarios (which is highly related to the difficult level of tasks), widely applied benchmarks in related work, commonly used measurements of continual learning. We then review related work of continual learning, including three main categories (regularization-based, architecture-based, replay-based) of approaches.
- Chapter 8 concludes all the proposed work in this thesis and discusses existing problems and possible future work.

1.3 Other Research Work

Besides the work listed in the above section, there are several publications I have coauthored but not included in the main contents of this thesis as they are not directly related to continual learning:

- Zijian Shi, Yu Chen, John Cartlidge. The LOB Recreation Model: Predicting the Limit Order Book from TAQ History Using an Ordinary Differential Equation Recurrent Neural Network, in *Proceeding of the 35th AAAI Conference on Artificial Intelligence (AAAI-2021)*. This work proposes a generative model based on Ordinary Differential Equation Recurrent Neural Network (ODE-RNN) for recreating Limited Order Book (LOB) data by Trades and Quotes (TAQ) data, where the LOB data is usually costly to obtain access.
- Song Liu, Takafumi Kanamori, Wittawat Jitkrittum, Yu Chen. Fisher Efficient Inference of Intractable Models, in *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS 2019)*. This work proposes a Discriminative Likelihood Estimator (DLE) based on Stein Density Ratio Estimation (SDRE) that is Fisher efficient for inferring models with intractable likelihood function.
- Yu Chen, Tom Diethe, Peter Flach. ADL^{TM} : A Topic Model for Discovery of Activities of Daily Living in a Smart Home, in *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016)*. This work proposes a novel topic model along with an segmentation algorithm which can effectively discover residents' activities in a smart home

by sequential and unlabeled sensor data. The experiments were conducted by sensor data from real smart homes and have shown potentials in real applications such as abnormal activity detection, monitoring sleep quality.

- Yu Chen, Peter Flach. SVR-based modelling for the MoReBikeS challenge: Analysis, visualisation and prediction, in *Proceedings of the International Conference on ECML-PKDD 2015 Discovery Challenge Volume 1526(pp. 19–27)*. This work is a wining solution of the ECML-PKDD 2015 Discovery Challenge [2]. The challenge is to predict the number of available bikes in every bike rental stations 3 hours in advance using the historical records of 200 stations. Our solution is a support vector regression model accompanying with effective feature selection and transformation which are motivated by carefully designed visualization at different time points.

[2]`http://reframe-d2k.org/Challenge#NEWS_AND_RESULTS`

CHAPTER 2

PRELIMINARIES: VARIATIONAL INFERENCE

In this chapter we provide a brief review of some preliminary techniques of Variational Inference (VI) for Probabilistic models including Bayesian models that are not specific for continual learning but are important to several related approaches and proposed methods in later chapters.

In Bayesian models, the posterior of a random variable $\mathbf{z}$ can be expressed through the Bayesian rule:

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})} \tag{2.1}$$

In continual learning, the prior $p(\mathbf{z})$ is often defined as the posterior learned in the previous task, which naturally introduces the previous learned knowledge to the current task. VI is a general framework for learning various Bayesian models and the formulation of it can easily be adapted to continual learning in such a straightforward way.

In the following we firstly introduce a few essential techniques of Stochastic Variational Inference (SVI) (Sec. 2.1) that are widely applied in deep Bayesian models and secondly demonstrate the applications of these techniques on several representative deep Bayesian models (Sec. 2.2).

2.1 Stochastic Variational Inference

To model unobserved structures of data, probabilistic models construct the model through a set of latent random variables, which we will collect together and denote using a single vector-valued variable $\mathbf{z}$ in the rest of this chapter. These variables represent different underlying factors that we want to understand. The objective of such models is often to infer the posterior of $\mathbf{z}$ given observations $\mathbf{x}$, where we have again collected all types of observation into a single vector. Unfortunately, for most models of interest we cannot compute the exact posterior $p(\mathbf{z}|\mathbf{x})$ of most models as it often involves infeasible integration when the dimensionality is too high or the integrand is too complex (Bishop, 2006). In addition, we often want further analysis about the posterior which requires computing expectations w.r.t. the posterior, leading to further infeasible integration problems. Therefore, we alternatively deploy approximation methods for inferring posteriors of Bayesian models. Variational Inference (VI) is a family of such approximation methods that has been well studied and widely applied in practice (Blei et al., 2017; Zhang et al., 2019). VI turns the learning objective into an optimization problem, which usually involves minimizing the Kullback–Leibler (KL) divergence. As a result it is then feasible to make use of existing optimization techniques, in particular Stochastic Gradient Descent (SGD). As SGD is commonly applied in Deep Neural Networks (DNNs), VI provides a way to efficiently train deep Bayesian models whilst making use of modern deep learning software packages and accelerated hardware.

The basic idea of VI is to define an approximation $q(\mathbf{z})$ of the true posterior $p(\mathbf{z}|\mathbf{x})$ and minimize the KL-divergence between them. Minimizing the KL-divergence in VI is commonly transformed to maximizing the Evidence Lower Bound (ELBO) to avoid the intractable computation of the marginal probability $\log p(\mathbf{x})$:

$$\begin{aligned} D_{KL}(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \mathbb{E}_q[\log q(\mathbf{z})] - \mathbb{E}_q[\log p(\mathbf{x},\mathbf{z})] + \log p(\mathbf{x}) \geq 0, \\ \rightarrow \quad \log p(\mathbf{x}) \geq \mathbb{E}_q[\log p(\mathbf{x},\mathbf{z})] - \mathbb{E}_q[\log q(\mathbf{z})] = \mathscr{L}_{\text{ELBO}} \end{aligned} \tag{2.2}$$

where $\log p(\mathbf{x})$ can be viewed as a constant w.r.t. $\mathbf{z}$. The variational posterior $q(\mathbf{z})$ is usually defined in a tractable form and parameterized by certain parameters, denoted as $q(\mathbf{z};\boldsymbol{\theta})$. $\boldsymbol{\theta}$ is also called the variational parameter and represents the (set of) parameter(s) that we want to optimize. Moreover, the joint distribution $p(\mathbf{x},\mathbf{z})$ can be obtained by $p(\mathbf{x},\mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z};\boldsymbol{\beta})$, where $p(\mathbf{z};\boldsymbol{\beta})$ is the prior of $\mathbf{z}$ and parameterized by some hyperparameters $\boldsymbol{\beta}$. The optimal $\boldsymbol{\theta}$ should satisfy $\nabla_{\boldsymbol{\theta}}\mathscr{L}_{\text{ELBO}} = 0$, but it is often difficult to obtain the closed form for it without strong assumptions. SVI relaxes this requirement by using gradient descent to update variables iteratively and obtain a solution that approximately satisfies $\nabla_{\boldsymbol{\theta}}\mathscr{L}_{\text{ELBO}} \approx 0$. However, applying SVI to deep Bayesian models often requires gradient back propagation and it may be difficult when the model involves DNNs. In the following sections, we will introduce several methods that are proposed to solve this problem.

2.1.1 Reparameterization trick

To directly compute the gradient of $\boldsymbol{\theta}$ for the first term $\mathbb{E}_q[\log p(\mathbf{x},\mathbf{z})]$ of $\mathscr{L}_{\text{ELBO}}$, $\mathbf{z}$ needs to be expressed as a deterministic and differentiable function of $\boldsymbol{\theta}$. Kingma & Welling (2013) introduced reparameterization trick to achieve this goal. It uses a differentiable transformation $g(\epsilon,\boldsymbol{\theta})$ with an auxiliary noise variable ϵ to transform $\mathbf{z}$:

$$\mathbf{z} = g(\epsilon,\boldsymbol{\theta}) \ \text{ where } \ \epsilon \sim p(\epsilon) \tag{2.3}$$

For example, if we define $q(\mathbf{z};\boldsymbol{\theta}) = \mathcal{N}(\mathbf{z};\boldsymbol{\mu},\boldsymbol{\sigma}^2)$, where $\boldsymbol{\theta} = \{\boldsymbol{\mu},\boldsymbol{\sigma}\}$, we can transform $\mathbf{z}$ as $\mathbf{z} = \boldsymbol{\mu} + \epsilon\boldsymbol{\sigma}$, $\epsilon \sim \mathcal{N}(0,1)$. There are three basic approaches to choose a particular $q(\mathbf{z};\boldsymbol{\theta})$ that can be reparameterized as Eq. (2.3):

i) Tractable inverse CDF: let $\epsilon \sim \mathcal{U}(0,I)$, let $g(\epsilon,\boldsymbol{\theta})$ be the inverse CDF of $q(\mathbf{z};\boldsymbol{\theta})$;

ii) Any "location-scale" family distributions, analogous to the Normal distribution example above we can set $g(\cdot) = \boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \epsilon$;

iii) Composition of transformations, such as Log-Normal (exponentiation of Normal distributions). In Sec. 2.1.2 we introduce an approach to automatically transform $\mathbf{z}$ in such a compositional way.

In addition, Figurnov et al. (2018) proposed a reparameterization method that does not need an invertible CDF of $q(\mathbf{z};\boldsymbol{\theta})$ and can be applied to a wider class of distributions, such as Gamma, Beta, Dirichlet distributions. As these reparameterization methods cannot handle discrete variables, Jang et al. (2017) proposed reparameterizing discrete random variables from a categorical distribution by a Gumbel-Softmax distribution.

In practice, the variational posterior $q(\mathbf{z}|\mathbf{x})$ of a continuous variable $\mathbf{z}$ is often defined according to its support. For example, when $\mathbf{z} \in [0,1]$, $q(\mathbf{z}|\mathbf{x})$ is often a Beta distribution; when $\mathbf{z} \in (0,+\infty)$, $q(\mathbf{z}|\mathbf{x})$ is often a Gamma distribution. Such rules are easy to process automatically and leads to a convenient SVI framework that will be introduced in the next section.

2.1.2 Automatic Differentiation Variational Inference

Kucukelbir et al. (2017) proposed a general framework, Automatic Differentiation VI (ADVI), for performing inference on various models by automatic transformation and differentiation. This method includes two steps:

1) The first step of ADVI is to automatically transform the support of the latent variable to an unconstrained space where the transformation $T(\cdot)$ is invertible and differentiable:

$$\hat{\mathbf{z}} = T(\mathbf{z}), \quad T : \mathrm{supp}(p(\mathbf{z})) \to \mathbb{R}^K \tag{2.4}$$

Then the joint density can be expressed by $\hat{\mathbf{z}}$:

$$p(\mathbf{x},\mathbf{z}) = p(\mathbf{x}, T^{-1}(\hat{\mathbf{z}})) \times |J_{T^{-1}}(\hat{\mathbf{z}})| \tag{2.5}$$

where $J_{T^{-1}}$ is the Jacobian matrix of the inverse of T.

2) The second step is to automatically compose a variational posterior by Gaussian distributions. There are two options, one is to posit a factorized (mean-field) Gaussian variational approximation.

$$q(\hat{\mathbf{z}};\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu},\boldsymbol{\sigma}^2 I) = \prod_{k=1}^{K} \mathcal{N}(\mu_k,\sigma_k^2), \quad \boldsymbol{\theta} = \{\boldsymbol{\mu},\boldsymbol{\sigma}\} \tag{2.6}$$

Here $\{\boldsymbol{\mu}, \boldsymbol{\sigma}\}$ are K-dimensional vectors, K is the dimension of $\mathbf{z}$. The elements in $\mathbf{z}$ are assumed to be independent in this case. Also we need to remove the constraint $\boldsymbol{\sigma} > 0$ by another transformation as Eq. (2.4).

Another option is to posit a full-rank Gaussian variational approximation, which generalizes the mean-field Gaussian by relaxing the independence assumption:

$$q(\hat{\mathbf{z}}; \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{L}\boldsymbol{L}^T) \tag{2.7}$$

Here $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{L}\}$, $\boldsymbol{\Sigma}$ is re-parametrized by the non-unique formulation of Cholesky factorization (Pinheiro & Bates, 1996), which relaxes the requirement of positive diagonal entries in the unique factorization. So, $\boldsymbol{L}$ is a lower triangular matrix with unconstrained elements in $R^{K(K+1)/2}$.

After the two steps the ELBO can then be optimized by SGD using automatic differentiation that has been commonly applied in DNNs. Although Kucukelbir et al. (2017) only proposed using Gaussian variational posteriors, it can actually approximate Beta or Gamma distribution by logit-Normal or Log-Normal distribution which are composition of transformations. Furthermore, this framework could be extended to more options of posterior distributions by applying other reparameterization methods.

2.1.3 Black-Box Variational Inference

Another general SVI method Black-Box VI (BBVI) (Ranganath et al., 2014) further relaxes the requirement of differentiable joint probability $p(\mathbf{x}, \mathbf{z})$ and uses the score function $\nabla_{\boldsymbol{\theta}} \log q(\mathbf{z}; \boldsymbol{\theta})$ to estimate the gradient $\nabla_{\boldsymbol{\theta}} \mathscr{L}_{\text{ELBO}}$. By assuming that $\boldsymbol{\theta}$ satisfies $\nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}, \mathbf{z}) = 0$, then we can have:

$$\nabla_{\boldsymbol{\theta}} \mathscr{L}_{\text{ELBO}} = \mathbb{E}_q \left[(\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}; \boldsymbol{\theta})) \cdot \nabla_{\boldsymbol{\theta}} \log q(\mathbf{z}; \boldsymbol{\theta}) \right] \tag{2.8}$$

Applying Monte Carlo integration, this gradient can be approximated by:

$$\nabla_{\boldsymbol{\theta}} \mathscr{L}_{\text{ELBO}} \approx \frac{1}{K} \sum_{k}^{K} (\log p(\mathbf{x}, \mathbf{z}^{(k)}) - \log q(\mathbf{z}^{(k)}; \boldsymbol{\theta})) \cdot \nabla_{\boldsymbol{\theta}} \log q(\mathbf{z}^{(k)}; \boldsymbol{\theta}) \tag{2.9}$$

where $\mathbf{z}^{(k)} \sim q(\mathbf{z}; \boldsymbol{\theta})$. BBVI is easy to apply without the need of $\nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}, \mathbf{z})$. However, a major drawback of this approach is the high variance of the approximated gradient. To reduce the high variance, control variates were proposed (Ranganath et al., 2014; Paisley et al., 2012). The basic idea is to construct a new function $\hat{f}(\mathbf{z}, \boldsymbol{\theta})$ with the same expectation as $\mathbb{E}_q[\hat{f}(\mathbf{z}, \boldsymbol{\theta})] = \mathbb{E}_q[f(\mathbf{z}, \boldsymbol{\theta})]$, where $f(\mathbf{z}, \boldsymbol{\theta}) = (\log p(\mathbf{x}, \mathbf{z}) - \log q(\mathbf{z}; \boldsymbol{\theta}))$, but having lower variance, and then replace $f(\mathbf{z}, \boldsymbol{\theta})$ by $\hat{f}(\mathbf{z}, \boldsymbol{\theta})$ in Eq. (2.9). For any model in with it is difficult to apply those reparameterization methods, this method can be an alternative for the inference.

2.2 Learning Deep Bayesian Models by Stochastic Variational Inference

In this section we introduce how to apply SVI methods to several representative deep Bayesian models, such as hierarchical probabilistic models, Variational Auto Encoders (VAEs), Bayesian Neural Networks (BNNs). Several related works (e.g. Variational Continual Learning (VCL) (Nguyen et al., 2018), Uncertainty-guided Continual Bayesian Neural Network (UCB) (Ebrahimi et al., 2020)) and proposed approaches (Chapters 4 and 6) deployed SVI in very similar ways.

2.2.1 Hierarchical Probabilistic Models with Latent Variables

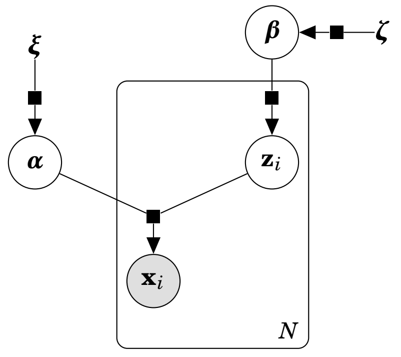


Figure 2.1: Factor graph of a general hierarchical probabilistic model. The grey circle represents observed data, white circles represent latent variables, variables without circles are hyperparameters, small black rectangles represent stochastic factors, and the rectangular plate represents replicates. In this model $\boldsymbol{\alpha}, \boldsymbol{\beta}$ are global latent variables while $\mathbf{z}_i$ are local latent variables.

Probabilistic models are a powerful and elegant framework that allows practitioners to perform inference over latent variables with specified interpretations, whilst dealing with uncertainty in a principled manner. Hierarchical probabilistic models can be cast in a form involving local and global latent variables, in which each local variable is defined for each observation whereas global variables are shared across observations. Local variables may depend on one or more global variables to reflect the structural relation between the latent variables. Such models are especially powerful when the underlying data structure is hierarchical. The factor graph (Kschischang et al., 2001) of a general hierarchical probabilistic model is shown in Fig. 2.1, where $\mathbf{z}_i$ represents local latent variables varying with observed data samples $\mathbf{x}_i$, N is the number of data samples. $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ represent global latent variables shared by all data samples, the difference is $\boldsymbol{\beta}$ governs the local variables $\mathbf{z}_i$ while $\boldsymbol{\alpha}$ does not. $\boldsymbol{\xi}$ and $\boldsymbol{\zeta}$ are hyperparameters of priors of $\boldsymbol{\alpha}, \boldsymbol{\beta}$. The definition of this model is as below:

$$\boldsymbol{\alpha} \sim p(\boldsymbol{\alpha}|\boldsymbol{\xi}), \;\; \boldsymbol{\beta} \sim p(\boldsymbol{\beta}|\boldsymbol{\zeta}), \;\; \mathbf{z}_i|\boldsymbol{\beta} \sim p(\mathbf{z}_i|\boldsymbol{\beta}), \;\; \mathbf{x}_i|\mathbf{z}_i, \boldsymbol{\alpha} \sim p(\mathbf{x}_i|\mathbf{z}_i, \boldsymbol{\alpha}) \tag{2.10}$$

A standard setting for VI assumes the latent variables are independent by conditioning on their own variational factors. Consequently, the variational posteriors can be written as the mean-field variational family (Blei et al., 2017). This technique can simplify the ELBO of hierarchical probabilistic models to a great extent. By such factorization, the ELBO of the model in Fig. 2.1 can be written as:

$$\mathscr{L}_{\text{ELBO}} = \mathbb{E}_q[\log p(\mathbf{x},\mathbf{z},\boldsymbol{\alpha},\boldsymbol{\beta})] - \mathbb{E}_q[\log q(\mathbf{z};\boldsymbol{\theta})] - \mathbb{E}_q[\log q(\boldsymbol{\alpha};\boldsymbol{\gamma})] - \mathbb{E}_q[\log q(\boldsymbol{\beta};\boldsymbol{\lambda})] \tag{2.11}$$

Here, $\mathbf{x} = \{\mathbf{x}_1,\ldots,\mathbf{x}_N\}$, $\mathbf{z} = \{\mathbf{z}_1,\ldots,\mathbf{z}_N\}$. $\Theta = \{\boldsymbol{\theta},\boldsymbol{\gamma},\boldsymbol{\lambda}\}$ where $\boldsymbol{\theta},\boldsymbol{\gamma},\boldsymbol{\lambda}$ denote parameters of the variational posteriors of $\mathbf{z},\boldsymbol{\alpha},\boldsymbol{\beta}$, respectively. The learning process is to find optimal Θ^* that maximizes $\mathscr{L}_{\text{ELBO}}$. Hoffman et al. (2013) proposed using SVI to optimize the ELBO of such a hierarchical model by natural gradients when the priors and posteriors are from the exponential family and are conjugate distributions. In addition to these conditions, the full conditional distribution of latent variables are also required. For example, to optimize $\boldsymbol{\lambda}$ in Eq. (2.11), the first term in Eq. (2.11) can be reduced to:

$$\begin{aligned} &\log p(\mathbf{x},\mathbf{z},\boldsymbol{\alpha},\boldsymbol{\beta}) = \log p(\boldsymbol{\beta}|\mathbf{x},\mathbf{z},\boldsymbol{\alpha}) + \log p(\mathbf{x},\mathbf{z},\boldsymbol{\alpha}) = \log p(\boldsymbol{\beta}|\mathbf{x},\mathbf{z}) + \text{const}, \\ &\mathscr{L}_{\text{ELBO}}(\boldsymbol{\lambda}) = \mathbb{E}_q[\log p(\boldsymbol{\beta}|\mathbf{x},\mathbf{z}) - \log q(\boldsymbol{\beta};\boldsymbol{\lambda})] \end{aligned} \tag{2.12}$$

Suppose the full conditional distribution and the variational posterior of the global variables $\boldsymbol{\beta}$ are in the following form of the exponential family:

$$\begin{aligned} p(\boldsymbol{\beta}|\mathbf{x},\mathbf{z}) &= h(\boldsymbol{\beta})\exp\{\eta(\mathbf{x},\mathbf{z})^T t(\boldsymbol{\beta}) - a(\eta(\mathbf{x},\mathbf{z}))\}, \\ q(\boldsymbol{\beta};\boldsymbol{\lambda}) &= h(\boldsymbol{\beta})\exp\{\boldsymbol{\lambda}^T t(\boldsymbol{\beta}) - a(\boldsymbol{\lambda})\} \end{aligned} \tag{2.13}$$

where $\eta(\cdot)$ is the natural parameter, $h(\cdot)$ is the base measure, $t(\cdot)$ are the sufficient statistics and $a(\cdot)$ is the log normalizer. In this case, the natural gradient of $\mathscr{L}_{\text{ELBO}}$ w.r.t. $\boldsymbol{\lambda}$ can be computed by:

$$\hat{\nabla}_{\boldsymbol{\lambda}}\mathscr{L}_{\text{ELBO}}(\boldsymbol{\lambda}) \triangleq \boldsymbol{G}^{-1}\nabla_{\boldsymbol{\lambda}}\mathscr{L}_{\text{ELBO}}(\boldsymbol{\lambda}) = \mathbb{E}_q[\eta(\mathbf{x},\mathbf{z})] - \boldsymbol{\lambda}, \tag{2.14}$$

where $\boldsymbol{G} = \mathbb{E}_q[(\nabla_{\boldsymbol{\lambda}}\log q(\boldsymbol{\beta};\boldsymbol{\lambda}))^2]$ is the Fisher information of $q(\boldsymbol{\beta};\boldsymbol{\lambda})$. The natural gradient uses a Riemannian metric to adjust the direction of the gradient, which accounts for the information geometry of its parameter space and is much easier to compute than the conventional gradient of $\mathscr{L}_{\text{ELBO}}$ in such a hierarchical model. Please refer to Sec. 4.1.1 for more detailed introduction of natural gradient. However, here computing natural gradient requires full conditional probabilities of all latent variables as in Eq. (2.13) which may not have a closed form or may be hard to compute.

To simplify inference over a hierarchical architecture, we can decouple latent variables based on their Markov blanket and construct optimization objectives for them separately as suggested in Winn & Bishop (2005), and the inference can be performed by gradient descent as introduced in the previous section. For instance, the ELBO of local variables $\mathbf{z}_i$ in Fig. 2.1 can be written as follows:

$$\max_{\boldsymbol{\theta}} \mathscr{L}_{\mathbf{z}}(\boldsymbol{\theta}) = \max_{\boldsymbol{\theta}} \sum_{i=1}^{N} [\mathbb{E}_{q(\mathbf{z}_i;\boldsymbol{\theta}_i)}[\log p(\mathbf{x}_i|\mathbf{z}_i,\boldsymbol{\alpha}) + \log p(\mathbf{z}_i|\boldsymbol{\beta}) - \log q(\mathbf{z}_i;\boldsymbol{\theta}_i)]] \tag{2.15}$$

We can construct $\mathscr{L}_{\boldsymbol{\alpha}}, \mathscr{L}_{\boldsymbol{\beta}}$ analogously.

$$
\begin{aligned}
&\max_{\boldsymbol{\lambda}} \mathscr{L}_{\boldsymbol{\beta}}(\boldsymbol{\lambda}) = \max_{\boldsymbol{\lambda}} \mathbb{E}_{q(\boldsymbol{\beta};\boldsymbol{\lambda})}[\sum_{i=1}^{N} \log p(\mathbf{z}_i|\boldsymbol{\beta}) + \log p(\boldsymbol{\beta}|\boldsymbol{\zeta}) - \log q(\boldsymbol{\beta};\boldsymbol{\lambda})], \\
&\max_{\boldsymbol{\gamma}} \mathscr{L}_{\boldsymbol{\alpha}}(\boldsymbol{\gamma}) = \max_{\boldsymbol{\gamma}} \mathbb{E}_{q(\boldsymbol{\alpha};\boldsymbol{\gamma})}[\sum_{i=1}^{N} \log p(\mathbf{x}_i|\mathbf{z}_i, \boldsymbol{\alpha}) + \log p(\boldsymbol{\alpha}|\boldsymbol{\xi}) - \log q(\boldsymbol{\alpha};\boldsymbol{\gamma})]
\end{aligned} \tag{2.16}
$$

For large scale data, the ELBO is usually estimated by mini-batches rather than the whole dataset. For computing the gradient for one mini-batch the approximation of the likelihood term would need an adjustment (Kingma & Welling, 2013). For example, the likelihood term of $\mathscr{L}_{\boldsymbol{\beta}}$ in Eq. (2.16) could be approximated by a mini-batch with size B as: $\frac{N}{B}\sum_i^B \log p(\mathbf{z}_i|\boldsymbol{\beta})$. The three objectives are optimized iteratively, i.e. when updating the local parameters $\boldsymbol{\theta}_i$ the global parameters $\boldsymbol{\lambda}, \boldsymbol{\gamma}$ are fixed. The dependency between the latent variables now relies on stochastic samples from their variational posteriors and the requirement of full conditional distributions can be avoided. Furthermore, different inference methods can be applied to different parts of the model, since they can be viewed as separate sub-models and are only connected by samples. By such a factorization, SVI can be applied to a wider range of hierarchical models as a general framework using automatic differentiation and gradient back propagation. We will demonstrate applying this technique to a hierarchical Item Response Theory (IRT) model in Chapter 6.

2.2.2 Variational Auto Encoder

The Variational Auto Encoder (VAE) (Kingma & Welling, 2013) is a popular generative model that combines neural networks and Bayesian inference. In the VAE a local variable is a latent representation of an observation. The posterior of the local latent variable $\mathbf{z}_i$ can be inferred by a deterministic function of $\mathbf{x}_i$, where the function is simulated by a neural network, such as $\mathbf{z}_i \sim q(\mathbf{z}_i;\boldsymbol{\theta}_i), \boldsymbol{\theta}_i = f_{\boldsymbol{\alpha}}(\mathbf{x}_i)$, where $\boldsymbol{\alpha}$ represents the parameters of the neural network f and it is the global variable over all $\mathbf{x}$. This inference technique is called amortized inference as the inference of individual $q(\mathbf{z}_i;\boldsymbol{\theta})$ is amortized by inferring a shared function $f_{\boldsymbol{\alpha}}$ for all local variables.

In the vanilla VAE, the posterior $q(\mathbf{z}_i;\boldsymbol{\theta}_i)$ is often a Gaussian distribution and then the output of the inference network is split into two parts: $\{\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i\} = f_{\boldsymbol{\alpha}}(\mathbf{x}_i) = \boldsymbol{\theta}_i$, by the reparameterization trick $\mathbf{z}_i = \boldsymbol{\mu}_i + \boldsymbol{\sigma}_i \epsilon, \epsilon \sim \mathcal{N}(0, I)$. Since $\mathbf{z}_i$ can be viewed as a latent representation of $\mathbf{x}_i$, $f_{\boldsymbol{\alpha}}(\cdot)$ plays the role like an encoder to transform $\mathbf{x}_i$ to $\mathbf{z}_i$. Analogously, a decoder is learned to recover $\mathbf{x}_i$ from $\mathbf{z}_i$. The decoder's objective is to maximize the log-likelihood $\log p(\mathbf{x}_i|\mathbf{z}_i)$, where $p(\mathbf{x}_i|\mathbf{z}_i)$ is often a Gaussian distribution as well, such as $p(\mathbf{x}_i|\mathbf{z}_i) = \mathcal{N}(g_{\boldsymbol{\beta}}(\mathbf{z}_i), \sigma_n^2 I)$. Here $\boldsymbol{\beta}$ is also a global variable representing the parameters of the decoder $g(\cdot)$, σ_n^2 is the variance of a Gaussian noise $\epsilon_n \sim \mathcal{N}(0, \sigma_n^2)$ which controls the smoothness of the loss. We can train the decoder and encoder simultaneously in an end-to-end fashion by optimizing the ELBO as the same as other Bayesian models:

$$\max_{\boldsymbol{\alpha},\boldsymbol{\beta}} \mathscr{L} = \sum_{i=1}^{N} \left[\mathbb{E}_{q(\mathbf{z}_i;\boldsymbol{\theta}_i)}[\log p(\mathbf{x}_i|\mathbf{z}_i)] - D_{KL}(q(\mathbf{z}_i;\boldsymbol{\theta}_i)||p(\mathbf{z}_i))\right], \text{ where } p(\mathbf{z}_i) = \mathcal{N}(\mu_0, \sigma_0^2 I),$$
$$p(\mathbf{x}_i|\mathbf{z}_i) = \mathcal{N}(g_{\boldsymbol{\beta}}(\mathbf{z}_i), \sigma_n^2 I), \ q(\mathbf{z}_i;\boldsymbol{\theta}_i) = \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2 I), \ \boldsymbol{\theta}_i = \{\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i\} = f_{\boldsymbol{\alpha}}(\mathbf{x}_i) \tag{2.17}$$

Here, the prior $p(\mathbf{z}_i)$ is set by pre-fixed hyperparameters for simplicity. Nevertheless, the prior can also be learned from the data in more advanced extensions to the vanilla VAE (van den Oord et al., 2017; Tomczak & Welling, 2018).

2.2.3 Bayesian Neural Networks

Bayesian Neural Networks (BNNs) introduce a Bayesian treatment to deep neural networks by imposing priors and posteriors on the network parameters (MacKay, 1992; Wang & Yeung, 2020) and VI can be applied to BNN through gradient back propagation by the reparameterization as well. In this case, the network parameters are the latent variables $\mathbf{z}$. In a mean-field Gaussian BNN, we usually define the prior and variational posterior of each parameter as a Gaussian distribution and assume the parameters are independent from each other, which is the simplest form of BNN.

$$\max_{\boldsymbol{\theta}} \mathscr{L} = \mathbb{E}_{q(\mathbf{z};\boldsymbol{\theta})}\left[\sum_{i=1}^{N} \log p(\mathbf{x}_i|\mathbf{z})] - D_{KL}(q(\mathbf{z};\boldsymbol{\theta})||p(\mathbf{z}))\right],$$
$$\text{where } p(\mathbf{z}) = \mathcal{N}(0, \sigma_0^2 I), \ q(\mathbf{z};\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2 I), \ \boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\sigma}\} \tag{2.18}$$

The log-likelihood could be the cross-entropy loss for usual classification tasks which is also the log-likelihood of a Categorical distribution. The prior can be interpreted as a regularization on the posterior. By the reparameterization trick (Eq. (2.3)) the variational parameters $\{\boldsymbol{\mu}, \boldsymbol{\sigma}\}$ can be optimized through gradient back propagation as we introduced above. The mean-field BNN may not be flexible enough to approximate a more complex posterior as it assumes all parameters are independent. However, to learn a full covariance matrix could be computationally prohibitive for a large model. A compromise could be assuming the parameters are layer-wise dependent and thus the covariance matrix is block-diagonal. We will introduce a method (Ritter et al., 2018) using such an idea in the next section.

Summary

In this chapter we introduced a few basic techniques of SVI for learning deep Bayesian models that are related to several chapters in the rest of this thesis. The formulation of ELBO in VI is also an important basis for Bayesian continual learning. In the following chapter we will introduce the related work of continual learning, which includes several Bayesian approaches such as VCL (Nguyen et al., 2018) and UCB (Ebrahimi et al., 2020) that are highly dependent on the methods we introduced in this chapter.

CHAPTER 3

Background and Related Work of Continual Learning

In this chapter we introduce the background and review related work of continual learning. We first summarize general application scenarios of continual learning and benchmark tasks that are widely applied in the literature in Sec. 3.1. We then provide definitions of popular evaluation metrics of continual learning in Sec. 3.2. In Sec. 3.3 we elaborate three main categories of continual learning approaches: regularization-based, architecture-based, and replay-based approaches. In addition, we give a brief review of generative models and theoretical analysis of continual learning which have drawn much less attention but may play an important role in future research. Through this chapter we provide an overview of the literature of continual learning and justify the motivations of our methods.

3.1 Application Scenarios of Continual Learning

An application scenario of continual learning can be described by three protocols: task, model, and training protocols. Different combinations of these protocols compose the scenario with various degrees of difficulty as they have different requirements of the task information. Certain types of approaches can only be applied to some of the scenarios because not all scenarios provide the task information that is necessary to them.

According to the problem setting of continual learning introduced in Chapter 1, i.e. tasks are received sequentially without full access to data of previous tasks, the ***task protocol*** of continual learning can be categorized in the following three types (van de Ven & Tolias, 2019) :

1. **Task-Incremental Learning (Task-IL)**: the model always has access to task identifiers during training and testing time. The tasks may or may not have overlapping domains and/or class;

2. **Domain-Incremental Learning (Domain-IL)**: all tasks are trying to solve the same problem with different input domain, for example, classifying the same set of classes with different types of instances of each class, the task identifiers are not provided during testing but may be available during training;
3. **Class-Incremental Learning (Class-IL)**: the model learns disjoint classes of data at each task which are exclusive to previously learned ones, the task identifiers are not provided during testing but may be available during training.

Except for the Task-IL case, the other two are required to be tested without knowing task identifiers. It is much easier to obtain better performance in the scenario of Task-IL since it is feasible to train task-specific components that are also valid during testing with the access to task identifiers. However, it is less practical in real applications as the task identifiers are often not available during testing time. In this sense, Domain-IL and Class-IL are more realistic but more difficult scenarios.

According to the task protocols, we can have two ***model protocols*** coping with different knowledge of task identifiers, which result in very different degrees of the difficulty of preserving the model performance on old tasks:

1. **Task-agnostic**: the model is shared across all learned tasks, which does not require task identifiers during testing time;
2. **Task-aware**: each task has a specific component that is not shared with other tasks, which requires task identifiers during training and testing time.

The task-agnostic model is a much more difficult setting than task-aware, even the simplest task-aware model, a multi-headed model (in which each task has a separate output layer and shares all other layers), can easily outperform carefully designed task-agnostic models as the output layer is usually the most vulnerable layer in terms of forgetting.

Moreover, according to the way of accessing the training data of the current task there are two different ***training protocols***:

1. **Offline-training**: we can access the entire training set of the current task during training time, i.e. the model can be trained on the training set with multiple epochs;
2. **Online-training**: we can only access to the training set of the current task in an online fashion, i.e. the model can only be trained with one epoch on the training set of each task. In practice, a model can be trained on a same batch by multiple iterations, or multiple batches can be sampled from a same buffer for multiple iterations. The point is that past samples can not be accessed again.

The training protocol does not depend on the task protocol or model protocol, however, awareness of task identifiers and task boundaries during training is optional to online-training but is necessary to offline-training.

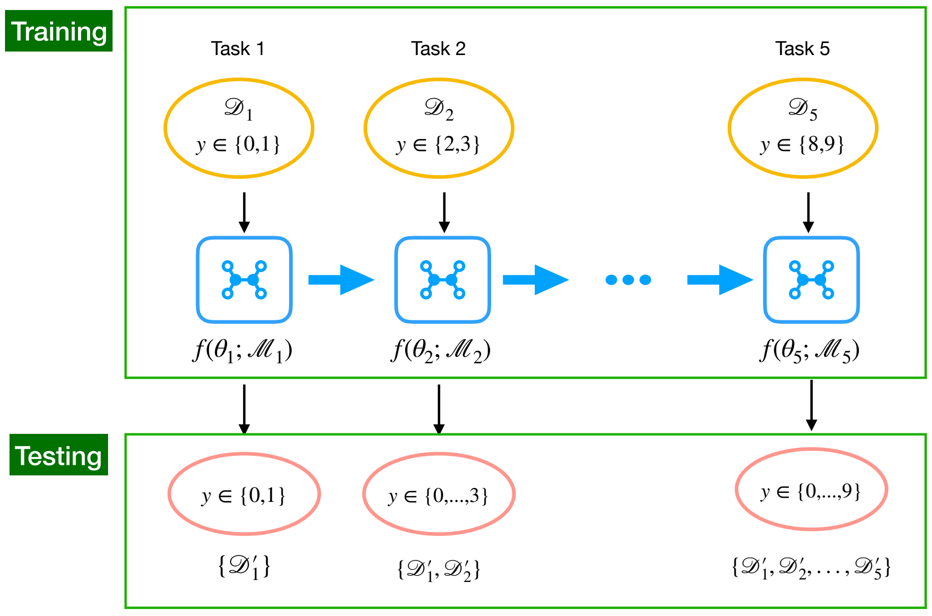
Table 3.1: Summary of the requirements of task information in different experimental scenarios in continual learning. The task identifiers are associated with every samples in the training or testing set. '✓' means mandatory, 'o' means optional, ✗means not available in the table.

	Task Protocol			Model Protocol		Training Protocol	
	Task-IL	Domain-IL	Class-IL	task-agnostic	task-aware	Offline	Online
Task identifiers (testing)	✓	✗	✗	✗	✓	o	o
Task identifiers (training)	✓	o	o	o	✓	✓	o

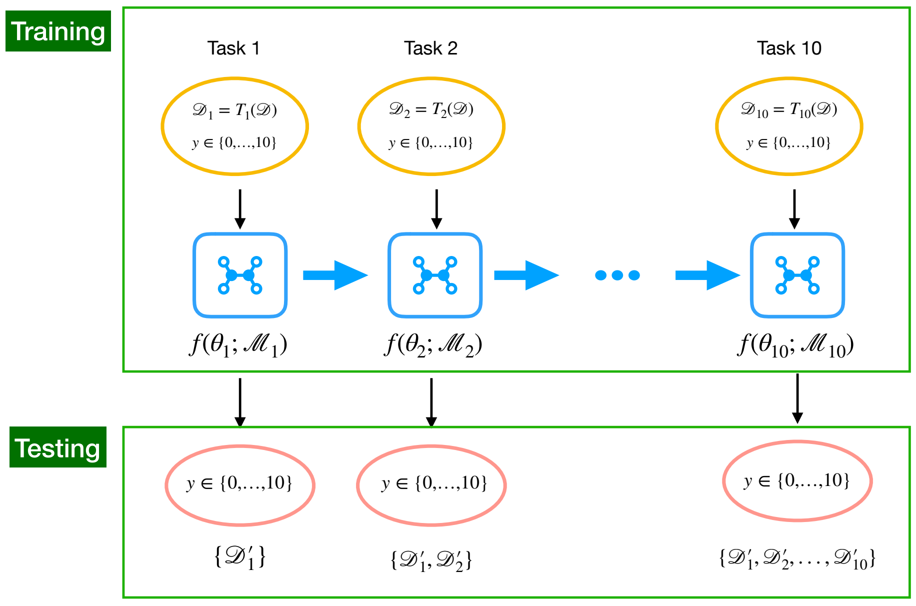
In general, the degree of difficulty of application scenarios in continual learning mainly depends on the task protocol which decides the model protocol. The training protocol is relatively independent but still relies on the availability of task identifiers. We summarize the requirements of all the protocols in Tab. 3.1. It is clear that task-aware models can only be deployed in Task-IL scenarios, and the online training is the only option when the task identifiers are not available during training. The most difficult scenario is that there is no access to task identifiers during training and testing, which indicates the only feasible option is task-agnostic models with online training and Task-IL is not applicable in such a circumstance. However, it may be the most pragmatic scenario for online applications with streaming data.

In the following, we will introduce common benchmarks in the literature of continual learning and demonstrate applying different protocols to these benchmarks. In most of the related work, split benchmarks (e.g. Split MNIST (Kirkpatrick et al., 2017)) and transformed benchmarks (e.g. Permuted MNIST (Kirkpatrick et al., 2017)) are commonly applied to evaluate the proposed approaches. The split benchmarks usually separate a dataset of a conventional classification task (e.g. MNIST, CIFAR10, CIFAR100) to several subsets, where each subset corresponds to a task and contains a subset of the classes in that dataset which is exclusive to other subsets. We demonstrate Split-MNIST as a typical split benchmark in Fig. 3.1a. In this benchmark, there are 5 tasks in total and each task contains 2 classes (digits) of the MNIST (LeCun et al., 2010) dataset. The split benchmarks are often deployed as Task-IL or Class-IL scenarios, depending on whether or not the task identifiers are available during testing time. The model protocols then can be chosen accordingly. The choice of training protocols mainly depends on if the benchmark is deployed to represent an online or offline application.

The transformed benchmarks generally consist of multiple tasks that are generated by different transformations of a dataset. We also demonstrate Permuted-MNIST as a typical transformed benchmark in Fig. 3.1b. It usually consists of 10 tasks (the number can be defined freely as it does not depend on the number of classes in the dataset), where each task contains the same 10 classes (digits) but with a different permutation of the MNIST images. The permutation can be replaced by other transformation functions, such as rotation. The transformed benchmarks represent applications that receives new instances of each class in a new task, which naturally fit the Domain-IL scenario.


Training
Task 1
Task 2
Task 5
$\mathscr{D}_1$ $y \in \{0,1\}$
$\mathscr{D}_2$ $y \in \{2,3\}$
$\mathscr{D}_5$ $y \in \{8,9\}$
$f(\theta_1; \mathscr{M}_1)$
$f(\theta_2; \mathscr{M}_2)$
$f(\theta_5; \mathscr{M}_5)$
Testing
$y \in \{0,1\}$
$y \in \{0,...,3\}$
$y \in \{0,...,9\}$
$\{\mathscr{D}'_1\}$
$\{\mathscr{D}'_1, \mathscr{D}'_2\}$
$\{\mathscr{D}'_1, \mathscr{D}'_2, \dots, \mathscr{D}'_5\}$

(a) Depiction of split benchmarks for Split-X tasks where X is the dataset. In this benchmark, there are 5 tasks in total and each task contains 2 classes (digits) of the dataset.


Training
Task 1
Task 2
Task 10
$\mathscr{D}_1 = T_1(\mathscr{D})$ $y \in \{0,\dots,10\}$
$\mathscr{D}_2 = T_2(\mathscr{D})$ $y \in \{0,\dots,10\}$
$\mathscr{D}_{10} = T_{10}(\mathscr{D})$ $y \in \{0,\dots,10\}$
$f(\theta_1; \mathscr{M}_1)$
$f(\theta_2; \mathscr{M}_2)$
$f(\theta_{10}; \mathscr{M}_{10})$
Testing
$y \in \{0,\dots,10\}$
$y \in \{0,\dots,10\}$
$y \in \{0,\dots,10\}$
$\{\mathscr{D}'_1\}$
$\{\mathscr{D}'_1, \mathscr{D}'_2\}$
$\{\mathscr{D}'_1, \mathscr{D}'_2, \dots, \mathscr{D}'_{10}\}$

(b) Depiction of transformed benchmarks for transformed-X tasks. In this benchmark, there are 10 tasks in total and each task contains the same 10 classes (digits) with different permutations of the features (column indices) of instances in the dataset.T_i represents the permutation applied in the i-th task.

Figure 3.1: Depiction of common benchmarks in continual learning. $\forall i \in \{1, 2, \dots, t\}, \mathscr{D}_i, \mathscr{D}'_i$ respectively denote the training and testing set of the i-th task, where t is the total number of tasks. $f(\cdot;\cdot)$ represents the model, $\boldsymbol{\theta}_i, \mathscr{M}_i$ denote the model parameter and episodic memory obtained after learning the i-th task.

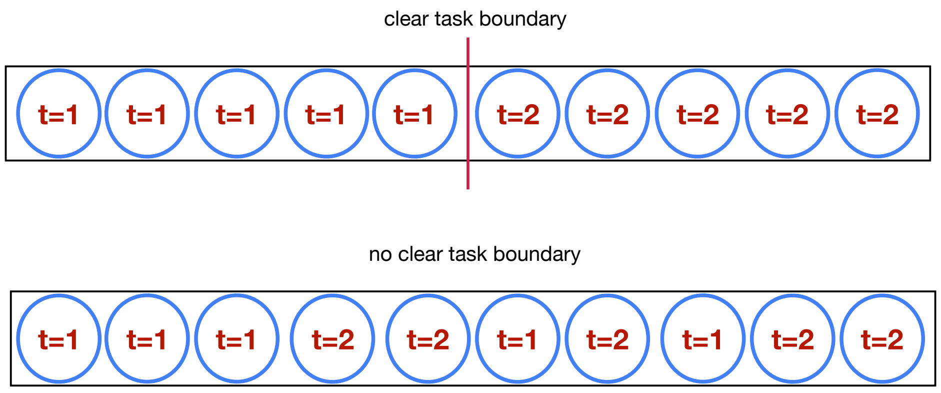


Figure 3.2: Demonstration of scenarios of task boundaries. Each circle represents one data sample of the data stream, t is the task identifier in the figure.

Nonetheless, split and transformed benchmarks can be applied with all three task protocols (van de Ven & Tolias, 2019). For example, Permuted-MNIST can be setup as a Class-IL application by treating the combination of each permutation and class as a new class.

Besides the awareness of task identifiers, the awareness of task boundaries during training is another source of difficulty to scenarios using online training. Many continual learning approaches rely on the availability of task boundaries to consolidate the knowledge of a learned task which we will see in Sec. 3.3. Note that knowing task identifiers during training does not mean knowing the task boundaries because there might be no clear split between tasks (i.e., when receiving data from a new task the model might still receive data from old tasks). We demonstrate the different scenarios of task boundaries in Fig. 3.2. In addition, the specific setting of episodic memories is also an important factor to the application scenarios. We will elaborate on this point along with the introduction of replay-based approaches in Sec. 3.3 because it highly depends on the applied strategy of experience replay.

Here we list the benchmarks we deployed in our experiments of Chapters 4 and 5, the Permuted MNIST is setup as Domain-IL, other split benchmarks are setup as Task-IL or Class-IL as specified in the latter chapters:

1) *Permuted MNIST*: 10 tasks using the MNIST dataset (LeCun et al., 2010), each task includes the same 10 classes with different permutation of features.

2) *Split MNIST*: 5 tasks using the MNIST dataset, each task includes 2 classes which are disjoint from the other tasks.

3) *Split Fashion-MNIST*: 5 tasks using the Fashion-MNIST dataset (Xiao et al., 2017), each task includes 2 classes which are disjoint from the other tasks.

4) *Split CIFAR-10*: 5 tasks using the CIFAR-10 dataset (Krizhevsky et al., 2009), each task includes 2 classes which are disjoint from other tasks.

5) *Split CIFAR-100*: 10 tasks using the CIFAR-100 dataset (Krizhevsky et al., 2009), each task includes 10 classes which are disjoint from other tasks.

6) *Split TinyImageNet*: 20 tasks using the TinyImageNet dataset (Le & Yang, 2015), each task includes 10 classes which are disjoint from other tasks.

3.2 Measurements of Continual Learning

As we introduced in Chapter 1, the stability of a model on old tasks is usually contradictory with the plasticity for accommodating new tasks. To get an overview on the trade-off between the stability and plasticity, an average performance over all learned tasks is the most convenient way to evaluate proposed approaches in continual learning. In the literature, the most prevalent choice for classification tasks is the average accuracy. Besides accuracy, other common metrics of classification could be options as well, such as precision, recall, F1 score. However, the average classification performance can not tell whether the stability or the plasticity is the main drawback of an approach. To obtain a more comprehensive evaluation in continual learning, the following three measurements are often used jointly for comparing different methods on a same benchmark task sequence:

Average accuracy, which is evaluated after learning all tasks. Let $a_{t,i}$ be the accuracy on task i after learning task t, where t is the index of the latest task, the definition of average accuracy is as follows:

$$\bar{a}_t = \frac{1}{t}\sum_{i=1}^{t} a_{t,i}, \tag{3.1}$$

.

Average forgetting (Chaudhry et al., 2018), which measures average accuracy drop of all tasks after learning the whole task sequence:

$$\bar{f}_t = \frac{1}{t-1}\sum_{i=1}^{t-1} \max_{j\in\{i,\dots,t-1\}} (a_{j,i} - a_{t,i}), \tag{3.2}$$

This metric quantifies forgetting and is a critical criterion for evaluating continual learning approaches since it is the main challenge in continual learning.

Average intransigence (Chaudhry et al., 2018), which measures the inability of a model to learn new tasks:

$$\bar{I}_t = \frac{1}{t}\sum_{i=1}^{t}(a_i^* - a_{i,i}), \tag{3.3}$$

where $a_{i,i}$ is the accuracy of task i at time i. a_i^* is the upper-bound accuracy of task i that could be obtained by a model trained solely on this task. Instead of training an extra model for each task, we use the best accuracy among all compared models as a_i^* in our experiments which avoids additional computational overhead. This metric quantifies intransigence and provides a complementary view of forgetting.

There are also some other metrics defined in a similar way, such as Backward Transfer (BWT) (Lopez-Paz & Ranzato, 2017) which is very similar to average forgetting. The difference is that BWT assumes $a_{i,i}$ is the highest accuracy of the i-th task given by the model which may not be always true. Another metric Forward Transfer (FWT) (Lopez-Paz & Ranzato, 2017) evaluates the initial performance on a new task before training on it, which shows if it is easy to transfer from the learned tasks. This metric does not reflect the actual performance on the new task and we consider the transferability is implied by the intransigence, hence, it is not adopted in our experiments.

Although average accuracy can provide an overview on the trade-off between forgetting and intransigence, it is not suitable for model selection in some cases. For example, when two models have the same average accuracy but different forgetting and intransigence, it is hard to tell which one is better if there is no extra criterion. In Chapter 6 we will introduce a novel measure which can provide richer insights than average accuracy for evaluating classifiers in continual learning.

For generative models in continual learning, the measurements used in related work are as the same as in static learning which use raw samples of all learned tasks to evaluate the generated samples. Regarding the particular setting of continual learning, i.e. with very limited or no access to samples of previous tasks, there is no prior work about how to evaluate generative models in such a case, and we will address this issue in Chapter 7.

3.3 Main Categories of Approaches in Continual Learning

There are two main routes of exploring ideas for alleviating forgetting in continual learning: *i*) preserving the knowledge of models of previous tasks; *ii*) preserving the knowledge of data distributions of previous tasks. Most existing work focuses on either way of solving the forgetting problem. In practice, a full solution to a real application could combine the strength of both ways.

In terms of preserving the knowledge of previous models, there are two main categories of approaches: *1*) regularization-based approaches, which focus on preserving important parameters of the previous tasks and the model usually remains the same architecture; *2*) architecture-based approaches, which evolve the model by separately building task-shared and task-specific blocks for each task; The architecture-based approaches are mostly restricted to the Task-IL scenario, i.e. task identifiers are necessary during training and testing. A simple example is the multi-headed model. In addition, architecture-based methods often deploy regularization-based methods in task-shared components for alleviating forgetting on old tasks. For instance, applying regularization-based methods to all shared layers of a multi-headed model except the multiple output layers.

On the other hand, replay-based approaches focus on preserving the knowledge of previous data distributions by replaying data samples stored in the episodic memory or generated by a generative model. In general, replay-based methods with episodic memories are more efficient than methods in other categories because this type of approach can provide competitive performance with much less computational cost. Particularly, when using a task-agnostic model in the Class-IL scenarios, a simple replay-based method with a small episodic memory, such as Experience Replay (ER) (Chaudhry et al., 2019b), can outperform SOTA methods in other categories (Prabhu et al., 2020). The benchmarks of Class-IL (e.g. split-MNIST, split-CIFAR) often have less similar tasks comparing with benchmarks of Domain-IL. Information shared through parameters of such tasks is not enough for preserving good performance on past tasks. In comparison, samples from past tasks are more informative in such cases. These results show that replay-based method has great potential for efficiently solving practical problems in real applications of continual learning. For this reason, we propose approaches mainly in this category, such as DRL (Chapter 5), Stein-gradient based episodic memories (Chapter 4).

In the following sections we will review several representative approaches of each aforementioned category in continual learning.

3.3.1 Regularization-based Approaches

The most intuitive cause of catastrophic forgetting is that model parameters tend to change towards fitting new data from a new task and thus deviate from optimal values of previously learned tasks. A straightforward intuition to solve this problem is finding a point that is closest to the optimal location of each task. Since the model is trained on a sequence of tasks, then it would be preferred that the search space for a new task is close to the optimal location of the previous task. This can be easily interpreted in a Bayesian way by viewing the preferred search space as the prior distribution of model parameters. Applying the principle of Maximum A Posteriori (MAP), we can obtain a general regularization term as the prior of model parameters to prevent parameters changing too much for preserving reasonable performance on old tasks.

$$p(\boldsymbol{\theta}|X) \propto p(X|\boldsymbol{\theta})p(\boldsymbol{\theta}) \quad \rightarrow \quad \boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}}(\log p(X|\boldsymbol{\theta}) + \log p(\boldsymbol{\theta})) \tag{3.4}$$

where X is observed data, $\boldsymbol{\theta}$ denotes model parameters, $p(\boldsymbol{\theta})$ represents the prior of $\boldsymbol{\theta}$. In regularization-based methods of continual learning, the prior is often defined according to the importance of parameters of previous tasks. For example, if the loss of previous tasks will increase a lot by slightly changing a parameter, then the penalty on changing that parameter would be large. The key point is how to identify and quantify the importance of parameters. The regularization-based methods explore different ways to achieve this goal. In principle, when we define the prior $p(\boldsymbol{\theta})$ by different forms we can get different regularization-based methods. In this section, we will elaborate technical details of several SOTA methods in this category.

Table 3.2: An overview of introduced approaches in the three main categories.

Category	Type	Approach	Authors & Year
Regularization based	Weight importance	EWC	Kirkpatrick et al. (2017)
		OEWC	Schwarz et al. (2018)
		Kronecker factored	Ritter et al. (2018)
		SI	Zenke et al. (2017)
		VCL	Nguyen et al. (2018)
		UCB	Ebrahimi et al. (2020)
	Node importance	AGS-CL	Jung et al. (2020)
		UCL	Ahn et al. (2019)
Architecture based	Adding task-specific nodes	P&C	Schwarz et al. (2018)
		DEN	Yoon et al. (2018)
		REC	Zhang et al. (2020b)
	Adding task-specific masks	HAT	Mallya et al. (2018)
		CLAW	Adel et al. (2020)
		APD	Yoon et al. (2020)
		Batch Ensemble	Wen et al. (2020)
Replay based	Sampling strategy for replay	ER	Chaudhry et al. (2019b)
		CBRS	Chrysakis & Moens (2020)
		MIR	Aljundi et al. (2019a)
	Gradient-based	GEM	Lopez-Paz & Ranzato (2017)
		A-GEM	Chaudhry et al. (2019a)
		OGD	Farajtabar et al. (2020)
		MEGA	Guo et al. (2020)
		ORTHOG-SUBSPACE	Chaudhry et al. (2020)
		MER	Riemer et al. (2019)
		GSS	Aljundi et al. (2019b)
	Knowledge distillation	iCaRL	Rebuffi et al. (2017)
		DER & DER++	Buzzega et al. (2020)

Importance-based weight regularization

We first introduce Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017) which can be viewed as a starting point of regularization-based approaches in continual learning. It approximates the prior as a Gaussian distribution centered at the previous optimum and the covariance matrix is estimated by the Hessian of parameters w.r.t. a previous loss.

$$\begin{aligned}&\mathscr{L}_\tau(\boldsymbol{\theta}) \approx \mathscr{L}_\tau(\boldsymbol{\theta}_\tau^*) + \mathscr{L}_\tau^{'}(\boldsymbol{\theta}_\tau^*)(\boldsymbol{\theta}-\boldsymbol{\theta}_\tau^*) + \frac{1}{2}\mathscr{L}_\tau^{''}(\boldsymbol{\theta}-\boldsymbol{\theta}_\tau^*)^2,\\ &\Delta\mathscr{L}_\tau = \mathscr{L}_\tau(\boldsymbol{\theta}) - \mathscr{L}_\tau(\boldsymbol{\theta}_\tau^*) \approx \frac{1}{2}\mathscr{L}_\tau^{''}(\boldsymbol{\theta}-\boldsymbol{\theta}_\tau^*)^2\end{aligned} \tag{3.5}$$

Here $\mathscr{L}_\tau(\boldsymbol{\theta})$ is the loss on the τ-th task and $\boldsymbol{\theta}_\tau^*$ is the optimal $\boldsymbol{\theta}$ that satisfies $\mathscr{L}_\tau^{'}(\boldsymbol{\theta}_\tau^*) = 0$. According to Eq. (3.5) we can see that the loss change on the τ-th task can be approximated by the Hessian and the distance to $\boldsymbol{\theta}_\tau^*$, which leads to the principle idea of EWC. The loss function of EWC for a

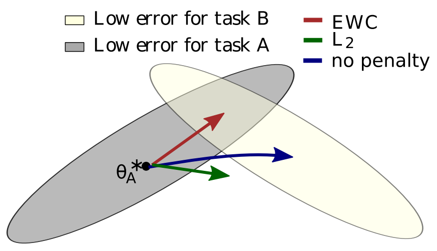


Figure 3.3: Demonstration of EWC by two tasks, the figure is from Kirkpatrick et al. (2017). The ellipses are parameter regions leading to good performance on task A (gray) and on task B (cream). The arrows point to the changed locations from the optimum of task A after training task B. With EWC, the new location is inside both ellipses and thus both tasks can obtain good performance. In comparison, without penalty the new location leads to good performance on task B but not good on task A, with L2 regularization it is not good for either task.

current task is defined as follows:

$$\mathscr{L}_{\text{EWC}}(\boldsymbol{\theta}) = \mathscr{L}_t(\boldsymbol{\theta}) + \frac{\lambda}{2}\sum_{\tau=1}^{t-1}\sum_{i=1}^{W} F_{\tau,i}\left(\boldsymbol{\theta}_i - \boldsymbol{\theta}^*_{\tau,i}\right)^2 \tag{3.6}$$

where $\boldsymbol{\theta}^*_{\tau,i}$ is the i-th parameter trained for a previous task τ; $\mathscr{L}_t(\boldsymbol{\theta})$ is a usual loss of the current task t, e.g., the cross entropy loss of classification tasks; W is the number of model parameters; λ is a scalar parameter for adjusting the trade-off between the performance of the new and old tasks. EWC proposes to approximate the diagonal of the Hessian with the Fisher information and omits the off-diagonal information in the Hessian, which makes the prior is equivalent to a mean-field Gaussian. This is done because computing the full Hessian is too expensive and prohibitive for large models. Therefore, EWC evaluates the importance of individual parameters by their Fisher information.

When moving to a new task, EWC will try to keep the network parameters close to the learned parameters of all previous tasks. Fig. 3.3 shows how the introduced regularization term differs from commonly used L2 regularization which can be interpreted as the importance of all parameters are the same. The L2 regularization may lead to a closest position in terms of Euclidean distance but outside of the optimal region of either task.

However, in Eq. (3.6) the Fisher information and the optimal parameters of previous tasks need to be stored, while learning more tasks the memory cost will increase linearly with the number of tasks. Schwarz et al. (2018) proposed Online EWC that approximates the overall posterior of all previous tasks by a moving sum. Instead of storing individual Fisher Information for each task, it only needs to store the optimal parameters of the last task and the moving sum of Fisher information of previous tasks:

$$\mathscr{L}_{\text{OEWC}}(\boldsymbol{\theta}) = \mathscr{L}_t(\boldsymbol{\theta}) + \frac{\lambda}{2}\sum_{i=1}^{W} F^*_{t-1,i}(\boldsymbol{\theta}_i - \boldsymbol{\theta}^*_{t-1,i})^2, \;\; F^*_t = \gamma F^*_{t-1} + F_t, \;\; \gamma < 1 \tag{3.7}$$

where t is task index, γ is a hyperparameter that can control the speed of forgetting on older tasks. This extension of EWC shows similar performance with original EWC using a constant memory cost.

Instead of the diagonal approximation of the Hessian by Fisher information, Ritter et al. (2018) proposed Kronecker-factored approximation of the Hessian which is a block-diagonal approximation. It can preserve layer-wise correlations between parameters as one block consists of parameters from one layer, and thus it is more accurate than the diagonal approximation with Fisher information as used in EWC and Online EWC.

$$H_l = \frac{\partial^2 \log p(X|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}_l \partial \boldsymbol{\theta}_l} = \mathcal{Q}_l \otimes \mathcal{H}_l, \quad \mathcal{Q}_l = \boldsymbol{a}_{l-1}\boldsymbol{a}_{l-1}^T, \quad \mathcal{H}_l = \frac{\partial^2 \log p(X|\boldsymbol{\theta})}{\partial \boldsymbol{h}_l \partial \boldsymbol{h}_l} \tag{3.8}$$

where H_l is one diagonal block of the estimated Hessian, l is the layer index, $\boldsymbol{h}_l$ and $\boldsymbol{a}_l$ are the pre-activation and activation output of the l-th layer respectively, $\boldsymbol{\theta}_l$ is the weight matrix of the l-th layer, $\otimes$ denotes Kronecker product. The computational complexity of this method is $O(\max_l(|\boldsymbol{\theta}_l|^2))$, where $|\boldsymbol{\theta}_l|$ is the number of parameters from the l-th layer. Although it is more tractable than a full approximation of the Hessian matrix, it may still be expensive to compute for a large model.

In addition to these methods that are based on the Hessian matrix, Synaptic Intelligence (SI) (Zenke et al., 2017) suggested another regularization formulation for preserving important parameters, which is inspired by biological neural networks:

$$\mathcal{L}_{\text{SI}}(\boldsymbol{\theta}) = \mathcal{L}_t(\boldsymbol{\theta}) + c\sum_i \Omega_i^t \left(\boldsymbol{\theta}_i - \boldsymbol{\theta}_{t-1,i}^*\right)^2, \quad \Omega_i^t = \sum_{j<t} \frac{\omega_i^j}{\left(\Delta_i^j\right)^2 + \xi} \tag{3.9}$$

where c is a hyperparameter with the same usage as λ in Eq. (3.6), and ξ is used to bound the expression in cases where $\Delta_i^j \to 0$. Ω_i^t is the importance measure determined by two terms that simulate the complexity of synapses in biological neural networks:

1) ω_i^j measures how much an individual parameter $\boldsymbol{\theta}_i$ contributed to a drop in the loss of task j, it is defined as follows:

$$\omega_i^j \triangleq -\int_{\tau^{j-1}}^{\tau^j} g_i(\boldsymbol{\theta}(\tau))\boldsymbol{\theta}_i'(\tau)d\tau, \quad g_i(\tau) = \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}_i}, \quad \boldsymbol{\theta}_i'(\tau) = \frac{d\boldsymbol{\theta}_i}{d\tau} \tag{3.10}$$

where τ represents time points of training. ω_i^j can be approximated online as a running sum during training.

2) $\Delta_i^j \triangleq \boldsymbol{\theta}_{j,i}^* - \boldsymbol{\theta}_{j-1,i}^*$ measures how far a parameter $\boldsymbol{\theta}_i$ moved for task j.

Since Ω_i^t is a sum over learned tasks as well, SI takes a constant memory cost as the same as online EWC. However, SI does not outperform EWC or Online EWC according to empirical results in related work (Nguyen et al., 2018; Prabhu et al., 2020).

Bayesian treatments of weight regularization

So far, the regularization-based approaches introduced above for common neural networks do not explicitly learn a posterior for each parameter. Instead, Nguyen et al. (2018) proposed a continual learning framework VCL for Bayesian neural networks that estimates posterior distributions of parameters by VI (Sec. 2.1). The loss of VCL is then formed by the ELBO (Sec. 2.1): let $\boldsymbol{\theta}$ denote the network parameters, define a variational posterior $q_t(\boldsymbol{\theta})$ of the parameters for the t-th task, the approximated posterior of its previous task ($q_{t-1}(\boldsymbol{\theta})$) is treated as the prior of $\boldsymbol{\theta}$, then the KL-divergence between the variational posterior and prior takes the role of regularization:

$$\mathscr{L}_{\text{VCL}}(\boldsymbol{\theta}) = -\mathbb{E}_{q_t(\boldsymbol{\theta})}[\log p(\mathscr{D}_t|\boldsymbol{\theta})] + D_{KL}(q_t(\boldsymbol{\theta})\|q_{t-1}(\boldsymbol{\theta})) \tag{3.11}$$

In this setting, $\boldsymbol{\theta}$ is a random variable and $q_t(\boldsymbol{\theta})$ is further parameterized. For instance, $q_t(\boldsymbol{\theta})$ is often defined as a Gaussian distribution and parameterized by its mean and covariance matrix. The optimization objective is to optimize such parameters of the variational posterior $q_t(\boldsymbol{\theta})$ as the same as we introduced in Sec. 2.2.3 for learning BNNs. When the posterior is assumed to be a mean-field Gaussian distribution, VCL explicitly defines the importance of parameters as their variance which can be easily inferred by Stochastic Variational Inference (SVI) (Sec. 2.1). However, using the conventional gradient in SVI changes the variational parameters in the steepest direction of the Euclidean space, which may cause a large difference in terms of distributions by a small change on the variational parameters. We propose using natural gradients in the gradient back-propagation of SVI in Chapter 4 to address this issue. Note that VCL can be a general framework for Bayesian models, including deep generative models as well, such as VAE, which we will introduce later in Sec. 3.4.

UCB (Ebrahimi et al., 2020) is another approach proposed for Bayesian neural networks in continual learning. It applies SVI for optimizing the model parameters as well. One difference is that UCB uses a pre-defined prior for all tasks which is a mixture of Gaussians with two components and both components are zero-centered with different variance. Such a prior was proposed by Blundell et al. (2015) to resemble a spike-and-slab prior.

$$\begin{aligned} &\mathscr{L}_{\text{UCB}}(\boldsymbol{\theta}) = -\mathbb{E}_{q_t(\boldsymbol{\theta})}[\log p(\mathscr{D}_t|\boldsymbol{\theta})] + D_{KL}(q_t(\boldsymbol{\theta})\|q_0(\boldsymbol{\theta})), \\ &q_t(\boldsymbol{\theta}) = \mathcal{N}(\mu,\sigma^2 I), \;\; q_0(\boldsymbol{\theta}) = \pi\mathcal{N}(0,\sigma_1^2 I) + (1-\pi)\mathcal{N}(0,\sigma_2^2 I), \;\; 0<\pi<1, \end{aligned} \tag{3.12}$$

Moreover, UCB introduces a scheduling of the learning rate by the standard deviation of the posterior estimated by the last task.

$$\begin{aligned} &\boldsymbol{\mu}^k \leftarrow \boldsymbol{\mu}^{k-1} - \alpha_{\boldsymbol{\mu}}\nabla\mathscr{L}_{UCB}, \;\; \sigma^k \leftarrow \sigma^{k-1} - \alpha_\sigma \nabla\mathscr{L}_{UCB}, \\ &\alpha^t_{\boldsymbol{\mu}} \leftarrow \alpha^{t-1}_{\boldsymbol{\mu}}\sigma^{t-1} \;\text{ or }\; \alpha^{t-1}_{\boldsymbol{\mu}}\sigma^{t-1}/|\boldsymbol{\mu}|, \;\; \alpha^t_\sigma \leftarrow \alpha^{t-1}_\sigma \end{aligned} \tag{3.13}$$

Here k is the index of update steps, t is the task index. This method uses the uncertainty of model weights learned by the last task to adjust the learning rate for current task. Intuitively, larger σ indicates the weight can vary in a wider range and thus can be changed faster when learning a new task. This adjustment of learning rate is also very similar to the natural gradient of mean-field Gaussian (Chapter 4).

Regularizing weights by node importance

Instead of introducing weight-wise uncertainty in Bayesian neural networks, Ahn et al. (2019) proposes Uncertainty-regularized Continual Learning (UCL) to estimate node-wise uncertainty and introduces several additional regularization terms for controlling stability and plasticity separately. UCL also deploys the Bayesian paradigm with variational inference similar to VCL, in which the parameter posteriors are approximated by mean-field Gaussian as well. In particular, the variance of a neuron (i.e., a hidden node) is shared by its incoming weights, and the KL-divergence term in Eq. (3.11) is replaced by several regularization terms which are extended from the KL-divergence term:

$$\begin{aligned}\mathscr{L}_{\text{UCL}} = -\log p(\mathscr{D}_t|\boldsymbol{\theta}) + \sum_{l=1}^{L} \Bigg[& \underbrace{\frac{1}{2}\left\| \Lambda^{(l)} \odot (\boldsymbol{\mu}_t^{(l)} - \boldsymbol{\mu}_{t-1}^{(l)}) \right\|_2^2}_{(a)} + \underbrace{(\sigma_{init}^{(l)})^2 \left\| \left(\frac{\boldsymbol{\mu}_{t-1}^{(l)}}{\boldsymbol{\sigma}_{t-1}^{(l)}} \right)^2 \odot (\boldsymbol{\mu}_t^{(l)} - \boldsymbol{\mu}_{t-1}^{(l)}) \right\|_1}_{(b)} \\ & + \underbrace{\frac{\beta}{2} \mathbf{1}^T \left(\left(\frac{\boldsymbol{\sigma}_t^{(l)}}{\boldsymbol{\sigma}_{t-1}^{(l)}} \right)^2 - \log \left(\frac{\boldsymbol{\sigma}_t^{(l)}}{\boldsymbol{\sigma}_{t-1}^{(l)}} \right)^2 + (\boldsymbol{\sigma}_t^{(l)})^2 - \log(\boldsymbol{\sigma}_t^{(l)})^2 \right)}_{(c)} \Bigg]\end{aligned} \tag{3.14}$$

where L is the number of layers, $\boldsymbol{\mu}_t^{(l)}, \boldsymbol{\sigma}_t^{(l)}$ denote the mean and standard deviation of weights in the l-th layer for task t. $\odot$ and division are element-wise product and division, respectively. The term (a) is to regularize changes on weights which connect to a node with small variance, where $\Lambda_{i,j}^{(l)} \triangleq \max\left(\frac{\boldsymbol{\sigma}_{init}^{(l)}}{\boldsymbol{\sigma}_{t-1,i}^{(l)}}, \frac{\boldsymbol{\sigma}_{init}^{(l-1)}}{\boldsymbol{\sigma}_{t-1,j}^{(l-1)}} \right)$, $\boldsymbol{\sigma}_{init}^{(l)}$ is the initial variance of l-th layer. The term (b) is to regularize changes on weights with large signal-to-noise ratio which are usually considered important in pruning techniques. The term (c) keeps $\boldsymbol{\sigma}_t^{(l)}$ close to $\sqrt{2}\boldsymbol{\sigma}_{t-1}^{(l)}$, which can increase the uncertainty of a node and hence keep the plasticity of the model by gradually forgetting. β is the hyperparameter for controlling the speed of forgetting. UCL introduces more flexible regularizations and reduces the memory cost by just storing node-wise uncertainty.

Adaptive Group Sparsity based Continual Learning (AGS-CL) (Jung et al., 2020) is another regularization-based method using node-wise importance but for non-Bayesian neural networks. The loss function of AGS-CL is as follows:

$$\mathscr{L}_{\text{AGS-CL}} = \mathscr{L}_t(\boldsymbol{\theta}) + \underbrace{\mu \sum_{l=1}^{L} \sum_{n_l \in \mathscr{G}_0^{t-1}} \|\boldsymbol{\theta}_{n_l}\|_2}_{\text{(a)}} + \underbrace{\lambda \sum_{l=1}^{L} \sum_{n_l \in \mathscr{G} \setminus \mathscr{G}_0^{t-1}} \Omega_{n_l}^{t-1} \|\boldsymbol{\theta}_{n_l} - \hat{\boldsymbol{\theta}}_{n_l}^{(t-1)}\|_2}_{\text{(b)}} \tag{3.15}$$

where n_l is a node in l-th layer, $\boldsymbol{\theta}_{n_l}$ is the vector of incoming weights of node n_l, $\mathscr{G}$ is the set of all nodes in the model and $\mathscr{G}_0^{t-1}$ is the set of unimportant nodes estimated at task $t-1$,

$\Omega_{n_l}^{t-1}$ represents the importance of node n_l at task $t-1$ which is the moving average of the activation output of that node. The unimportant nodes are identified as $\mathcal{G}_0^{t-1} \triangleq \{n_l : \Omega_{n_l}^{t-1} = 0\}$. The term (a) introduces a group lasso penalty on unimportant nodes which attempts to preserve plasticity for new tasks. The term (b) penalizes deviation from important nodes and hence preserves performance on previous tasks. In addition, AGS-CL proposes a strategy to re-initialize connected weights of an unimportant node: a) fix the outgoing weights of the node to be 0, which prevent backward propagation to upper layers from this node and hence reduce interference caused by future tasks; b) re-initialize a part of incoming weights which are randomly chosen with a probability ρ so that the plasticity of the model can be improved by re-activating some unimportant nodes for future tasks.

Summary of regularization-based approaches

As we can see, regardless whether a regularization-based approach is designed for non-Bayesian or Bayesian neural networks, the key is to provide effective measures for evaluating the weight importance, e.g. the Fisher information, the uncertainty of a weight, the magnitude of a weight, the importance of its connected node, etc.. Most of these methods have a common requirement during training: awareness of task boundaries, because the weight or node importance for regularization needs to be consolidated when the training on one task is done. This requirement increases difficulties of regularization-based methods in some application scenarios, for instance, in online training without clear task boundaries an extra strategy for detecting and deciding task boundaries would be required.

3.3.2 Architecture-based Approaches

Applying regularization-based approaches, the model capacity can become a main problem when all tasks share all the parameters of the model (i.e., the model capacity is fixed). In such a case, getting a reasonable trade-off between the stability and plasticity will become more and more difficult while the number of tasks increases because more weights have become important for learned tasks and less weights can be altered for a new task. Architecture-based approaches in continual learning solve this issue by adding task-specific components to the model for each task, hence the model capacity can increase along with the number of tasks. Consequently, the model consists of task-shared and task-specific components. However, the model capacity cannot be expanded freely in most cases due to the limitation of available resources. To constrain the expansion of the model, most architecture-based approaches aim to minimize the task-specific components and enhancing the generalization ability of task-shared components. In addition, to select the task-specific component for an input requires the task identifier of that input, which restricts the application scenarios of architecture-based methods. We will introduce several representative methods of this category in this section.

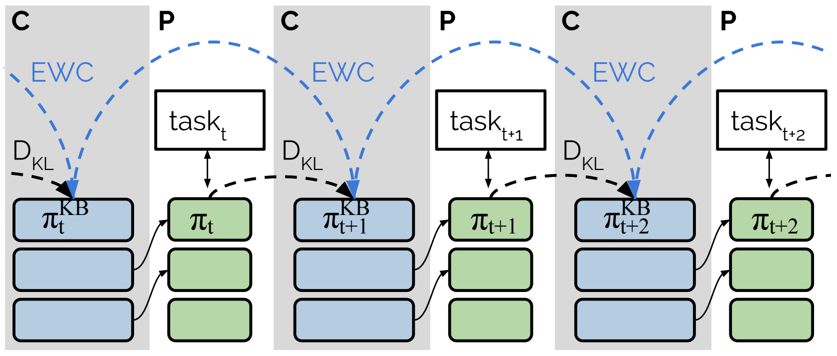


Figure 3.4: Illustration of the architecture of P&C framework, the figure is reproduced from Schwarz et al. (2018). The grey plates denote *compress* phases (C), the white plates denote *progress* phases (P). π_t^{KB} means outputs of the knowledge base, π_t means outputs of the active column. The previously learned parameters of the knowledge base are preserved by EWC, the newly learned parameters are imported to the knowledge base by knowledge distillation.

Increasing the model capacity by expansion and compression

Schwarz et al. (2018) proposed a Progress & Compress (P&C) framework for continual learning that consists of two components: a) a knowledge base for preserving knowledge of learned tasks and transferring knowledge to a new task; b) an active column for learning the new task. The architecture of P&C framework is illustrated in Fig. 3.4. The two components are learned in an interleaved manner. The active column is optimized during the *Progress* phase while the knowledge base is fixed; the knowledge base is updated during the *Compress* phase while the active column remains unchanged.

As shown in Fig. 3.4, the active column receives input from the knowledge base as an explicit forward transfer:

$$\boldsymbol{h}_i = \sigma(W_i \boldsymbol{h}_{i-1} + \boldsymbol{\alpha}_i \odot U_i \sigma(V_i \boldsymbol{h}_{i-1}^{KB} + \mathbf{c}_i) + \mathbf{b}_i) \tag{3.16}$$

where $\boldsymbol{h}_i$ is the output of i-th layer in the active column, W_i, U_i, V_i are weight matrices, $\mathbf{b}_i, \mathbf{c}_i$ are biases, σ denotes the activation function, $\odot$ means element-wise product, $\boldsymbol{\alpha}_i$ is a trainable vector for adapting the strength from the knowledge base. After completing a *Progress* phase, the knowledge base will be updated by: 1) a knowledge distillation objective for consolidating the knowledge of the latest task; 2) a regularization term for preserving knowledge of old tasks.

$$\mathscr{L}_{\text{P\&C}}^{\text{KB}}(\boldsymbol{\theta}) = \mathbb{E}_{p(\mathbf{x})}[KL(\pi_t(\cdot|\mathbf{x})||\pi_{t+1}^{KB}(\cdot|\mathbf{x}))] + \mathscr{R}(\boldsymbol{\theta}_{t+1}^{KB}, \boldsymbol{\theta}_t^{KB}) \tag{3.17}$$

where $\pi_t(\cdot)$ and $\pi^{KB}(\cdot)$ are the outputs of the active column and the knowledge base respectively, $\mathscr{R}(\boldsymbol{\theta}_{t+1}^{KB}, \boldsymbol{\theta}_t^{KB})$ is the regularization term derived from EWC (Eq. (3.6)) or Online EWC (Eq. (3.7)). As an early example of architecture-based approaches, the P&C framework demonstrates a feasible way to apply regularization-based methods into architecture-based methods. However, the progression expands the model by a fixed size which grows the model linearly with the number of tasks. Some following work address these issues in a more flexible way which we will introduce below.

Dynamically Expandable Networks (DEN) (Yoon et al., 2018) tries to dynamically increase the model capacity by three steps:

1) *Selective retraining* – in this step, DEN trains each layer of the model from top to bottom (output to input) by a regular loss of the current task with L1 regularization, while training one layer, the lower layers are fixed to optimal parameters of the last task. After this round of training, the non-zero weights are selected to be retrained for the current task again. This step forces the sparsity of the model and identifies important weights for the current task.
2) *Dynamic network expansion* – If after the first step the loss of the current task is larger than a threshold, then add k new neurons to each layer and optimize the new parameters with group sparsity regularization where the incoming weights of a neuron compose a group. After training, unnecessary new neurons will be removed. This step increases the model capacity when it is necessary. In addition, the task index of adding a group of parameters is kept and used at inference time for selecting task-specific weights, which is equivalent to a task-specific component.
3) *Network split/duplication* – If after the second step a neuron has drifted too much from its previous value (measured by the L2 distance of its incoming weights), a duplication of it will be added into the model and then the model needs to be retrained with a L2 regularization term for preserving previous weights.

DEN made the attempt to dynamically expand the model when it is necessary, however, the procedure is quite complex and requires several runs of full training. More importantly, the connections between the three steps are rather loose, for example, the important weights identified at the first step are of no use for the later two steps and the model expansion is split into two steps which is a bit cumbersome.

Zhang et al. (2020b) proposed a clearer process – Regularize, Expand and Compress (REC) – for continual learning, which consists of two steps:

1) *Expansion* – in this step, techniques of Neural Architecture Search (NAS) are employed to expand the network for accommodating a new task, meanwhile, regularization-based methods can be used to prevent changes on important weights for previous tasks. Zhang et al. (2020b) also proposed a regularization-based method, Regularized Weight Consolidation (RWC), which adds a $L_{2,1}$-norm (the sum of L2 norm of each column) regularization term in addition to the EWC regularization for capturing layer-wise important parameters, and also adds a L_1 regularization to enforce sparsity in newly joined parameters. Using

NAS approaches for continual learning also has been proposed in Li et al. (2019) which is followed by a fine-tuning process instead of compression (step 2).

2) *Compression* – after growing the model size in the expansion phase, techniques of network compression are employed to reduce the model size in this step. REC also uses a knowledge distillation method to compress the model to the initial size which keeps the model non-expansive. The exact non-expansive restriction may not be necessary in which case a dynamic expansion process can be adopted instead.

The two steps of REC compose a general process for continual learning that can depoly various approaches of NAS and model compression in each step according to the needs of application scenarios. Nontheless, NAS is computationally expensive and REC does not gain significant improvement on performance comparing with standalone RWC. Moreover, using knowledge distillation for compression may not perform well when there is no access to previous data, as the training of student network is only based on data of current task.

Preventing forgetting by task-specific masks

Instead of adding new units to expand the model, several methods propose to learn task-specific masks that can force each task to only use partial units of the network. Hard Attention to the Task (HAT) (Mallya et al., 2018) learns task-specific attentions on hidden units of each layer.

$$\boldsymbol{h}_l^t = \boldsymbol{a}_l^t \odot \boldsymbol{h}_l, \;\; \boldsymbol{a}_l^t = \sigma(s\mathbf{e}_l^t) \tag{3.18}$$

where $\boldsymbol{h}_l$ is the output of l-th layer, $\boldsymbol{a}_l^t$ is learned attentions on $\boldsymbol{h}_l$ for task t and $\boldsymbol{h}_l^t$ is the task adaptive output, $\sigma(\cdot) \in [0,1]$ is a gate function (e.g. sigmoid function), s is a positive scalar, $\mathbf{e}_l^t$ is the learned task embedding of the l-th layer. To prevent performance degradation on previous tasks, HAT apply an inverse attention on the gradient by a cumulative attention vector $\boldsymbol{a}_l^{\le t}$ which is the maximum attention among task 1 to t:

$$\hat{\boldsymbol{g}}_{l,ij}^t = \left[1 - \min\left(\boldsymbol{a}_{l,i}^{\le t}, \boldsymbol{a}_{l-1,j}^{\le t}\right)\right] \boldsymbol{g}_{l,ij}, \;\; \boldsymbol{a}_l^{\le t} = \max\left(\boldsymbol{a}_l^t, \boldsymbol{a}_l^{\le t-1}\right) \tag{3.19}$$

This inverse attention is used to reduce the changes on parameters associated with high attentions. In addition, HAT introduces sparsity of the task-specific component by a L1 regularization on attentions as well.

Besides these non-Bayesian methods, Adel et al. (2020) proposed a Bayesian approach, Continual Learning with Adaptive Weights (CLAW), that learns binary attentions on hidden units. In CLAW the attention is defined as a random variable for selecting task-specific weights:

$$\boldsymbol{h}_l^t = (1 + b^t \boldsymbol{\alpha}_l^t) \odot \boldsymbol{h}_l, \;\; \boldsymbol{\alpha}_{l,i}^t \sim \mathscr{B}(p_{l,i}) \tag{3.20}$$

where b^t is a variable controlling the strength of attention vectors, $\mathscr{B}(p_{l,i})$ is approximated by a Normal distribution $\mathcal{N}(p_{l,i}, p_{l,i}(1-p_{l,i}))$ that can leverage the reparameterization trick (Sec. 2.1.1) in SVI.

According to Eq. (3.20), CLAW actually learns an additive decomposition of task-shared and task-specific neurons for each task. Analogously, a recent work called Additive Parameter Decomposition (APD) (Yoon et al., 2020) proposed a hierarchical additive decomposition for learning sparse task-specific weight matrices. It firstly decomposes the model parameters to two weight matrices: $\boldsymbol{\theta}_t = \boldsymbol{\sigma} \odot \mathscr{M}_t + \boldsymbol{\tau}_t$, where $\boldsymbol{\sigma}$ is a task-shared weight matrix and $\boldsymbol{\tau}_t$ is a task-adaptive weight matrix, $\mathscr{M}_t$ is a weight mask matrix used for adopting weights of the task-shared matrix. The loss function of APD is as follows:

$$\mathscr{L}_{\text{APD}}(\boldsymbol{\theta}_t) = \mathscr{L}_t(\boldsymbol{\theta}_t) + \lambda_1 \|\boldsymbol{\tau}_t\|_1 + \lambda_2 \|\boldsymbol{\sigma} - \boldsymbol{\sigma}^{(t-1)}\|_2^2 \tag{3.21}$$

Similar to other architecture-based approaches, the L1 regularization of $\boldsymbol{\tau}_t$ enforces sparsity of task-adaptive weights which minimizes increased model size for each task. As usual, the L2 regularization minimizes the changes on the task-shared weights to prevent performance degradation on previous tasks. Furthermore, APD introduces a new process, hierarchical knowledge consolidation, to further decompose the task-adaptive matrix $\boldsymbol{\tau}_t$ by a local-shared matrix and a sparser task-adaptive matrix, which leverages the hierarchical shared information across tasks by clustering task-adaptive matrices. This process further restricts the model expansion for each task and then reduces memory cost.

Furthermore, Wen et al. (2020) introduced another decomposition of the weight matrix for continual learning, in which the task-specific weight matrix is a rank-one matrix that can be decomposed as the product of two vectors.

$$W_t = W \odot F_t, \quad F_t = \mathbf{r}_t \mathbf{s}_t^T \tag{3.22}$$

Here, W is the task-shared weight matrix that will be learned at the first task; F_t is the task-specific weight matrix that will be learned through $\mathbf{r}_t$ and $\mathbf{s}_t$ for each task. This method gives another option to reduce the incremental memory cost besides enforcing sparsity into the task-specific weight matrix. However, the generalization ability of the shared weights solely depends on the expressiveness of the first task, which could be highly limited for later tasks. And the rank-one task-spesific matrix could be less capable of capturing representations of complex tasks.

Summary of architecture-based approaches

The architecture-based approaches are very similar to many methods in Multi-task learning (Ruder, 2017), which also have task-shared and task-specific components. The main difference is that in multi-task learning a model has access to training sets of all tasks and it mainly attempts to learn more general representations through task-shared components and hence improve performance on every individual tasks. The architecture-based approaches in continual learning focus on efficiently expanding the model capacity when it is necessary for accommodating a new task. Methods in this category often rely on adding task-specific components and require keeping the model architecture of previous tasks unchanged. Hence, conditioning on task identifiers during training and testing time is usually mandatory. As a result, the architecture-based approaches are generally more computationally expensive than regularization-based methods and less flexible than replay-based methods in terms of application scenarios. For real applications, architecture-based methods may become more practical if their requirement of task identifiers is eliminated by working with replay-based methods, which could be a promising avenue for future work.

3.3.3 Replay-based Approaches

Like humans having memories, replay-based methods allow limited access to data of previously learned tasks as episodic memories or having a generator to simulate data from previous tasks. We will introduce generative replay in Sec. 3.4 and focus on replaying episodic memories in this section. The replayed-based methods aim to efficiently utilize samples from the memories to prevent forgetting and facilitate knowledge transfer while learning a new task, which can be realized by different sampling strategies, refining gradients, and enforcing knowledge distillation using the memorized samples. The main restriction of replay-based methods is the memory size, which is often limited to be a small portion of the training set. The memory may or may not be allowed to grow with the number of seen tasks depending on the application requirement. In general, replay-based methods prefer more information preserved in a smaller memory which leads to another important direction of research work in this category. Replay-based approaches are feasible for most application scenarios and are usually more robust than other types of approaches when using task-agnostic models.

Sampling strategies for experience replay in the online setting

The most straightforward replay-based approaches explore various sampling strategies for replaying memorized samples, such as Experience Replay (ER) (Chaudhry et al., 2019b), Class-Balancing Reservoir Sampling (CBRS)(Chrysakis & Moens, 2020), and Maximally Interfered Retrieval (MIR) (Aljundi et al., 2019a). One advantage of these methods is the capability of handling online training. There are two memory-update strategies often used with these sampling strategies in the setting of online continual learning:

1) **Reservoir sampling** (Vitter, 1985), which selects each incoming data sample with a probability $\frac{|\mathcal{M}|}{n}$ into the memory buffer, where $|\mathcal{M}|$ is the memory size and n is the number of observed data samples so far. This method is often used to sample a subset from a data stream with unknown length.
2) **Ring buffer**, which allocates the memory buffer equally to each seen class and selects data samples into the memory by a First In First Out (FIFO) strategy.

Chaudhry et al. (2019b) provided an empirical study of these strategies. According to the experimental results, Reservoir sampling performs better when the memory is not too small (at least more than a few samples per class) and a hybrid strategy which switches Reservoir sampling to ring buffer when memory size of each class is lower than a threshold n_c performs constantly well (i.e., guarantees at least n_c samples per class). Chaudhry et al. (2019b) also suggeted a simple sampling strategy called Experience Replay (ER) for composing the training batch (the batch used for one gradient update). ER divides the training batch equally by samples from the episodic memory and samples from a current task where samples from the memory are randomly sampled. According to their experimental results, ER can outperform some SOTA methods, such as EWC (Serra et al., 2018) and Averaged-Gradient Episodic Memory (GEM) (A-GEM) (Chaudhry et al., 2019a) with a small size of the memory (e.g. 10 samples per class).

CBRS (Chrysakis & Moens, 2020) includes a memory-update strategy that is similar to the hybrid strategy suggested in Chaudhry et al. (2019b) and capable of handling imbalanced data. It keeps the memory equally divided by seen classes and selects samples for each class by Reservoir sampling. Note that the memory may not be exactly equally divided when there are no enough samples encountered for some classes. Chrysakis & Moens (2020) also suggested to form the training batch as the same as in ER but use a weighted loss to combine the loss computed by new samples and memorized samples, i.e.:

$$\mathscr{L}_{\text{CBRS}} = \alpha\mathscr{L}_{\text{new}} + (1-\alpha)\mathscr{L}_{\text{mem}}, \quad \alpha = 1/n_c, \quad n_c \text{ is the number of seen classes.} \tag{3.23}$$

The combined loss generally put more weight on memorized samples in comparison with the loss of ER and has shown better performance on forgetting than ER.

Besides uniformly randomly sampling from the memory buffer, Aljundi et al. (2019a) proposed Maximally Interfered Retrieval (MIR) that selects top-k samples in the memory which would get largest increment of the loss if the model parameters are updated by the batch of new samples. The criterion is defined as follows:

$$s_{\text{MIR}}(\mathbf{x},\mathbf{y}) = \ell(f_{\boldsymbol{\theta}^v}(\mathbf{x}),\mathbf{y}) - \ell(f_{\boldsymbol{\theta}}(\mathbf{x}),\mathbf{y}), \ \ \boldsymbol{\theta}^v = \boldsymbol{\theta} - \alpha\nabla\mathscr{L}(f_{\boldsymbol{\theta}}(X_t),Y_t) \tag{3.24}$$

where (x,y) is a sample from the memory, (X_t,Y_t) is the batch of new samples, $\boldsymbol{\theta}$ is the current parameters, α is the learning rate, ℓ and $\mathscr{L}$ are the loss of one sample and one batch of samples respectively. The intuition is to choose the samples of previous tasks that would be most interfered (whose performance is most negatively impacted) by new samples into the training batch and thus prevent their performance from further degrading. Applying MIR to ER has shown significant improvement on ER in terms of average accuracy and forgetting, which indicates it is effective in alleviating forgetting without sacrificing the performance on intransigence.

Utilizing gradients produced by the episodic memory

Another line of replay-based approaches focuses on utilising the gradients produced by samples from the episodic memory. More specifically, those gradients are used to confine the gradients produced by new samples. For example, Gradient Episodic Memory (GEM) (Lopez-Paz & Ranzato, 2017), Averaged-GEM (A-GEM) (Chaudhry et al., 2019a), Orthogonal Gradient Descent (OGD)

(Farajtabar et al., 2020) are all in this line. The basic idea of such gradient-based methods is to reduce the interference on old tasks by forcing the inner product of the two gradients to be non-negative:

$$\langle \boldsymbol{g}_t, \boldsymbol{g}_k \rangle = \left\langle \frac{\partial \mathscr{L}(\mathbf{x}_t, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}, \frac{\partial \mathscr{L}(\mathbf{x}_k, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right\rangle \geq 0, \;\; \forall k < t \tag{3.25}$$

where t and k are time indices, $\mathbf{x}_t$ denotes a new sample from the current task, and $\mathbf{x}_k$ denotes a sample from the episodic memory. Thus, the updates of parameters are forced to preserve the performance on previous tasks as much as possible. In GEM (Lopez-Paz & Ranzato, 2017), g_t is projected to a direction that is closest to it in L_2-norm whilst also satisfying Eq. (3.25):

$$g_{\text{GEM}} = \arg\min_{\tilde{\boldsymbol{g}}} \frac{1}{2} ||\boldsymbol{g}_t - \tilde{\boldsymbol{g}}||_2^2, \;\; s.t. \langle \tilde{\boldsymbol{g}}, \boldsymbol{g}_k \rangle \geq 0, \;\; \forall k < t \tag{3.26}$$

Optimization of this objective requires a high-dimensional quadratic program and thus is computationally expensive. A-GEM (Chaudhry et al., 2019a) alleviates the computational burden of GEM by using the averaged gradient over a batch of samples from the episodic memory instead of individual gradients of samples from the memory.

$$g_{\text{A-GEM}} = \arg\min_{\tilde{\boldsymbol{g}}} \frac{1}{2} ||\boldsymbol{g} - \tilde{\boldsymbol{g}}||_2^2, \;\; s.t. \;\; \langle \tilde{\boldsymbol{g}}, \boldsymbol{g}ref \rangle \geq 0, \tag{3.27}$$

where $\boldsymbol{g}_{ref}$ is the average gradients produced by a batch of memorized samples. It can be solved by $\tilde{\boldsymbol{g}} = g - \frac{\boldsymbol{g}^T \boldsymbol{g}_{ref}}{\boldsymbol{g}_{ref}^T \boldsymbol{g}_{ref}}$ which only involves computing inner products of gradients. This not only simplifies the computation, but also obtains comparable performance with GEM.

Guo et al. (2020) proposed two adaptive schemes (MEGA-I & MEGA-II) based on A-GEM in order to utilize the loss information during training for a better balanced loss between old and new tasks.

$$\begin{aligned} &\boldsymbol{\theta}_{k+1}^t \leftarrow \boldsymbol{\theta}_k^t - \eta \left(\alpha_1(\boldsymbol{\theta}_k^t) \nabla \ell_t(\boldsymbol{\theta}_k^t) + \alpha_2(\boldsymbol{\theta}_k^t) \nabla \ell_{ref}(\boldsymbol{\theta}_k^t) \right), \\ &\text{MEGA-I}: \begin{cases} \alpha_1(\boldsymbol{\theta}) = 1, \;\; \alpha_2(\boldsymbol{\theta}) = \ell_{ref}(\boldsymbol{\theta})/\ell_t(\boldsymbol{\theta}), \;\; \text{if} \;\; \ell_t(\boldsymbol{\theta}) > \epsilon; \\ \alpha_1(\boldsymbol{\theta}) = 0, \;\; \alpha_2(\boldsymbol{\theta}) = 1, \;\; \text{if} \;\; \ell_t(\boldsymbol{\theta}) \leq \epsilon. \end{cases} \end{aligned} \tag{3.28}$$

where ℓ_t and ℓ_{ref} are the expectation of loss over new samples and memorized samples, respectively. MEGA-I adapts the weights of ℓ_t and ℓ_{ref} dynamically by preventing overfitting on new samples with a loss threshold. MEGA-II tries to find a direction of gradients that maximize:

$$\text{MEGA-II}: \;\; \max_{\tilde{\boldsymbol{g}}} \;\; \ell_t(\boldsymbol{\theta}) \frac{\langle \tilde{\boldsymbol{g}}, \boldsymbol{g}_t \rangle}{||\tilde{\boldsymbol{g}}||_2 \cdot ||\boldsymbol{g}_t||_2} + \ell_{ref}(\boldsymbol{\theta}) \frac{\langle \tilde{\boldsymbol{g}}, \boldsymbol{g}_{ref} \rangle}{||\tilde{\boldsymbol{g}}||_2 \cdot ||\boldsymbol{g}_{ref}||_2} \tag{3.29}$$

which adaptively chooses a direction of gradients that closer to the direction obtained by the larger loss among $\ell_t(\boldsymbol{\theta})$ and $\ell_{ref}(\boldsymbol{\theta})$. Both schemes have shown improvements compared with A-GEM by similar computational cost.

Orthogonal Gradient Descent (OGD) (Farajtabar et al., 2020) instead projects the gradient of new samples to the direction that is orthogonal to gradients of samples from previous tasks, which attempts to keep the model performance of previous tasks unchanged while learning a new task.

$$\tilde{g} \perp \nabla f_j(\mathbf{x};\boldsymbol{\theta}_k^*), \;\; \forall \mathbf{x} \in \mathcal{M}_k, \;\; j \in \{1 \ldots c\}, \;\; k \leq t, \tag{3.30}$$

where f_j is the j-th logit that corresponds to the true class of $\mathbf{x}$, c is the number of classes, k is the task index, $\mathcal{M}_k$ and $\boldsymbol{\theta}_k^*$ is the episodic memory and optimized parameters for the k-th task respectively. Although this method shows higher accuracy than A-GEM, it incurs higher memory cost as it needs to store gradients $\nabla f_j(\mathbf{x};\boldsymbol{\theta}_k^*)$ of all learned tasks which could easily be prohibitive when the model is large. In addition, OGD also requires the knowledge of task boundaries since it needs to save the optimized gradients when finishing a task.

Chaudhry et al. (2020) proposed ORTHOG-SUBSPACE for learning orthonomal weight matrices to obtain orthogonal subspaces of learned tasks. This method first ensures the gradients of current and previous tasks are orthogonal by constructing a projection matrix P_t for each task that satisfies:

$$P_t^T P_t = I, \;\; P_t^T P_k = 0, \;\; \forall k \neq t, \tag{3.31}$$

where t and k are task indices; and then projects the output of the final hidden layer (the representation to the linear layer) as $\phi_t = P_t h_L$. In this way $g_L^t \perp g_L^{k \neq t}$ and orthonormal weight matrices can preserve the inner product of gradients in each layer. Then the objective becomes learning orthonormal weight matrices of each layer which is an optimization problem over Stiefel Manifold (Bonnabel, 2013) and can be solved iteratively using the Cayley transform (Li et al., 2020). For each layer $l \in \{1,\ldots,L\}$ the weights is updated by the following steps:

$$\begin{aligned}
&A = g_l (W_l^k)^T - W_l^k g_l^T, \;\; U = A W_l^k, \\
&Y^0 = W_l^k - \tau U, \;\; \tau = \min(\alpha, 2q/(\|W_l^k\| + \epsilon)), \\
&Y^i = W_l^k - \frac{\tau}{2} A (W_l^k + Y^{i-1}), \;\; i \in \{1,\ldots,s\}, \\
&W_l^{k+1} = Y^s
\end{aligned} \tag{3.32}$$

where τ is the adaptive learning rate and α, q, ϵ, s are hyperparameters. This method achieves a similar goal as OGD without storing gradients of previous tasks. Although there are no experimental results comparing these two methods directly, ORTHOG-SUBSPACE might perform better as it shows larger improvements over A-GEM.

Another way to adjust the gradient direction is updating the parameters by a meta-learning style. Meta-Experience Replay (MER) (Riemer et al., 2019) is such a method using multiple batches to generate one-step update as Reptile (Nichol et al., 2018) which first optimizes across s batches sequentially with SGD and learning rate α and then takes a final update with a learning rate β after training on all batches: $\boldsymbol{\theta}_0 = \boldsymbol{\theta}_0 + \beta(\boldsymbol{\theta}_s - \boldsymbol{\theta}_0)$. MER further serializes the updates within each batch which approximately optimizes:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{x}_{ij},\mathbf{y}_{ij} \sim \mathcal{D}_t \cup \mathcal{M}_{t-1}} \left[2 \sum_{i=1}^{s} \sum_{j=1}^{b} [\mathcal{L}(\mathbf{x}_{ij},\mathbf{y}_{ij}) - \sum_{q=1}^{i-1} \sum_{r=1}^{j-1} \alpha \langle \frac{\partial \mathcal{L}(\mathbf{x}_{ij},\mathbf{y}_{ij})}{\partial \boldsymbol{\theta}}, \frac{\partial \mathcal{L}(\mathbf{x}_{qr},\mathbf{y}_{qr})}{\partial \boldsymbol{\theta}} \rangle] \right] \tag{3.33}$$

where b is batch size and s is number of batches, α is the learning rate applied to updates within each batch.

In addition, Aljundi et al. (2019b) propose Gradient-based Sample Selection (GSS), which selects samples that produce most diverse gradients with other samples into the episodic memory. Here the diversity is measured by the cosine similarity between gradients. Since the cosine similarity is computed using the inner product of two normalized gradients, GSS embodies the same principle as other gradient-based approaches introduced above.

$$\begin{aligned} \widehat{\mathcal{M}_i} &= \operatorname*{arg\,min}_{\mathcal{M}} \sum_{n,m\in\mathcal{M}} \frac{\langle g_n, g_m\rangle}{||g_n||\cdot||g_m||}, \\ &s.t. \;\; \mathcal{M} \subset (\widehat{\mathcal{M}_{i-1}} \cup \mathcal{B}_i), \;\; |\mathcal{M}| = M \end{aligned} \tag{3.34}$$

Here M is the memory size, i is the iteration index, and $\mathcal{B}_i$ is the i-th batch of the current task. However, due to the limited size of episodic memories, GSS faces difficulties to construct memories that are representative enough for all seen tasks. We will have a further discussion about this in Chapter 5.

The gradient-based approaches require at least $O(W)$ computational complexity in addition to the usual gradient back-propagation, where W is the number of model parameters, which is much more expensive than other replay-based methods when the model is large. And the obtained performance is not significantly better compared with other replay-based methods. We make the connection between the diversity of gradients and the discriminativeness of representations in Chapter 5, which leads to a more efficient way to reduce the gradient diversity across tasks.

Knowledge distillation using the episodic memory

Another way to utilize the episodic memory is knowledge distillation, a technique was originally proposed for knowledge compression and transferring (Hinton et al., 2015). Li & Hoiem (2017) introduced Learning without Forgetting (LwF), a method adding the loss of knowledge distillation into continual learning by feeding new samples to the previous model:

$$\begin{aligned} &\mathcal{L}_{\text{LwF}} = \lambda_o \mathcal{L}_{ce}(\hat{Y}_o, \tilde{Y}_o) + \mathcal{L}_{ce}(Y_n, \tilde{Y}_n) + \mathcal{R}, \\ &\hat{Y}_o = S(f(X_n; \boldsymbol{\theta}_s, \boldsymbol{\theta}_o)), \;\; \tilde{Y}_o = S(f(X_n; \hat{\boldsymbol{\theta}}_s, \hat{\boldsymbol{\theta}}_o)), \;\; \tilde{Y}_n = S(f(X_n; \hat{\boldsymbol{\theta}}_s, \hat{\boldsymbol{\theta}}_n)), \end{aligned} \tag{3.35}$$

where $\mathcal{L}_{ce}$ is the cross entropy loss, $S(\cdot)$ is the softmax function, $\mathcal{R}$ is the regularization loss, f represents the model function, $\boldsymbol{\theta}_s$ is the shared parameter learned by the previous task, $\boldsymbol{\theta}_o$ is the specific parameter of old tasks, $\hat{\boldsymbol{\theta}}_s$ and $\hat{\boldsymbol{\theta}}_o$ are the updated $\boldsymbol{\theta}_s$ and $\boldsymbol{\theta}_o$ by the current task, and $\hat{\boldsymbol{\theta}}_n$ denotes new parameters added for the current task. X_n, Y_n represents new samples and their true labels of the current task. LwF is not a typical replay-based method as it does not replay samples of old tasks. The loss of LwF is more like adding a regularization term by the knowledge distillation loss. However, the knowledge distillation loss is easily adapted to experience replay as was done in Incremental Classifier and Representation Learning (iCaRL) (Rebuffi et al., 2017):

$$\begin{aligned} &\mathscr{L}_{\text{iCaRL}} = \mathscr{L}_{ce}(\hat{Y}_o, \tilde{Y}_o) + \mathscr{L}_{ce}(Y_n, \tilde{Y}_n), \\ &\tilde{Y}_o = S(f(X_o;\boldsymbol{\theta}_n)),\ \ \tilde{Y}_n = S(f(X_n;\boldsymbol{\theta}_n)),\ \ \hat{Y}_o = S(f(X_o;\boldsymbol{\theta}_o)) \end{aligned} \tag{3.36}$$

where $\boldsymbol{\theta}_o$ and $\boldsymbol{\theta}_n$ are model parameters learned by old and new tasks, respectively. X_o denotes samples (or exemplars) from old tasks and $\hat{Y}_o$ denotes their logits generated by the previous model. In this case, $X_o, \hat{Y}_o$ are stored in the episodic memory and the memory is updated at the end of each task to include newly learned classes.

Buzzega et al. (2020) introduced Dark Experience Replay (DER) which includes an additional L2-distance term for the knowledge distillation as follows:

$$\begin{aligned} &\mathscr{L}_{\text{DER}} = \mathscr{L}_{ce}(Y_t, \tilde{Y}_t) + \alpha \mathbb{E}_{k<t}[\|Z_k - \tilde{Z}_k\|_2^2], \\ &\tilde{Y}_t = S(f(X_t;\boldsymbol{\theta}_t)),\ \ \tilde{Z}_k = f(X_k;\boldsymbol{\theta}_t),\ \ Z_k = f(X_k;\boldsymbol{\theta}_k), \end{aligned} \tag{3.37}$$

where X_t denotes samples of the t-th task and Y_t is the true label of X_t, t and k are task indices, Z_k and $\tilde{Z}_k$ are logits of X_k produced by the model at the k-th task and t-th task, respectively. In this case X_k, Z_k are stored in the episodic memory by a Reservoir sampling strategy while training on the k-th task. DER++ improves DER by an additional loss over memorized samples as usual experience replay:

$$\mathscr{L}_{\text{DER++}} = \mathscr{L}_{ce}(Y_t, \tilde{Y}_t) + \alpha \mathbb{E}_{k<t}[\|Z_k - \tilde{Z}_k\|_2^2] + \beta \mathbb{E}_{k<t}[\mathscr{L}_{ce}(Y_k, \tilde{Y}_k)],\ \ \tilde{Y}_k = S(f(X_k;\boldsymbol{\theta}_t)) \tag{3.38}$$

where Y_k are true labels of X_k which also needs to be stored in the memory. DER++ combines the knowledge distillation and experience replay in a rather simple way yet has shown strong performance in its empirical results.

Summary of replay-based approaches

The replay-based methods have the flexibility to cope with various application scenarios and generally show strong performance with relatively low computational cost in terms of time and RAM usage. The main drawback of replay-based methods is that the memory cost may increase while learning more tasks, otherwise, the allocated memory could be not enough to guarantee a minimum performance on learned tasks. To this end, the memory cost should be considered as a performance measurement for replay-based methods too. We will introduce a weighted combination of measurements including the memory cost in sec. 5.6, which is applied in a continual learning challenge. As a result, the main challenge of replay-based methods is how to make more efficient use of the memory and how to make the memorized information more representative. We will show that this conclusion also aligns with the theoretical analysis of continual learning in Sec. 3.5.

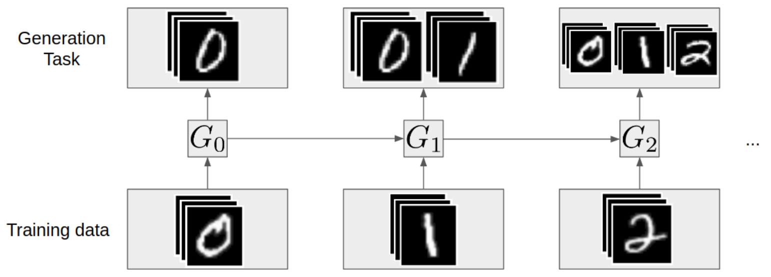


Figure 3.5: Demonstration of the training protocol of generative models in continual learning. At task t the training set consists of samples of category $t-1$ and samples generated by the model at the previous task, and the model is to generate samples from all previously seen categories (figure reproduced from (Lesort et al., 2018)).

3.4 Generative Models in Continual Learning

The previously introduced methods mostly focus on classification tasks in continual learning. In comparison, generative models in continual learning have different training protocols and can be applied in unsupervised learning scenarios, such as representation learning. The main difference in training is that we can draw samples from a generative model trained in previous tasks and use those samples as replay memories of previous tasks, i.e., the generative model itself can work like a replay buffer. A simplified scenario for generative models in continual learning is depicted in Fig. 3.5, where the goal is to learn a generative model for one category (digit) per task while still being able to generate samples of all previous categories. The training dataset of task t consists of real samples of category t and samples of task $1\cdots t-1$ generated by the previous model.

Besides the standalone generative tasks in continual learning, generative models can be applied to replay-based methods as a memory generator for classification tasks as well. For example, Shin et al. (2017) propose using GANs as a generative replay component in classification tasks. Moreover, Wu et al. (2018) show that a generative model can be jointly trained with a classifier as a memory generator.

$$\begin{aligned} &\text{Objective of generator}: \min_{\boldsymbol{\theta}^G}\left(\mathscr{L}^G_{\text{GAN}}(\boldsymbol{\theta}^G,\mathscr{D})+\mathscr{L}^G_{\text{CLS}}(\boldsymbol{\theta}^G,\mathscr{D})\right),\\ &\text{Objective of discriminator and classifier}: \min_{\boldsymbol{\theta}^D,\boldsymbol{\theta}^C}\left(\mathscr{L}^D_{\text{GAN}}(\boldsymbol{\theta}^D,\mathscr{D})+\mathscr{L}^D_{\text{CLS}}(\boldsymbol{\theta}^C,\mathscr{D})\right),\\ &\mathscr{L}^G_{\text{CLS}}(\boldsymbol{\theta}^G,\mathscr{D})=-\mathbb{E}_{z\sim p_z,c\sim p_c}[\mathbf{y}_c\log p_{\boldsymbol{\theta}^C}(G_{\boldsymbol{\theta}^G}(z,c))] \end{aligned} \tag{3.39}$$

Where $\mathscr{L}^G_{\text{GAN}}$ and $\mathscr{L}^D_{\text{GAN}}$ are the conventional loss of the generator and the discriminator of the GAN, respectively. $\boldsymbol{\theta}^G,\boldsymbol{\theta}^D,\boldsymbol{\theta}^C$ are parameters of the generator, discriminator, and the classifier, respectively. $\mathscr{D}$ denotes the training set, $p_z=\mathscr{N}(0,1)$ is the prior of z and $p_c=\mathscr{U}\{1,\dots,K\}$ is the prior of classes. $\mathscr{L}^D_{\text{CLS}}(\boldsymbol{\theta}^C,\mathscr{D})$ is the usual cross-entropy loss of classifiers. $\mathscr{L}^G_{\text{CLS}}(\boldsymbol{\theta}^G,\mathscr{D})$ is a loss w.r.t. $\boldsymbol{\theta}^G$ to maximize the likelihood of generated samples. In addition, the regularization-based methods of continual learning, such as EWC, can also be applied to generative models as demonstrated in (Wu et al., 2018).

When a generative model generates samples with higher fidelity it would be more resistant to forgetting as the replayed samples would be more similar to raw samples. For high-dimensional data using generative replay might be better than using episodic memories as the number of replayed samples is not limited, however, the generative model needs to be able to simulate the true data distribution which is also a challenging task. Lesort et al. (2018) provide a comprehensive study on generative models in continual learning, which focuses on GANs and VAEs. The results show that GANs generate samples of higher quality than VAEs in most experiments. Nonetheless, VAEs can also be applied to continual representation learning as introduced in Rao et al. (2019) since the latent variable space of VAEs can be explicitly learned as representations. In principle, other types of generative models can be applied in continual learning as well, like flow-based generative models (Rezende & Mohamed, 2015; Kingma & Dhariwal, 2018). For Bayesian generative models, the framework of VCL can be used for transferring knowledge through priors, such as for VAEs (Nguyen et al., 2018):

$$\mathscr{L}_{\text{VCL}-\text{VAE}} = \mathbb{E}_{q_t(\boldsymbol{\theta})}[\mathscr{L}_{\text{VAE}}(\boldsymbol{\theta})] - D_{KL}(q_t(\boldsymbol{\theta})|q_{t-1}(\boldsymbol{\theta})) \tag{3.40}$$

Here, $\boldsymbol{\theta}$ denotes parameters of a VAE which encoder and decoder are Bayesian neural networks, $q_t(\boldsymbol{\theta})$ and $q_{t-1}(\boldsymbol{\theta})$ are the posterior of $\boldsymbol{\theta}$ at task t and $t-1$. This framework is general to Bayesian models as long as the posterior of parameters is able to be approximated. We adapt a hierarchical probabilistic model into continual learning by this framework which will be introduced in Chapter 6.

3.5 Theoretical Analysis of Continual Learning

Most existing work in continual learning proposes methods for alleviating forgetting motivated by heuristic or empirical analysis. A few authors have attempted to provide some theoretical guidance for developing approaches in continual learning. In traditional online learning the theoretical bounds are often developed with the assumption that the unseen data are from the same distribution of the seen data or a distribution conditioned on the seen data. In contrast, continual learning pursues the learning ability that is capable of learning from a task sequence without any pre-conditioning. Although we expect the tasks are correlated to some extent but the correlation is not assumed to be foreseen.

Knoblauch et al. (2020) develop a theoretical treatment of continual learning by set theory which makes no assumption about the tasks. It shows that optimal algorithms in continual learning can be interpreted as solving a version of the set intersection decision problem, which is generally NP-hard. Moreover, it concludes that the optimal algorithms must have perfect memories under mild regularities where a perfect memory contains information that can be used to reconstruct at least one optimal solution of all seen tasks. These theoretical findings lead to an explanation of the better performance of replay-based methods compared with regularization-based methods, where the latter narrows the searching space of an overall optimal solution in a neighborhood of the solution of a previous task. In practice, it is more easily to reconstruct an overall solution by data representations of tasks than parameters, especially for less homogeneous tasks. It also explains why some regularization-based methods (such as LwF) use data to form regularization instead of using parameters. From this point of view, there are two factors that are critical to the efficiency of a continual learning algorithm: 1) how efficiently it constructs a memory that close to the perfect memory; and 2) how efficiently it builds a solution based on the memory and training set that close to the optimal solution for all seen tasks. We follow this lead and propose several approaches which will be introduced in the following chapters. In particular, we propose Gaussian Natural Gradient (GNG) for more efficiently updating parameters in Bayesian continual learning and Stein Gradient-based Episodic Memories (SGEM) for constructing the episodic memory along with posterior approximation in Chapter 4. In Chapter 5 we propose a replay-based method DRL that is more efficient than gradient-based approaches with better performance.

3.6 Summary

In this chapter we introduced three main categories of continual learning approaches and compared the pros and cons of each category regarding different settings of application scenarios. We also made connections between related approaches and gave an overview of specified branches in each category. In addition, we introduced several methods for training generative models continously as they can be complementary to classification tasks. Moreover, we aligned empirical results in those related work with a theoretical analysis of continual learning which have shown the importance of episodic memories for solving catastrophic forgetting.

CHAPTER 4

Natural Gradients and Stein Gradients for Bayesian Continual Learning

In this chapter, we propose the approaches to improve efficiency in Bayesian continual learning. When using Bayesian models in continual learning, knowledge from previous tasks can be retained in two forms: (i) as posterior distributions over the parameters, containing the knowledge of the model gained from previous tasks, which then serve as the priors for the following task; (ii) as episodic memories (referred as coresets in Nguyen et al. (2018)), containing knowledge of data distributions of previous tasks. To cope with both forms, we propose a regularization-based and a replay-based approach for Bayesian continual learning in this chapter, which are built upon the Variational Continual Learning (VCL) framework and utilize the natural gradient and Stein Variational Gradient Descent (SVGD) respectively. We first introduce the regularization-based approach: Gaussian Natural Gradient (GNG), and then introduce the replay-based approach: Stein Gradient-based Episodic Memories (SGEM). Finally, we provide experimental results for both approaches along with the combination of them, which demonstrate that our methods effectively bring improvements based on VCL.

4.1 Gaussian Natural Gradient for Bayesian continual learning

In Chapter 3 we introduced several regularization-based approaches for continual learning, including Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017), Synaptic Intelligence (SI) (Zenke et al., 2017), Variational Continual Learning (VCL) (Nguyen et al., 2018), and Uncertainty-guided Continual Bayesian Neural Network (UCB) (Ebrahimi et al., 2020), which adaptively regularize the parameters when learning a new task by the importance of previous parameters. Those approaches use different ways to measure the importance of previous parameters including non-Bayesian and Bayesian treatments. In particular, VCL is a framework that makes use of Variational Inference (VI) to facilitate continual learning for Bayesian Neural Networks (BNNs). We recall the objective of VCL in Eq. (4.1) as it is the basis of our approach:

$$\mathscr{L}_{\text{VCL}} = \mathbb{E}_{q_t(\boldsymbol{\theta})}[\log p(\mathscr{D}_t|\boldsymbol{\theta})] - D_{KL}(q_t(\boldsymbol{\theta})\|q_{t-1}(\boldsymbol{\theta})). \tag{4.1}$$

where t is the index of tasks and D_t is the training data of task t, $q_t(\boldsymbol{\theta})$ represents the approximated posterior of parameters $\boldsymbol{\theta}$ at task t. We can parameterize $q_t(\boldsymbol{\theta})$ as $q(\boldsymbol{\theta};\boldsymbol{\beta}_t)$ if we assume the posteriors of all tasks are from the same functional family. $q_{t-1}(\boldsymbol{\theta})$ here takes the role as the prior of $q_t(\boldsymbol{\theta})$ as in VI. The regularization term is the KL-divergence between the posteriors of the current and the previous task, which encourages the posterior of the current task being closer to the previous task. VCL shows promising performance compared to EWC (Kirkpatrick et al., 2017) and SI (Zenke et al., 2017), which demonstrates the effectiveness of Bayesian continual learning.

However, conventional gradient methods give the direction of steepest descent of parameters in Euclidean space, which might cause a large difference in terms of distributions following a small change in terms of parameters. We posit that natural gradient methods may be a better choice than the conventional gradient descent. The natural gradient is in the direction of steepest descent in Riemannian space rather than Euclidean space (Pascanu & Bengio, 2014) as it adjusts the vanilla gradient by Fisher information, which would prefer the smallest change in terms of distribution while optimizing some objective function. Prior works (Honkela et al., 2007; Hoffman et al., 2013) have been done for applying the natural gradient in VI in static learning, which have shown effectiveness of the natural gradient for learning Bayesian models. According to Eq. (4.1) we would want to keep the two posteriors $q(\boldsymbol{\theta};\boldsymbol{\beta}_t)$, $q(\boldsymbol{\theta};\boldsymbol{\beta}_{t-1})$ as close as possible while optimizing the likelihood $\mathbb{E}_{q_t(\boldsymbol{\theta})}[\log p(\mathscr{D}_t|\boldsymbol{\theta})]$. The natural gradient could be a more suitable choice than the vanilla gradient for such an objective as it updates parameters with less changes on the posterior. For a better understanding, we will provide more technique details of natural gradient in the next section.

4.1.1 Preliminary: Natural Gradient

Let $\boldsymbol{\beta}$ be the parameter of the posterior of $\boldsymbol{\theta}$ and $\Delta\boldsymbol{\beta}$ denote a small change on $\boldsymbol{\beta}$ that is optimized to satisfy:

$$\Delta\boldsymbol{\beta}^* = \arg\min_{\Delta\boldsymbol{\beta}} \mathscr{L}(\boldsymbol{\beta}+\Delta\boldsymbol{\beta}), \quad s.t. \;\; D_{KL}(q(\boldsymbol{\theta};\boldsymbol{\beta})||q(\boldsymbol{\theta};\boldsymbol{\beta}+\Delta\boldsymbol{\beta})) = const \tag{4.2}$$

Assuming $\Delta\boldsymbol{\beta} \to 0$, the KL-divergence can be expanded and approximated by the second-order Taylor series (Pascanu & Bengio, 2014):

$$\begin{aligned}
&D_{KL}(q(\boldsymbol{\theta};\boldsymbol{\beta})||q(\boldsymbol{\theta};\boldsymbol{\beta}+\Delta\boldsymbol{\beta})) = \mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\beta})}[\log q(\boldsymbol{\theta};\boldsymbol{\beta})] - \mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\beta})}[\log q(\boldsymbol{\theta};\boldsymbol{\beta}+\Delta\boldsymbol{\beta})] \\
&\approx \mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\beta})}[\log q(\boldsymbol{\theta};\boldsymbol{\beta})] - \mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\beta})}[\log q(\boldsymbol{\theta};\boldsymbol{\beta}) + \nabla_{\boldsymbol{\beta}} \log q(\boldsymbol{\theta};\boldsymbol{\beta})\Delta\boldsymbol{\beta} + \frac{1}{2}\Delta\boldsymbol{\beta}^T \nabla^2_{\boldsymbol{\beta}} \log q(\boldsymbol{\theta};\boldsymbol{\beta})\Delta\boldsymbol{\beta}] \\
&= \frac{1}{2}\Delta\boldsymbol{\beta}^T \mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\beta})}[-\nabla^2_{\boldsymbol{\beta}} \log q(\boldsymbol{\theta};\boldsymbol{\beta})]\Delta\boldsymbol{\beta}
\end{aligned} \tag{4.3}$$

Here we use $\mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\beta})}[\nabla_{\boldsymbol{\beta}} \log q(\boldsymbol{\theta};\boldsymbol{\beta})\Delta\boldsymbol{\beta}] = \Delta\boldsymbol{\beta}\mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\beta})}[\nabla_{\boldsymbol{\beta}} \log q(\boldsymbol{\theta};\boldsymbol{\beta})] = \Delta\boldsymbol{\beta}\int \nabla_{\boldsymbol{\beta}} q(\boldsymbol{\theta};\boldsymbol{\beta})d\boldsymbol{\theta} = 0$. In addition,

$$\begin{aligned}
\mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\beta})}[-\nabla^2_{\boldsymbol{\beta}} \log q(\boldsymbol{\theta};\boldsymbol{\beta})] &= \mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\beta})}\left[-\frac{\nabla^2_{\boldsymbol{\beta}} q(\boldsymbol{\theta};\boldsymbol{\beta})}{q(\boldsymbol{\theta};\boldsymbol{\beta})} + \left(\frac{\nabla_{\boldsymbol{\beta}} q(\boldsymbol{\theta};\boldsymbol{\beta})}{q(\boldsymbol{\theta};\boldsymbol{\beta})}\right)^T \left(\frac{\nabla_{\boldsymbol{\beta}} q(\boldsymbol{\theta};\boldsymbol{\beta})}{q(\boldsymbol{\theta};\boldsymbol{\beta})}\right)\right] \\
&= \mathbb{E}_{q(\boldsymbol{\theta};\boldsymbol{\beta})}[\nabla_{\boldsymbol{\beta}} \log q(\boldsymbol{\theta};\boldsymbol{\beta})^T \nabla_{\boldsymbol{\beta}} \log q(\boldsymbol{\theta};\boldsymbol{\beta})] = \boldsymbol{F}_{\boldsymbol{\beta}}
\end{aligned}$$

$\boldsymbol{F}_{\boldsymbol{\beta}}$ is the Fisher information of $\boldsymbol{\beta}$. Applying the first-order Taylor series on $\mathscr{L}(\boldsymbol{\beta}+\Delta\boldsymbol{\beta})$ and using Lagrangian relaxation with a positive scalar λ to solve Eq. (4.2) gives:

$$\Delta\boldsymbol{\beta}^* \approx \underset{\Delta\boldsymbol{\beta}}{\arg\min}\, \mathscr{L}(\boldsymbol{\beta}) + \nabla_{\boldsymbol{\beta}}\mathscr{L}(\boldsymbol{\beta})\Delta\boldsymbol{\beta} + \frac{1}{2}\lambda\Delta\boldsymbol{\beta}^T \boldsymbol{F}_{\boldsymbol{\beta}}\Delta\boldsymbol{\beta}, \quad \lambda > 0. \tag{4.4}$$

Solving Eq. (4.4) by taking the derivative w.r.t $\Delta\boldsymbol{\beta}$, we have $\Delta\boldsymbol{\beta}^* = -\frac{1}{\lambda}\boldsymbol{F}_{\boldsymbol{\beta}}^{-1}\nabla_{\boldsymbol{\beta}}\mathscr{L}(\boldsymbol{\beta})$, which gives the definition of the natural gradient as follows:

$$\hat{\nabla}_{\boldsymbol{\beta}}\mathscr{L}(\boldsymbol{\beta}) \triangleq \boldsymbol{F}_{\boldsymbol{\beta}}^{-1}\nabla_{\boldsymbol{\beta}}\mathscr{L}(\boldsymbol{\beta}), \tag{4.5}$$

In the rest of this chapter, we use $\hat{\nabla}$ to denote the natural gradient. Through the derivation of natural gradient we can see that it is obtained by an extra constraint that minimizes the KL-divergence between $q(\boldsymbol{\theta};\boldsymbol{\beta})$ and $q(\boldsymbol{\theta};\boldsymbol{\beta}+\Delta\boldsymbol{\beta})$, which is more reasonable when we want each update of the parameters leading to less change on their distributions. This property of natural gradient helps with obtaining a more efficient update trajectory of parameters when optimizing the objective of VCL.

4.1.2 Gaussian Natural Gradient and the Adam optimizer

In the simplest (and most common) formulation of BNNs, the weights are drawn from Gaussian distributions with a mean-field factorization which assumes that the weights are independent. Hence, we have an approximate posterior for each weight $q(\theta_i;\mu_i,\sigma_i) = \mathcal{N}(\mu_i,\sigma_i^2)$, where μ_i,σ_i are variational parameters, and their Fisher information has an analytic form:

$$F_{\mu_i} = 1/\sigma_i^2, \quad F_{v_i} = 2, \quad \text{where } v_i = \log\sigma_i. \tag{4.6}$$

Let $q_t(\theta_i) = \mathcal{N}(\mu_{i,t},\sigma_{i,t}^2)$, we have:

$$\mathscr{L}_{\text{VCL}}(\theta_i) = \mathbb{E}_{q_t(\theta_i)}[\log p(\mathscr{D}_t|\theta_i)] - \left(\log\frac{\sigma_{i,t-1}}{\sigma_{i,t}} + \frac{1}{2}\left(\frac{\sigma_{i,t}^2 + (\mu_{i,t}-\mu_{i,t-1})^2}{\sigma_{i,t-1}^2} - 1\right)\right) \tag{4.7}$$

The natural gradient of the mean $\mu_{i,t}$ can be computed as follows:

$$\begin{aligned} &\widehat{g}_{\mu_{i,t}} = \widehat{\nabla}_{\mu_{i,t}} \mathscr{L}_{\text{VCL}}(\theta_i) = \sigma_{i,t}^2 \left(g_i - (\sigma_{i,t-1})^{-2} (\mu_{i,t} - \mu_{i,t-1}) \right), \\ &\text{where } \; g_i \triangleq \nabla_{\theta_i} \mathbb{E}_{q_t(\theta_i)}[\log p(\mathscr{D}_t|\theta_i)], \;\; \theta_i = \mu_{i,t} + \sigma_{i,t}\epsilon_0, \;\; \epsilon_0 \sim \mathscr{N}(0,1) \end{aligned} \tag{4.8}$$

Eq. (4.8) indicates that small $\sigma_{i,t}$ can cause the magnitude of natural gradient to be much reduced. However, BNNs usually need very small variances in initialization to obtain a reasonable performance at prediction time, which brings difficulties of tuning learning rates when applying a vanilla SGD optimizer to this Gaussian Natural Gradient (GNG). As shown in Figs. 4.1c and 4.2c we can see how the scale of variance in initialization changes the magnitude of GNG. To tackle this issue, we consider applying the Adam optimizer (Kingma & Ba, 2014) to GNG. Since F_{v_i} is a constant, its natural gradient is just the Euclidean gradient multiplied by a scalar, which makes no difference in the Adam optimizer. We will explain this below.

The Adam optimizer (Kingma & Ba, 2014) provides a method which update step-size is approximately bounded by the learning rate. It mostly ignores the scale of gradients in update steps, which could compensate the drawback of GNG in BNNs. More precisely, the Adam optimizer updates parameters as follows:

$$\theta_{k+1} \leftarrow \theta_k - \alpha * \widehat{m}_k \,/\, (\sqrt{\widehat{v}_k} + \epsilon), \tag{4.9}$$

where k is index of update steps, $\widehat{m}_k, \widehat{v}_k$ are the moving average of the first and second moment of the gradients, respectively. ϵ is a small number to prevent division from zero. α is the learning rate. We can see that if re-scale the gradients by a scalar c the update step-size is invariant because $\Delta\theta_k \approx \alpha_k(c\widehat{m}_k \,/\, \sqrt{c^2\widehat{v}_k}) = \alpha_k(\widehat{m}_k \,/\, \sqrt{\widehat{v}_k})$, when $\epsilon \to 0$.

Assume $k \to +\infty$, $\widehat{m}_k \approx \mathbb{E}_k[g_k]$ and $\widehat{v}_k \approx \mathbb{E}_k[g_k^T g_k]$, where $\mathbb{E}_k$ indicates the expectation is over k updates. Considering the first and second moments of the natural gradient $\hat{g}_{\mu_i,k}$,

$$\begin{aligned} &\mathbb{E}_k[\hat{g}_{\mu_i,k}] = \mathbb{E}_k[\sigma_{i,k}^2]\mathbb{E}_k[g_{\mu_i,k}] + \text{cov}(\sigma_{i,k}^2, g_{\mu_i,k}) \\ &\mathbb{E}_k[\hat{g}_{\mu_i,k}^2] = (\mathbb{E}_k[\sigma_{i,k}^2]^2 + \text{var}(\sigma_{i,k}^2))\mathbb{E}_k[g_{\mu_i,k}^2] + \text{cov}(\sigma_{i,k}^4, g_{\mu_i,k}^2) \end{aligned} \tag{4.10}$$

We can see that only when $\text{var}(\sigma_{i,k}^2) = 0$ and $g_{\mu_i,k}$ are independent from $\sigma_{i,k}^2$, the updates of GNG are equal to the updates by Euclidean gradients with the Adam optimizer. In addition, larger variance of the gradient will result in smaller updates when applying Adam optimization since $\widehat{v}_k \approx \widehat{m}_k^2 + \text{var}_k(\widehat{g}_{\mu_{i,k}})$.

Figs. 4.1 and 4.2 demonstrate how the optimization method and scale of σ_i affect parameter updates in an 1-dimensional Bayesian linear regression model trained by 5 tasks continuously. These tasks are independent and have different optimal solutions. The model is defined as $y \sim \mathscr{N}(wx + b, 0.1), w \sim \mathscr{N}(\mu_w, \sigma_w^2), b \sim \mathscr{N}(\mu_b, \sigma_b^2)$. The initialization of σ_w and σ_b is set to 0.1 and 0.001 in Figs. 4.1 and 4.2, respectively. The update steps in Fig. 4.1d are smaller than in Fig. 4.2d, even when its initialization of the variance is larger, which is because larger value of initialization σ_0 results in a larger variance of gradients (see the difference between Fig. 4.1a

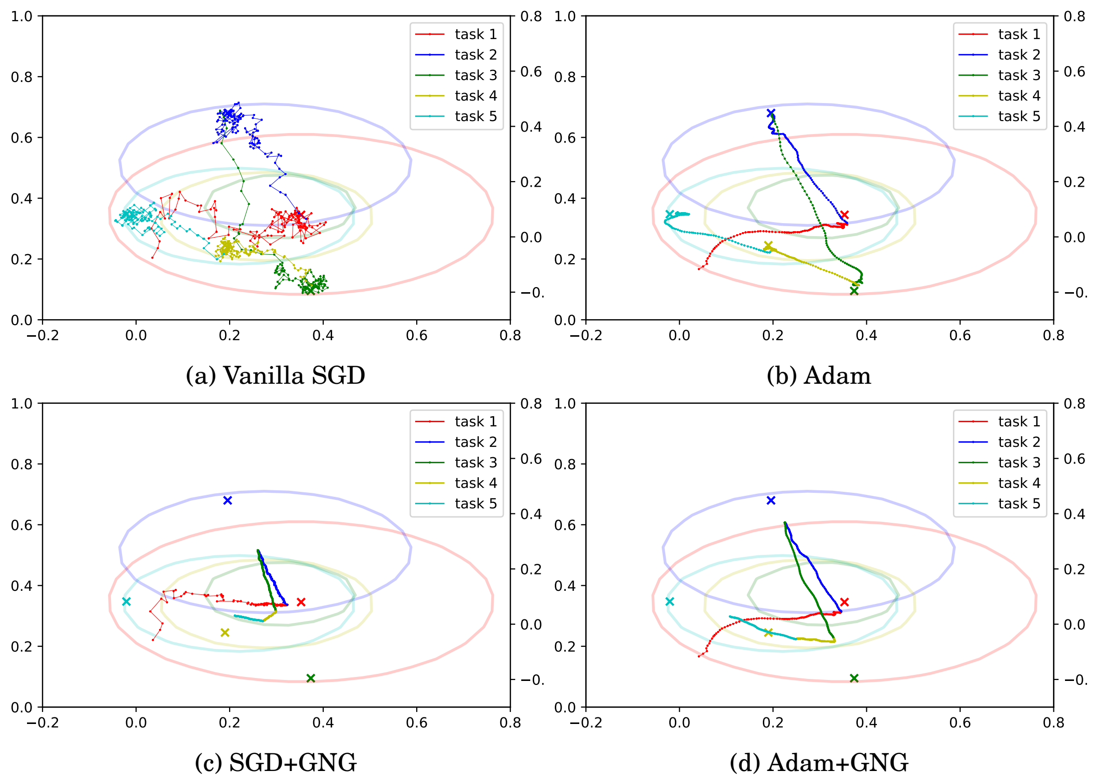


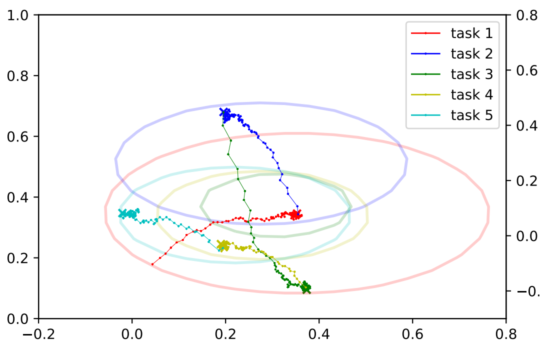
Figure 4.1: Updating trajectory of parameters of 1-dimensional Bayesian linear regression in continual learning. The x-axis is μ_w and y-axis is μ_b. The contour depicts the same level of average MSE over seen tasks, to get equal or better performance on seen tasks, the model needs to find an optimum inside the contour, so the area of the contour becomes smaller and smaller when the model has learned more tasks. The cross-mark indicates the position of true parameters of each task, different colours represent different tasks. The learning rate is set to 0.001 for vanilla SGD and 0.01 for all other methods. The initialization of σ_w and σ_b is set to $\sigma_0 = 0.1$.

and Fig. 4.2a) due to the reparameterization trick we applied to approximate the gradients in VI. Since we draw samples of w and b to compute the gradients at each step, the variance of gradients highly relate to the variance of both variables. Consequently, the step size of parameter updates decreases according to Eq. (4.9). In general, GNG shows lower variance in parameter updates, and it works better with Adam than with SGD.

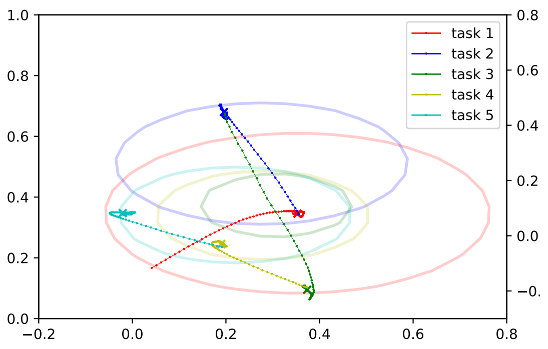
4.1.3 Related work

As we introduced in Sec. 3.3.1, EWC and Online EWC (OEWC) also utilize Fisher information but in a different way. In OEWC, the prior of the parameter $\boldsymbol{\theta}$ at task t can be viewed as approximated by $p(\boldsymbol{\theta}) \approx \mathcal{N}(\boldsymbol{\theta}^*_{t-1}, (F^*_{t-1})^{-1})$ and the regularization term is derived from the perspective of MAP. According to Eq. (3.7), the gradient of a parameter θ_i in OEWC is:

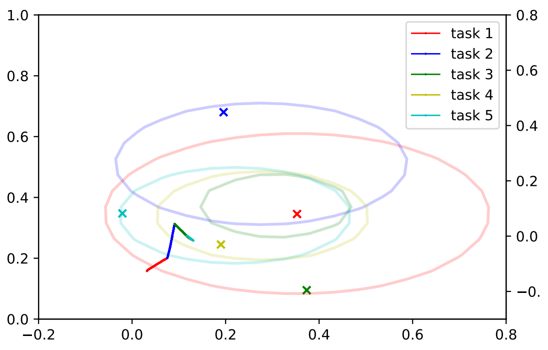
$$\nabla_{\theta_i} \mathscr{L}_{\text{OEWC}} = \nabla_{\theta_i} \mathscr{L}_t(\theta_i) + \lambda F^*_{t-1,i}(\theta_i - \theta^*_{t-1,i}) \tag{4.11}$$



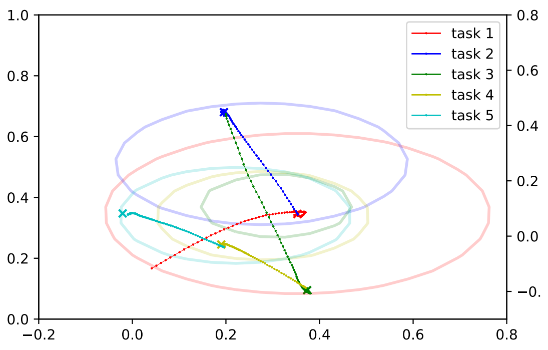
(a) Vanilla SGD



(b) Adam



(c) SGD+GNG



(d) Adam+GNG

Figure 4.2: Parameter trajectory of 1-dimensional Bayesian linear regression in continual learning. All configurations are the same as in Fig. 4.1 except $\sigma_0 = 0.001$.

We can see that this is in a very different form in comparison with GNG (Eq. (4.8)). More specifically, we optimize the variational parameters of the posteriors for BNNs instead of directly optimizing the network weights as in EWC and OEWC.

A concurrent work (Tseran et al., 2018) proposes Vadam VCL which also utilising Fisher information based on VCL. It deploys Vadam (Khan et al., 2018) in a similar way with VCL as follows:

$$\tilde{g}_{\mu_{i,t}} = \sigma_{i,t}^{-1}\left[g_i - \sigma_{i,t-1}^{-2}(\mu_{i,t} - \mu_{i,t-1})\right], \;\; \sigma_{i,t}^{-2} \leftarrow s_t + \sigma_{i,t-1}^{-2}, \;\; s_t \leftarrow \lambda s_t + (1-\lambda) g_i^2 \tag{4.12}$$

We can see that it complies with the formulation of VI but without learning specific parameters of σ_i. Instead, it estimates the variance by a moving average of the diagonal of the Hessian matrix (which can also be viewed as the Fisher information) and update the mean in a similar way as Adam. UCB (Ebrahimi et al., 2020) is subsequent work that adapts the conventional gradients by the variance of the posterior in a similar way with GNG but without the full treatment of the natural gradient. In addition, it deploys a mixture Gaussian distribution as the prior of all tasks instead of using the previous posterior $q_{t-1}(\theta_i)$. The gradient approximated in UCB is as below:

$$\tilde{g}_{\mu_{i,t}} = \Omega_{i,t}^{-1} \nabla_{\mu_{i,t}} \mathscr{L}_{\text{UCB}}, \;\; \Omega_{i,t} = 1/\sigma_{i,t} \;\text{ or }\; |\mu_{i,t}|/\sigma_{i,t} \tag{4.13}$$

The loss of UCB ($\mathscr{L}_{\text{UCB}}$) is given in Eq. (3.12). UCB adjusts the gradient in a similar way with natural gradient. However, it solely relies on this adjustment to keep closer to important previous parameters since it fixes the prior for all tasks with zero means. Ebrahimi et al. (2020) provided experimental results comparing UCB with EWC, Vadam, and GNG, which shows UCB outperforms these methods.

In the next section we will introduce a method of composing episodic memories based on the framework VCL and in Sec. 4.3 we will show experimental results of combining GNG with it.

4.2 Stein Gradient-based Episodic Memories

In the context of continual learning, "episodic memories" (also referred as "coresets" in VCL (Nguyen et al., 2018)) are small collections of data samples of every learned task, used for experience replay over learned tasks when learning a new task. The motivation is to retain summarized information of the data distribution of learned tasks so that we can use this information to obtain an optimal solution for old tasks along with solving a new task. On the other hand, the information of a learned task is contained not only in its data samples but also in its trained parameters. To this end we consider constructing the episodic memory with the use of approximated posteriors. In this section we introduce an approach based on Stein Variational Gradient Descent (SVGD) to compose the episodic memory not only including information of the data distribution but also entailing information of the model parameters.

4.2.1 Preliminary: Stein Variational Gradient Descent

We have introduced several methods of Stochastic Variational Inference (SVI) in Chapter 2 which approximate the posterior by an explicitly parameterized form. Besides approximating an explicit posterior, we can also approximate samples from the variational posterior without an explicit form of it. Liu & Wang (2016) provides a way to achieve this goal, which is Stein Variational Gradient Descent (SVGD). Instead of optimizing the variational parameters of the approximated posterior, SVGD transforms random samples from an initial distribution to samples from the true posterior. Suppose we have a series of samples $\mathbf{x}$ from an initial distribution $q(\mathbf{x})$ and transform them as follows:

$$\mathbf{z} = T(\mathbf{x}) = \mathbf{x} + \epsilon\phi_{q,p}(\mathbf{x}), \quad \mathbf{x} \sim q(\mathbf{x})$$

We denote the density of $\mathbf{z}$ as $q_{[T]}(\mathbf{z})$. Here $\boldsymbol{\phi}_{q,p}(\cdot)$ is in the ball of a Reproducing Kernel Hilbert Space (RKHS) $\mathscr{H}^d$: $\mathscr{B} = \{\phi \in \mathscr{H}^d : ||\phi||_{\mathscr{H}^d} \leq \mathbb{D}(q,p)\}$, $\mathbb{D}(q,p)$ is the Kernelized Stein Discrepancy (KSD) defined as:

$$\mathbb{D}(q,p) \triangleq \arg\max_{\phi \in \mathscr{H}^d} \{\mathbb{E}_{\mathbf{x}\sim q(\mathbf{x})}[\text{trace}(\mathscr{A}_p \phi(\mathbf{z}))], \quad s.t. \quad \|\phi\|_{\mathscr{H}^d} \leq 1\} \tag{4.14}$$

Choose the optimal ϕ to decrease the KL-divergence between a target distribution $p(\mathbf{z})$ and $q_{[T]}(\mathbf{z})$ in the steepest direction:

$$\phi^*_{q,p}(\cdot) \triangleq \arg\min_{\phi} \nabla_\epsilon D_{KL}(q_{[T]}||p)|_{\epsilon=0} \tag{4.15}$$

Theorem 3.1 in Liu & Wang (2016) shows:

$$\nabla_\epsilon D_{KL}(q_{[T]}||p)|_{\epsilon=0} = -\mathbb{E}_{q(\mathbf{x})}[\text{trace}(\mathscr{A}_p \phi(\mathbf{x}))] \tag{4.16}$$

where $\mathscr{A}_p$ is the Stein operator that satisfies the Stein Identity:

$$\mathbb{E}_p[\mathscr{A}_p \phi(\mathbf{x})] = 0, \quad \mathscr{A}_p \phi(\mathbf{x}) = \phi(\mathbf{x}) \nabla_{\mathbf{x}} \log p(\mathbf{x})^T + \nabla_{\mathbf{x}} \phi(\mathbf{x})$$

Lemma 3.2 in Liu & Wang (2016) gives:

$$\phi^*_{q,p}(\cdot) = \mathbb{E}_{\mathbf{x}\sim q(\mathbf{x})}[\kappa(\mathbf{x},\cdot)\nabla_{\mathbf{x}} \log p(\mathbf{x}) + \nabla_{\mathbf{x}} \kappa(\mathbf{x},\cdot)], \tag{4.17}$$

$\kappa(\cdot,\cdot)$ is a definite positive kernel in the Stein class of p (i.e., $\int \nabla_{\mathbf{x}}(\kappa(\mathbf{x},\cdot)p(\mathbf{x}))d\mathbf{x} = 0$) (Liu et al., 2016). Eq. (4.17) suggests how to iteratively transform $\mathbf{x}$ from $q(\mathbf{x})$ to samples from the target distribution $p(\mathbf{z})$ as at each step k we apply:

$$\mathbf{z}_{k+1} = T_t(\mathbf{z}_k) = \mathbf{z}_k + \epsilon_{t-1} \phi^*_{q_{[T_k]},p}(\mathbf{z}_k), \ \ k \geq 1, \ \ \mathbf{z}_0 = \mathbf{x}, \ \ q_{[T_0]} = q \tag{4.18}$$

which iteratively decreases the KL-divergence in the steepest direction with sufficiently small $\{\epsilon_k\}$. Thus, the Stein gradient of each sample can be computed as follows:

$$\phi^*_{q_{[T_k]},p}(\mathbf{z}_k^{(i)}) = \frac{1}{M} \sum_{j=1}^{M} \left[\kappa(\mathbf{z}_k^{(j)}, \mathbf{z}_k^{(i)}) \nabla_{\mathbf{z}_k^{(j)}} \log p(\mathbf{z}_k^{(j)}) + \nabla_{\mathbf{z}_k^{(j)}} \kappa(\mathbf{z}_k^{(j)}, \mathbf{z}_k^{(i)}) \right] \tag{4.19}$$

We continue to show how to take advantage of SVGD to generate episodic memories for Bayesian continual learning.

4.2.2 Stein Gradient-based Episodic Memories

Since Stein gradients Liu & Wang (2016) can be used to generate samples of a target distribution, we consider to compose episodic memories by using Stein gradients to generate samples from the joint distribution $p(\mathbf{x}, \boldsymbol{\theta}|\mathbf{y})$, where $\mathbf{x}$ denotes the input data of the model, $\mathbf{y}$ denotes the true class of $\mathbf{x}$, and $\boldsymbol{\theta}$ denotes model parameters. As the joint distribution also includes information of the parameters, the memory could be more helpful to prevent parameters' drifting as well. Suppose the posterior of parameters $\boldsymbol{\theta}$ are learned over the whole training set. As the true posterior, $p(\boldsymbol{\theta}|\mathscr{D})$ ($\mathscr{D} = \{X, Y\}$ represents the training set) is usually not available, we approximate it by the variational posterior $q(\boldsymbol{\theta})$ in VI and the joint distribution can be approximated by $\tilde{p}(\mathbf{x}, \mathbf{y}, \boldsymbol{\theta}) = p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}) q(\mathbf{x}) q(\boldsymbol{\theta})$, $q(\mathbf{x})$ is the empirical marginal distribution of $\mathbf{x}$. Here we assume $q(\mathbf{x})$ and $q(\boldsymbol{\theta})$ are independent because $q(\boldsymbol{\theta})$ is learned in a separate process and fixed during the memory updates. $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})$ is given by the model output, $q(\mathbf{x}|\mathbf{y})$ denotes the empirical conditional distribution of $\mathbf{x}$ given $\mathbf{y}$. We can apply Stein gradient to update $\mathbf{x}$ of a given class $\mathbf{y}$ as follows.

Theorem 4.1. *Let $\boldsymbol{\theta} \sim q(\boldsymbol{\theta})$, $\tilde{p}(\mathbf{x},\mathbf{y},\boldsymbol{\theta}) = p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})q(\mathbf{x})q(\boldsymbol{\theta})$, $\tilde{p}(\mathbf{x},\boldsymbol{\theta}|\mathbf{y}) = \tilde{p}(\mathbf{x},\mathbf{y},\boldsymbol{\theta})/p(\mathbf{y})$, define the transformation $T(\mathbf{x}) = \mathbf{x} + \epsilon\phi(\mathbf{x})$, when $\mathbf{x} \sim q(\mathbf{x}|\mathbf{y})$, denote $T(\mathbf{x}) \sim q_{[T]}(\mathbf{x}|\mathbf{y})$, assume $p(\mathbf{y}) = 1/K$, $q(\mathbf{x}) = 1/N$, where K is the number of classes, N is the total number of samples of all classes, given $\mathbf{y}$ we have:*

$$\begin{aligned}\phi^*(\mathbf{x}|\mathbf{y}) &\triangleq \operatorname*{argmin}_{\phi} \nabla_\epsilon D_{KL}(q(\boldsymbol{\theta})q_{[T]}(\mathbf{x}|\mathbf{y})||\tilde{p}(\mathbf{x},\boldsymbol{\theta}|\mathbf{y}))|_{\epsilon=0}\\ &= \mathbb{E}_{\mathbf{x}\sim q_{[T]}(\mathbf{x}|\mathbf{y})}[\kappa(\mathbf{x},\cdot)\nabla_{\mathbf{x}}\mathbb{E}_{q(\boldsymbol{\theta})}[\log p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})] + \nabla_{\mathbf{x}}\kappa(\mathbf{x},\cdot)]\end{aligned}$$

Proof.

$$\begin{aligned}\nabla_\epsilon D_{KL}(q(\boldsymbol{\theta})q_{[T]}(\mathbf{x}|\mathbf{y})||\tilde{p}(\mathbf{x},\boldsymbol{\theta}|\mathbf{y})) &= \nabla_\epsilon \int\int q(\boldsymbol{\theta})q_{[T]}(\mathbf{x}|\mathbf{y})\log\frac{q(\boldsymbol{\theta})q_{[T]}(\mathbf{x}|\mathbf{y})p(\mathbf{y})}{p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})q(\mathbf{x})q(\boldsymbol{\theta})}d\boldsymbol{\theta}d\mathbf{x}\\ &= \nabla_\epsilon \int q(\boldsymbol{\theta})\int q_{[T]}(\mathbf{x}|\mathbf{y})\log\frac{q_{[T]}(\mathbf{x}|\mathbf{y})p(\mathbf{y})}{p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})q(\mathbf{x})}d\mathbf{x}d\boldsymbol{\theta}\\ &= \nabla_\epsilon \int q(\boldsymbol{\theta})\int q_{[T]}(\mathbf{x}|\mathbf{y})\log\frac{\frac{1}{K}q_{[T]}(\mathbf{x}|\mathbf{y})}{p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})q(\mathbf{x})}d\mathbf{x}d\boldsymbol{\theta}\\ &= \int q(\boldsymbol{\theta})\nabla_\epsilon \int q_{[T]}(\mathbf{x}|\mathbf{y})\left(\log\frac{q_{[T]}(\mathbf{x}|\mathbf{y})}{p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})q(\mathbf{x})} - \log K\right)d\mathbf{x}d\boldsymbol{\theta}\\ &= \mathbb{E}_{q(\boldsymbol{\theta})}\left[\nabla_\epsilon D_{KL}(q_{[T]}(\mathbf{x}|\mathbf{y})||p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})q(\mathbf{x}))\right]\end{aligned}$$

The last equality above is because $\nabla_{\boldsymbol{\epsilon}}\int q_{[T]}(\mathbf{x}|\mathbf{y})\log K d\mathbf{x} = \nabla_{\boldsymbol{\epsilon}}\log K = 0$. According to Eq. (4.16):

$$\begin{aligned}&\nabla_\epsilon D_{KL}(q(\boldsymbol{\theta})q_{[T]}(\mathbf{x}|\mathbf{y})||\tilde{p}(\mathbf{x},\boldsymbol{\theta}|\mathbf{y}))|_\epsilon = \mathbb{E}_{q(\boldsymbol{\theta})}[\mathbb{E}_{q_{[T]}(\mathbf{x}|\mathbf{y})}[\text{trace}(\mathscr{A}_p\phi(\mathbf{x}|\mathbf{y}))]],\\ &\text{where } \mathscr{A}_p\phi(\mathbf{x}|\mathbf{y}) = \phi(\mathbf{x}|\mathbf{y})\nabla_{\mathbf{x}}\log(p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})q(\mathbf{x})) + \nabla_{\mathbf{x}}\phi(\mathbf{x}|\mathbf{y})\end{aligned} \tag{4.20}$$

Because $q(\mathbf{x}) = 1/N$, $\nabla_{\mathbf{x}}\log(p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})q(\mathbf{x})) = \nabla_{\mathbf{x}}\log p(\mathbf{y}|\mathbf{x},\boldsymbol{\theta})$. By **Lemma 3.2** in Liu & Wang (2016) (Eq. (4.17)) we prove the theorem. □

Suppose we have a series of samples $\mathbf{x}$ from the empirical distribution $q(\mathbf{x}|\mathbf{y})$, and we update them iteratively to move closer to samples from the joint distribution $\tilde{p}(\mathbf{x},\boldsymbol{\theta}|\mathbf{y})$ by $\mathbf{x}_{k+1} = \mathbf{x}_k + \epsilon\phi^*(\mathbf{x}_k|\mathbf{y})$, according to Theorem 4.1, the Stein gradient can be computed by:

$$\phi^*(\mathbf{x}_k^{(i)}|\mathbf{y}) = \frac{1}{M}\sum_{j=1}^{M}\left[\kappa(\mathbf{x}_k^{(j)},\mathbf{x}_k^{(i)})\frac{1}{S}\sum_{s=1}^{S}\left(\nabla_{\mathbf{x}_k^{(j)}}\log p(\mathbf{y}|\mathbf{x}_k^{(j)},\boldsymbol{\theta}_s)\right) + \nabla_{\mathbf{x}_k^{(j)}}\kappa(\mathbf{x}_k^{(j)},\mathbf{x}_k^{(i)})\right],\ \ \boldsymbol{\theta}_s \sim q(\boldsymbol{\theta}) \tag{4.21}$$

In the mean-field BNNs, $q(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu},\sigma^2 I)$. We update the samples selected into the memory by Stein gradients along with the procedure of VI when $q(\boldsymbol{\theta})$ is also updated iteratively. The updates by Stein gradients do not affect the usual parameter updates and can work with other approaches for Bayesian continual learning.

4.2.3 Related work

There are some existing approaches of Bayesian coreset construction for scalable machine learning Huggins et al. (2016); Campbell & Broderick (2018), which have a similar goal with composing the episodic memory in continual learning, i.e., using as few samples as possible to represent a large dataset. In continual learning, the memory cost often increases with the number of tasks, hence, we would prefer the memory size used for each task as small as possible. The basic idea of Bayesian coresets is to find a sparse weighted subset of data to approximate the likelihood over the whole dataset. In their problem setting the coreset construction is an objective combined with the posterior approximation, and the computational cost is at least $O(MN)$ Campbell & Broderick (2018), where M is the coreset size and N is the dataset size. In Bayesian continual learning, the memory construction does not play a role in the posterior approximation of a task as we learn a new task over the whole training set. Hence, we can construct the episodic memory in different ways without affecting the posterior approximation. Moreover, the computational complexity of the Stein gradient method is $O(M^2)$, which is significantly cheaper than $O(MN)$ when $M << N$.

On the other hand, there is some related work on composing the episodic memory in continual learning. For example, Nguyen et al. (2018) propose selecting top K-centres of the training data into the memory. Aljundi et al. (2019b) introduce using the gradient diversity to select most difficult samples into the memory and Chaudhry et al. (2019b) also show that randomly sampled memory can perform well on various tasks. We also found that randomly sampled memory can outperform the one selected by gradient diversity and will provide detailed analysis in Chapter 5. In the next section, we will present experimental results of SGEM in comparison with episodic memories composed by K-centres and random samples.

4.3 Experiments

In all experiments we applied a Bayesian Neural Network (BNN) with two hidden layers, each layer with 100 hidden units and ReLu activations, all split tasks tested using multi-head models which follows the setting in Nguyen et al. (2018). We tested our methods in the framework of VCL (Nguyen et al., 2018) on three benchmarks: permuted MNIST, split MNIST, and split fashion MNIST tasks, which are introduced in Sec. 3.1. The results are displayed in Fig. 4.3, the left column shows results of GNG with Adam and the right column shows results of SGEM. The error bars are from 5 runs by different random seeds.

We applied a RBF kernel to the Stein gradients which follows the setting described in Liu & Wang (2016). For experiments with episodic memories, the memory size is 200 per task in permuted MNIST and 40 in split tasks, which is the same as used in Nguyen et al. (2018). We tested two different usages of the episodic memories. The first is training a predictive model solely by memorized samples of each seen task to perform prediction for that task. The predictive model is fine-tuned based on the model trained at the latest task and this setting is used in Nguyen et al. (2018). The loss function of the predictive model is as follows:

$$\hat{\mathscr{L}}_t = \mathbb{E}_{q_t(\boldsymbol{\theta})}[\log p(C_t|\boldsymbol{\theta})] - \mathrm{KL}\left(q_t(\boldsymbol{\theta})\|q_t^*(\boldsymbol{\theta})\right). \tag{4.22}$$

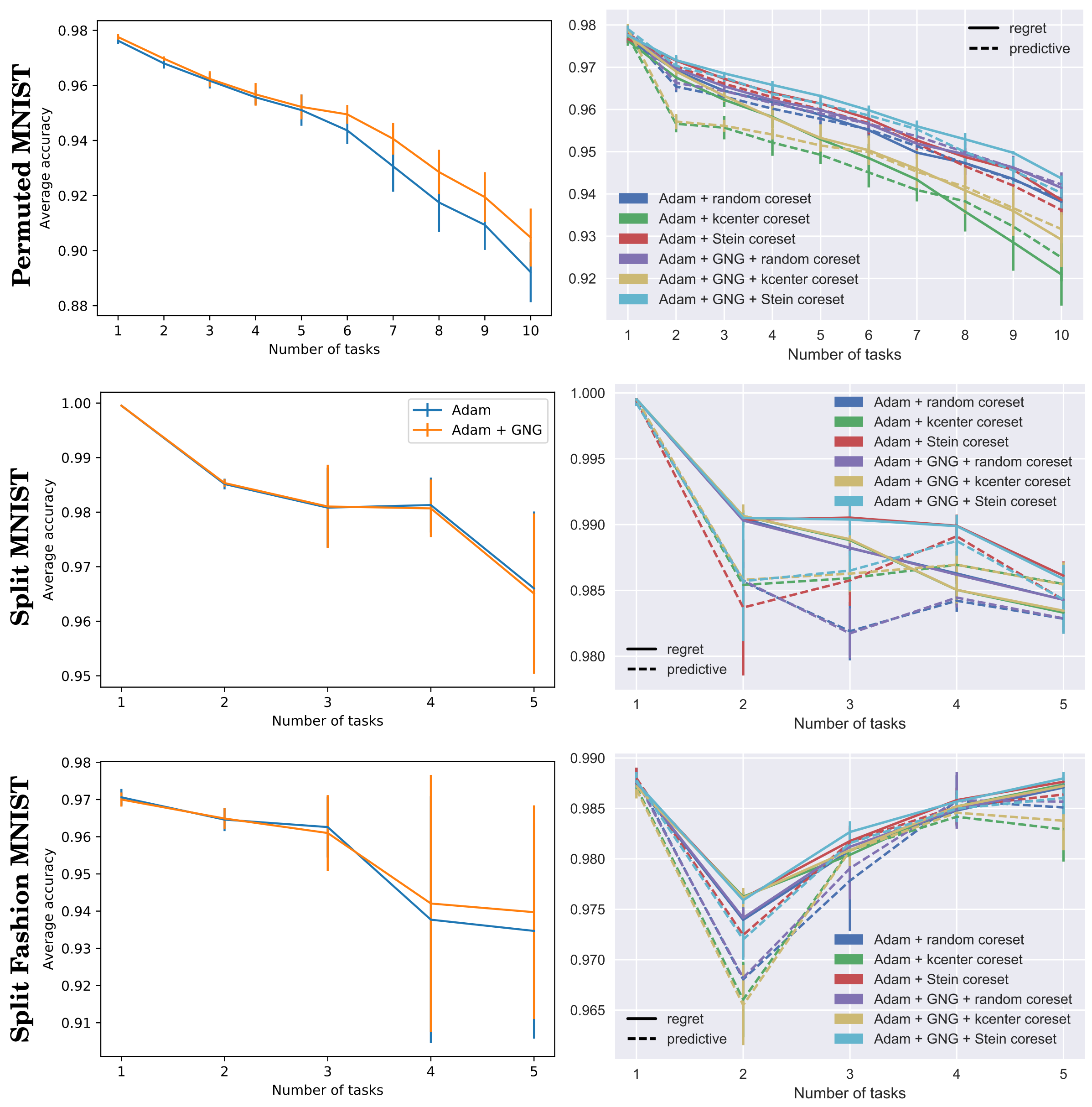


Figure 4.3: Average accuracy on permuted and split tasks without (left) and with (right) episodic memories. All methods are based on VCL.

As shown in Eq. (4.22), $C_t = \{c_1, c_2, \ldots, c_t\}$ represents the collection of episodic memories of learned tasks at time t and $q_t^*(\boldsymbol{\theta})$ is the optimal posterior obtained by the usual training procedure over the t-th task without using the episodic memory. The second usage is to add a regret loss using the episodic memory to the objective function of VCL, which does not require a separate predictive model:

$$\hat{\mathscr{L}}_t = \mathbb{E}_{q_t(\boldsymbol{\theta})}[\log p(\mathscr{D}_t|\boldsymbol{\theta})] + \mathbb{E}_{q_t(\boldsymbol{\theta})}[\log p(C_{t-1}|\boldsymbol{\theta})] - \mathrm{KL}\left(q_t(\boldsymbol{\theta}) \| q_{t-1}^*(\boldsymbol{\theta})\right), \tag{4.23}$$

where the second term C_{t-1} in Eq. (4.23) is the regret loss constructed by memorized samples of previous tasks. The results of the first usage (predictive) are displayed in dotted line and the second usage (regret) in solid line in the right column of Fig. 4.3.

The regret usage gives better performance in general, and SGEM outperforms other two methods in most cases. Moreover, SGEM with GNG shows better performance than others in permuted MNIST tasks as in the permuted MNIST task GNG with Adam outperforms standalone Adam when not using episodic memories. There is no significant difference in split tasks between with and without GNG, and we provide some further analysis in the following.

As one model has a limited capacity, and each different task contains some different information, the ideal case for continual learning is that each new task shares as much information as possible with previous tasks, and occupying as little extra capacity of the model as possible. This is analogous to model compression Louizos et al. (2017), but one key difference is we want more *free* parameters instead of parameters that are frozen to zero. For example, assume there are k parameters in a model with a mean-field prior and the log-likelihood of the current task is as:

$$\log p(D_t|\theta_1,\theta_2,\dots,\theta_k) = \log p(D_t,\theta_1,\theta_2,\dots,\theta_k) - \sum_{i=1}^{k} \log p(\theta_i). \tag{4.24}$$

If θ_1 is absolutely free for this task, it indicates the log-likelihood is a constant w.r.t. θ_1, then we have:

$$\nabla_{\theta_1} \log p(D_t,\theta_2,\dots,\theta_k|\theta_1) = 0, \quad \forall\theta_1. \tag{4.25}$$

In this case, θ_1 is free to change without affecting other parameters or the likelihood of the data. Hence, no matter what value of θ_1 is set to in future tasks, it will not affect the loss of previously learned tasks. In realistic situations, θ_1 is unlikely to be absolutely free. However, it is feasible to maximize the entropy of θ_1, larger entropy indicating more freedom of θ_1. For instance, minimizing KL divergence includes maximizing the entropy of parameters:

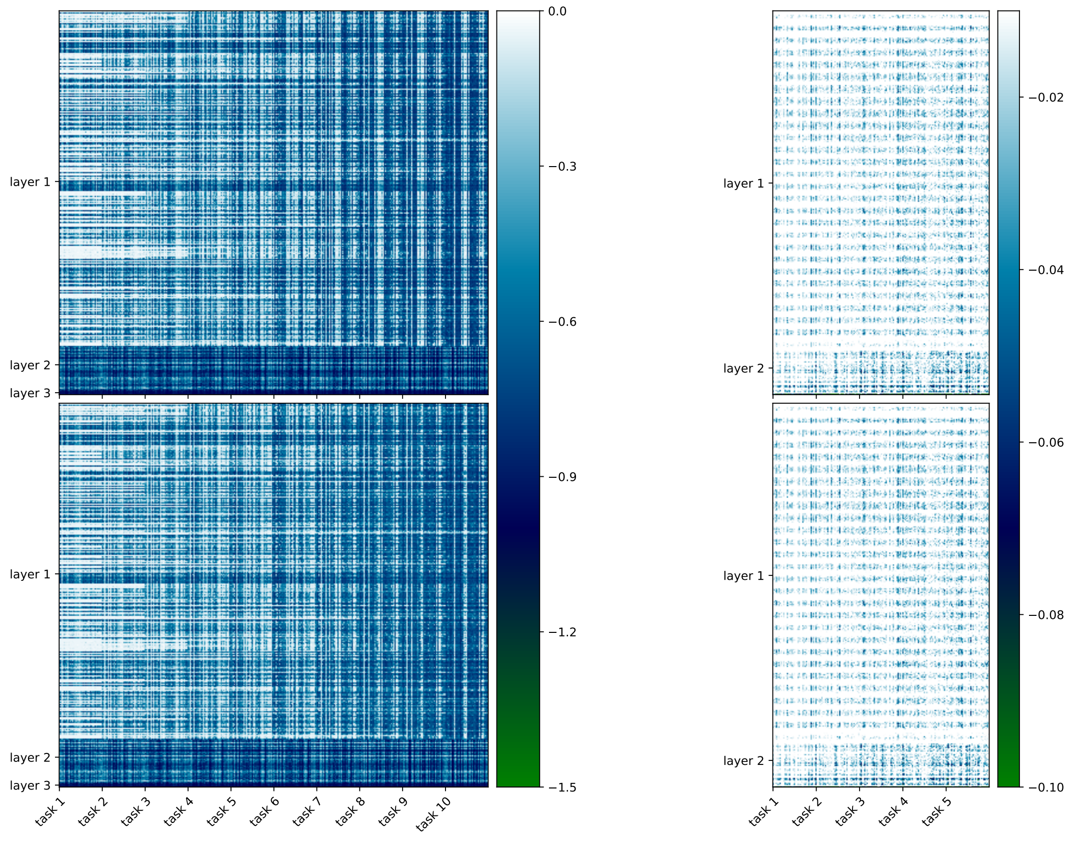
$$D_{KL}(q_t(\theta)||q_{t-1}(\theta)) = -\mathbb{E}_{q_t}[\log q_{t-1}(\theta)] - H_{q_t}(\theta). \tag{4.26}$$

On the other hand, it is undesirable to change parameters with lower entropy instead of those with higher entropy while learning a new task, since it could cause performance degradation on previous tasks.

The entropy of a Gaussian distribution is determined by its variance alone. In this sense, a larger decrease of the variance indicates larger decrease of the entropy of a parameter. To understand why GNG works better on permuted MNIST tasks, we visualized how the variances of parameters change in Fig. 4.4 where all changed values are normalized as below for a better scaling in the visualization:

$$\Delta\sigma_{i,t} = \frac{\sigma_{i,t} - \max_i \sigma_{i,1}}{\max_i \sigma_{i,1}}, \tag{4.27}$$

In the above equation $\max_i \sigma_{i,1}$ is the maximal variance among all parameters at the first task. When the variance of parameters is decreased by learning a new task, the entropy of the model



(a) permuted MNIST (b) split MNIST

Figure 4.4: Variance changes w.r.t. first task, top row is from models trained by Adam, bottom row is from models trained by Adam + GNG, tested on permuted and split MNIST without episodic memories. The x-axis is concatenated by tasks, the y-axis is concatenated by BNN layers, as split tasks are tested on multi-head models, so there is no layer 3 in Fig. 4.4b.

is decreased as well. We can think of it as new information written into the model, so when the model has learned more tasks, more parameters have reduced variance as shown in Fig. 4.4. The darker colour indicates larger decrease of the variance. In an ideal case, a parameter with larger variance should be chosen to write new information preferentially to avoid erasing information of previous tasks. Therefore, it would be preferred if the dark colour spread more evenly in latter tasks in Fig. 4.4, and Adam + GNG appears to have this property for the permuted MNIST task (Fig. 4.4a). However, there is no notable difference caused by GNG for split MNIST tasks (Fig. 4.4b), which is consistent with their performance in terms of average accuracy over tasks. We posit there are some parameters important across split tasks that are constantly rewritten for a new task. The reason could be that split tasks are less homogeneous than permuted tasks and thus less parameters can be shared (staying invariant) across tasks, therefore, the changes over parameters are less evenly spread even with GNG.

4.4 Summary

In this chapter we proposed two methods to obtain improvements over VCL for Bayesian continual learning: one is a regularization-based method utilising natural gradient to adjust the gradient of variational parameters of the approximated posterior in BNNs; the other is a replay-based method utilising SVGD to compose the episodic memory along with the posterior approximation for BNNs. The regularization-based method GNG has shown notable improvements on permuted MNIST tasks, however, on split tasks it has similar performance with just applying the Adam optimizer to VCL. On the other hand, the replay-based method SGEM also shows more gains on permuted MNIST tasks than on split tasks. Since both methods attempt to preserve more information of the posteriors of previous parameters, the results suggest that keeping closer to previous posteriors may not be beneficial to less homogeneous tasks as the optimal posterior for multiple tasks may not be close to the one learned previously. This is also an intrinsic issue of regularization-based methods for Non-Bayesian models. In general, regularization-based methods result in much worse performance on split tasks in single-headed models (van de Ven & Tolias, 2019) because the final linear layer tends to be largely changed by learning a new task. In the next chapter we will introduce an approach to alleviate the conflicts between tasks through the perspective of gradients, which can efficiently reduce diversity of gradients between tasks with much less computational cost than directly adjusting the gradients during training.

CHAPTER 5

Discriminative Representation Loss for Continual Learning

Through Chapter 4 we found that constraining parameters to be close to previous values in continual learning may not be satisfactory for less homogeneous tasks. As replay-based methods exhibit more robust performance in such cases, we turned to explore in this direction for a more general solution. In Sec. 3.3 we introduced several gradient-based replay approaches that constrain the gradients resulting from new samples with those from memorized samples, aiming to reduce the diversity of gradients from different tasks. In this chapter, we investigate the relation between diversity of gradients and discriminativeness of representations, demonstrating connections between Deep Metric Learning (DML) and continual learning. Based on these findings, we propose a simple yet highly efficient method, called Discriminative Representation Loss (DRL), for continual learning. In comparison with several state-of-the-art methods, DRL shows effectiveness with low computational cost on multiple benchmark experiments in the setting of online continual learning. In the following sections, we first briefly review the problem background in Sec. 5.1, then we show the connections between gradients and representations empirically and theoretically in Sec. 5.2 which lead to the definition of our method in Sec. 5.3. In Sec. 5.5, we present comprehensive experimental results including an ablation study on DRL. Finally, we provide our experimental results for the CLVision challenge hosted by the Continual Learning Workshop of CVPR 2020 in Sec. 5.6.

5.1 Introduction

Gradient-based approaches using episodic memories have been receiving increasing attention recently in the literature of continual learning. As we introduced in Sec. 3.3.3, the essential idea is to use gradients produced by samples from episodic memories to constrain the gradients produced by new samples, *e.g.* by ensuring the inner product of the pair of gradients is non-negative (Lopez-Paz & Ranzato, 2017) as follows:

$$\langle \boldsymbol{g}_t, \boldsymbol{g}_k \rangle = \left\langle \frac{\partial \mathscr{L}(\mathbf{x}_t, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}}, \frac{\partial \mathscr{L}(\mathbf{x}_k, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right\rangle \geq 0, \quad \forall k < t \tag{5.1}$$

where t and k are time indices, $\mathbf{x}_t$ denotes a new sample from the current task, and $\mathbf{x}_k$ denotes a sample from the episodic memory. Thus, the parameter updates are forced to preserve the performance on previous tasks as much as possible. GEM (Lopez-Paz & Ranzato, 2017), A-GEM (Chaudhry et al., 2019a), and OGD (Farajtabar et al., 2020) aim to re-project the gradients produced by new samples to a direction close to or orthogonal to gradients produced by memorized samples. GSS (Aljundi et al., 2019b) selects samples that produce most dissimilar gradients with other samples into episodic memories. Although GSS suggests the samples with most diverse gradients are important for generalization across tasks, Chaudhry et al. (2019b) show that the average gradient over a small set of random samples may obtain good generalization as well.

In this chapter, we answer the following questions: *i*) Which samples tend to produce diverse gradients that strongly conflict with other samples and why are such samples able to help with generalization? *ii*) Why does a small set of randomly chosen samples also help with generalization? *iii*) Can we reduce the diversity of gradients in a more efficient way? Our answers to these questions shed light on the relation between diversity of gradients and discriminativeness of representations, and further lead to connections between Deep Metric Learning (DML) (Kaya & Bilge, 2019; Roth et al., 2020) and continual learning. Drawing on these findings we propose a new approach, Discriminative Representation Loss (DRL), for classification tasks in continual learning. Our method shows improved performance with relatively low computational cost in terms of time and RAM cost when compared to several state-of-the-art (SOTA) methods across multiple benchmark tasks in the setting of online continual learning.

5.2 A New Perspective of Reducing Diversity of Gradients

According to the basic idea of gradient-based approaches (Eq. (5.1)), negative inner product between gradients produced by current and previous tasks results in conflicting parameter update which may increase the loss of either current or previous task, and hence leads to worse performance in continual learning. Liu et al. (2020) suggest that the variance of gradients relates to the Gradient Signal to Noise Ratio (GSNR) , which plays a crucial role in the model's generalization ability:

$$\text{GSNR} = \mathbb{E}^2_{p(\mathbf{x})}[\boldsymbol{g}]/\text{Var}_{p(\mathbf{x})}[\boldsymbol{g}], \quad \boldsymbol{g} = \nabla_{\boldsymbol{\theta}} \mathscr{L}(\mathbf{x}, \boldsymbol{\theta}).$$

Intuitively, when more of the gradients point in diverse directions, the variance will be larger, leading to a smaller GSNR and worse generalization, which indicates that reducing the diversity of gradients can increase GSNR and then improve generalization. This finding leads to the conclusion that samples with the most diverse gradients (i.e., the gradients have largely negative similarities with other samples) contain the most critical information for generalization, which is consistent with in Aljundi et al. (2019b).

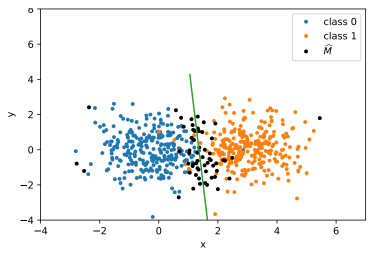
5.2.1 The relation between gradients and representations

We first conducted a simple experiment on classification tasks of 2-D Gaussian distributions, aimed to identify samples with most diverse gradients in the 2-D feature space. We trained a linear model on the first task to discriminate between two classes (blue and orange dots in Fig. 5.1a). We then applied the algorithm Gradient-based Sample Selection with Integer Quadratic Programming (GSS-IQP) (Aljundi et al., 2019b) to select 10% of the samples of training data that produce gradients with the lowest similarity (black dots in Fig. 5.1a), which is defined as:

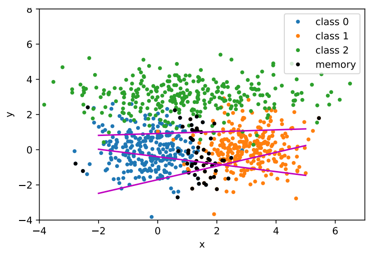
$$\widehat{M} = \underset{M}{\arg\min} \sum_{i,j \in M} \frac{\langle \boldsymbol{g}_i, \boldsymbol{g}_j \rangle}{||\boldsymbol{g}_i|| \cdot ||\boldsymbol{g}_j||} \tag{5.2}$$

It is clear from Fig. 5.1a that the samples in $\widehat{M}$ are mostly around the decision boundary between the two classes. Increasing the size of $\widehat{M}$ results in the inclusion of samples that trace the outer edges of the data distributions from each class. It indicates that gradients can be strongly opposed when samples from different classes are very similar. Samples close to decision boundaries are most likely to exhibit this characteristic. This result of the first task suggests that ***more similar representations in different classes result in more diverse gradients.*** An extreme example would be two samples with same features having different labels. In this case, if one sample results in a minimal loss, then the other will result in a maximal loss, and hence the gradient to decrease the loss of the latter sample must increase the loss of the former.

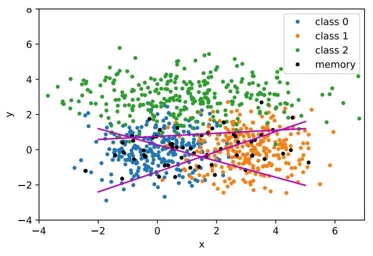
Intuitively, storing the decision boundaries of previously learned classes should be an effective way to preserve classification performance on those classes. However, why randomly chosen samples can help the model generalize well? To answer this question, we introduced a second task - training the model above on a third class (green dots). We tested two strategies to select samples from the first task into the episodic memory: one is storing samples selected by $\widehat{M}$ (black dots in Fig. 5.1b), which is the GSS-IQP method ((Aljundi et al., 2019b)); the other is storing randomly chosen samples (black dots Fig. 5.1c). We display the decision boundaries (purple lines in Figs. 5.1b and 5.1c, which split the feature space in a one *vs.* all manner) learned by the model after learning task 2 with the two memory strategies. The model with random memory shows better performance than the one with GSS-IQP as it learns more accurate decision boundaries. It is because samples in $\widehat{M}$ have much less variance along the x-axis than the true data distributions of the first two classes, which increases difficulties of learning new boundaries when adding the third class. Through these experiments, we can see that if the episodic memory only includes samples representing the learned boundaries, it may miss important information when the model



(a) Samples with most diverse gradients ($\widehat{M}$) after learning task 1, the green line is the decision boundary.



(b) Learned decision boundaries (purple lines) after task 2. Here the episodic memory includes samples in $\widehat{M}$.



(c) Learned decision boundaries (purple lines) after task 2. Here the episodic memory consists of random samples.

Figure 5.1: 2-D classification examples, the x and y axis are the coordinates (also features) of samples. We sequentially train a logistic regression model on two tasks: the first task is to classify two classes as shown in (a); the second class is to incrementally classify a third class as shown in (b) and (c). The solid lines are decision boundaries between classes.

is required to incrementally learn new classes. It explains why randomly selected memories may generalize better in continual learning. Ideally, with $\widehat{M}$ large enough, the model can remember all edges of each class, and hence learn much more accurate decision boundaries sequentially. However, memory size is often limited in practice, especially for high-dimensional data. Under such a restriction, we would prefer storing samples that contain more information of the true data distribution. These experimental results suggest that: ***less compact representations within classes help with learning new boundaries incrementally.***

Now we formalize the connection between the diversity of gradients and the discriminativeness of representations for the linear model. **Notations**: ***Negative pair*** represents two samples from different classes. ***Positive pair*** represents two samples from a same class. Let $\mathscr{L}$ represent the softmax cross entropy loss, $\mathbf{W} \in \mathbb{R}^{D\times K}$ is the weight matrix of the linear model, and $\mathbf{x}_n \in \mathbb{R}^D$ denotes the input data, $\mathbf{y}_n \in \mathbb{R}^K$ is a one-hot vector that denotes the label of $\mathbf{x}_n$, D is the dimension of representations, K is the number of classes. Let $\boldsymbol{p}_n = softmax(\mathbf{o}_n)$, where $\mathbf{o}_n = \mathbf{W}^T\mathbf{x}_n$, the gradient $\boldsymbol{g}_n = \nabla_{\mathbf{W}}\mathscr{L}(\mathbf{x}_n, \mathbf{y}_n; \mathbf{W})$. $\mathbf{x}_n, \mathbf{x}_m$ are two different samples when $n \neq m$.

Lemma 5.1. *Let $\boldsymbol{\epsilon}_n = \boldsymbol{p}_n - \mathbf{y}_n$, we have: $\langle \boldsymbol{g}_n, \boldsymbol{g}_m \rangle = \langle \mathbf{x}_n, \mathbf{x}_m \rangle \langle \boldsymbol{\epsilon}_n, \boldsymbol{\epsilon}_m \rangle$,*

Proof. Let $\boldsymbol{\ell}'_n = \partial\mathscr{L}(\mathbf{x}_n, \mathbf{y}_n; \mathbf{W})/\partial\mathbf{o}_n$, by the chain rule, we have $\langle \boldsymbol{g}_n, \boldsymbol{g}_m \rangle = \langle \mathbf{x}_n, \mathbf{x}_m \rangle \langle \boldsymbol{\ell}'_n, \boldsymbol{\ell}'_m \rangle$. By the definition of softmax cross-entropy loss $\mathscr{L}$, we can find $\boldsymbol{\ell}'_n = \boldsymbol{p}_n - \mathbf{y}_n = \boldsymbol{\epsilon}_n$. □

Theorem 5.2. *Suppose $\mathbf{y}_n \neq \mathbf{y}_m$, and let c_n denote the class index of $\mathbf{x}_n$ (i.e. $\mathbf{y}_{n,c_n} = 1, \mathbf{y}_{n,i} = 0, \forall i \neq c_n$). Let $\beta \triangleq \boldsymbol{p}_{n,c_m} + \boldsymbol{p}_{m,c_n}$ and $s_p \triangleq \langle \boldsymbol{p}_n, \boldsymbol{p}_m \rangle$, then:*

$$\Pr\Big(sign(\langle \boldsymbol{g}_n, \boldsymbol{g}_m \rangle) = sign(-\langle \mathbf{x}_n, \mathbf{x}_m \rangle)\Big) = \Pr(\beta > s_p),$$

Proof. According to Lemma 5.1 and $\mathbf{y}_n \neq \mathbf{y}_m$, we have

$$\langle \boldsymbol{\epsilon}_n, \boldsymbol{\epsilon}_m \rangle = \langle \boldsymbol{p}_n, \boldsymbol{p}_m \rangle - \boldsymbol{p}_{n,c_m} - \boldsymbol{p}_{m,c_n} = s_p - \beta$$

When $\beta > s_p$, we must have $\langle \boldsymbol{\epsilon}_n, \boldsymbol{\epsilon}_m \rangle < 0$. According to Lemma 5.1, we prove this theorem. □

Theorem 5.2 says that for samples from different classes, $\langle \boldsymbol{g_n}, \boldsymbol{g_m} \rangle$ gets an opposite sign of $\langle \mathbf{x}_n, \mathbf{x}_m \rangle$ with a probability that depends on the predictions $\boldsymbol{p}_n$ and $\boldsymbol{p}_m$. This probability of flipping the sign especially depends on β which reflects how likely the model will misclassify both samples to its opposite class in the pair.

Theorem 5.3. *Suppose $\mathbf{y}_n = \mathbf{y}_m$, when $\langle \boldsymbol{g}_n, \boldsymbol{g}_m \rangle \neq 0$, we have:*

$$sign(\langle \boldsymbol{g}_n, \boldsymbol{g}_m \rangle) = sign(\langle \mathbf{x}_n, \mathbf{x}_m \rangle)$$

Proof. Because $\sum_{k=1}^{K} \boldsymbol{p}_{n,k} = 1$, $\boldsymbol{p}_{n,k} \geq 0, \forall k$, and $c_n = c_m = c$,

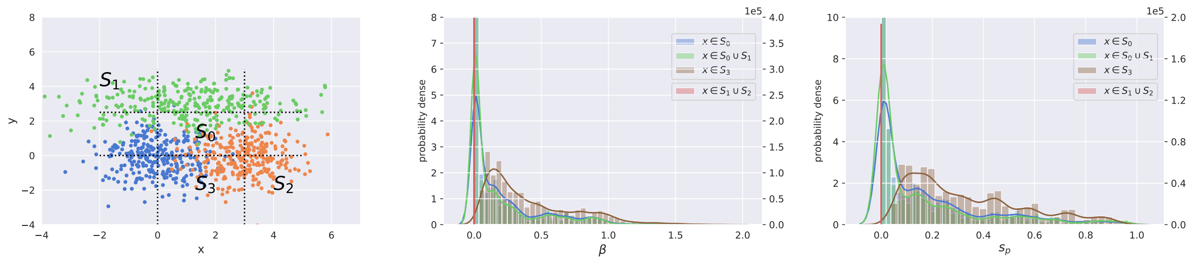
$$\langle \boldsymbol{\epsilon}_n, \boldsymbol{\epsilon}_m \rangle = \sum_{k \neq c}^{K} \boldsymbol{p}_{n,k} \boldsymbol{p}_{m,k} + (\boldsymbol{p}_{n,c} - 1)(\boldsymbol{p}_{m,c} - 1) \geq 0 \tag{5.3}$$

According to Lemma 5.1, we prove the theorem. □

Theorem 5.3 says that $\langle \boldsymbol{g}_n, \boldsymbol{g}_m \rangle$ has the same sign as $\langle \mathbf{x}_n, \mathbf{x}_m \rangle$ when the two samples are from the same class.

For a better understanding of the theorems, we conduct empirical study by partitioning the feature space of three classes into several subsets as shown in Fig. 5.2a and examine four cases of pairwise samples by these subsets: 1). $\mathbf{x} \in S_0$, both samples in a pair are near the intersection of the three classes; 2). $\mathbf{x} \in S_0 \cup S_1$, one sample is close to decision boundaries and the other is far away from the boundaries; 3). $\mathbf{x} \in S_3$, both samples close to the decision boundary between their true classes but away from the third class; 4). $\mathbf{x} \in S_1 \cup S_2$, both samples are far away from the decision boundaries.

By training a linear model, we show the empirical distributions of β and s_p upon the four subsets in Figs. 5.2b and 5.2c, respectively. In general, s_p shows similar behaviors with β in the four cases but in a smaller range, which makes $\beta > s_p$ tends to be true except when β is close to zero. Basically, a subset including more samples close to decision boundaries leads to more probability mass on larger values of β, and the case of $\mathbf{x} \in S_3$ results in largest mass on larger values of β because the predicted probabilities mostly concentrate on the two classes in a pair. As shown in Tab. 5.1, more mass on larger values of β leads to larger probabilities of flipping the sign. These results demonstrate that samples with most diverse gradients (i.e., gradients that have largely negative similarities with other samples) are close to decision boundaries because the model tends to classify such samples as the opposite class and hence the predictions result in a large β. When $\mathbf{x} \in S_1 \cup S_2$, $\langle \boldsymbol{g_n}, \boldsymbol{g_m} \rangle$ is also mostly negative as shown in Tab. 5.1, because $\langle \mathbf{x}_n, \mathbf{x}_m \rangle$ is negative and the probability of flipping the sign is mostly zero due to β concentrates



(a) Splitting samples into several subsets in a 3-class classification task. Dots in different colors are from different classes.

(b) Estimated distributions of β when drawing negative pairs from different subsets.

(c) Estimated distributions of s_p when drawing negative pairs from different subsets.

Figure 5.2: Illustration of how $\Pr(\beta > s_p)$ in Theorem 5.2 behaves in various cases by drawing negative pairs from different subsets of a 3-class feature space. The subsets are displayed in Fig. 5.2a. The classifier is a linear model. y-axis in the right side of (b) & (c) is for the case of $x \in S_1 \cup S_2$. We see that s_p behaves in a similar way with β but in a smaller range which makes β the key in studying $\Pr(\beta > s_p)$. In the case of $x \in S_3$ the distribution of β has more mass on larger values than other cases because the predicted probabilities are mostly on the two classes in a pair, and it causes all $\langle \boldsymbol{g}_n, \boldsymbol{g}_m \rangle$ having the opposite sign of $\langle \mathbf{x}_n, \mathbf{x}_m \rangle$ as shown in Tab. 5.1.

Table 5.1: Illustration of the Theorems by drawing pairs from different subsets that are defined in Fig. 5.2a. We obtain the gradients and predictions by a linear model and a MLP with two hidden layers (16 units for each) and ReLU (or tanh) activations. The gradients are computed using all parameters of the model. We can see that the non-linear models exhibit similar behaviors with the linear model as described in the theorems. One exception is that the MLP with ReLU activations gets much less negative $\langle \boldsymbol{g}_n, \boldsymbol{g}_m \rangle$ in the case of $S_1 \cup S_2$ for negative pairs, we consider the difference is caused by representations to the final linear layer always being positive in this case due to ReLU activations.

		Negative pairs (Thm. 1)				Positive pairs (Thm.2)			
		S_0	$S_0 \cup S_1$	S_3	$S_1 \cup S_2$	S_0	$S_0 \cup S_1$	S_3	$S_1 \cup S_2$
	$\Pr(\langle \mathbf{x}_n, \mathbf{x}_m \rangle > 0)$	1.	0.877	1.	0.	1.	0.99	1.	1.
Linear	$\Pr(\langle \boldsymbol{g}_n, \boldsymbol{g}_m \rangle < 0)$	0.727	0.725	1.	0.978	0.	0.007	0.	0.
	$\Pr(\beta > s_p)$	0.727	0.687	1.	0.	–	–	–	–
MLP (ReLU)	$\Pr(\langle \boldsymbol{g}_n, \boldsymbol{g}_m \rangle < 0)$	0.72	0.699	1.	0.21	0.013	0.01	0.	0.
	$\Pr(\beta > s_p)$	0.746	0.701	1.	0.	–	–	–	–
MLP (tanh)	$\Pr(\langle \boldsymbol{g}_n, \boldsymbol{g}_m \rangle < 0)$	0.745	0.744	1.	0.993	0.004	0.007	0.	0.
	$\Pr(\beta > s_p)$	0.766	0.734	1.	0.	–	–	–	–

around zero. $\langle \boldsymbol{g_n}, \boldsymbol{g_m} \rangle$ are also close to zero in this case according to Lemma 5.1 as the predictions are close to true labels, hence, such samples are not considered with most diverse gradients.

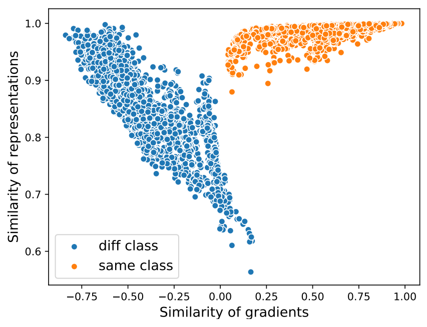
We can see that the results of positive pairs in Tab. 5.1 match Theorem 5.3. In the case of $S_0 \cup S_1$ the probabilities from the linear model do not add up to exactly 1 because the implementation of cross-entropy loss in tensorflow smooths the function by a small value for preventing numerical issues which slightly changes the gradients. As $\langle \mathbf{x}_n, \mathbf{x}_m \rangle$ is mostly positive for positive pairs, $\langle \boldsymbol{g}_n, \boldsymbol{g}_m \rangle$ hence is also mostly positive. On the other hand, if $\langle \mathbf{x}_n, \mathbf{x}_m \rangle$ is negative then $\langle \boldsymbol{g}_n, \boldsymbol{g}_m \rangle$

will be negative, which indicates representations within a class are expected to have non-negative inner products.

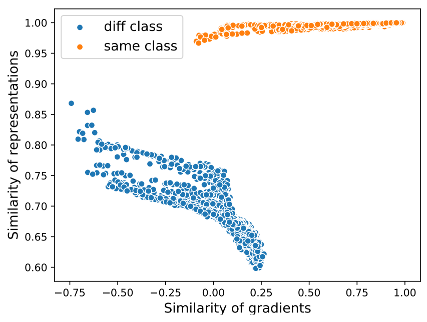
Extending the theoretical analysis based the a linear model, we also provide empirical study of non-linear models (MLPs). As demonstrated in Tab. 5.1, $\Pr(\beta > s_p)$ in MLPs are very similar with the linear model since it only depends on the predictions and all models have learned reasonable decision boundaries. $\Pr(\langle \boldsymbol{g}_n, \boldsymbol{g}_m \rangle < 0)$ is also similar with the linear model except in the case of $S_1 \cup S_2$ for negative pairs, in which case the MLP with ReLU gets much less negative $\langle \boldsymbol{g}_n, \boldsymbol{g}_m \rangle$. As MLP with tanh activations is still consistent with the linear model in this case, we consider the difference is caused by the representations always being positive due to ReLU activations. These results demonstrate that non-linear models exhibit similar behaviors with linear models and mostly align with the theorems.

Since only negative $\langle \boldsymbol{g}_n, \boldsymbol{g}_m \rangle$ causes conflicts when updating parameters, we consider reducing the diversity of gradients by reducing negative $\langle \boldsymbol{g}_n, \boldsymbol{g}_m \rangle$, which could have two ways: 1) minimizing the representation inner product of negative pairs, which pushes its value to be negative or zero (for positive representations); 2) decreasing the probability of flipping the sign ($\Pr(\beta > s_p)$). According to Theorem 5.2, minimizing the representation similarity of negative pairs may not only help with the first way but also the second way since more discriminative representations are easier for prediction. In addition, according to Fig. 5.2 and Tab. 5.1 larger s_p likely accompanies larger $\Pr(\beta > s_p)$. For example, $\mathbf{x} \sim S_3$ gets larger prediction similarities than $\mathbf{x} \sim S_0$ due to the predictions put most probability mass on both classes of a pair, which results in larger β and hence larger $\Pr(\beta > s_p)$. For this reason, we consider to minimize the prediction similarity as well, hence, we also include logits in the representations.

We verify the relation between gradient similarity and representation similarity by training two binary classifiers for two groups of MNIST classes ($\{0,1\}$ and $\{7,9\}$). The classifiers have two hidden layers each with 100 hidden units and ReLU activations. We randomly chose 100 test samples from each group to compute the pairwise cosine similarities. Representations are obtained by concatenating the output of all layers (including logits) of the neural network. We concatenate outputs of all layers as the representation. Gradients are computed by all parameters of the model. We display the similarities in Figs. 5.3a and 5.3b. The correlation coefficients between the gradient and representation similarities of negative pairs are -0.86 and -0.85 respectively, which of positive pairs are 0.71 and 0.79 respectively. In all cases, the similarities of representations show strong correlations with the similarities of gradients. The classifier for class 0 and 1 gets smaller representation similarities and much less negative gradient similarities for negative pairs (blue dots). It also gains a higher accuracy than the other classifier (99.95% vs. 96.25%), which illustrates the potential of reducing the gradient diversity by decreasing the representation similarity of negative pairs.



(a) Similarities of gradients *vs.* representations (class 7 & 9)



(b) Similarities of gradients *vs.* representations (class 0 & 1)

Figure 5.3: Similarities of gradients and representations of two classes in the MNIST dataset. The x and y axis are the cosine similarity of gradients and representations, respectively. Blue dots indicate the similarity of *negative pairs*, while orange dots indicate that of *positive pairs*.

5.2.2 Connection with Deep Metric Learning

Reducing the representation similarity between classes shares the same concept as learning larger margins which has been an active research area for a few decades. For example, Kernel Fisher Discriminant analysis (KFD) (Mika et al., 1999) and distance metric learning (Weinberger et al., 2006) aim to learn kernels that can obtain larger margins in an implicit representation space, whereas Deep Metric Learning (DML) (Kaya & Bilge, 2019; Roth et al., 2020) leverages deep neural networks to learn embedding that maximize margins in an explicit representation space. In this sense, DML has the potential to help with reducing the diversity of gradients in continual learning.

However, the usual concepts in DML may not entirely be appropriate for continual learning, as they also aim to learn compact representations within classes (Schroff et al., 2015; Wang et al., 2017; Deng et al., 2019). In continual learning, the unused information for the current task might be important for a future task, e.g. the results of Figs. 5.1b and 5.1c indicate that compact representations of a current task might omit important information in the data distribution for a future task. Even if we store diverse samples into the memory, the learned representations may be difficult to generalize on future tasks when the necessary dimensions are omitted in early tasks. It is because the omitted dimensions can only be relearned by using limited samples in the memory. We demonstrate this by training a model with and without L1 regularization at the first task of split-MNIST and split-Fashion MNIST. In order to verify the influence of compact representations more precisely, we remove the L1 regularization in the later tasks so that the model is flexible to relearn omitted dimensions without extra penalty. The results are shown in Tab. 5.2. We see that with L1 regularization the model learns much more compact representations in the first task and gives a similar performance with the one not adding L1 loss. However, the performance suffers from larger and larger degradation when more tasks have been encountered. The results suggest that less compact representation space may be beneficial to preserve necessary information for future tasks and it is difficult to enrich representations

Table 5.2: Demonstration of performance degradation in continual learning by compact representations. We test tasks of split MNIST and split Fashion-MNIST by training a MLP (2 hidden layers with 100 units per layer and ReLU activations) with and without L1 regularization at the first task. The memory is formed by 300 samples that are randomly chosen. Representations are outputs of hidden layers. We identify active dimensions of the representation space after learning task 1 by selecting the hidden units that have a mean activation larger than 0.5 over all learned classes.

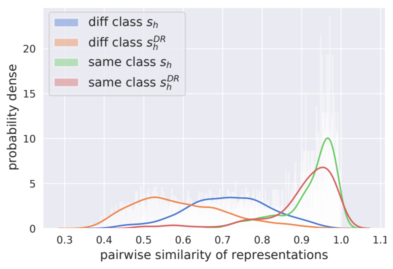
	L1 (t=1)	# Act. Dim. (t=1)	Avg. Accuracy (in %)				
			t=1	t=2	t=3	t=4	t=5
Split-MNIST	no	51	99.9	97.4	93.7	90.6	85.4
	yes	5	99.7	93.0	87.1	68.4	53.5
Split-Fashion	no	66	98.3	90.3	83.1	74.4	77.3
	yes	8	97.4	88.9	75.7	56.1	50.2

over learned tasks with small memories when learning a new task. Therefore, we propose an opposite way to DML regarding the within-class compactness: minimizing the similarities within the same class for obtaining less compact representation space.

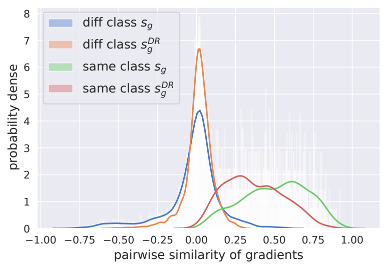
Roth et al. (2020) proposed a ρ-spectrum metric to measure the information entropy contained in the representation space: $\rho = D_{KL}(\mathscr{U} || S_{\Phi_{\mathscr{X}}})$. The ρ-spectrum computes the KL-divergence between a discrete uniform distribution $\mathscr{U}$ and the spectrum of data representations $S_{\Phi_{\mathscr{X}}}$, where $S_{\Phi_{\mathscr{X}}}$ is normalized and sorted singular values of the representation matrix $\Phi(\mathscr{X})$, Φ denotes the representation extractor (e.g. a neural network) and $\mathscr{X}$ is input data samples. Lower values of ρ indicate higher variance of the representations and hence more information entropy retained. Roth et al. (2020) also introduced a ρ-regularization method to restrain over-compression of representations. The ρ-regularization method randomly replaces negative pairs by positive pairs with a pre-selected probability p_ρ. Nevertheless, switching pairs is inefficient and may be detrimental to the performance in an online setting because some samples may never be learned in this way. Thus, we propose a different way to restrain the compression of representations which will be introduced in the following section.

5.3 Discriminative Representation Loss

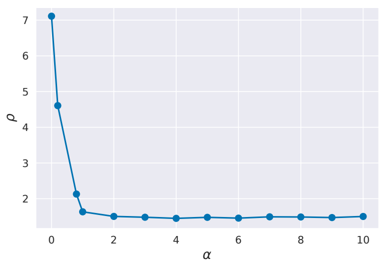
Based on our findings in the above section, we propose an auxiliary objective Discriminative Representation Loss (DRL) for classification tasks in continual learning, which is straightforward and efficient. Instead of explicitly re-projecting gradients during training process, DRL helps with decreasing gradient diversity by optimizing the representations. As defined in Eq. (5.4), DRL consists of two parts: one is for minimizing the inner products of representations from different classes ($\mathscr{L}_{bt}$) which can reduce the diversity of gradients from different classes, the other is for minimizing the inner products of representations from a same class ($\mathscr{L}_{wi}$) which helps preserve



(a) Similarities of representations with and without $\mathscr{L}_{DRL}$



(b) Similarities of gradients with and without $\mathscr{L}_{DRL}$



(c) Relation between α and ρ-spectrum.

Figure 5.4: Effects of $\mathscr{L}_{DRL}$ on reducing diversity of gradients and ρ-spectrum. (a) and (b) display distributions of similarities of representations and gradients. s_h^{DR} and s_h denote similarities of representations with and without $\mathscr{L}_{DRL}$, respectively, s_g^{DR} and s_g denote similarities of gradients with and without $\mathscr{L}_{DRL}$, respectively. (c) demonstrates increasing α in $\mathscr{L}_{DRL}$ can reduce ρ effectively.

discriminative information for future tasks in continual learning.

$$\mathscr{L}_{DRL} = \mathscr{L}_{bt} + \alpha \mathscr{L}_{wi}, \quad \alpha > 0,$$

$$\mathscr{L}_{bt} = \frac{1}{N_{bt}} \sum_{i=1}^{B} \sum_{j \neq i, y_j \neq y_i}^{B} \langle h_i, h_j \rangle, \quad \mathscr{L}_{wi} = \frac{1}{N_{wi}} \sum_{i=1}^{B} \sum_{j \neq i, y_j = y_i}^{B} \langle h_i, h_j \rangle, \tag{5.4}$$

B is training batch size. N_{bt}, N_{wi} are the number of negative and positive pairs, respectively. α is a hyperparameter controlling the strength of $\mathscr{L}_{wi}$, h_i is the representation of $\mathbf{x}_i$, $\mathbf{y}_i$ is the label of $\mathbf{x}_i$. The final loss function combines the commonly used softmax cross entropy loss for classification tasks ($\mathscr{L}$) with DRL ($\mathscr{L}_{DRL}$) as below:

$$\widehat{\mathscr{L}} = \mathscr{L} + \lambda \mathscr{L}_{DRL}, \quad \lambda > 0, \tag{5.5}$$

where λ is a hyperparameter controlling the strength of $\mathscr{L}_{DRL}$. We provide experimental results of an ablation study on $\mathscr{L}_{bt}$ and $\mathscr{L}_{wi}$ in Tab. 5.8 in Sec. 5.5, according to which $\mathscr{L}_{bt}$ and $\mathscr{L}_{wi}$ both have shown effectiveness on improving the performance. We also show the correlation between ρ-spectrum and the model performance in Tab. 5.6 in Sec. 5.5.

We verify the effects of $\mathscr{L}_{DRL}$ by training a model with/without $\mathscr{L}_{DRL}$ on Split-MNIST tasks: Fig. 5.4a shows that $\mathscr{L}_{DRL}$ notably reduces the representation similarity of negative pairs while making representations within a class less similar; Fig. 5.4b shows more probability mass of the gradient similarity of negative pairs are pushed around zero and more probability mass of positive pairs are pushed towards zero. Fig. 5.4c demonstrates increasing α can effectively decrease ρ-spectrum to a low-value level, where lower values of ρ indicate higher variance of the representations and hence more information entropy retained. These results confirm that DRL can achieve our expectation for reducing negative gradient similarities and preventing over compactness of representations. We will provide experimental results for a more comprehensive evaluation of model performance in sec. 5.5.

Algorithm 1: Ring Buffer Update with Fixed Buffer Size

```
Input: $\mathbb{B}_t$ - current data batch,
       $\mathbb{C}_t$ - the set of classes in $\mathbb{B}_t$,
       $\mathcal{M}$ - memory buffer,
       $\mathbb{C}$ - the set of classes in $\mathcal{M}$,
       $K$ - memory buffer size.
for $c$ in $\mathbb{C}_t$ do
    Get $\mathbb{B}_{t,c}$ - samples of class $c$ in $\mathbb{B}_t$,
    $\mathcal{M}_c$ - samples of class $c$ in $\mathcal{M}$,
    if $c$ in $\mathbb{C}$ then
        $\mathcal{M}_c = \mathcal{M}_c \cup \mathbb{B}_c$
    else
        $\mathcal{M}_c = \mathbb{B}_c, \quad \mathbb{C} = \mathbb{C} \cup \{c\}$
    end if
end for
$R = |\mathcal{M}| + |\mathbb{B}| - K$
while $R > 0$ do
    $c' = \arg\max_c |\mathcal{M}_c|$
    remove the first sample in $\mathcal{M}_{c'}$,
    $R = R - 1$
end while
return $\mathcal{M}$
```

Algorithm 2: Balanced Experience Replay

```
Input: $\mathcal{M}$ - memory buffer,
       $\mathbb{C}$ - the set of classes in $\mathcal{M}$,
       $B$ - training batch size,
       $\Theta$ - model parameters,
       $\mathcal{L}_\Theta$ - loss function,
       $\mathbb{B}_t$ - current data batch,
       $\mathbb{C}_t$ - the set of classes in $\mathbb{B}_t$,
       $K$ - memory buffer size.
$\mathcal{M} \leftarrow$ MemoryUpdate($\mathbb{B}_t, \mathcal{C}_t, \mathcal{M}, \mathbb{C}, K$)
$n_c, \mathbb{C}_s, \mathbb{C}_r \leftarrow$ ClassSelection($\mathbb{C}_t, \mathbb{C}, B$)
$\mathbb{B}_{train} = \emptyset$
for $c$ in $\mathbb{C}_s$ do
    if $c$ in $\mathbb{C}_r$ then
        $m_c = n_c + 1$
    else
        $m_c = n_c$
    end if
    Get $\mathcal{M}_c \lhd$ samples of class $c$ in $\mathcal{M}$,
    $\mathbb{B}_c \overset{m_c}{\sim} \mathcal{M}_c \lhd$ sample $m_c$ samples from $\mathcal{M}_c$
    $\mathbb{B}_{train} = \mathbb{B}_{train} \cup \mathbb{B}_c$
end for
$\Theta \leftarrow$ Optimizer($\mathbb{B}_{train}, \Theta, \mathcal{L}_\Theta$)
```

The computational complexity of DRL is $O(B^2H)$, where B is training batch size, H is the dimension of representations. B is usually small (not larger than 20) in related work with the online setting (Chaudhry et al., 2019a,b; Aljundi et al., 2019b), and commonly $H \ll W$, where W is the number of network parameters. In comparison, the computational complexity of A-GEM (Chaudhry et al., 2019a) and GSS-greedy (Aljundi et al., 2019b) are $O(B_rW)$ and $O(BB_mW)$, respectively, where B_r is the reference batch size in A-GEM and B_m is the memory batch size in GSS-greedy. The computational complexity discussed here is additional to the cost of common back-propagation. We compare the training time of all methods in Tab. 5.7, which shows the representation-based methods are much faster than gradient-based approaches.

Since DRL depends on the negative and positive pairs in the training batch, we suggest a bit more sophisticated replay strategy than vanilla ER to work with DRL and introduce it in the next section.

5.4 Online memory update and Balanced Experience Replay

We follow the *online setting* of continual learning as was done for other gradient-based approaches with episodic memories (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019a; Aljundi

Algorithm 3: Class Selection for Balanced Experience Replay (BER)

```
Input: $\mathbb{C}_t$ - the set of classes in current data batch $\mathbb{B}_t$,
       $\mathbb{C}$ - the set of classes in the memory $\mathcal{M}$,
       $B$ - training batch size,
       $m_p$ - minimum number of positive pairs ($m_p \in \{0,1\}$) .
$n_c = \lfloor B/|\mathbb{C}| \rfloor, \ r_c = B \mod |\mathbb{C}|$,
if $B > |\mathbb{C}|$ or $m_p == 0$ then
    $\mathbb{C}_r \overset{r_c}{\sim} \mathbb{C}$  $\lhd$ sample $r_c$ classes from all seen classes without replacement.
    $\mathbb{C}_s = \mathbb{C}$
else
    $\mathbb{C}_r = \emptyset$, $n_c = 1$, $n_s = B - |\mathbb{C}_t|$, $\lhd$ ensure the training batch including samples
        from the current task.
    $\mathbb{C}_s \overset{n_s-1}{\sim} (\mathbb{C} - \mathbb{C}_t)$ $\lhd$ sample $n_s - 1$ classes from all seen classes except classes in $\mathbb{C}_t$.
    $\mathbb{C}_s = \mathbb{C}_s \bigcup \mathbb{C}_t$,
    $\mathbb{C}_r \overset{1}{\sim} \mathbb{C}_s$ $\lhd$ sample one class to have a positive pair
end if
Return: $n_c, \mathbb{C}_s, \mathbb{C}_r$
```

et al., 2019b), in which the model only trained with one epoch on the training data. We update the episodic memories by the basic ring buffer strategy: keep the last n_c samples of class c in the memory buffer, where n_c is the memory size of a seen class c. We have deployed the episodic memories with a fixed size, implying a fixed budget for the memory cost. Further, we maintain a uniform distribution over all seen classes in the memory. The buffer may not be evenly allocated to each class before enough samples are acquired for newly arriving classes. We show pseudo-code of the memory update strategy in Alg. 1 for a clearer explanation. For class-incremental learning, this strategy can work without clear task boundaries, i.e., the model may still see samples from previous classes when a new class has joined.

Since DRL and methods of DML depend on the pairwise similarities of samples, we would prefer the training batch to include as wide a variety of different classes as possible to obtain sufficient discriminative information. Hence, we adjust the ER strategy (Chaudhry et al., 2019b) for the needs of such methods. The idea is to uniformly sample from seen classes in the memory buffer to form a training batch, so that this batch can contain as many seen classes as possible. Moreover, we ensure the training batch includes at least one positive pair to enable the parts computed by positive pairs in the loss when the number of learned classes is much larger than the training batch size. In addition, we also ensure the training batch includes at least one class from the current task. We call this Balanced Experience Replay (BER). The pseudo code is in Alg. 2. Note that we update the memory and form the training batch based on the task ID instead of class ID for Domain-IL tasks (e.g. permuted MNIST tasks), as in this case each task always includes the same set of classes.

We provide the details of online ring buffer update and Balanced Experience Replay (BER) in

Algs. 1 to 3. We directly load new data batches into the memory buffer without a separate buffer for the current task. The memory buffer works like a sliding window for each class in the data stream and we draw training batches from the memory buffer instead of directly from the data stream. In this case, one sample may not be seen only once as long as it stays in the memory buffer. This strategy is a more efficient use of the memory when $|\mathbb{B}| < n_c$, where $|\mathbb{B}|$ is the loading batch size of the data stream (i.e., the number of new samples added into the memory buffer at each iteration), we set $|\mathbb{B}|$ to 1 in all experiments and we will have a discussion of this in the following section.

5.5 Experiments

In this section we provide comprehensive experimental results for evaluating our method DRL and we also provide an ablation study on DRL from various aspects to obtain more insights on it. The results show that DRL is efficient to alleviate forgetting in the online setting with limited training data and without the need of task identifiers during testing, which is the most difficult situation among the application scenarios of continual learning.

5.5.1 Comparing DRL with other baselines

We evaluate our methods on multiple benchmark tasks by comparing with several baseline methods in the setting of online continual learning.

We have conducted experiments on the following *Benchmark tasks* that have been introduced in Sec. 3.1, here we give the settings of training size and memory size. The setting of tasks on MNIST and CIFAR10 follows (Aljundi et al., 2019b):

1) Permuted MNIST: the training size is 1000 samples per task and memory size is 300;

2) Split MNIST: the training size is 1000 samples per task and memory size is 300;

3) Split Fashion-MNIST: the training size is 1000 samples per task and memory size is 300;

4) Split CIFAR-10: the training size is 2000 samples per task and memory size is 1000;

5) Split CIFAR-100: the training size is 5000 samples per task and memory size is 5000.

6) Split TinyImageNet: the training size is 5000 samples per task and memory size is 5000.

N.B.: We use *single-head* (shared output) models in all of our experiments, meaning that we comply with the task-agnostic model protocol (no task identifiers during testing) and the online training protocol. Such settings are more difficult for continual learning but more practical in real applications. Our experimental results show worse performance than those in some related works (Delange et al., 2021; Requeima et al., 2019) due to these settings as well.

We compare our methods with 6 *Baselines*: two gradient-based baselines (*A-GEM* (Chaudhry et al., 2019a) and *GSS-greedy* (Aljundi et al., 2019b)), two standalone experience replay methods

(*ER* (Chaudhry et al., 2019b) and *BER*), two SOTA methods of DML (*Multisimilarity* (Wang et al., 2019) and *R-Margin* (Roth et al., 2020)). These methods are introduced in Sec. 3.3 except for the last two. We provide a brief introduction of *Multisimilarity* and *R-Margin* in the following:

Multisimilarity (Wang et al., 2019): A SOTA method of DML which has shown outstanding performance in a comprehensive empirical study of DML (Roth et al., 2020). We adopt the loss function of Multisimilarity as an auxiliary objective in classification tasks of continual learning, the batch mining process is omitted because we use labels for choosing positive and negative pairs. The loss of Multisimilarity is defined as below:

(5.6)
$$\mathcal{L}_{multi} = \frac{1}{B}\sum_{i=1}^{B}\left[\frac{1}{\alpha}\log[1+\sum_{j\neq i, y_j=y_i}\exp(-\alpha(s_c(h_i,h_j)-\gamma))] + \frac{1}{\beta}\log[1+\sum_{y_j\neq y_i}\exp(\beta(s_c(h_i,h_j)-\gamma))]\right]$$

where $s_c(\cdot,\cdot)$ is cosine similarity, α, β, γ are hyperparameters.

R-Margin (Roth et al., 2020): A SOTA method of DML which deploy the ρ regularization method for Margin loss (Wu et al., 2017) and has shown outstanding performance in Roth et al. (2020). We similarly deploy R-Margin for continual learning as an auxiliary objective, which uses the Margin loss (Wu et al., 2017) with the ρ regularization (Roth et al., 2020) as introduced in Sec. 5.2.2. The loss is defined as below:

$$\mathcal{L}_{margin} = \sum_{i=1}^{B}\sum_{j=1}^{B}\gamma + \mathbf{I}_{j\neq i, y_j=y_i}(d(h_i,h_j)-\beta) - \mathbf{I}_{y_j\neq y_i}(d(h_i,h_j)-\beta) \tag{5.7}$$

where $d(\cdot,\cdot)$ is Euclidean distance,$\mathbf{I}$ denotes the indicator function, β is a trainable variable and γ is a hyperparameter.

N.B.: We deploy the losses of Multisimilarity and R-Margin as auxiliary objectives as the same as DRL because using standalone such losses causes difficulties of convergence in our experimental settings.

We use the *Average accuracy*, *Average forgetting*, *Average intransigence* as performance measures, the definitions of which are provided in Sec. 3.2.

Regarding the *Experimental settings* of the models, we use the vanilla SGD optimizer for all experiments without any scheduling; all networks are trained from scratch without any preprocessing of data except normalization. We use a MLP with two hidden layers and ReLU activations for tasks on MNIST and Fashion-MNIST, and each layer has 100 hidden units. For tasks on CIFAR data sets and TinyImageNet we use the same reduced Resnet18 as used in Chaudhry et al. (2019a). We concatenate outputs of all layers as representation in MLPs. For Resnet18, representations are the concatenation of outputs of the final hidden layer and the linear layer. We consider the outputs of hidden layers behave like different levels of the representation, and when higher layers (layers closer to the input) generate more discriminative representations it would be easier for lower layers to learn more discriminative representations as well. It improves the performance of MLPs in our observations. For ResNet18 we found that including outputs of higher hidden layers performs almost as the same as only including the final representation, so we just include the final hidden layer for lower computational cost, and this also a common setting for conventional neural networks in the literature of DML. We deploy BER (Alg. 2) as the replay strategy for DRL, Multisimilarity, and R-Margin. The standard deviation shown in all results are evaluated over 10 runs with different random seeds. We use 10% of training set as validation set for choosing hyperparameters by cross validation. To make a fair comparison of all methods, the configurations of GSS-greedy are as suggested in Aljundi et al. (2019b), with batch size set to 10 and each batch receives multiple iterations. For the other methods, we use the ring buffer memory as described in Alg. 1, the loading batch size is set to 1, following with one iteration, the training batch size is provided in Tab. 5.10. More hyperparameters are given in Tab. 5.10 as well.

Table 5.3: Average accuracy (in %), the higher the better, the bold font indicates the best performance on this criterion

	P-MNIST	S-MNIST	Fashion	CIFAR10	CIFAR100	TinyImageNet
DRL	**80.5 ± 0.4**	**88.1 ± 0.6**	**77.9 ± 0.8**	**40.4 ± 1.5**	**19.3 ± 0.5**	**8.3 ± 0.2**
BER	79.2 ± 0.3	85.2 ± 1.1	77.0 ± 0.7	37.3 ± 1.4	18.2 ± 0.4	6.7 ± 0.8
ER	78.2 ± 0.6	83.2 ± 1.5	75.8 ± 1.4	39.4 ± 1.6	18.3 ± 0.3	7.6 ± 0.6
A-GEM	76.7 ± 0.5	84.5 ± 1.1	66.4 ± 1.5	25.7 ± 3.3	16.5 ± 1.2	2.0 ± 0.8
GSS	77.1 ± 0.3	82.8 ± 1.8	72.5 ± 0.9	33.6 ± 1.7	13.9 ± 1.0	3.3 ± 0.2
Multisim	79.5 ± 0.6	86.3 ± 1.1	77.2 ± 0.6	38.9 ± 3.2	18.4 ± 0.5	6.8 ± 1.0
R-Margin	78.0 ± 0.3	85.6 ± 0.9	76.6 ± 0.8	36.7 ± 1.7	18.3 ± 0.5	6.4 ± 0.3

Table 5.4: Average forgetting (in %), the lower the better, the bold font indicates the best performance on this criterion

	P-MNIST	S-MNIST	Fashion	CIFAR10	CIFAR100	TinyImageNet
DRL	**4.5 ± 0.2**	**8.9 ± 0.7**	**16.6 ± 1.9**	41.4 ± 3.2	29.0 ± 1.0	27.0 ± 1.4
BER	5.1 ± 0.3	13.0 ± 1.5	18.2 ± 2.7	52.3 ± 1.6	35.9 ± 1.0	26.0 ± 1.2
ER	7.1 ± 0.6	17.2 ± 1.9	24.0 ± 2.7	50.3 ± 1.9	37.0 ± 1.5	28.5 ± 2.0
A-GEM	5.4 ± 0.4	12.6 ± 1.3	37.0 ± 1.9	40.4 ± 4.6	25.3 ± 1.4	24.4 ± 0.6
GSS	7.6 ± 0.2	17.9 ± 2.4	27.4 ± 2.2	**27.6 ± 4.0**	**18.6 ± 0.7**	**11.3 ± 0.7**
Multisim	5.0 ± 0.7	12.0 ± 1.4	18.8 ± 2.2	48.0 ± 3.8	35.8 ± 0.6	37.6 ± 0.4
R-Margin	5.4 ± 0.3	12.5 ± 1.4	17.1 ± 3.0	50.5 ± 2.6	35.1 ± 0.5	38.1 ± 0.7

As we follow the online setting of training on limited data with one epoch, we either use a small loading-batch size or iterate on one batch several times to obtain necessary steps for gradient optimization. We chose a small batch size with one iteration instead of larger batch size with multiple iterations because by our memory update strategy (Alg. 1) it achieves similar performance without tuning the number of iterations. Since GSS-greedy has a different strategy for updating memories, we use its reported settings in (Aljundi et al., 2019b).

Table 5.5: Average intransigence (in %), the lower the better, the bold font indicates the best performance on this criterion

	P-MNIST	S-MNIST	Fashion	CIFAR10	CIFAR100	TinyImageNet
DRL	2.0 ± 0.5	2.8 ± 0.3	6.9 ± 1.0	9.9 ± 1.5	9.0 ± 1.0	25.0 ± 7.5
BER	3.0 ± 0.2	2.5 ± 0.3	7.7 ± 1.2	4.3 ± 0.6	4.0 ± 0.6	28.3 ± 0.5
ER	$\mathbf{1.8 \pm 0.4}$	1.2 ± 0.1	4.1 ± 0.6	$\mathbf{3.8 \pm 1.0}$	$\mathbf{2.8 \pm 1.0}$	11.8 ± 2.5
A-GEM	7.0 ± 0.7	3.5 ± 0.3	$\mathbf{1.0 \pm 0.3}$	25.4 ± 2.1	15.2 ± 1.0	21.4 ± 1.3
GSS	7.6 ± 0.2	$\mathbf{0.8 \pm 0.3}$	27.4 ± 2.2	27.9 ± 2.0	22.3 ± 1.4	45.0 ± 0.1
Multisim	2.6 ± 0.2	2.2 ± 0.3	6.2 ± 1.0	6.1 ± 1.1	3.8 ± 0.6	4.0 ± 1.2
R-Margin	3.6 ± 0.4	2.4 ± 0.3	10.5 ± 1.7	6.3 ± 2.1	4.5 ± 0.7	$\mathbf{3.9 \pm 0.4}$

Table 5.6: Correlation between model performance and ρ-spectrum on all benchmark tasks

Coefficient	P-MNIST	S-MNIST	Split Fashion	CIFAR10	CIFAR100	TinyImageNet
Avg. Acc.	−0.3997	−0.2870	−0.8189	−0.3214	−0.6833	−0.8138
Avg. Forg.	0.2684	0.0959	0.7649	−0.4076	−0.7127	−0.4296
Avg. Intran.	0.1264	0.3831	−0.5035	0.4550	0.8112	0.4532

Tabs. 5.3 to 5.5 give the average accuracy, forgetting, and intransigence of all methods on all benchmark tasks, respectively. TinyImageNet gets much worse performance than other benchmarks because it is the most difficult one as it has more classes (200), a longer task sequence (20 tasks), and higher feature dimensions ($64 \times 64 \times 3$). As we can see, the forgetting and intransigence often contradictory to each other which is a common phenomenon in continual learning. Our method DRL is able to get a better trade-off between them and thus outperforms other methods over most benchmark tasks in terms of average accuracy. We provide an ablation study of the two components in DRL in Tab. 5.8 which shows $\mathscr{L}_{wi}$ helps with obtaining an improved intransigence and $\mathscr{L}_{bt}$ brings an improved forgetting in most cases. The two terms in DRL are complementary to each other and combining them brings benefits on both sides. In addition, Multisimilarity and R-Margin both have shown relatively good performance, which indicates learning a better representation could be a more efficient way than direct gradient re-projection.

As shown in Tab. 5.6 the ρ-spectrum shows strong correlation to average accuracy on several benchmarks. In addition, it always has a negative correlation to average accuracy and mostly a positive correlation to average intransigence. The reason could be the ρ-spectrum helps with learning new decision boundaries across tasks which facilitates accommodating new tasks in most cases. On Split Fashion-MNIST the ρ-spectrum has a positive correlation to average forgetting and a negative correlation to average intransigence which could be because the learned classes have more influence on new decision boundaries in this case. In general, the ρ-spectrum is the smaller the better because it indicates the representations are more informative. However, it may be detrimental to the performance when ρ is too small because the learned representations are too noisy to classify as demonstrated in (Roth et al., 2020).

Tab. 5.7 compares the training time of all methods on several benchmarks. We can see that simply memory-replay methods ER and BER are faster than others; representation-based methods DRL, Multisimilarity, and R-Margin take similar training time with memory-replay methods; the gradient-based methods A-GEM and GSS are much slower than others, especially on a larger model. The experiments with the MLP have been tested on a laptop with an 8-core Intel CPU and 32G RAM, the experiments with the reduced Resnet18 have been tested on a server with a NVIDIA TITAN V GPU.

Table 5.7: Training time (in seconds) of the whole task sequence of several benchmarks.

	DRL	BER	ER	A-GEM	GSS	Multisim	R-Margin
P-MNIST (MLP)	12.48 ± 0.16	11.17 ± 0.18	$\mathbf{10.38 \pm 0.05}$	28.0 ± 0.09	33.98 ± 0.6	12.91 ± 0.13	13.45 ± 0.14
S-MNIST (MLP)	5.6 ± 0.14	5.29 ± 0.08	$\mathbf{5.25 \pm 0.02}$	13.41 ± 0.47	19.07 ± 1.31	5.89 ± 0.09	6.29 ± 0.43
CIFAR10 (r. Resnet18)	265.1 ± 0.9	264.3 ± 0.6	$\mathbf{261.5 \pm 0.3}$	1067.1 ± 5.6	5289.4 ± 7.3	286.4 ± 0.7	281.7 ± 0.8

5.5.2 Ablation study on DRL

In this section we provide results of an ablation study on DRL. We have conducted a series of experiments to obtain more insights on DRL, including comparing the two terms of within/between-classes in DRL, comparing different memory-replay strategies and memory cost with DRL.

Comparing the two terms of DRL: Tab. 5.8 provides the results of comparing the effects of the two terms in DRL. In general, both of them show improvements on standalone BER in most cases. $\mathscr{L}_{bt}$ gets more improvements on forgetting, $\mathscr{L}_{wi}$ gets more improvements on intransigence. Overall, combining the two terms obtains a better trade-off between forgetting and intransigence. It indicates preventing over-compact representations while maximizing margins can improve the learned representations that are easier for generalization over previous and new tasks. Regarding the weights on the two terms, a larger weight on $\mathscr{L}_{wi}$ is for less compact representations within classes, but a too dispersed representation space may include too much noise. We set α to 1 or 2 in our experiments (Tab. 5.10). A larger weight on $\mathscr{L}_{bt}$ is more resistant to forgetting but may be less capable of transferring to a new task. In addition, we notice that with standalone $\mathscr{L}_{bt}$ we can only use a smaller λ than with both terms otherwise the gradients will explode in a high probability. We show the hyperparameters of all methods in Tab. 5.10.

Comparing different replay strategies: We compare DRL with different memory replay strategies in Tab. 5.9 which shows DRL has general improvements based on the applied replay strategy. BER has consistently shown better performance on forgetting whereas ER has shown better performance on intransigence. The performance of DRL obviously correlates to the applied replay strategy, e.g., DRL+BER gets better performance on forgetting than DRL+ER. We see

Table 5.8: Comparing the performance with or without the regularization terms ($\mathscr{L}_{bt}$, $\mathscr{L}_{wi}$) in DRL. All criteria are in percentage. The bold font indicates the best performance of a criterion.

		BER	DRL		
		with none	with $\mathscr{L}_{bt}$	with $\mathscr{L}_{wi}$	with both
P-MNIST	Avg. Accuracy	79.2 ± 0.3	79.6 ± 0.4	80.1 ± 0.4	$\mathbf{80.5 \pm 0.4}$
	Avg. Forgetting	5.1 ± 0.3	4.8 ± 0.3	4.9 ± 0.5	$\mathbf{4.5 \pm 0.2}$
	Avg. Intransigence	2.9 ± 0.2	2.6 ± 0.2	$\mathbf{1.9 \pm 0.3}$	$\mathbf{1.9 \pm 0.5}$
S-MNIST	Avg. Accuracy	85.2 ± 1.1	86.5 ± 0.9	86.6 ± 0.7	$\mathbf{88.1 \pm 0.6}$
	Avg. Forgetting	13.0 ± 1.5	11.3 ± 1.3	11.7 ± 1.1	$\mathbf{8.9 \pm 0.7}$
	Avg. Intransigence	1.4 ± 0.3	1.4 ± 0.3	$\mathbf{1.0 \pm 0.2}$	1.7 ± 0.3
Fashion	Avg. Accuracy	77.0 ± 0.7	77.3 ± 0.8	77.1 ± 0.8	$\mathbf{77.9 \pm 0.8}$
	Avg. Forgetting	18.2 ± 2.7	18.1 ± 2.4	18.5 ± 2.2	$\mathbf{16.6 \pm 1.9}$
	Avg. Intransigence	3.7 ± 1.8	3.6 ± 1.5	$\mathbf{3.4 \pm 1.1}$	4.1 ± 1.4
CIFAR10	Avg. Accuracy	37.3 ± 1.4	$\mathbf{40.4 \pm 1.8}$	39.4 ± 0.8	$\mathbf{40.4 \pm 1.6}$
	Avg. Forgetting	52.3 ± 1.6	45.7 ± 2.1	46.6 ± 2.1	$\mathbf{41.4 \pm 3.2}$
	Avg. Intransigence	$\mathbf{3.5 \pm 0.6}$	5.6 ± 1.0	5.9 ± 1.8	9.0 ± 1.5
CIFAR100	Avg. Accuracy	18.2 ± 0.4	18.7 ± 0.3	18.5 ± 0.4	$\mathbf{19.3 \pm 0.5}$
	Avg. Forgetting	35.9 ± 1.0	34.7 ± 0.5	36.2 ± 0.5	$\mathbf{29.0 \pm 1.0}$
	Avg. Intransigence	3.1 ± 0.6	3.7 ± 0.3	$\mathbf{2.5 \pm 0.4}$	8.1 ± 1.0
TinyImageNet	Avg. Accuracy	6.7 ± 0.8	7.2 ± 0.5	6.7 ± 0.4	$\mathbf{8.3 \pm 0.2}$
	Avg. Forgetting	$\mathbf{26.0 \pm 1.2}$	39.6 ± 0.7	40.4 ± 0.3	27.0 ± 1.4
	Avg. Intransigence	17.3 ± 1.9	3.9 ± 1.0	$\mathbf{3.7 \pm 0.5}$	14.8 ± 1.5

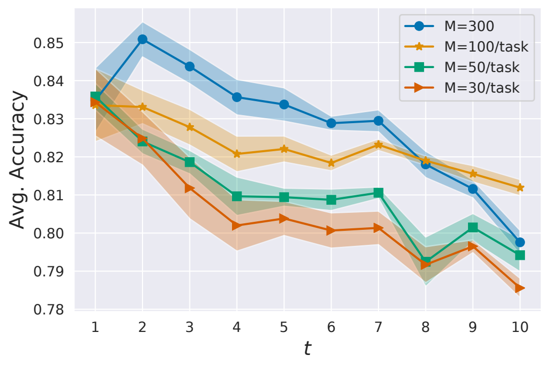
that on benchmarks which are more difficult in terms of intransigence DRL+ER obtains better performance than DRL+BER, e.g., on TinyImageNet benchmarks.

Comparing different memory cost: Fig. 5.5 compares average accuracy of DRL+BER on MNIST tasks with different memory cost. The fixed memory size (M = 300) getting very similar average accuracy with memory M = 50/class in Split MNIST while it takes less cost of the memory after task 3. Meanwhile, the fixed memory size (M = 300) gets much better performance than M = 50/task in most tasks of Permuted MNIST and it takes less cost of the memory after task 6. Since the setting of fixed memory size takes larger memory buffer in early tasks, the results indicate better generalization of early tasks can benefit later tasks, especially for more homogeneous tasks such as Permuted MNIST. The results also align with findings in (Chaudhry et al., 2019b) and a hybrid memory strategy could bring improvements as suggested in (Chaudhry et al., 2019b).

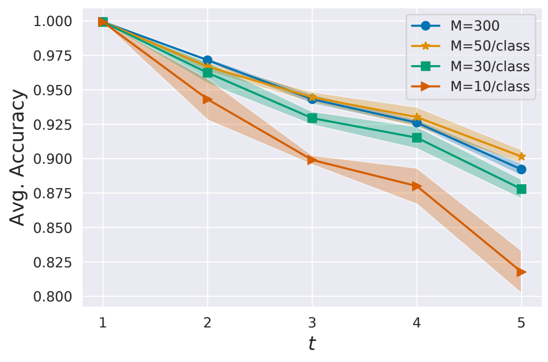
We provide the chosen hyper-parameters of all methods on all benchmarks in Tab. 5.10. We set the original hyperparameters of Multisimilarity recommended as the same as in (Roth et al., 2020) which generally perform well on multiple complex data sets. For hyperparameters of Multisimilarity (Eq. (5.6)), $\alpha = 2$, $\beta = 40$, $\gamma = 0.5$; for R-Margin (Eq. (5.7)), the initial value of β is 0.6, $\gamma = 0.2$, $p_\rho = 0.2$. In Tab. 5.11 we give the search range of these hyperparameters.

Table 5.9: Comparing DRL with different memory replay strategies, all criteria are in percentage.

		DRL+ BER	DRL + ER	BER	ER
P-MNIST	Avg. Accuracy	**80.5 ± 0.4**	78.9 ± 0.4	79.2 ± 0.3	78.2 ± 0.6
	Avg. Forgetting	**4.5 ± 0.2**	6.7 ± 0.3	5.1 ± 0.3	7.1 ± 0.6
	Avg. Intransigence	2.2 ± 0.5	**1.5 ± 0.2**	3.1 ± 0.2	1.9 ± 0.4
S-MNIST	Avg. Accuracy	**88.1 ± 0.6**	84.2 ± 1.4	85.2 ± 1.17	83.2 ± 1.5
	Avg. Forgetting	**8.9 ± 0.7**	16.1 ± 1.8	13.0 ± 1.5	17.2 ± 1.9
	Avg. Intransigence	2.4 ± 0.3	**0.6 ± 0.1**	2.1 ± 0.3	0.8 ± 0.1
Fashion	Avg. Accuracy	**77.9 ± 0.8**	76.5 ± 1.0	77.0 ± 0.7	75.8 ± 1.4
	Avg. Forgetting	**16.6 ± 1.9**	23.5 ± 1.9	18.2 ± 2.7	24.0 ± 2.7
	Avg. Intransigence	6.5 ± 1.4	**2.5 ± 1.2**	6.2 ± 1.8	2.7 ± 1.4
CIFAR10	Avg. Accuracy	**40.4 ± 1.6**	36.0 ± 1.7	37.3 ± 1.4	39.4 ± 1.7
	Avg. Forgetting	**41.4 ± 3.2**	49.6 ± 4.6	52.3 ± 1.6	50.3 ± 1.9
	Avg. Intransigence	9.8 ± 1.5	7.6 ± 1.8	4.2 ± 0.6	**3.7 ± 1.0**
CIFAR100	Avg. Accuracy	19.3 ± 0.5	**19.6 ± 1.0**	18.2 ± 0.4	18.3 ± 0.3
	Avg. Forgetting	**29.0 ± 1.0**	34.9 ± 1.1	35.9 ± 1.0	37.0 ± 1.4
	Avg. Intransigence	9.6 ± 1.0	4.0 ± 0.9	4.6 ± 0.6	**3.4 ± 1.0**
TinyImageNet	Avg. Accuracy	8.3 ± 0.2	**11.3 ± 1.0**	6.7 ± 0.8	7.6 ± 0.6
	Avg. Forgetting	27.0 ± 1.4	30.0 ± 0.9	**26.0 ± 1.2**	28.5 ± 2.0
	Avg. Intransigence	10.8 ± 1.5	**5.1 ± 0.8**	13.3 ± 1.9	10.1 ± 2.5



(a) Permuted MNIST



(b) Split MNIST

Figure 5.5: Average accuracy of DRL+BER with different memory cost. The x axis is the index of tasks, the shaded area is plotted by standard deviation of 10 runs.

5.6 Experiments for CLVision Challenge

The CLVision Challenge [1] was hold by the Continual Learning workshop in CVPR 2020. It consists of three tracks (each is a different benchmark) based on the CORe50 dataset (Lomonaco & Maltoni, 2017). DRL is a wining solution for this challenge as it shows consistently good performance on all benchmarks (Lomonaco et al., 2020). We provide experimental results of DRL for the CLVision Challenge in this section. In these experiments we used a pre-trained model, a

[1]`https://sites.google.com/view/clvision2020/challenge?authuser=0`

Table 5.10: Hyperparameters of all methods

	P-MNIST	S-MNIST	Fashion	CIFAR-10	CIFAR-100	TinyImageNet
training batch size	20	10	10	10	10	5
learning rate	0.1	0.02	0.02	0.1	0.05	0.1
learning rate (A-GEM)	0.02	0.001	0.001	0.1	0.05	0.01
ref batch size (A-GEM)	256	256	256	512	1500	1000
α of DRL	2	2	2	1	1	2
λ of DRL	1×10^{-3}	1×10^{-2}	1×10^{-2}	2×10^{-3}	2×10^{-3}	5×10^{-4}
λ of Multisim	5	1	1	2	1	5
λ of R-Margin	2×10^{-5}	1×10^{-3}	1×10^{-3}	1×10^{-4}	1×10^{-3}	1×10^{-3}
λ of standalone $\mathscr{L}_{bt}$	1×10^{-4}	5×10^{-4}	5×10^{-4}	2×10^{-4}	2×10^{-4}	5×10^{-5}

Table 5.11: The search range of hyperparameters

	The grid-search range
training batch size	[5,10, 20, 50, 100]
learning rate	[0.001, 0.01, 0.02,0.05, 0.1, 0.2]
ref batch size (A-GEM)	[128, 256, 512, 1000, 1500, 2000]
α of DRL	[0.1, 0.2, 0.5, 1, 2, 4]
λ of DRL	$[1,2,5]\times[10^{-5},10^{-4},10^{-3},10^{-2},10^{-1}]$
λ of Multisim	[10, 8, 6, 5, 4, 3, 2, 1, 0.5, 0.2, 0.1, 0.05]
λ of R-Margin	$[1,2,5]\times[10^{-5},10^{-4},10^{-3},10^{-2},10^{-1}]$

ResNet50 (He et al., 2016) or ResNeSt50 (Zhang et al., 2020a) trained on ImageNet (Russakovsky et al., 2015). In all experiments we set $\alpha = 1$ in DRL.

CORe50 is a image dataset collected for object recognition. It consists of 164,866 images with 128×128 pixels, which are photos of 50 domestic objects from 10 categories. For each object the dataset includes 11 video sessions (~300 frames recorded with a Kinect 2 at 20 fps) characterized by relevant variations in terms of lighting, background, pose and occlusions. More information about this dataset can be found in Lomonaco & Maltoni (2017). In this challenge each object is treat as a class which means there are 50 classes in total for all tracks. The three tracks are defined as below:

1. NI: it has 8 tasks each of which includes the same 50 classes encountered over time. Each task is composed of different images collected in different environmental conditions. It is a typical Domain-IL scenario.
2. Multi-Task-NC: the 50 classes are split into 9 different tasks where 10 classes in the first task and 5 classes in the other 8. The task identifier is provided during training and test which means it is a Task-IL scenario and we used a multi-headed model in experiments of

this track.

3. NIC: it is composed of 391 tasks each of which containing 300 images of a single class. No task identifier is provided and each task may contain images of a class seen before as well as a completely new class. This is a mixture scenario of Domain-IL and Class-IL that is likely happen in practice.

The evaluation applied in this challenge is a weighted sum over several metrics which aim to measure the efficiency and accuracy of each solution jointly. Such a setting emphasizes the scalability of continual learning approaches for real applications in practice. The individual metrics are defined as below (Lomonaco et al., 2020):

1. *Final accuracy on the test set*[2]: computed only at the end of the training procedure.
2. *Average accuracy over time on the validation set*: computed at the end of every task.
3. *Total training/test time*: total running time from start to end of the main function (in minutes).
4. *RAM usage*: total memory occupation of the process and its eventual sub-processes. It is computed at every epoch (in MB).
5. *Disk usage*: only of additional data produced during training (like replay patterns) and additionally stored parameters. It is computed at every epoch (in MB).

The final aggregation metric (CL_{score}) is the weighted average of the 1-5 metrics (0.3, 0.1, 0.15, 0.125, 0.125 respectively).

Tabs. 5.12 to 5.14 show our experimental results of each track of the challenge. The test accuracy of some tests are from submissions we submitted to the challenge and those without test accuracy are results we have not submitted. We have tested three different replay strategies: BER, ER, and shuffle. Shuffle is a basic replay strategy that shuffles the memorized samples with new training data to obtain a mixture training set. BER gives higher accuracy than ER in NI track, whereas ER gets better accuracy than BER in Multi-Task-NC track. The difference is the NI track uses a single-headed model and the Multi-Task-NC uses a multi-headed one. Hence, Multi-Task-NC suffers much less on forgetting and ER gets better performance on intransigence. Interestingly, the basic shuffle strategy works better for NIC. It is probably because the task sequence is very long (391 tasks) and the training set is quite small (300 samples) for each task. In such a case, the memorised samples from previous tasks will compose the majority of the training samples in latter tasks. By the shuffle strategy all the memorised samples will be selected into the training batch for certain, whereas ER and BER may missed some samples during training due to the random sampling from the memory. In general, ResNeSt50 works better than ResNet50 and applying DRL obtains better performance than without it.

[2]Accuracy in CORe50 is computed on a fixed test set. Rationale behind this choice is explained in Lomonaco & Maltoni (2017)

Table 5.12: Experiment results of NI

	Mem. size	Batch size	ER type	λ_{DR}	Test Acc.	Avg. Valid Acc.	RAM Usage (Mb)	Time (m)
ResNet50	100	32	Shuffle	0.	0.79	0.74	19678.42	6.
	100	16	Shuffle	0.	N/A	0.75	16573.91	8.62
	100	16	ER	0.	N/A	0.76	16560.55	9.92
	100	16	BER	0.	N/A	0.75	16611.49	19.71
	100	16	ER	0.0002	0.81	0.79	16558.13	10.1
	100	16	BER	0.0002	0.82	0.78	16625.19	19.88
	300	16	BER	0.0002	0.83	0.77	16447.77	19.56
	300	16	BER	0.001	0.83	0.79	16502.56	19.88
	2000	16	BER	0.001	N/A	0.77	17506.92	20.48
ResNeSt50	1500	16	BER	0.	N/A	0.77	18340.47	40.95
	1500	16	BER	0.001	0.89	0.81	18342.86	21.25

Table 5.13: Experiment results of Multi-Task-NC

	Mem. size	Batch size	ER type	λ_{DR}	Test Acc.	Avg. Valid Acc.	RAM Usage (Mb)	Time (m)
ResNet50	20	32	Shuffle	0.	0.9	0.51	23863.71	5.66
	100	32	Shuffle	0.	N/A	0.50	24121.09	6.14
	100	16	Shuffle	0.	N/A	0.52	19400	8.73
	100	16	BER	0.0002	0.94	0.54	19583.7	20.92
	100	16	ER	0.0002	N/A	0.53	19579	10.0
	500	16	ER	0.0002	0.95	0.54	19047.21	9.7
ResNeSt50	500	16	ER	0.0002	0.97	0.54	19167.67	13.47

Table 5.14: Experiment results of NIC

	Mem. size	Batch size	ER type	λ_{DR}	Test Acc.	Avg. Valid Acc.	RAM Usage (Mb)	Time (m)
ResNet50	10	32	Shuffle	0.	0.81	0.53	13683.84	65.08
	50	32	Shuffle	0.0002	0.84	0.55	21275.17	126.76
	50	16	Shuffle	0.0002	N/A	0.49	22183.93	264.84
	50	16	ER	0.	N/A	0.04	15017.92	22.11
	50	16	ER	0.0002	N/A	0.04	15074.38	22.37
	50	2	BER	0.0002	N/A	0.18	12088.44	220.98
ResNeSt50	50	32	Shuffle	0.00001	0.89	0.57	21571.89	154.56

Tabs. 5.15 to 5.17 shows the results of the top 5 teams of the three tracks. The average performance in the last row of all three tables is the average over all finalists (11 teams entered in the final phase) in the challenge. Yc14600 is our team using DRL as the solution for all final submissions. All the top solutions used pre-trained models and most of them included a preprocessing procedure whereas we did not. In addition to experimental results in the previous section, these results demonstrate that our method is efficient with pre-trained model in practical scenarios as well. Please refer to Lomonaco et al. (2020) for more details.

Table 5.15: NI track results for the top 5 finalists of the challenge.

team name	test acc (%)	avg. val acc (%)	run time (m)	avg. ram (mb)	max ram (mb)	avg. disk (mb)	max disk (mb)	CL_{score}
UT_LG	0.91	0.90	63.78	11429.83	11643.63	0	0	0.692
Yc14600	0.88	0.85	22.58	17336.38	18446.90	0	0	0.648
ICT_VIPL	0.95	0.93	113.70	2459.42	2460.16	421.875	750	0.629
Jodelet	0.84	0.85	3.11	18805.60	18829.96	0	0	0.612
Soony	0.85	0.81	25.57	16662.73	17000.10	0	0	0.602
avg	0.82	0.78	100.74	18987.80	21135.96	30.13	53.57	0.52

5.7 Summary

Gradient-based approaches have shown that the diversity of gradients computed on data from different tasks is a key to generalization over these tasks. In this chapter, we formally connect the diversity of gradients to discriminativeness of representations, which leads to an alternative way to reduce the diversity of gradients in continual learning. We subsequently exploit ideas from DML for learning more discriminative representations, and furthermore identify the shared and different interests between continual learning and DML. In continual learning we would prefer larger margins between classes the same as in DML. The difference is that continual learning requires less compact representations for better compatibility with future tasks. Based on these findings, we provide a simple yet efficient approach to achieve better performance for classification tasks in continual learning.

Table 5.16: NC track results for the top 5 finalists of the challenge.

team name	test acc (%)	avg. val acc (%)	run time (m)	avg. ram (mb)	max ram (mb)	avg. disk (mb)	max disk (mb)	CL_{score}
Ar1	0.93	0.53	16.02	10263.19	14971.72	0	0	0.693
UT_LG	0.95	0.55	19.02	13793.31	16095.20	0	0	0.691
Yc14600	0.97	0.54	11.81	15870.62	19403.57	0	0	0.686
Soony	0.97	0.55	55.02	14005.91	16049.12	0	0	0.679
Jodelet	0.97	0.55	2.55	17893.58	23728.84	0	0	0.679
avg	0.85	0.51	63.77	17624.83	21773.26	43.27	43.27	0.60

Table 5.17: NIC track results for the top 5 finalists of the challenge.

team name	test acc (%)	avg. val acc (%)	run time (m)	avg. ram (mb)	max ram (mb)	avg. disk (mb)	max disk (mb)	CL_{score}
UT_LG	0.91	0.58	123.22	6706.61	7135.77	0	0	0.706
Jodelet	0.83	0.54	14.12	10576.67	11949.16	0	0	0.694
Ar1	0.71	0.48	28.19	3307.62	4467.64	0	0	0.693
ICT_VIPL	0.90	0.56	91.29	2485.95	2486.03	192.187	375	0.625
Yc14600	0.89	0.57	160.24	16069.91	21550.97	0	0	0.586
avg	0.72	0.47	134.03	10606.70	13249.99	19.22	37.50	0.56

Chapter 6

Measuring Classifiers in Continual Learning by β^3-IRT Model

We have proposed several approaches for classification tasks in continual learning in the preceding chapters. How to evaluate the approaches of continual learning is still an open question as it relates to performance in terms of different aspects, such as forgetting and intransigence. In this chapter, we propose a new evaluation method for classifiers in continual learning, which is based on Item Response Theory (IRT). IRT aims to assess latent abilities of respondents based on the correctness of their answers to aptitude test items with different difficulty levels. We propose the β^3-IRT model, which models continuous responses using a more expressive formulation and enables a new metric for evaluating classifiers. As β^3-IRT is a Bayesian probabilistic model, it can be easily fitted in the framework of VCL for evaluating classifiers in continual learning. We show that the assessed ability by β^3-IRT is a joint measure regarding both forgetting and intransigence which is adaptive to the context. It may provide more insights when average accuracy fails to tell the difference between classifiers.

In the following, we first introduce the background and related work of IRT model in Secs. 6.1 and 6.2. We then provide technique details of our method β^3-IRT in Sec. 6.3. Finally, we show the effectiveness of β^3-IRT for evaluating classifiers by two scenarios: *i*) assess the ability of binary classifiers in static learning, and further analyze the inferred difficulty and discrimination of data instances (Sec. 6.4); *ii*) assess the ability of multi-class classifiers in continual learning and further analyze the difference between the ability and average accuracy in continual learning (Sec. 6.5).

6.1 Introduction

Unlike classical test theory, which is concerned with performance at the test level, IRT focuses on the finer detail of test items, by modeling responses (answers) given by respondents with different abilities (Embretson & Reise, 2013). Item Response Theory (IRT) is widely adopted in psychometrics for estimating human latent ability in tests. The concept of item depends on the application, and can represent for test questions, judgments or choices in exams. In practice, IRT models estimate latent difficulties of the items and the latent abilities of the respondents based on observed responses in a test, and have been commonly applied to assess performance of students in exams.

Recently, IRT was adopted to analyze Machine Learning (ML) classification tasks (Martínez-Plumed et al., 2016). Here, items correspond to instances in a dataset, while respondents are classifiers. The responses are the outcomes of classifiers on the test instances (right or wrong decisions collected in a cross-validation experiment) which are limited to be binary. In addition, an Item Characteristic Curve (ICC) is estimated for each item (instance), which is a logistic function that returns the probability of a correct response for the item based on the respondent ability. This ICC is determined by two item parameters: difficulty, which is the location parameter of the logistic function; and discrimination, which affects the slope of the ICC. Binary IRT models are limited when the techniques of interest return continuous responses (e.g. class probabilities). Continuous IRT models have been developed and applied in psychometrics (Noel & Dauvier, 2007) but the flexibility is still limited since ICCs are limited to logistic functions.

To address the issues of binary and logistic-based IRT models, we propose a new IRT model called β^3-IRT. β^3-IRT is defined by a new parameterization of IRT models such that: a) the observed responses are continuous instead of binary; b) the resulting ICCs are not limited to logistic curves, different shapes can be obtained depending on the item parameters, which allows more flexibility when fitting responses for different items; c) abilities and difficulties are in the $[0,1]$ range, which gives a unified scale for easier interpretation and evaluation.

We demonstrate that this model provides a method for assessing the 'ability' of classifiers by means of an aggregated metric weighted by instance-wise 'difficulty'. Several metrics (Sec. 3.2) have been proposed for evaluating classifiers in continual learning as classification tasks have drawn most attentions in continual learning. However, most of them are average measures which can easily be skewed by a single task. And some metrics can only evaluate one specific aspect of the performance in continual learning, such as forgetting, intransigence, which leaves a problem of model selection: what is the appropriate way to combine the evaluations given by these metrics. We demonstrate that the 'ability' is capable of jointly measuring different aspects of classification performance in continual learning, which provides an effective way for model selection in continual learning.

6.2 Related work

In Item Response Theory, the probability of a correct response for an item depends on the latent respondent ability and the item difficulty. Most previous studies on IRT have adopted binary models, in which the responses are either correct or incorrect. Such models assume a binary response x_{ij} of the i-th respondent to the j-th item. In the IRT model with two item parameters (2PL), the probability of a correct response ($x_{ij} = 1$) is defined by a logistic function with location parameter δ_j and shape parameter a_j. Responses are modelled by the Bernoulli distribution with parameter p_{ij} as follows:

$$x_{ij} = Bern(p_{ij}), \quad p_{ij} = \sigma(-a_j d_{ij}), \quad d_{ij} = \theta_i - \delta_j \tag{6.1}$$

where $\sigma(\cdot)$ is the logistic function. The 2PL model gives a logistic Item Characteristic Curve (ICC) mapping ability θ_i to expected response as follows:

$$\mathbb{E}[x_{ij}|\theta_i, \delta_j, a_j] = p_{ij} = \frac{1}{1 + e^{-a_j(\theta_i - \delta_j)}} \tag{6.2}$$

At $\theta_i = \delta_j$ the expected response is 0.5. The slope of the ICC (the derivative w.r.t. $\theta_i - \delta_j$) at $\theta_i = \delta_j$ is $a_j/4$. If $a_j = 1, \forall j$, a simpler model is obtained, known as 1PL (one item parameter logistic), which describes items solely by their difficulties. Generally, discrimination a_j indicates how the probability of correct responses changes as the ability increases. High discriminations lead to steep ICCs at the point where ability equals difficulty, with small changes in ability producing significant changes in the probability of correct response.

Despite their wide use in psychometrics, binary IRT models are of limited use when responses are naturally produced on continuous scales. Particularly in ML, binary models are not adequate if the responses to evaluate are class probability estimates. There have been earlier approaches to IRT with continuous approaches. In particular, Noel & Dauvier (2007) proposed an IRT model which adopts the Beta distribution with parameters m_{ij} and n_{ij} as follows:

$$\begin{aligned} &m_{ij} = e^{(\theta_i - \delta_j)/2}, \quad n_{ij} = e^{-(\theta_i - \delta_j)/2} = m_{ij}^{-1}, \\ &p_{ij} \sim Beta(m_{ij}, n_{ij}). \end{aligned}$$

In this model p_{ij} is the continuous response given by respondent i to item j. Similar to standard IRT, parameter $\boldsymbol{\theta}_i$ is the ability of the respondent and $\boldsymbol{\delta}_j$ is the difficulty of the item. m_{ij} and n_{ij} can be interpreted as *acceptance* and *refusal* parameters to the response, which are defined as a function of ability-difficulty distance on some latent range. This model also gives a logistic ICC mapping ability to expected response for item j of the form:

$$\mathbb{E}[p_{ij}|\theta_i, \delta_j] = \frac{m_{ij}}{m_{ij} + n_{ij}} = \frac{1}{1 + e^{-(\theta_i - \delta_j)}} \tag{6.3}$$

Although the β^3-IRT model also assumes the responses are from a Beta distribution, there are several crucial distinctions:

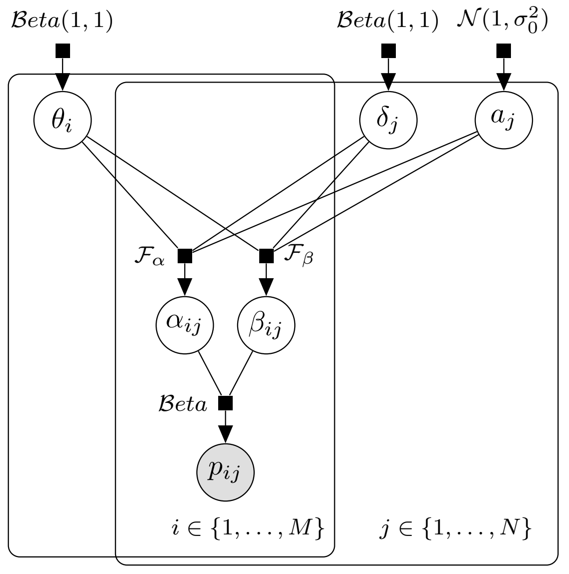


Figure 6.1: Factor Graph of β^3-IRT Model. The grey circle represents observed data, white circles represent latent variables, small black rectangles represent stochastic factors, and the rectangular plates represent replicates. M is number of respondents and N is number of items, p_{ij} is the observed response of respondent i to item j.

- Eq. (6.3) does not have a discrimination parameter which is similar to the standard 1PL (One-Parameter Logistic) IRT model. The ICC therefore has a fixed slope of 0.25 at $\theta_i = \delta_j$ as it assumes that all items have the same discrimination $a_j = 1$;

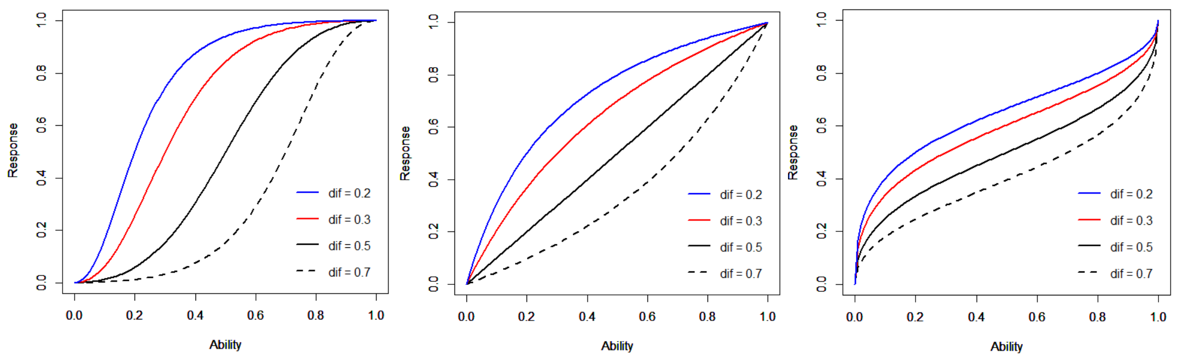
- Eq. (6.3) assumes a real-valued scale for abilities and difficulties, whereas β^3-IRT uses a $[0,1]$ scale to avoid issues with interpreting extreme values;

- Eq. (6.3) is still fixed in the form of logistic function whereas β^3-IRT opens the door to more options including non-sigmoidal ICCs as depicted in Figure 6.2.

We will elaborate on the details of β^3-IRT in the following section.

6.3 The β^3-Item Response Theory model

In this section, we introduce the parameterization and inference method of β^3-IRT in details, highlighting the differences with existing IRT models.



(a) $a_j = 2$. (b) $a_j = 1$. (c) $a_j = 0.5$.

Figure 6.2: Examples of Beta ICCs for Different Values of Difficulty and Discrimination. Higher discriminations lead to steeper ICCs, higher difficulties need higher abilities to achieve higher responses.

6.3.1 Model description

β^3-IRT can be depicted by a factor graph as shown in Fig. 6.1 and Eq. (6.4) below accordingly gives the model definition:

$$p_{ij} \sim Beta(\alpha_{ij}, \beta_{ij}), \quad \alpha_{ij} = \mathscr{F}_\alpha(\theta_i, \delta_j, a_j) = \left(\frac{\theta_i}{\delta_j}\right)^{a_j}, \quad \beta_{ij} = \mathscr{F}_\beta(\theta_i, \delta_j, a_j) = \left(\frac{1-\theta_i}{1-\delta_j}\right)^{a_j},$$
$$\theta_i \sim Beta(1,1), \quad \delta_j \sim Beta(1,1), \quad a_j \sim \mathcal{N}(1, \sigma_0^2) \tag{6.4}$$

In Fig. 6.1, M is the number of respondents and N is number of items, p_{ij} is the observed response of respondent i to item j, which is drawn from a Beta distribution. The parameters of the Beta distribution α_{ij}, β_{ij} are computed from θ_i (the ability of participant i), δ_j (the difficulty of item j), and a_j (the discrimination of item j). θ_i and δ_j are also drawn from Beta distributions and their priors are set to $Beta(1,1)$ in a general setting. In specific applications, the priors can be adjusted according to the prior knowledge of the data. Moreover, the default priors could be parameterized by further hyperparameters when there is further prior information available. The discrimination a_j is drawn from a Normal distribution with prior mean 1 and variance σ_0^2, where σ_0 is a hyperparameter of the model. The default prior mean of a_j is set to 1 rather than 0 because the discrimination a_j is a power factor here.

When comparing probabilities we often use ratios (e.g. the likelihood ratio). Similarly, here we use the ratio of ability to difficulty since in our model these are measured on a $[0,1]$ scale as well: a ratio smaller/larger than 1 means that ability is lower/higher than difficulty. These ratios are positive reals and hence map directly to α and β in Eq. (6.4). Importantly, the new parameterization enables us to obtain non-logistic ICCs. In this model the ICC is defined by the expectation of $Beta(\alpha_{ij}, \beta_{ij})$ and then obtains the form:

$$\mathbb{E}[p_{ij} | \theta_i, \delta_j, a_j] = \frac{\alpha_{ij}}{\alpha_{ij} + \beta_{ij}} = \frac{1}{1 + \left(\frac{\delta_j}{1-\delta_j}\right)^{a_j} \left(\frac{\theta_i}{1-\theta_i}\right)^{-a_j}} \tag{6.5}$$

The expectation of responses is 0.5 when $\theta_i = \delta_j$ and the curve has slope $a_j/(4\delta_j(1-\delta_j))$ at that point. Fig. 6.2 shows examples of Beta ICCs for different regimes depending on a_j:

$$\begin{aligned} a_j > 1: &\qquad \text{a logistic shape similar to standard IRT,} \\ a_j = 1: &\qquad \text{parabolic curves with vertex at 0.5,} \\ 0 < a_j < 1: &\qquad \text{anti-logistic behaviour.} \end{aligned}$$

Note that the model allows for negative discrimination, which indicates items that are somehow harder for more able respondents. Negative a_j can be divided similarly:

$$\begin{aligned} -1 < a_j < 0: &\qquad \text{decreasing anti-logistic,} \\ a_j = -1: &\qquad \text{decreasing parabolic curves with vertex at 0.5,} \\ a_j < -1: &\qquad \text{decreasing logistic.} \end{aligned}$$

We will see examples in Sec. 6.4 that negative discrimination can in fact be useful for identifying ‘noisy’ items, where higher abilities obtain worse responses. For instance, if an item in class 1 is wrongly labelled as in class 2, a well-trained classifier could make a prediction that is contradictory to the label but a random classifier might get a more ‘correct’ response by a random guess.

6.3.2 Model inference

The method we applied to learn the latent variables in β^3-IRT is meanfield Bayesian VI (Bishop, 2006), i.e. the variables are assumed to be independent. The conventional Maximum Likelihood Estimation (MLE) can also be applied to the model and we have conducted experiments using MLE as well. Such experiments use data formed by student answers from an online platform for response prediction, applying the likelihood function shown in Eq. (6.5). They are not included in this thesis because they are not related to classifier evaluation. Please refer to Chen et al. (2019) for more details.

As we introduced in Sec. 2.1, the optimization objective of VI is the lower bound of the KL divergence between the true posterior and variational posterior of latent variables. The three latent variables $(\theta_i, \delta_j, a_j)$ in β^3-IRT are on the same level of the model structure, which makes the model highly non-identifiable (Nishihara et al., 2013) (i.e., easily result in undesirable combinations of these variables): for instance, when p_{ij} is close to 1 (high correctness), it usually indicates $\alpha_{ij} > 1$ and $\beta_{ij} < 1$, which can arise when $\theta_i > \delta_j$ (ability larger than difficulty) with positive a_j, or $\theta_i < \delta_j$ (ability smaller than difficulty) with negative a_j. To prevent negative discrimination in such a case, we employ the coordinate ascent method of VI (Blei et al., 2017), keeping a_j fixed while optimizing θ_i and δ_j and vice versa (see also Alg. 4). In addition, we set the prior of discrimination as $\mathcal{N}(1,1)$ to reflect the assumption that discrimination is more often

Algorithm 4: Variational Inference for β^3-IRT

```
1 Set number of iterations $L_{iter}$, randomly initialize $\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\lambda}$;
2 for t in range($T_{iter}$) do
3     for k in range($K_{iter}$) do
4         Compute $\nabla_\Theta \mathscr{L}_{\text{local}}$ according to Eq. (6.6), $\Theta = \{\boldsymbol{\phi}, \boldsymbol{\psi}\}$;
5         $\Theta_k \leftarrow \Theta_{k-1} - \eta \nabla_\Theta \mathscr{L}_{\text{local}}$
6     end
7     Compute $\nabla_{\boldsymbol{\lambda}} \mathscr{L}_{\text{global}}$ according to Eq. (6.6);
8     $\boldsymbol{\lambda}_t \leftarrow \boldsymbol{\lambda}_{t-1} - \zeta \nabla_{\boldsymbol{\lambda}} \mathscr{L}_{\text{global}}$
9 end
```

positive than negative. According to the adjustment of updating variables, we separately define two loss functions for local and global variables:

$$\mathscr{L}_{\text{local}} = \sum_{i=1}^{M}\sum_{j=1}^{N} \mathbb{E}_q[\log p(p_{ij}|\theta_i,\delta_j)] + \sum_{i=1}^{M} \mathbb{E}_q[\log p(\theta_i) - \log q(\theta_i|\boldsymbol{\phi}_i)] + \sum_{j=1}^{N} \mathbb{E}_q[\log p(\delta_j) - \log q(\delta_j|\boldsymbol{\psi}_j)]$$

$$\mathscr{L}_{\text{global}} = \sum_{j=1}^{N} \mathbb{E}_q[\log p(p_{ij}|a_j) + \log p(a_j) - \log q(a_j|\boldsymbol{\lambda}_j)] \tag{6.6}$$

Here, $\boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\lambda}$ are parameters of the variational posteriors of θ, δ, a, respectively. In order to apply the reparameterization trick (Kingma & Welling, 2013), we use Logit-Normal to approximate the Beta distribution in variational posteriors. The steps to perform VI of the model are shown in Alg. 4. η and ζ denote the learning rate for updating local and global variables and they can be different. Any gradient descent optimization algorithm can be applied to this algorithm, such as the Adam optimizer (Kingma & Ba, 2014) that was used in our experiments. We implemented Alg. 4 based on the probabilistic programming library Edward (Tran et al., 2016).

6.4 Measuring Classifiers in Static Learning

We now apply β^3-IRT for measuring classifiers in static learning (i.e., all classifiers are just trained on a single task). Here, each 'respondent' is a different classifier and items are instances from a classification dataset. Note that an entire dataset could be considered as an item too, with associated difficulty, the classifiers then could be evaluated over multiple datasets. The responses are the probabilities that a classifier assigned to the *correct* class of each instance. Specifically, the observed response is given by $p_{ij} = \sum_k \mathbb{I}(y_j = k) p_{ijk}$, where $\mathbb{I}(\cdot)$ is the indicator function, y_j is the label of item j, k is the index of each class, and p_{ijk} is the predicted probability of item j belonging to class k given by classifier i.

The following steps are deployed to obtain the responses from M classifiers on a dataset:

1. Train all M classifiers on the training set $\mathscr{D}_{\text{train}}$;

2. Use all trained classifiers to predict the class probabilities p_{ijk} for each data instance in a validation set $\mathscr{D}_{\text{val}}$, which gives $p_{ij},\ \ \forall j \in \{1,2,\ldots,N\},\ \ N = |\mathscr{D}_{\text{val}}|$.

Inference is then performed for the β^3-IRT model as demonstrated in Alg. 4, using the responses of the M classifiers to N validation instances.

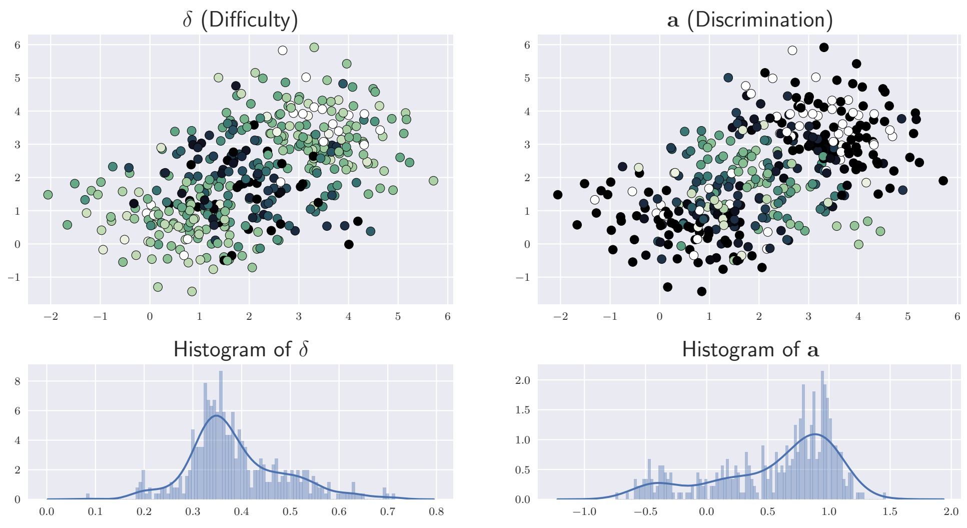
6.4.1 Experimental setup

We first applied the β^3-IRT model on two synthetic binary classification datasets, MOONS and CLUSTERS, which are chosen because they are convenient for visualization. Both datasets are available in scikit-learn (Pedregosa et al., 2011). Each dataset is divided into training and validation sets, each set with 400 instances. We also conducted experiments on classes 3 vs 5 from the MNIST dataset (LeCun et al., 2010), chosen as they are similar and contain difficult instances. For the MNIST dataset, the training and validation sets have 1000 instances. The classes are balanced in each dataset. We inject noise in the validation set by flipping the label y_j for 20% of randomly chosen data instances. The hyperparameter of discrimination σ_0 is set to 1 across all tests unless specified explicitly.

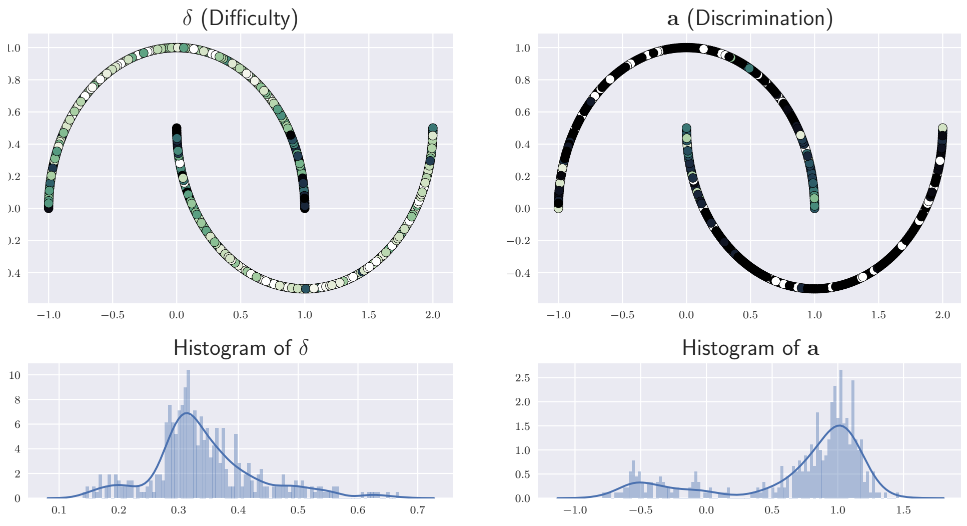
We tested 12 classifiers in this experiment: (i) Naive Bayes; (ii) MLP (two hidden layers with 256 and 64 units); (iii) AdaBoost (Hastie et al., 2009); (iv) Logistic Regression (LR); (v) k-Nearest Neighbours (KNN) (K=3); (vi) Linear Discriminant Analysis (LDA) (Friedman et al., 2001); (vii) Quadratic Discriminant Analysis (QDA) (Friedman et al., 2001); (viii) Decision Tree (Breiman et al., 1984); (ix) Random Forest (Breiman, 2001); (x) Calibrated Constant Classifier (assign probability $p = 1/K$ to all instances, K is the number of classes); (xi) Positive classifier (always assign positive class to instances, i.e. $p_{ij1} = 1$); (xii) Negative classifier (always assign negative class to instances, i.e. $p_{ij0} = 1$). The last two are specific for binary classification cases and the two classes are denoted as positive ($k = 1$) and negative ($k = 0$) class, respectively. All except the last three are taken from scikit-learn (Pedregosa et al., 2011) using default configuration unless specified explicitly. We add the last three classifiers to the group because they can represent incapable respondents, giving a reference to the inferred abilities of the classifiers.

6.4.2 Exploring item parameters

Figs. 6.3a and 6.3b show the empirical distributions of the mean of inferred difficulties and discriminations in the CLUSTERS and MOONS datasets. Instances near the true decision boundary have higher difficulties and lower discriminations for both datasets, whereas instances far away from the decision boundary have lower difficulties and higher discriminations. Fig. 6.4 illustrates ICCs fitting different abilities to items with different combinations of difficulty and discrimination. Our model provides more flexibility than logistic-shape ICC in both non-noisy and noisy cases.



(a) Dataset CLUSTERS

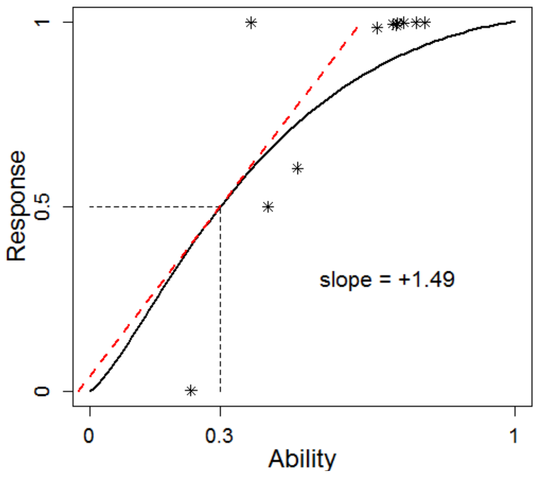


(b) Dataset MOONS.

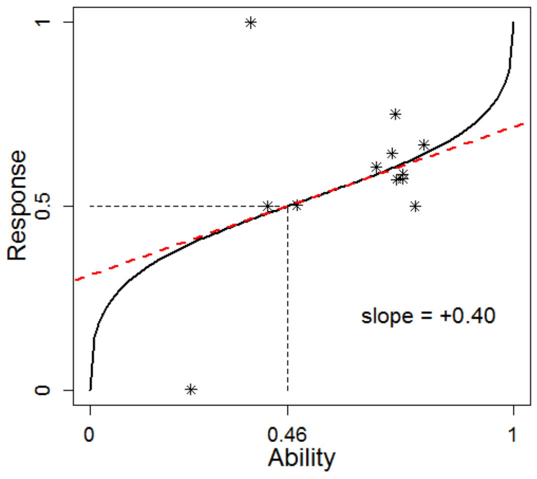
Figure 6.3: Inferred Latent Variables of Items of Two Synthetic Datasets: CLUSTERS and MOONS. Darker colour indicates higher value. Items closer to the class boundary get higher difficulty and lower discrimination.

There are some items inferred to have negative discrimination: these are mostly items with incorrect labels, as shown in Fig. 6.5a. The negative discrimination fits the case when a low-valued response (correctness) is given by a classifier with high ability to an item with low difficulty. Figs. 6.5a and 6.5b shows that negative discrimination flips high difficulty to low difficulty in comparison with the results where the discrimination is fixed. Figs. 6.5c and 6.5d show that when there are no noisy items, no negative discriminations are inferred by the model.

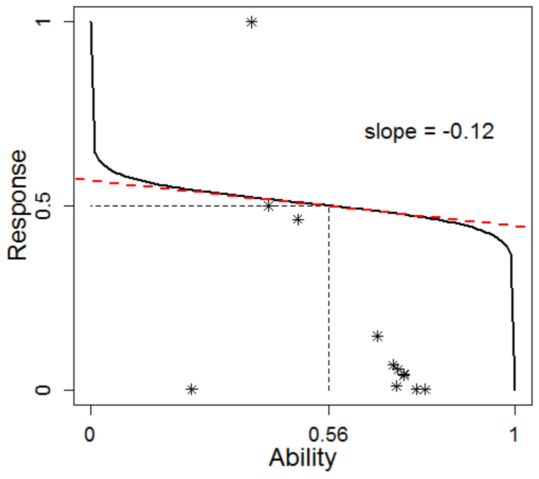
However, unlike the common setting of label noise detection (Frénay & Verleysen, 2014), using negative discrimination to identify noisy labels requires that the training set can only include a few noisy examples. It is because the negative discriminations are inferred by items



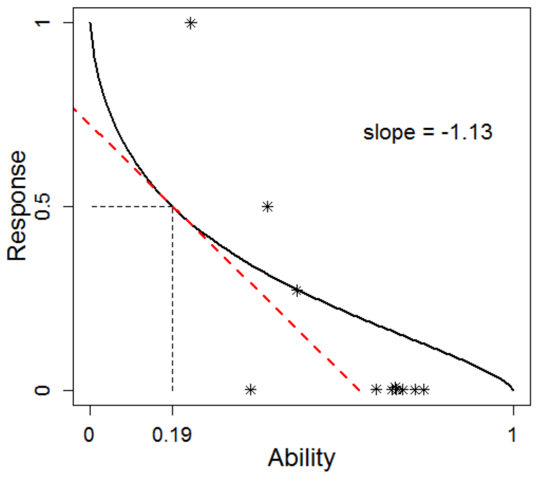
(a) Difficulty low, positive discrimination high. Instance is non-noisy item far from decision boundary.



(b) Difficulty high, positive discrimination low. Instance is non-noisy item close to decision boundary.



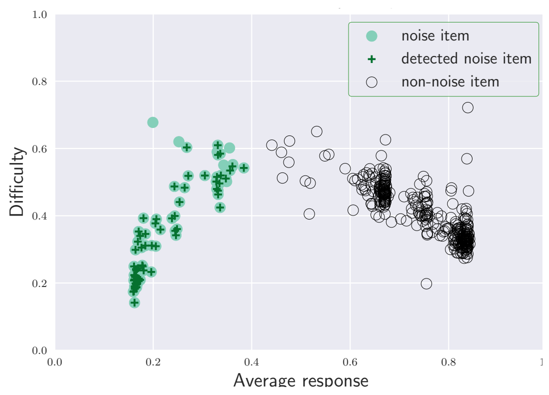
(c) Difficulty high, negative discrimination low. Instance is noisy item close to decision boundary.



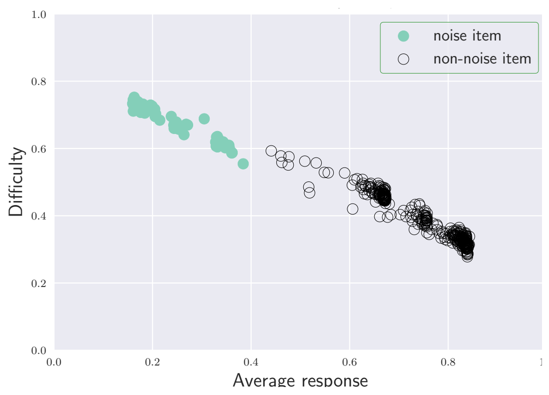
(d) Difficulty low, negative discrimination high. Instance is noisy item far from decision boundary.

Figure 6.4: Examples of ICC in the CLUSTERS Dataset. Stars are the actual classifier responses fit by the ICCs.

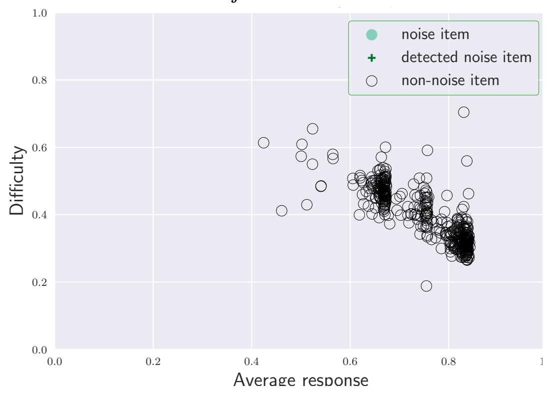
receiving low responses from classifiers with high abilities. Noise in the training set introduces noise to classifiers' abilities (especially for methods like KNNs). Therefore, the model will not be able to infer negative discrimination correctly since the responses given by the classifiers are not reliable. The experimental results of such cases are compared in Fig. 6.6. This is a common issue in ensemble-based approaches for noise detection, which has been addressed for instance by Sluban & Lavrač (2015). Our model can be trained on a small noise-free training set and then updated incrementally with identified non-noisy items, which could still be practical in real applications. In contrast, in experiments using human responses, such as student exams,



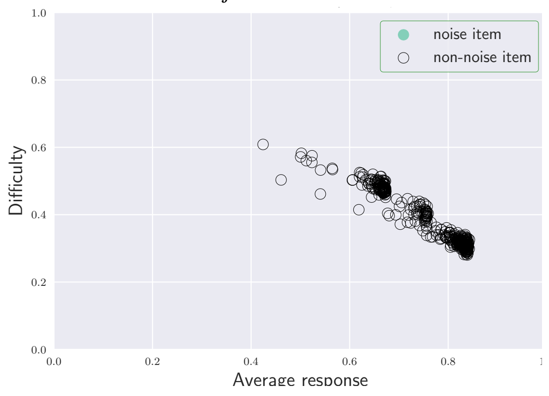
(a) a_j is learned.



(b) a_j is fixed to 1.

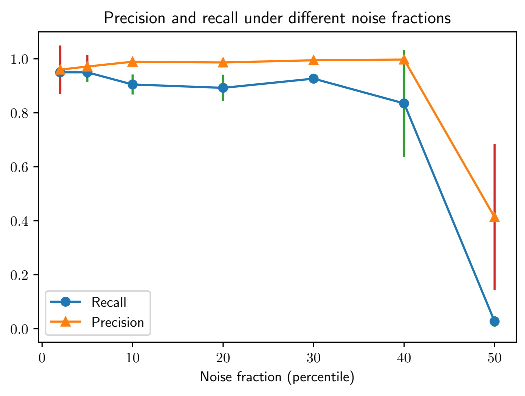


(c) a_j is learned.

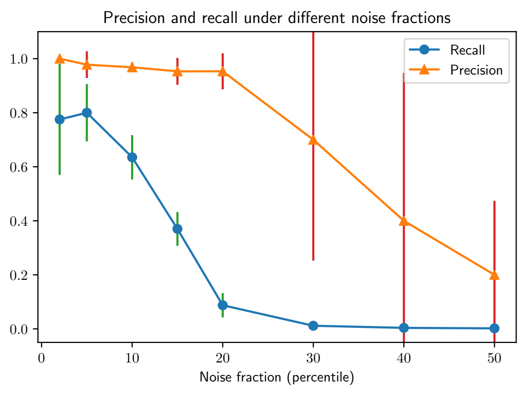


(d) a_j is fixed to 1.

Figure 6.5: Correlation between Average Response and Difficulty Changes. (Under different settings of discrimination, shown for classes 3 vs 5 of MNIST dataset. (a), (b) are from validation data with 20% injected noise; (c), (d) no noise.)



(a) Noise only in validation set



(b) Noise in training and validation set

Figure 6.6: Denoising Performance of Negative Discrimination in Different Settings. Tested on MNIST dataset, means and standard deviations over 5 runs of all combinations of any two classes.

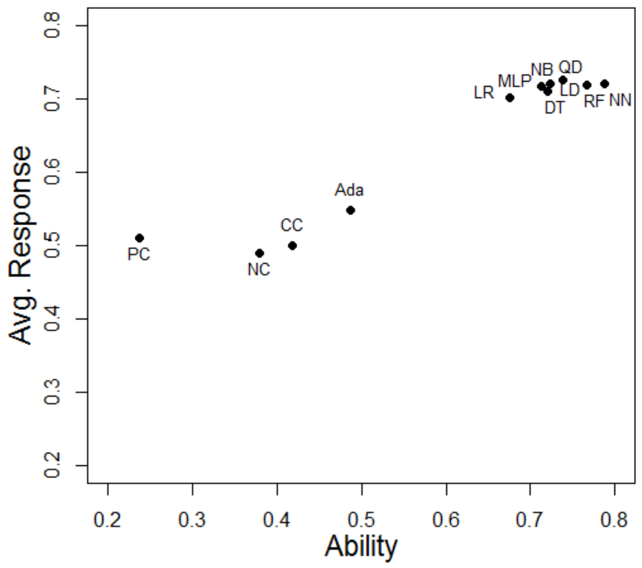


Figure 6.7: Ability vs Average Response in the CLUSTERS dataset. The classifiers in the top right getting similar avg, response around 0.7, but their abilities are diverse from 0.65 to 0.8.

there is no separate training set to build students' abilities before getting their responses to questions because the students are assumed to be trained by formal education already. In such cases, β^3-IRT can be a good option for detecting suspicious questions that might have wrong standard answers.

6.4.3 Assessing the ability of classifiers

Fig. 6.7 shows a monotonic increasing relation between ability and average response except the top right and bottom left of the figure. However, most classifiers are in the top right part, with ability between 0.7 and 0.8 and avg. response around 0.72, and the highest ability does not correspond to the highest avg response. This is because the inferred ability can be considered as an element-wise weighted mean which we will discuss now.

Tabs. 6.1 and 6.3 shows the comparison between abilities and several popular classifier evaluation metrics on the MNIST and CLUSTERS dataset, respectively. Tabs. 6.2 and 6.4 give the Spearman's rank correlation between these metrics for the two datasets. We can see that ability behaves differently from the other metrics because it is not using uniform aggregates of predicted probabilities (unlike log-loss, Brier score), instead, the aggregation is weighted by the difficulties and discriminations of corresponding items. This can be seen from the equation below:

$$\left(\frac{1}{\bar{p}_{ij}}-1\right)^{1/a_j}\left(\frac{1}{\delta_j}-1\right)=\frac{1}{\theta_i}-1,\quad \bar{p}_{ij}=\frac{\alpha_{ij}}{\alpha_{ij}+\beta_{ij}} \tag{6.7}$$

Here, $\bar{p}_{ij}$ is the expected response of item j given by classifier i, α_{ij} and β_{ij} are defined in Eq. (6.4). For example, a low $\bar{p}_{ij}$ for a difficult instance will not give high penalty to the ability θ_i.

Assume the discrimination $a_j = 1$, according to Eq. (6.7):

$$\left(\frac{1}{\bar{p}_{ij}} - 1\right)\left(\frac{1}{\delta_j} - 1\right) + 1 = \frac{1}{\theta_i},$$

We have four typical scenarios of a classifier:

1) giving a high response (high correctness) to an easy item, e.g., $\bar{p}_{i,j} = 0.8,\ \ \delta_j = 0.2 \rightarrow \theta_i = 0.5$;
2) giving a low response (low correctness) to a difficult item, e.g., $\bar{p}_{i,j} = 0.2,\ \ \delta_j = 0.8 \rightarrow \theta_i = 0.5$;
3) giving a high response to a difficult item, e.g., $\bar{p}_{i,j} = 0.8,\ \ \delta_j = 0.8 \rightarrow \theta_i = \frac{16}{17} \approx 0.94$;
4) giving a low response to an easy item, e.g., $\bar{p}_{i,j} = 0.2,\ \ \delta_j = 0.2 \rightarrow \theta_i = \frac{1}{17} \approx 0.06$.

The first two scenarios are likely to happen for most classifiers and the model tends to infer the ability as mediocre. If we initialize the ability as 0.5, then both cases will not make notable change on it. The third and fourth scenarios will change the ability towards 1 and 0, respectively, which means giving a difficult item a correct response will gain most on the ability and the vice versa. Moreover, when $\frac{1}{\bar{p}_{ij}} - 1 < 1$ (response is high), if $0 < a_j < 1$ then the strength of increasing θ_i will be stronger; when $\frac{1}{\bar{p}_{ij}} - 1 > 1$ (response is low), if $a_j > 1$ then the strength of decreasing θ_i will be weaker. As demonstrated in Fig. 6.3, we see that when difficulty δ_j is high the discrimination a_j is often close to zero, and vice versa. It indicates with non-fixed discrimination getting higher response on a difficult item has stronger strength to change the ability than getting lower response on an easy item. According to these results we see that the difficulty of each item plays an important role for inferring the ability of a classifier. In addition, the difficulty of an item can be viewed as the *relative* difficulty to a specific group of respondents since it depends on the average response over all respondents (Fig. 6.5). Consequently, the assessed ability is *relative* ability among this certain group as well.

The ability learned by β^3-IRT provides a new scaled measurement for classifiers, which evaluates their performance on a weighted instance-wise basis. This characteristic also results in adaptive behavior of the ability when the classifiers are evaluated on different datasets, for instance, the Spearman's rank correlation between ability and several other measures (e.g. accuracy, F1, Brier score, log-loss) are very different on the MNIST (Tab. 6.2) and CLUSTERS (Tab. 6.4) datasets. It helps to identify the classifier that is most capable of classifying difficult items, which can be very helpful in an ensemble or crowd-sourcing scenario. Another advantage of ability as a measurement of classifiers is that it is robust to noisy validation data. Fig. 6.8 demonstrates that the inferred abilities of the classifiers stay nearly constant as the noise fraction in the validation set is increased until half of the validation points are incorrectly labeled. In contrast, other measures will be highly affected by noisy items since they are determined by the given labels, e.g., average accuracy will monotonically decreasing when increasing noisy items.

Table 6.1: Comparison between Ability and other Metrics (MNIST). ↑ indicates the higher the better and vice versa.

	Avg. Resp. ↑	Ability ↑	Accuracy ↑	F1 score ↑	Brier score ↓	Log loss ↓	AUC ↑
DT	0.7398	0.7438	0.7425	0.7297	0.2337	1.1537	0.7776
NB	0.6439	0.7423	0.6425	0.6951	0.3533	10.6097	0.6799
MLP	0.7826	0.8384	0.7825	0.774	0.2086	2.457	0.7951
Ada.	0.5621	0.4887	0.775	0.7656	0.2036	0.5959	0.79
RF	0.7215	0.7395	0.7725	0.7573	0.1926	3.4979	0.8119
LDA	0.7185	0.8052	0.72	0.7098	0.2714	5.1328	0.7276
QDA	0.5948	0.5892	0.595	0.6611	0.405	13.9884	0.5854
LR	0.7699	0.8001	0.7775	0.7688	0.2059	1.4246	0.7939
KNN	0.7645	0.8228	0.7675	0.7572	0.2111	6.3816	0.8011

Table 6.2: Spearman's Rank Correlation between Ability and other Metrics (MNIST)

	Avg. Resp.	Ability	Accuracy	F1	Brier socre	Log loss	AUC
Avg. Resp.	1.0	0.8333	0.6	0.6	0.2833	0.2	0.6
Ability	0.8333	1.0	0.3333	0.3333	-0.05	-0.05	0.35
Accuracy	0.6	0.3333	1.0	1.0	0.8333	0.7	0.75
F1	0.6	0.3333	1.0	1.0	0.8333	0.7	0.75
Brier	0.2833	-0.05	0.8333	0.8333	1.0	0.6833	0.8333
Log loss	0.2	-0.05	0.7	0.7	0.6833	1.0	0.3667
AUC	0.6	0.35	0.75	0.75	0.8333	0.3667	1.0

Table 6.3: Comparison between Ability and other Metrics (CLUSTERS)

	Avg. Resp. ↑	Ability ↑	Accuracy ↑	F1 ↑	Brier score ↓	Log loss ↓	AUC ↑
DT	0.7113	0.7226	0.7175	0.7154	0.2456	1.0114	0.7596
NB	0.7217	0.7388	0.75	0.7487	0.2221	1.0718	0.7682
MLP	0.7195	0.7263	0.7375	0.7342	0.2233	1.0092	0.7652
Ada.	0.5478	0.4623	0.725	0.7277	0.2148	0.6204	0.7571
RF	0.7206	0.7741	0.7275	0.7241	0.2304	5.7305	0.7648
LDA	0.7251	0.7488	0.745	0.745	0.2244	1.1623	0.7683
QDA	0.7255	0.7549	0.7475	0.7469	0.2242	1.1802	0.768
LR	0.7017	0.6848	0.7375	0.7328	0.2141	0.8071	0.7677
KNN	0.7212	0.7899	0.7325	0.7332	0.2389	6.8574	0.7582

6.5 Measuring Classifiers in Continual Learning

We have demonstrated that the ability assessed by β^3-IRT provides a new measure for classifiers which is adaptively weighted by item difficulties. In continual learning, we can consider the difficulty of a data instance having two extra sources:

1) how much the performance on it will drop if it is from an old task (i.e., the tendency of forgetting),
2) how much the performance on it will be inhibited if it is from a new task (i.e., the tendency

Table 6.4: Spearman's Rank Correlation between Ability and other Metrics (CLUSTERS)

	Avg. Resp.	Ability	Accuracy	F1	Brier score	Log loss	AUC
Avg. Resp.	1.0	0.75	0.7197	0.7333	-0.3167	-0.7333	0.6667
Ability	0.75	1.0	0.2678	0.2833	-0.6	-0.9833	0.1667
Accuracy	0.7197	0.2678	1.0	0.954	0.3766	-0.1925	0.8619
F1	0.7333	0.2833	0.954	1.0	0.3167	-0.2	0.7333
Brier	-0.3167	-0.6	0.3766	0.3167	1.0	0.6833	0.2167
Log loss	-0.7333	-0.9833	-0.1925	-0.2	0.6833	1.0	-0.1333
AUC	0.6667	0.1667	0.8619	0.7333	0.2167	-0.1333	1.0

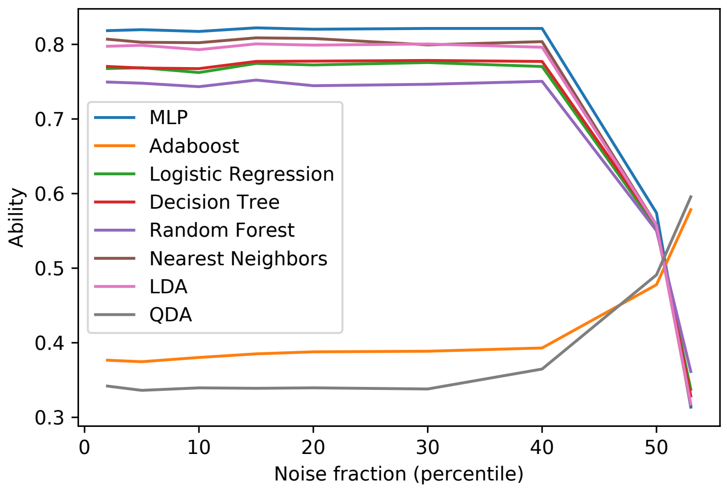


Figure 6.8: Ability of classifiers with different noise fractions of validation data, which shows the ability is robust to noisy validation data.

of intransigence).

In the case of continual learning, the ability can be viewed as an aggregated measure that jointly evaluates the forgetting and intransigence over all instances. The average intransigence and forgetting (Chaudhry et al., 2018) are often contradictory to each other as shown in Sec. 5.5 which is a common phenomenon in continual learning, i.e., a lower forgetting usually accompanies with a higher intransigence. It is due to the intrinsic resource limitation of continual learning since we want the model continuously learn more tasks with very limited resource expansion (small increase of the memory or model parameters). A model with a better trade-off between forgetting and intransigence often obtains a higher average accuracy. However, the average accuracy cannot tell the difference between some cases, for instance, a model with high forgetting and low intransigence may have the same average accuracy with a model with low forgetting and high intransigence. The ability is capable of giving more insights in such cases by an adaptive manner, which depends on the difficulties caused by forgetting and intransigence. We will demonstrate it in the remainder of this section.

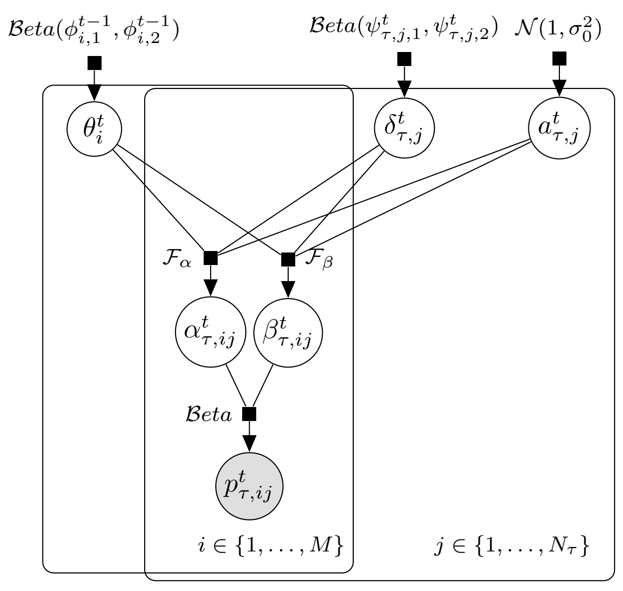


Figure 6.9: Factor graph of continual β^3-IRT. t is the index of the current task, τ is the task index that satisfies $\tau \leq t$, N_τ is the number of items from the τ-th task. The prior of ability θ_i in the t-th task is the posterior of the $(t-1)$-th task. The prior of difficulty δ in the t-th task is the empirical prior computed by the number of correct and incorrect predictions given by all classifiers after learning the t-th task.

The experiments in the previous section are all based on binary classification tasks in static learning. In this section, we apply β^3-IRT to multi-classification tasks in continual learning using a different way of generating responses. To align with a more general setting across related work, we train each classifier multiple times (such as with different random seeds) and collect the predictions on validation sets of each time. The final response is computed as $p_{ij} = \sum_{s=1}^{S} \mathbb{I}(y_j = \hat{y}_{ij,s})/S$, where y_j is the true class of the j-th instance, $\hat{y}_{ij,s}$ is the predicted class given by the i-th classifier after s-th training, and S is the total number of training times which is set to 10 in our experiments. In this way, p_{ij} can be viewed as the probability of the i-th classifier giving the correct prediction on the j-th item.

We apply the β^3-IRT model to evaluate a set of classifiers in continual learning which follows the conventional settings: at task t, we train M classifiers by the training set of the t-th task and generate responses by the validation sets of all learned tasks, where $t \in \{1, 2, \ldots, T\}$ and T is the total number of tasks. Analogously, we can also train the β^3-IRT model at each task after the classifiers have been trained in a continuous manner. Since it is a probabilistic model, we can apply the framework of VCL by simply changing the priors of ability θ_i at task t to the inferred posteriors at task $t-1$, which we call Continual β^3-IRT (CBIRT). Since the prior and initialization of ability is inherited from the previous task, the ability is learned under an implicit assumption that it is the same variable across tasks. In other words, the ability defined in CBIRT reflects the overall performance on the whole sequence of tasks rather than the performance on individual tasks. The factor graph of CBIRT is depicted in Fig. 6.9, where τ is index of a learned task and t is the index of the latest task, hence $\tau \leq t$. N_τ is the number of instances of the τ-th task. $p^t_{\tau,ij}$ denotes the response from classifier i to j-th instance of the τ-th task after learning t-th task . θ^t_i denotes the *ability* of classifier i at task t. $\delta^t_{\tau,j}$, $a^t_{\tau,j}$ denote the *difficulty* and *discrimination* of the j-th item of the τ-th task evaluated at task t, respectively. We set the prior of difficulty ($Beta(\psi^t_{\tau,j,1}, \psi^t_{\tau,j,2})$) by an empirical estimation, where $\psi^t_{\tau,j,1}$ and $\psi^t_{\tau,j,2}$ is the number of classifiers that have incorrectly and correctly classify this instance, respectively. We configure the prior of difficulty in this way because the difficulty of an instance is often linearly correlated to the average response of it (Fig. 6.5). We do not set the previous posterior as the prior of item parameters (difficulty and discrimination) because such parameters can change significantly in continual learning due to catastrophic forgetting. The model inference can be done as the same as Eqs. (6.6) and (6.6). When $t = 1$ the priors are as the same as in Fig. 6.1.

We apply the same experimental settings as in Sec. 5.5 with Split MNIST and Split Fashion-MNIST tasks, i.e., all experiments comply with the online training with a single-headed model. The training set has 1000 samples and all methods have the access to a memory buffer with 300 samples. We have tested CBIRT on 14 classifiers including 10 approaches of continual learning: OEWC (Schwarz et al., 2018), A-GEM (Chaudhry et al., 2019a), iCaRL (Rebuffi et al., 2017), MER (Riemer et al., 2019), ER (Chaudhry et al., 2019b), GSS-greedy (Aljundi et al., 2019b), Multisimilarity (Wang et al., 2019), R-Margin (Roth et al., 2020), BER (Sec. 5.2), DRL (Sec. 5.2). As OEWC performs very poorly in the single-headed setting, we enhance it by adding the replay loss of ER to it so that it can make the usage of the memory buffer as well. In addition, we deploy 4 simulated classifiers to obtain better insights on the assessed ability: Calibrated Constant Classifier (the response $p^t_{\tau,ij} = 1/K_t$ where K_t is the number of classes at task t), Perfect Classifier (the response $p^t_{\tau,ij} = 1$), Perfect First Task Classifier (which has perfect responses on the first task but random guesses on other tasks $p^t_{1,ij} = 1, p^t_{\tau,ij} = 1/K_t, \forall \tau \neq 1$), Perfect Latest Task Classifier (which has perfect responses on the latest task but random guesses on other tasks $p^t_{t,ij} = 1, p^t_{\tau,ij} = 1/K_t, \forall \tau \neq t$). The results are shown in Tabs. 6.5 and 6.6.

We compare the assessed ability of all classifiers with the usual measurements of continual learning (Sec. 3.2): average accuracy, average forgetting, and average intransigence. We rescale the ability by min-max normalization:

$$\tilde{\theta}_i = (\theta_i - \min_{i'} \theta_{i'})/(\max_{i'} \theta_{i'} - \min_{i'} \theta_{i'})$$

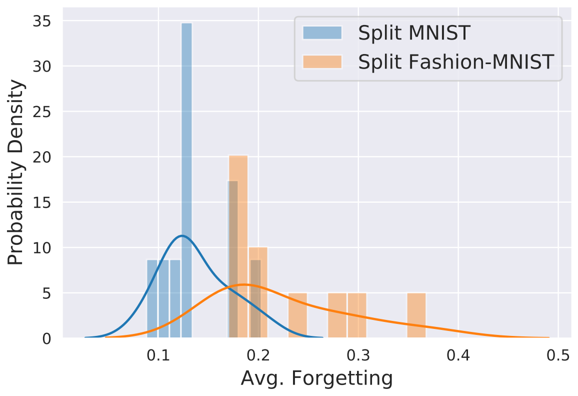
Thus, the Calibrated Constant Classifier gets 0 ability and the Perfect Classifier gets 1 ability on both benchmarks. The learned ability of the Perfect Classifier is not 1 because not all items push it's ability to 1 as we explained in the previous section, and the final result also depends on the learning rate and iteration numbers. And the Calibrated Constant Classifier gets an analogous situation. Nonetheless, the Perfect Classifier and the Calibrated Constant Classifier always get the highest and lowest ability, respectively. Hence, the min-max normalization can rescale the

Table 6.5: Comparing ability with other measurements on Split MNIST tasks, the numbers in the parentheses are rankings under each measurement. ↑ indicates the higher the better and vice versa.

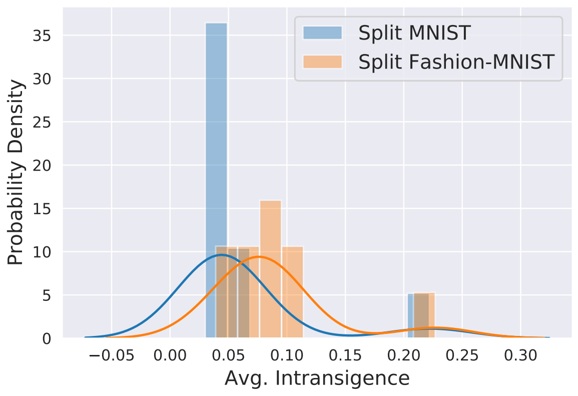
	Ability ↑	Avg. Acc.↑	Avg. Forget. ↓	Avg. Intrans. ↓
A-GEM	0.908 (6)	0.846 (7)	0.124 (6)	0.055 (10)
DRL	0.927 (2)	0.882 (2)	0.088 (3)	0.048 (9)
iCaRL	0.764 (11)	0.676 (11)	0.129 (8)	0.222 (12)
ER	0.904 (8)	0.834 (8)	0.170 (12)	0.030 (3)
OEWC	0.884 (10)	0.797 (10)	0.203 (13)	0.041 (5)
Multisim	0.921 (3)	0.866 (3)	0.117 (5)	0.041 (6)
BER	0.912 (5)	0.850 (5)	0.133 (9)	0.044 (8)
GSS	0.900 (9)	0.830 (9)	0.169 (11)	0.034 (4)
R-Margin	0.916 (4)	0.857 (4)	0.127 (7)	0.042 (7)
MER	0.906 (7)	0.849 (6)	0.109 (4)	0.064 (11)
Constant	0.000 (14)	0.100 (14)	0.160 (10)	0.772 (14)
Perfect	1.000 (1)	1.000 (1)	0.000 (1)	0.000 (1)
Perfect First	0.338 (12)	0.280 (12)	0.060 (2)	0.672 (13)
Perfect Latest	0.281 (13)	0.280 (12)	0.900 (14)	0.000 (1)

Table 6.6: Comparing ability with other measurements on Split Fashion-MNIST tasks, the numbers in the parentheses are rankings under each measurement.

	Ability	Avg. Acc.	Avg. Forget.	Avg. Intrans.
A-GEM	0.738 (10)	0.666 (10)	0.368 (13)	0.039 (3)
DRL	0.824 (2)	0.778 (2)	0.170 (4)	0.086 (9)
iCaRL	0.717 (11)	0.631 (11)	0.178 (7)	0.227 (12)
ER	0.809 (7)	0.757 (7)	0.241 (10)	0.051 (4)
OEWC	0.745 (9)	0.692 (9)	0.298 (12)	0.072 (6)
Multisim	0.823 (3)	0.769 (3)	0.191 (8)	0.078 (7)
BER	0.819 (4)	0.765 (5)	0.192 (9)	0.084 (8)
GSS	0.785 (8)	0.718 (8)	0.274 (11)	0.063 (5)
R-Margin	0.817 (6)	0.765 (6)	0.170 (5)	0.099 (11)
MER	0.819 (4)	0.767 (4)	0.171 (6)	0.096 (10)
Constant	0.000 (14)	0.100 (14)	0.160 (3)	0.772 (14)
Perfect	1.000 (1)	1.000 (1)	0.000 (1)	0.000 (1)
Perfect First	0.398 (12)	0.280 (12)	0.060 (2)	0.672 (13)
Perfect Latest	0.399 (13)	0.280 (12)	0.900 (14)	0.000 (1)



(a) Distributions of average forgetting



(b) Distributions of average intransigence

Figure 6.10: The empirical distribution of average forgetting (a) and intransigence (b) of 10 continual learning methods on both benchmarks. The two benchmarks have very different distributions of the average forgetting but similar distributions of the average intransigence.

values to match our understanding of the classifiers' ability. The 10 approaches of continual learning have lower abilities on split Fashion-MNIST than on split MNIST, which is consistent with the average accuracy. We can see that the ability gives very similar rankings with the average accuracy, but there exist a few cases having different rankings, such as A-GEM and MER in split-MNIST, BER and MER in split Fashion-MNIST. These cases have a common property: one classifier shows better performance on forgetting but worse performance on intransigence than the other.

To obtain better insights on the difference between ability and average accuracy, we provide further analysis based on the results of the Perfect First Classifier and Perfect Latest Classifier in the following. As we can see, the Perfect First Classifier and Perfect Latest Classifier have the same average accuracy in both cases, and their average forgetting and intransigence remain unchanged on different benchmarks due to their definitions. However, the Perfect First Classifier gets much higher ability than the Perfect Latest Classifier on split MNIST tasks. This is because the ability assessed by β^3-IRT is actually the *relative* ability among all classifiers which depends on the *relative* difficulty of each items as we analyzed in the previous section. We see that most classifiers show lower average forgetting on split MNIST than on split Fashion-MNIST tasks which gives the Perfect Latest Classifier a larger ability gap with other classifiers on split MNIST. We demonstrate this by the empirical distributions of the average forgetting and intransigence of the 10 continual learning methods in Fig. 6.10. Most classifiers have average forgetting below 0.15 on split MNIST but exhibit much higher forgetting on split Fashion-MNIST, whereas the Perfect Latest Classifier has 0.9 forgetting on both benchmarks. This indicates that on split MNIST more easy items of old tasks receiving low responses from the Perfect Latest Classifier, which pushes its ability towards 0 in a larger extent. On the other hand, the distributions of average intransigence on both benchmarks are similar and hence the Perfect Latest Classifier has not obtained enough gains by its low intransigence to outperform the Perfect First Classifier on either benchmark. The results demonstrate that the ability is able to evaluate classifiers in a more sensitive way than average accuracy. It combines the performance on forgetting and intransigence in an instance-wise weighted manner which makes it a more informative criterion in continual learning.

6.6 Summary

In this chapter we introduce the β^3-IRT model that can produce a new family of ICCs including logistic and anti-logistic curves by adopting a new formulation of IRT. This model enables more expressive class of response patterns and provides a new measurement for assessing performance of classifiers in both static and continual learning.

The experimental results showed that item parameters inferred by the β^3-IRT model can provide useful insights for difficult or noisy instances, and the inferred latent ability variable serves to evaluate classifiers on an instance-wise basis regarding the difficult levels of each instance. And in continual learning, the difficulty can also be caused by forgetting and intransigence, which makes the ability evaluate classifiers in a more flexible and informative way than the average accuracy. Note that the assessed ability depends on the context of the evaluation, i.e., the group of classifiers, which indicates it could be incompatible to the ability assessed over other groups. Adding the Calibrated Constant Classifier and Perfect Classifier to rescale the abilities could be a solution to this issue as we have done it for experiments in continual learning (Tabs. 6.5 and 6.6).

Besides classification tasks in continual learning, generative tasks in continual learning have different training process and evaluation criteria. In the next chapter, we will introduce a new method for evaluating generative models under the restrictive setting of continual learning, i.e., there is no access to the training data of learned tasks.

CHAPTER 7

MEASURING GENERATIVE MODELS IN CONTINUAL LEARNING BY CONTINUAL DENSITY RATIO ESTIMATION

In this chapter we introduce a novel method, Continual Density Ratio Estimation (CDRE), which can be used for evaluating generative models in continual learning. CDRE approximates density ratios between the initial and current distributions (p/q_t) of a data stream in an iterative fashion without the need of storing past samples, where q_t is shifting away from p over time t. CDRE can be applied in scenarios of online learning, such as importance weighted covariate shift, tracing dataset changes for better decision making. Particularly, CDRE enables the evaluation of generative models under the rigorous setting of continual learning. To the best of our knowledge, there is no existing method that can evaluate generative models in continual learning without storing samples from the original distribution. This chapter is structured as follows. Sec. 7.1 introduces the problem background of our method and compares several related works. In Sec. 7.2 we firstly provides technique details of preliminaries of CDRE, including the formal problem setting of CDRE, and a brief review of KLIEP which is a basic method for direct density ratio estimation. We secondly elaborate the technique details of CDRE and demonstrate the instantiation of CDRE by KLIEP. We finally provide the theoretical analysis of asymptotic normality of this method. In Sec. 7.3 we demonstrate several online applications of CDRE and Sec. 7.4 shows experimental results of evaluating generative models by CDRE in continual learning. Finally, we give some further discussion about CDRE and its applications Sec. 7.5.

7.1 Introduction

Online applications are ubiquitous in practice since large amounts of data are generated and processed in a streaming manner. There are two types of machine learning scenarios commonly deployed for such streaming data:

1) train a model online on the streaming data (*e.g.* online learning (Shalev-Shwartz et al., 2012) and continual learning (Parisi et al., 2019)) – in this case the training set may be shifting over time;
2) train a model offline and deploy it online – in this case the test set may be shifting over time.

In both cases, the main problem is dataset shift, i.e. the data distribution changes gradually over time. Awareness of how far the training or test set has shifted can be crucial to the performance of the model. For example, when the training set is shifting, the latest model may become less accurate on test samples from earlier data distributions. In this case, we can apply 'backward' covariate shift (Shimodaira, 2000) (i.e. swapping the training and testing distributions in covariate shift) and 'rollback' the model by adding important weights to the loss function: $\mathbb{E}_{p_1(x)}[\mathscr{L}(x)] = \mathbb{E}_{p_t(x)}[\frac{p_1(x)}{p_t(x)}\mathscr{L}(x)]$, where the importance weights are density ratios $\frac{p_{t-1}(x)}{p_t(x)}$. In the other case, the performance of a pre-trained model may gradually degrade when the test set shifts away from the training set over time. It will be beneficial to trace the distribution difference caused by dataset shift so that we can decide when to update the model for preventing the performance from degradation.

Density Ratio Estimation (DRE) (Sugiyama et al., 2012) is a method for estimating the ratio between two probability distributions which can reflect the difference between the two distributions. In particular, it can be applied to settings in which only samples of the two distributions are available, which is usually the case in practice. However, under certain restrictive conditions in online applications – e.g., unavailability of historical samples in an online data stream – existing DRE methods are no longer applicable. Moreover, DRE exhibits difficulties for accurate estimations when there exists significant differences between the two distributions (Sugiyama et al., 2012; McAllester & Stratos, 2020; Rhodes et al., 2020). In this chapter, we propose a new framework of density ratio estimation called Continual Density Ratio Estimation (CDRE) which is capable of coping with the online scenarios and gives better estimation than standard DRE when the two distributions are less similar.

Moreover, CDRE can trace the differences between distributions by estimating their f-divergences and thus it provides a new option for evaluating generative models in continual learning. The scenario of the training set shifting over time matches the problem setting of continual learning (Parisi et al., 2019) in which a single model is trained by a set of tasks sequentially with no (or very limited) access to the data from past tasks, and yet is able to perform on all learned tasks. The existing methods of evaluating generative models are to estimate the difference between the original data distribution and the distribution of model samples (Heusel et al., 2017; Bińkowski et al., 2018). All those methods require the samples from the original data distribution which may not be possible in some applications of continual learning. For instance, when we train a generative model to simulate the residents' activities in a smart house by camera data and due to privacy regulations, we can only access to the data of the latest 7 days. It may be difficult to evaluate longer periodic activities generated by the model with the existing methods, yet CDRE can fit in such a situation.

7.2 Continual Density Ratio Estimation

In this section we first formally introduce the problem setting of CDRE, then we briefly review the formulation of KLIEP which we use for instantiating CDRE. After that we describe the basic formulation of CDRE and demonstrate instantiating it by KLIEP which we call CKLIEP. We further provide theoretical analysis about the asymptotic normality of CKLIEP. Finally, we discuss some techniques for dimensionality reduction in applications of CDRE.

7.2.1 The problem setting of CDRE

Suppose we want to estimate density ratios between two distributions $r_{\tau,t}(x) = p_\tau(x)/q_{\tau,t}(x), t \geq \tau$, where τ is the initial time index of starting tracing the distribution $p_\tau(x)$, t denotes the current time index. We refer to $p_\tau(x)$ as the *original distribution* and $q_{\tau,t}(x)$ as the *dynamic distribution* of $p_\tau(x)$. The dynamic distribution is assumed to be shifting away from its original distribution gradually over time. When $t > \tau$ the samples of $p_\tau(x)$ are not available any longer. For example, let $\tau = 1$, at $t = 1$ we have access to samples of both p_τ and $q_{\tau,t}$, we can directly estimate $r_{\tau,t}$ by standard DRE; and when $t > 1$, we no longer have access to samples of p_τ, instead, we only have access to samples of $q_{\tau,t-1}$ and $q_{\tau,t}$. CDRE is proposed to estimate $r_{\tau,t}$ in such a situation.

Note that CDRE is able to estimate ratios of multiple pairs of the original and dynamic distributions by a single estimator. It avoids building separated estimators for tracing different original distributions in an application (e.g. seasonal data), which also fits the common setting of continual learning that we will introduce in the latter sections. For convenience, we initially introduce the formulation with a single pair of original and dynamic distributions. We then give a more general formulation for multiple pairs of original and dynamic distributions.

7.2.2 Kullback–Leibler Importance Estimation Procedure (KLIEP)

KLIEP is a basic method for density ratio estimation introduced in Sugiyama et al. (2008). We will deploy it as an example of the basic estimator of CDRE in Sec. 7.2.4.

Let $r^*(x) = p(x)/q(x)$ be the (unknown) true density ratio, then $p(x)$ can be estimated by $\tilde{p}(x) = r(x)q(x)$, where $r(x)$ is an estimation of $r^*(x)$. Hence, we can optimize $r(x)$ by minimizing the KL-divergence between $p(x)$ and $\tilde{p}(x)$ with respect to r:

$$\begin{aligned} &D_{KL}(p(x)||\tilde{p}(x)) = \int p(x) \log \frac{p(x)}{\tilde{p}(x)} dx = \int p(x) \log r^*(x) dx - \int p(x) \log r(x) dx, \\ &s.t. \;\; r(x) > 0, \;\; \int r(x)q(x)dx = \int \tilde{p}(x)dx = 1 \end{aligned} \tag{7.1}$$

As the first term of the right-hand side in Eq. (7.1) is constant w.r.t. $r(x)$, the empirical objective of optimizing $r(x)$ is as follows:

$$J_r = \max_r \frac{1}{N}\sum_{i=1}^{N} \log r(x_i), \;\; x_i \sim p(x), \qquad s.t. \;\; \frac{1}{M}\sum_{j=1}^{M} r(x_j) = 1, \;\; x_j \sim q(x), \;\; r(x) \geq 0. \tag{7.2}$$

The expectations are estimated by Monte Carlo integration so the convergence of this method is highly related to the number of samples. It results in a drawback of such method that the estimation may have large variance with a small number of samples. Unlike importance sampling in which the denominator distribution can be chosen with interests and the density function of the nominator distribution is often available, in DRE both distributions are pre-decided and only samples of them are available. It is also the reason that DRE becomes more difficult when the two distributions are less similar.

One convenient way of parameterizing $r(x)$ is by using a log-linear model with normalization, which then automatically satisfies the constraints in Eq. (7.2):

$$r(x;\beta) = \frac{\exp(\psi_\beta(x))}{\frac{1}{M}\sum_{j=1}^{M}\exp(\psi_\beta(x_j))}, \quad x_j \sim q(x), \quad \psi_\beta : \mathbb{R}^D \to \mathbb{R}, \tag{7.3}$$

where ψ_β can be any deterministic function: we use a neural network as ψ_β in our implementations, β then representing parameters of the neural network.

7.2.3 The basic form of CDRE

For simplicity of notation, we assume $\tau = 1$ in the case of estimating density ratios between a *single* pair of original and dynamic distributions, and then omit τ in the basic formulations. Thus, the density function of the original distribution is denoted as $p(x)$ and its samples are unavailable when $t > 1$. Similarly, $q_t(x)$ denotes the density function of the dynamic distribution at time t. The true density ratio $r_t^*(x) \triangleq p(x)/q_t(x)$ can be decomposed as follows:

$$r_t^*(x) = \frac{q_{t-1}(x)}{q_t(x)}\frac{p(x)}{q_{t-1}(x)} = r_{s_t}^*(x) r_{t-1}^*(x), \;\; t > 1, \tag{7.4}$$

where $r_{s_t}^*(x) \triangleq q_{t-1}(x)/q_t(x)$ represents the true density ratio between the two latest dynamic distributions. Using this decomposition we can estimate $p(x)/q_t(x)$ in an iterative manner without the need of storing samples from $p(x)$ when t increases. The key point is that we can estimate $r_t^*(x)$ by estimating $r_{s_t}^*(x)$ when the estimation of $r_{t-1}^*(x)$ is known. In particular, it introduces one extra constraint:

$$\int r_{s_t}^*(x) q_t(x) dx = \int \frac{r_t^*(x)}{r_{t-1}^*(x)} q_t(x) dx = 1 \tag{7.5}$$

Existing methods of DRE can be applied to estimating the initial ratio $r_1^*(x) = p(x)/q_1(x)$ and the latest ratio $r_{s_t}^*(x), \forall t > 1$, as the basic ratio estimator of CDRE. Let $r_t(x)$ be the estimation of

$r_t^*(x)$, where r_{t-1} is already obtained, then the objective of CDRE can be expressed as:

$$J_{CDRE}(r_t) = J_{DRE}\left(\frac{r_t}{r_{t-1}}\right), \quad s.t. \ \frac{1}{N}\sum_{n=1}^{N}\frac{r_t(x_n)}{r_{t-1}(x_n)} = 1, \ \ x_n \sim q_t(x). \tag{7.6}$$

where J_{DRE} can be the objective of any method used for standard DRE, such as KLIEP (Sugiyama et al., 2008).

7.2.4 An instantiation of CDRE: CKLIEP

We now demonstrate how to instantiate CDRE by KLIEP, which we call Continual KLIEP (CKLIEP). Define $r_t(x), r_{t-1}(x)$ by the log-linear form as in Eq. (7.3), let $N_t = N_{t-1} = N$ as the sample size of each distribution, then r_{s_t} is as follows:

$$\begin{aligned} r_{s_t}(x) &= \frac{r_t(x)}{r_{t-1}(x)} = \exp\{\psi_{\beta_t}(x) - \psi_{\beta_{t-1}}(x)\} \times \frac{\frac{1}{N}\sum_{j=1}^{N}\exp\{\psi_{\beta_{t-1}}(x_{t-1,j})\}}{\frac{1}{N}\sum_{i=1}^{N}\exp\{\psi_{\beta_t}(x_{t,i})\}}, \\ & x_{t,i} \sim q_t(x), \quad x_{t-1,j} \sim q_{t-1}(x). \end{aligned} \tag{7.7}$$

where β_t, β_{t-1} represent parameters of $r_t(x), r_{t-1}(x)$, respectively. When the constraint in Eq. (7.6) is satisfied, we have the following equality by substituting Eq. (7.7) into the constraint:

$$\frac{\sum_{i=1}^{N}\exp\{\psi_{\beta_t}(x_{t,i})\}}{\sum_{j=1}^{N}\exp\{\psi_{\beta_{t-1}}(x_{t-1,j})\}} = \frac{1}{N}\sum_{i=1}^{N}\exp\{\psi_{\beta_t}(x_{t,i}) - \psi_{\beta_{t-1}}(x_{t,i})\} \tag{7.8}$$

r_{s_t} can then be rewritten in the same log-linear form of Eq. (7.3) by substituting Eq. (7.8) into Eq. (7.7):

$$r_{s_t} = \frac{\exp\{\psi_{\beta_t}(x) - \psi_{\beta_{t-1}}(x)\}}{\frac{1}{N}\sum_{i=1}^{N}\exp\{\psi_{\beta_t}(x_{t,i}) - \psi_{\beta_{t-1}}(x_{t,i})\}} = \frac{\exp\{\phi_{\beta_t}(x)\}}{\frac{1}{N}\sum_{i=1}^{N}\exp\{\phi_{\beta_t}(x_i)\}}, \quad \phi_{\beta_t}(x) \triangleq \psi_{\beta_t}(x) - \psi_{\beta_{t-1}}(x). \tag{7.9}$$

Now we can instantiate J_{DRE} in Eq. (7.6) by the objective of KLIEP (Eq. (7.2)) and adding the equality constraint (Eq. (7.8)) into the objective with a hyperparameter λ_c, which gives the objective of CKLIEP as follows:

$$\begin{aligned} &\mathscr{L}_t^*(\beta_t) = \max_{\beta_t}\frac{1}{N}\sum_{j=1}^{N}\log r_{s_t}(x_{t-1,j}) + \lambda_c\left(\frac{\Psi_t(x_t)}{\Phi_t(x_t)\Psi_{t-1}(x_{t-1})} - 1\right)^2, \\ &\Phi_t(x_t) \triangleq \frac{1}{N}\sum_{i=1}^{N}\exp\{\phi_{\beta_t}(x_{t,i})\}, \ \ \Psi_t(x_t) \triangleq \frac{1}{N}\sum_{i=1}^{N}\exp\{\psi_{\beta_t}(x_{t,i})\} \\ &\text{where } \ t > 1, \ \ x_{t,i} \sim q_t(x), \ \ x_{t-1,j} \sim q_{t-1}(x), \end{aligned} \tag{7.10}$$

Here β_{t-1} is the estimated parameter of $r_{t-1}(x)$ and hence a constant in the objective.

7.2.5 Asymptotic normality of CKLIEP

Define $\hat{\beta}_t$ as the estimated parameter that satisfies:

$$\mathscr{L}_t'(\hat{\beta}_t) \triangleq \nabla_{\beta_t}\mathscr{L}_t(\beta_t)\big|_{\beta_t=\hat{\beta}_t} = 0 \tag{7.11}$$

Assume $\phi_{\beta_t}(x)$ (Eq. (7.9)) includes the correct function that there exists β_t^* recovers the true ratio over the population:

$$r^*_{s_t}(x) = \frac{q_{t-1}(x)}{q_t(x)} = \frac{\exp\{\phi_{\beta_t^*}(x)\}}{\mathbb{E}_{q_t}[\exp\{\phi_{\beta_t^*}(x)\}]}, \text{ where } \phi_{\beta_t^*}(x) = \psi_{\beta_t^*}(x) - \psi_{\beta_{t-1}}(x), \tag{7.12}$$

Notations: $\rightsquigarrow$ and $\xrightarrow{P}$ mean convergence in distribution and convergence in probability, respectively.

Assumptions: We assume $q_t(x)$ and $q_{t-1}(x)$ are independent, $n_t = n_{t-1} = n$, where n_t is the sample size of $q_t(x)$. Let S_t be the support of q_t, we assume $S_{t-1} \subseteq S_t$ in all cases.

Lemma 7.1. *Let $\ell_r'(\beta_t^*) \triangleq \frac{1}{n}\sum_{j=1}^n \nabla_{\beta_t} \log r_{s_t}(x_{t-1,j})|_{\beta_t=\beta_t^*}$, we have $\sqrt{n}\ell_r'(\beta_t^*) \rightsquigarrow \mathcal{N}(0,\sigma^2)$, where*

$$\sigma^2 = Cov_{q_{t-1}}[\nabla_{\beta_t}\phi_{\beta_t^*}(x)] + \frac{Cov_{q_t}[\nabla_{\beta_t}\exp\{\phi_{\beta_t^*}(x)\}]}{\mathbb{E}_{q_t}[\exp\{\phi_{\beta_t^*}(x)\}]^2}$$

Proof. Because

$$\nabla_{\beta_t} \log r_{s_t}(x) = \nabla_{\beta_t}\phi_{\beta_t}(x) - \frac{\sum_i^n \nabla_{\beta_t}\exp\{\phi_{\beta_t}(x_{t,i})\}}{\sum_i^n \exp\{\phi_{\beta_t}(x_{t,i})\}} \tag{7.13}$$

then

$$\sqrt{n}\ell_r'(\beta_t) = \frac{\sqrt{n}}{n}\sum_{j=1}^n \nabla_{\beta_t}\phi_{\beta_t}(x_{t-1,j}) - \sqrt{n}\frac{\frac{1}{n}\sum_i^n \nabla_{\beta_t}\exp\{\phi_{\beta_t}(x_{t,i})\}}{\frac{1}{n}\sum_i^n \exp\{\phi_{\beta_t}(x_{t,i})\}} \tag{7.14}$$

By the central limit theorem we have:

$$\begin{aligned}
&\frac{1}{n}\sum_{j=1}^n \nabla_{\beta_t}\phi_{\beta_t}(x_{t-1,j}) \rightsquigarrow \mathcal{N}\left(\mathbb{E}_{q_{t-1}}[\nabla_{\beta_t}\phi_{\beta_t}(x)], \frac{Cov_{q_{t-1}}[\nabla_{\beta_t}\phi_{\beta_t}(x)]}{n}\right),\\
&\frac{1}{n}\sum_i^n \nabla_{\beta_t}\exp\{\phi_{\beta_t}(x_{t,i})\} \rightsquigarrow \mathcal{N}\left(\mathbb{E}_{q_t}[\nabla_{\beta_t}\exp\{\phi_{\beta_t}(x)\}], \frac{Cov_{q_t}[\nabla_{\beta_t}\exp\{\phi_{\beta_t}(x)\}]}{n}]\right),
\end{aligned} \tag{7.15}$$

and by the weak law of large numbers:

$$\frac{1}{n}\sum_i^n \exp\{\phi_{\beta_t}(x_{t,i})\} \xrightarrow{P} \mathbb{E}_{q_t}[\exp\{\phi_{\beta_t}(x)\}] \tag{7.16}$$

Because $q_t(x)$ and $q_{t-1}(x)$ are assumed independent, combine the above results we get:

$$\begin{aligned}
&\sqrt{n}\ell_r'(\beta_t^*) \rightsquigarrow \mathcal{N}(\mu,\sigma^2),\\
&\mu = \sqrt{n}\left(\mathbb{E}_{q_{t-1}}[\nabla_{\beta_t}\phi_{\beta_t^*}(x)] - \frac{\mathbb{E}_{q_t}[\nabla_{\beta_t}\exp\{\phi_{\beta_t^*}(x)\}]}{\mathbb{E}_{q_t}[\exp\{\phi_{\beta_t^*}(x)\}]}\right),\\
&\sigma^2 = Cov_{q_{t-1}}[\nabla_{\beta_t}\phi_{\beta_t^*}(x)] + \frac{Cov_{q_t}(\nabla_{\beta_t}\exp\{\phi_{\beta_t^*}(x)\})}{\mathbb{E}_{q_t}[\exp\{\phi_{\beta_t^*}(x)\}]^2}
\end{aligned} \tag{7.17}$$

Taking derivatives from both sides of $1 = \int r^*_{s_t}(x) q_t(x) dx$:

$$
\begin{aligned}
0 &= \nabla_{\beta_t} \mathbb{E}_{q_t}[r^*_{s_t}(x)] = \int \nabla_{\beta_t} r^*_{s_t}(x) q_t(x) dx = \int \frac{\nabla_{\beta_t} r^*_{s_t}(x)}{r^*_{s_t}(x)} r^*_{s_t}(x) q_t(x) dx \\
&= \int \nabla_{\beta_t} \log r^*_{s_t}(x) q_{t-1}(x) dx = \mathbb{E}_{q_{t-1}}[\nabla_{\beta_t} \log r^*_{s_t}(x)] \\
&= \mathbb{E}_{q_{t-1}}[\nabla_{\beta_t} \phi_{\beta_t^*}(x)] - \frac{\mathbb{E}_{q_t}[\nabla_{\beta_t} \exp\{\phi_{\beta_t^*}(x)\}]}{\mathbb{E}_{q_t}[\exp\{\phi_{\beta_t^*}(x)\}]}
\end{aligned}
\tag{7.18}
$$

which gives $\mu = 0$. This completes the proof. □

Lemma 7.2. *Let $\ell''_r(\beta_t^*) \triangleq \frac{1}{n}\sum_{j=1}^n \nabla^2_{\beta_t} \log r_{s_t}(x_{t-1,j})|_{\beta_t=\beta_t^*}$, then $\ell''_r(\beta_t^*) \xrightarrow{P} -I_{\beta_t^*}$, where $I_{\beta_t^*} \triangleq Cov_{q_{t-1}}[\nabla_{\beta_t}\phi_{\beta_t^*}(x)]$.*

Proof. According to Eq. (7.13)

$$
(\nabla_{\beta_t} \log r_{s_t}(x))^2 = \left(\frac{\sum_i^n \nabla_{\beta_t} \exp\{\phi_{\beta_t}(x_{t,i})\}}{\sum_i^n \exp\{\phi_{\beta_t}(x_{t,i})\}}\right)^2 - 2\nabla_{\beta_t}\phi_{\beta_t}(x) \frac{\sum_i^n \nabla_{\beta_t} \exp\{\phi_{\beta_t}(x_{t,i})\}}{\sum_i^n \exp\{\phi_{\beta_t}(x_{t,i})\}} + (\nabla_{\beta_t}\phi_{\beta_t}(x))^2
\tag{7.19}
$$

By the law of large numbers,

$$
\begin{aligned}
\frac{1}{n}\sum_{j=1}^n (\nabla_{\beta_t} \log r_{s_t}(x_{t-1,j}))^2 &\xrightarrow{P} \left(\frac{\mathbb{E}_{q_t}[\nabla_{\beta_t} \exp\{\phi_{\beta_t}(x_{t,i})\}]}{\mathbb{E}_{q_t}[\exp\{\phi_{\beta_t}(x_{t,i})\}]}\right)^2 \\
&- 2\mathbb{E}_{q_{t-1}}[\nabla_{\beta_t}\phi_{\beta_t}(x)] \frac{\mathbb{E}_{q_t}[\nabla_{\beta_t} \exp\{\phi_{\beta_t}(x_{t,i})\}]}{\mathbb{E}_{q_t}[\exp\{\phi_{\beta_t}(x_{t,i})\}]} + \mathbb{E}_{q_{t-1}}[(\nabla_{\beta_t}\phi_{\beta_t}(x))^2]
\end{aligned}
\tag{7.20}
$$

Substituting Eq. (7.18) to the right side of the above equation, we can get:

$$
\begin{aligned}
\frac{1}{n}\sum_{j=1}^n (\nabla_{\beta_t} \log r_{s_t}(x_{t-1,j}))^2|_{\beta_t=\beta_t^*} &\xrightarrow{P} \mathbb{E}_{q_{t-1}}[(\nabla_{\beta_t}\phi_{\beta_t^*}(x))^2] - \mathbb{E}_{q_{t-1}}[\nabla_{\beta_t}\phi_{\beta_t^*}(x)]^2 \\
&= Cov_{q_{t-1}}[\nabla_{\beta_t}\phi_{\beta_t^*}(x)] = I_{\beta_t^*}
\end{aligned}
\tag{7.21}
$$

Because

$$
\nabla^2_{\beta_t} \log r_{s_t}(x) = \frac{\nabla^2_{\beta_t} r_{s_t}(x)}{r_{s_t}(x)} - (\nabla_{\beta_t} \log r_{s_t}(x))^2,
\tag{7.22}
$$

then according to Eq. (7.21)

$$
\ell''_r(\beta_t^*) \xrightarrow{P} \mathbb{E}_{q_{t-1}}\left[\frac{\nabla^2_{\beta_t} r^*_{s_t}(x)}{r^*_{s_t}(x)}\right] - I_{\beta_t^*} = \int \nabla^2_{\beta_t} r^*_{s_t}(x) q_t(x) dx - I_{\beta_t^*}
\tag{7.23}
$$

Under mild assumptions we can interchange the integral and derivative operators:

$$
\int \nabla^2_{\beta_t} r^*_{s_t}(x) q_t(x) dx = \nabla^2_{\beta_t} \int r^*_{s_t}(x) q_t(x) dx = \nabla^2_{\beta_t} \int \frac{q_{t-1}(x)}{q_t(x)} q_t(x) dx = 0
\tag{7.24}
$$

which completes the proof. □

Lemma 7.3. *3 Let $\ell_c(\beta_t) \triangleq \lambda_c \left(\frac{\Psi_t(x_t)}{\Phi_t(x_t)\Psi_{t-1}(x_{t-1})} - 1\right)^2$, and $\ell_c^{'}(\beta_t^*) \triangleq \nabla_{\beta_t}\ell_c(\beta_t)|_{\beta_t=\beta_t^*}$, if we set $\lambda_c = \frac{A}{\sqrt{n}}$, where A is a positive constant, then $\sqrt{n}\ell_c^{'}(\beta_t^*) \xrightarrow{P} 0$.*

Proof.

$$\sqrt{n_{t-1}}\ell_c^{'}(\beta_t) = 2A\left(\frac{\Psi_t(x_t)}{\Phi_t(x_t)\Psi_{t-1}(x_{t-1})} - 1\right) \times \left(\nabla_{\beta_t}\frac{\Psi_t(x_t)}{\Phi_t(x_t)\Psi_{t-1}(x_{t-1})}\right)$$

By the law of large numbers,

$$\frac{\Psi_t(x_t)}{\Phi_t(x_t)\Psi_{t-1}(x_{t-1})}\Big|_{\beta_t=\beta_t^*} \xrightarrow{P} \frac{\mathbb{E}_{q_t}[\exp\{\psi_{\beta_t^*}(x)\}]}{\mathbb{E}_{q_t}[\exp\{\phi_{\beta_t^*}(x)\}]\mathbb{E}_{q_{t-1}}[\exp\{\psi_{\beta_{t-1}}(x)\}]}$$

Define $\tilde{r}_{t-1}(x) \triangleq \frac{\exp\{\psi_{\beta_{t-1}}(x)\}}{\mathbb{E}_{q_{t-1}}[\exp\{\psi_{\beta_{t-1}}(x)\}]}$, by the definition of $r_{s_t}^*(x)$ (Eq. (7.12)):

$$\int \tilde{r}_{t-1}(x)r_{s_t}^*(x)q_t(x)dx = \int \tilde{r}_{t-1}(x)\frac{q_{t-1}(x)}{q_t(x)}q_t(x)dx = \int \tilde{r}_{t-1}(x)q_{t-1}(x)dx = 1$$

Substituting the right side of Eq. (7.12) into the left side of the above equation, we can get:

$$\frac{\mathbb{E}_{q_t}[\exp\{\psi_{\beta_t^*}(x)\}]}{\mathbb{E}_{q_t}[\exp\{\phi_{\beta_t^*}(x)\}]\mathbb{E}_{q_{t-1}}[\exp\{\psi_{\beta_{t-1}}(x)\}]} = 1 \tag{7.25}$$

which completes the proof. □

Theorem 7.4. *Suppose $\lambda_c = \frac{A}{\sqrt{n}}$, where A is a positive constant, assume $\ell_c^{''}(\beta_t^*) = o_p(1)$, $\hat{\beta}_t - \beta_t^* = o_p(1)$, $\mathcal{L}_t^{'''}(\tilde{\beta}_t) = O_p(1)$, where $\tilde{\beta}_t$ is a point between $\hat{\beta}_t$ and β_t^* that satisfies the Taylor series approximation, then $\sqrt{n}(\hat{\beta}_t - \beta_t^*) \rightsquigarrow \mathcal{N}(0, v^2)$, where*

$$v^2 = I_{\beta_t^*}^{-1} + \mathbb{E}_{q_t}[\exp\{\phi_{\beta_t^*}(x)\}]^{-2} \times I_{\beta_t^*}^{-1}Cov_{q_t}[\nabla_{\beta_t}\exp\{\phi_{\beta_t^*}(x)\}]I_{\beta_t^*}^{-1} \tag{7.26}$$

Proof. Applying Taylor expansion to Eq. (7.11) around β_t^*:

$$\begin{aligned} 0 = \mathcal{L}_t^{'}(\hat{\beta}_t) &= \mathcal{L}_t^{'}(\beta_t^*) + (\hat{\beta}_t - \beta_t^*)\mathcal{L}_t^{''}(\beta_t^*) + \frac{1}{2}(\hat{\beta}_t - \beta_t^*)^2\mathcal{L}_t^{'''}(\tilde{\beta}_t), \\ \sqrt{n}(\hat{\beta}_t - \beta_t^*) &= \frac{-\sqrt{n}\mathcal{L}_t^{'}(\beta_t^*)}{\mathcal{L}_t^{''}(\beta_t^*) + \frac{1}{2}(\hat{\beta}_t - \beta_t^*)\mathcal{L}_t^{'''}(\tilde{\beta}_t)} \end{aligned} \tag{7.27}$$

where

$$\mathcal{L}_t(\beta_t) = \ell_r(\beta_t) + \ell_c(\beta_t) \tag{7.28}$$

As we assume $\ell_c^{''}(\beta_t^*) = o_p(1)$, according to Lemma 7.2, we get $\mathcal{L}_t^{''}(\beta_t^*) \xrightarrow{P} -I_{\beta_t^*}$. Combining the results of Lemma 7.1 to 7.3 proves the theorem. □

Theorem 7.5. *Suppose $p(x)$ and $q_t(x)$ ($\forall t$) are from the exponential family, define $r_{s_t}^*(x) = \exp\{\phi_{\beta_t^*}(x)\}$, $\phi_{\beta_t^*}(x) = \beta_t^* T(x) + C$, $T(x)$ is a sufficient statistic of x, C is a constant, then $\sqrt{n}(\hat{\beta}_t - \beta_t^*) \rightsquigarrow \mathcal{N}(0, v_e^2)$, where $T(x)$ is a column vector, $T(x)^2 = T(x)T(x)^T$, $I_{\beta_t^*} = Cov_{q_{t-1}}[T(x)]$:*

$$v_e^2 = I_{\beta_t^*}^{-1} + I_{\beta_t^*}^{-1}(\mathbb{E}_{q_{t-1}}[r_{s_t}^*(x)T(x)^2] - \mathbb{E}_{q_{t-1}}[T(x)]^2)I_{\beta_t^*}^{-1} \tag{7.29}$$

Proof. Because $\phi_t^*(x) \triangleq \psi_{\beta_t^*}(x) - \psi_{\beta_{t-1}^*}(x)$, then $\nabla_{\beta_t}\phi_{\beta_t}^*(x) = T(x)$, we have

$$I_{\beta_t^*} = Cov_{q_{t-1}}[\nabla_{\beta_t}\phi_{\beta_t}^*(x)] = Cov_{q_{t-1}}[T(x)], \tag{7.30}$$

Because $r_{s_t}^*(x) = \exp\{\phi_t^*(x)\}$,

$$\mathbb{E}_{q_t}[\exp\{\phi_{\beta_t}^*(x)\}] = \mathbb{E}_{q_t}[r_{s_t}^*(x)] = 1,$$

In addition,

$$Cov_{q_t}[\nabla_{\beta_t}\exp\{\phi_{\beta_t}^*(x)\}] = Cov_{q_t}[r_{s_t}^*(x)T(x)] = \mathbb{E}_{q_t}[(r_{s_t}^*(x)T(x))^2] - \mathbb{E}_{q_t}[r_{s_t}^*(x)T(x)]^2 \tag{7.31}$$

where

$$\begin{aligned} \mathbb{E}_{q_t}[(r_{s_t}^*(x)T(x))^2] &= \int q_t(x)(r_{s_t}^*(x)T(x))^2 dx = \int q_{t-1}(x)r_{s_t}^*(x)T(x)^2 dx = \mathbb{E}_{q_{t-1}}[r_{s_t}^*(x)T(x)^2], \\ \mathbb{E}_{q_t}[r_{s_t}^*(x)T(x)] &= \int q_t(x)r_{s_t}^*(x)T(x)dx = \int q_{t-1}(x)T(x)dx = \mathbb{E}_{q_{t-1}}[T(x)] \end{aligned} \tag{7.32}$$

Substitute above results into Theorem 7.4, this proves the theorem. □

Theorem 7.5 shows how the covariance matrix v_e^2 depends on the latest density ratio ($r_{s_t}^*$) when the distributions are from the exponential family. Since a smaller variance is better for convergence, we would prefer $r_{s_t}^*(x) = q_{t-1}(x)/q_t(x)$ is not large, which means when $q_{t-1}(x)$ is large $q_t(x)$ should be also large. In this case, $r_{s_t}^*$ is less likely to explode and thus the variance of the estimated parameter would be likely confined. We demonstrate this by experiments with 1-D Gaussian distributions. We fix $q_{t-1}(x) = \mathcal{N}(0,1)$, testing different $q_t(x) = \mathcal{N}(\mu_t, 1)$, where $\mu_t = \delta k, \delta = 0.1, k \in \{0,1,\dots,20\}$. When μ_t is larger, q_t is farther to q_{t-1}. In this case, $T(x) = \{x, x^2\}$, $\beta_t = \{\beta_{t,1}, \beta_{t,2}\}$. We display the diagonal of v_e^2 (variance of $\beta_{t,1}, \beta_{t,2}$) in Fig. 7.1. It is clear that the variance of β_t is getting larger when q_t is farther to q_{t-1}.

When $\beta_t = \beta_t^*, n \to \infty$, we have $r_t(x) = r_{s_t}^*(x)r_{t-1}(x)$, then $\log r_t^*(x) - \log r_t(x) = \log r_{t-1}^*(x) - \log r_{t-1}(x)$, which means the error inherited from $r_{t-1}(x)$ will be the intrinsic error for estimating $r_t^*(x)$ due to the objective of CDRE is iterative. It indicates that smaller difference between each intermediate q_τ and $q_{\tau-1}$ ($\forall 1 < \tau \le t$) leads to a better estimation. We demonstrate in Sec. 7.3.2 that it is often the case in practice even when $\psi_{\beta_t^*}(x)$ is approximated by a non-linear model. Moreover, Fig. 7.1 shows the variance of the estimation may grow rapidly when the difference between the two distributions exceeds a certain value. CKLIEP could prevent such an issue by ensuring the difference between any intermediate pairs of distributions is relatively small, e.g., when the data distribution changes fast we could set smaller time intervals for collecting samples to give smaller changes at each step. We will demonstrate this in Sec. 7.3.2 as well.

7.2.6 Multiple original distributions in CDRE

Now we consider tracing multiple original distributions in CDRE, in which case a new pair of original and dynamic distributions will be added into the training process of the estimator at

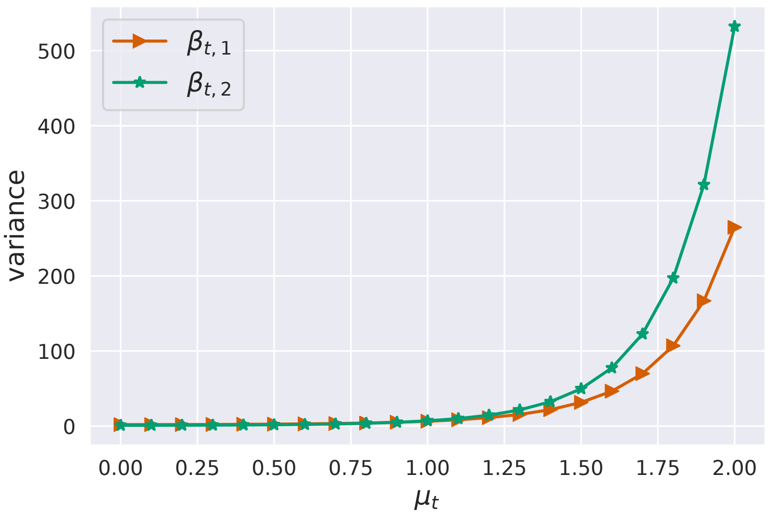


Figure 7.1: Demonstration of the variance of estimated parameters in Theorem 7.5 by 1-D Gaussian distributions: fix $q_{t-1}(x) = \mathcal{N}(0,1)$ and vary $q_t(x) = \mathcal{N}(\mu_t, 1)$ by setting $\mu_t = \delta k, \delta = 0.1, k \in \{0, 1, \ldots, 20\}$. When μ_t is larger the two distributions are less similar and the variance is larger, which aligns with Theorem 7.5.

some time point. As introduced in sec. 7.2.1 we refer to an original distribution as $p_\tau(x)$, where τ is the time index of starting tracing the original distribution. And samples of $p_\tau(x)$ are not available when $t > \tau$. Similarly, $q_{\tau,t}(x)$ denotes the dynamic distribution that corresponding to $p_\tau(x)$ at time t:

$$r^*_{\tau,t}(x) = \frac{p_\tau(x)}{q_{\tau,t}(x)} = \frac{q_{\tau,t-1}(x)}{q_{\tau,t}(x)} \frac{p_\tau(x)}{q_{\tau,t-1}(x)} = r^*_{s_{\tau,t}}(x) r^*_{\tau,t-1}(x),$$

Where $r^*_{s_{\tau,t}}(x) = q_{\tau,t-1}(x)/q_{\tau,t}(x)$. In this case, we optimize the estimator at time t by an averaged objective:

$$\max_{\beta_t} \bar{\mathcal{L}}_t(\beta_t) = \max_{\beta_t} \frac{1}{|\mathbb{T}|} \sum_{\tau \in \mathbb{T}} \mathcal{L}_t(\beta_t; \tau) \tag{7.33}$$

where $\mathbb{T}$ is the set of time indices of adding original distributions, $|\mathbb{T}|$ is the size of $\mathbb{T}$. $\mathcal{L}_t(\beta_t; \tau)$ is as the same as the loss function of a single original distribution (Eq. (7.10)) for a given τ. Further, $r_{s_{\tau,t}}(x)$ is also defined by the same form of Eq. (7.9), the difference is that $\psi_{\beta_t}(x)$ becomes $\psi_{\beta_t}(x; \tau)$:

$$r_{s_{\tau,t}} = \frac{\exp\{\phi_{\beta_t}(x;\tau)\}}{\frac{1}{N}\sum_{i=1}^N \exp\{\phi_{\beta_t}(x_i;\tau)\}}, \quad \text{where} \quad \phi_{\beta_t}(x;\tau) \triangleq \psi_{\beta_t}(x;\tau) - \psi_{\beta_{t-1}}(x;\tau). \tag{7.34}$$

In our implementation, we concatenate the time index τ to each data sample as the input of the ratio estimator. In addition, we set the output of $\psi_{\beta_t}(\cdot)$ as a $|\mathbb{T}|$-dimensional vector $\{o_1, \ldots, o_i, \ldots, o_{|\mathbb{T}|}\}$ where o_i corresponds to the output of $\psi_{\beta_t}(x; \tau = \mathbb{T}_i)$. Thus, we can avoid learning separate ratio estimators for multiple original distributions. Note that with CDRE we have the flexibility to extend the model architecture since the latest estimator function ϕ_{β_t} only needs the output of the previous estimator function $\psi_{\beta_{t-1}}$. This can be beneficial when the model capacity becomes a bottleneck of the performance.

7.2.7 Dimensionality reduction in applications of CDRE

The main concern of estimating density ratios stems from high-dimensional data. Several methods for dimensionality reduction in DRE have been introduced in (Sugiyama et al., 2012). A fundamental assumption of these methods is that the difference between two distributions can be confined to a subspace, which means $p(z)/q(z) = p(x)/q(x)$ where z is a lower-dimensional representation of x. This aims for exact density ratio estimation in a subspace but incurs a high computational cost. In most applications, a model (e.g. classifiers) is often trained upon a representation extractor, indicating a surrogate feature space of high-dimensional data can be used for the density ratio estimation in practice. For instance, most measurements of generative models estimate the difference between two distributions in a surrogate feature space, e.g., the inception feature defined for the Inception Score (Salimans et al., 2016) is extracted by a neural network, and this method is also applied in many other measurements of generative models, e.g. Fréchet Inception Distance (Heusel et al., 2017) and Kernel Inception Distance (Bińkowski et al., 2018).

In prior work, a pre-trained classifier is often used to generate surrogate features of high-dimensional image data (such as inception features (Salimans et al., 2016)). However, it may be difficult to train such a classifier in online applications because: a) a homogeneous dataset for all unseen data or tasks may not be available in advance; b) labeled data may not be available in generative tasks. In order to cope with such circumstances, we introduce Continual Variational Auto Encoder (CVAE) in a pipeline with CDRE. The loss function of CVAE is defined as follows:

$$\begin{aligned}
&\mathscr{L}_{CVAE}(\theta_t, \vartheta_t) = NLL + \mathscr{D}_{KL}(q_t(z) || q_{t-1}(z)), \\
&NLL = -\frac{1}{t}\Big[\sum_{\tau=1}^{t-1} \mathbb{E}_{q_{\tau,t-1}(x)}[\mathbb{E}_{q_t(z)}[\log p(x|z;\vartheta_t)]] + \mathbb{E}_{p_t(x)}[\mathbb{E}_{q_t(z)}[\log p(x|z;\vartheta_t)]]\Big]
\end{aligned} \tag{7.35}$$

where $q_t(z) = \mathcal{N}(\mu_{\theta_t}(x), \sigma_{\theta_t}(x))$, θ_t and ϑ_t denote parameters of the encoder and decoder of CVAE, respectively. NLL is the negative log likelihood term as the same as in vanilla VAE (Kingma & Welling, 2013), the training data set includes samples of all dynamic distributions at the previous step ($q_{\tau,t-1}(x), \forall \tau < t$) and the samples of the current original distribution ($p_t(x)$). The KL-divergence term in the loss function serves a regularization term: the current encoder is expected to give similar z for a similar x comparing with the previous encoder. This term forces the consistency between inputs of the previous and current ratio estimators (i.e. inputs of $\psi_{\beta_{t-1}}$ and ψ_{β_t} in Eq. (7.10)). It is different from VAEs proposed with VCL in Nguyen et al. (2018) because the posterior and prior are defined w.r.t. parameters in Nguyen et al. (2018) whereas they are w.r.t. latent variables of the encoder in CVAE. CVAE can generate effective features without requiring labels or pre-training, nevertheless, other commonly used methods (e.g. pre-trained classifiers) are also applicable to CDRE.

There are existing methods for detecting changing points online by direct DRE (Kawahara & Sugiyama, 2009; Liu et al., 2013; Bouchikhi et al., 2018), which estimate density ratios between distributions of two consecutive time intervals. In contrast, CDRE estimates density ratios between distributions of the initial and latest time intervals without storing historical samples.

A concurrent work (Rhodes et al., 2020) has developed Telescoping Density Ratio Estimation (TRE) by a consecutive decomposition that is similar with our method (Eq. (7.4)) but with the following main differences:

1) TRE simultaneously optimizes m ratio estimators as below:

$$\mathscr{L}_{TRE}(r) = \frac{1}{m}\sum_{k=1}^{m}\mathscr{L}(r_k), \quad r = \frac{p_0}{p_m} = \frac{p_0}{p_1}\frac{p_1}{p_2}\cdots\frac{p_{m-1}}{p_m}, \quad r_k = \frac{p_{k-1}}{p_k}, \quad k \in \{1,\dots,m\}$$

where m is the number of decomposed ratios of the target ratio (p_0/p_m). The intermediate distributions ($\{p_1,\dots,p_{m-1}\}$) are designed by gradually changing from p_0 to p_m. $\mathscr{L}(r_k)$ is the loss function of estimating the k-th ratio $r_k = p_k/p_{k+1}$. Estimating the intermediate ratios would be easier than directly estimating p_0/p_m because the two adjacent distributions are more similar to each other. TRE attempts to estimate the m ratios at the same time. In comparison, CDRE estimates p_0/p_m in an iterative fashion (Eq. (7.4)) as p_m is changing over time and at each time step ($\forall m \geq 1$) CDRE only optimizes the latest ratio estimator $r_m = p_{m-1}/p_m$;

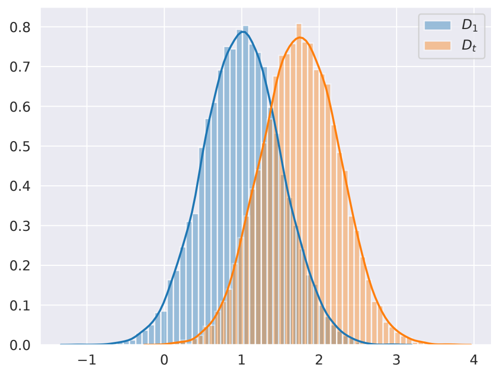
2) According to the optimization objective, TRE requires samples of all intermediate distributions as well as the original distribution, in contrast CDRE only requires samples of the two latest distributions.

Note that CDRE can also be applied to applications of TRE by deploying the sampling methods suggested in Rhodes et al. (2020) to generate samples of the intermediate distributions. However, TRE **cannot** be applied to online applications of CDRE due to the off-line nature of its algorithm.

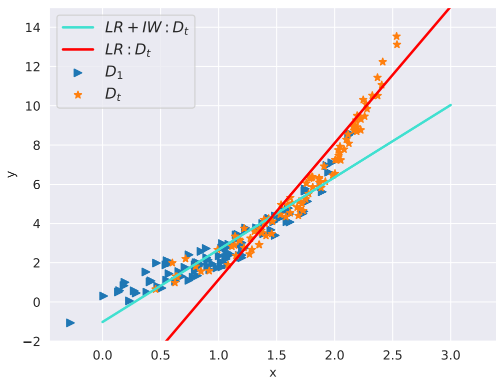
Another related work (Stojanov et al., 2019) is in the scope of dimensionality reduction for estimating density ratios in covariate shift. It aims to find a low dimensional representation space according to the relevance of features X to the predicted target Y, which supposes the representation mapping function $h(X): R^D \to R^d$ can satisfy $X \perp\!\!\!\perp Y | h(X)$ and $D \gg d$. This method relies on the target variable Y in supervised learning whereas the proposed CVAE does not need Y. Furthermore, we will show that CDRE can be applied to backward covariate shift in the next section and this method can be an option for dimensionality reduction in such a case as well.

7.3 Online Applications

In this section we illustrate several online applications of CDRE, including: 1) backwards covariate shift; 2) tracing distribution shifts by KL-divergence; and 3) monitoring real stock data for a regression model. We demonstrate that CDRE can provide reliable estimations by synthetic data as we can compare the results with true ratios. To show the effectiveness of CDRE in real-world applications, we make an example by real stock data as well. In all of our experiments, $\psi(\cdot)$ is a



(a) the data distribution at $\tau = 1$ (D_1) and $\tau = t$ (D_t)



(b) linear regression trained by D_t for predicting test samples from D_1

Figure 7.2: Demo experiment of backward covariate shift. (a) shows the data distribution of training set at $\tau = 1$ and $\tau = t$. (b) displays the regression lines learned by the model at $\tau = t$, the cyan and red lines are fitted by D_t with and without importance weights, respectively.

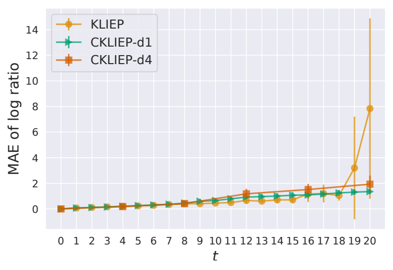
neural network with two and three dense layers for a single and multiple original distributions respectively, each layer having 256 hidden units and ReLU activations. λ_c (the hyperparameter used for controlling the strength of the constraint in the objective in Eq. (7.10)) is set to 10 for experiments with a single original distribution, and set to $100k$ for experiments with the multiple original distributions, where k is the number of joined original distributions at each time step. We apply CKLIEP to instantiate CDRE in all of our experiments.

7.3.1 Backwards covariate shift

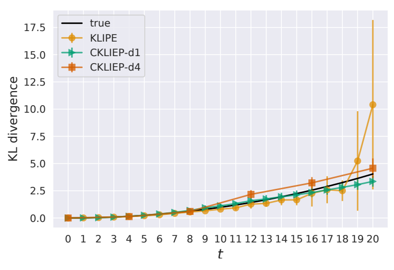
We demonstrate that CDRE can be applied in *backward covariate shift*, where the training set shifts and the test set is from a previous distribution. It just swaps the situation of training and test set in the scenario of common covariate shift (Sugiyama et al., 2008; Shimodaira, 2000). We assume a linear regression model defined as $\hat{y} = wx + b + \epsilon_0$, where the noise $\epsilon_0 \sim \mathcal{N}(0, 0.01)$. At time τ, the training data $x \in D_\tau$ and D_τ shifts away from D_1 gradually, where $\tau \in \{1, 2, \dots, t\}$ and $t = 10$ is the latest time index. Fig. 7.2a displays the data distribution at time $\tau = 1$ and $\tau = t$ in which we can see there exists notable difference between the two distributions. When the model is trained by D_t, it will not be able to accurately predict on test samples from D_1 unless we adjust the loss function by importance weights (i.e. density ratios) as in handling covariate shift:

$$\mathcal{L} = \mathbb{E}_{x \sim q_t(x)} \left[\frac{q_1(x)}{q_t(x)} (y - \hat{y})^2 \right]$$

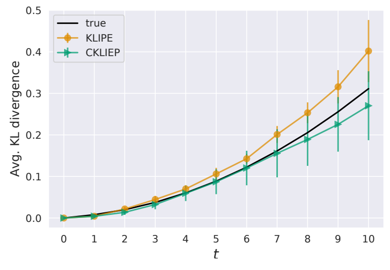
Fig. 7.2b shows the regression lines learned by the model at $\tau = t$ with and without the importance weights, where the weights $q_1(x)/q_t(x)$ are estimated by CKLIEP. We can see that the line learned with importance weights fits D_1 more accurately than the one without the weights. This enables



(a) Comparing MAE of log ratios estimated by CKLIEP and KLIEP in the scenario of a single original distribution



(b) Comparing KL-divergence estimated by CKLIEP and KLIEP in the scenario of a single original distribution



(c) Comparing average KL-divergence estimated by CKLIEP and KLIEP in the scenario of multiple original distributions

Figure 7.3: Comparing the performance of CKLIEP and KLIEP by synthetic data in the scenarios of the single and multiple original distributions. (a) & (b) compare the Mean Absolute Error (MAE) of log ratios and estimated KL-divergences for a single original distribution, (c) compares the average KL-divergences for multiple original distributions. The true values of KL-divergences are computed by true ratios. The error bar is the standard deviation of 10 runs.

the model to make reasonable predictions on test samples from D_1 when the training set of D_1 is not available.

7.3.2 Tracing distribution shifts via KL-divergence

By estimating density ratios, we can approximate f-divergences between two distributions:

$$\mathscr{D}_f(p||q) = \mathbb{E}_q\left[f\left(\frac{p(x)}{q(x)}\right)\right] \approx \frac{1}{N}\sum_{i=1}^{N} f(r(x_i)) \tag{7.36}$$

where $x_i \sim q(x)$, $f(\cdot)$ is a convex function and $f(1) = 0$. Thus, we can trace the distribution shifts by using CDRE to approximate the f-divergences between $p_\tau(x)$ and $q_{\tau,t}(x)$ in the online setting described in Sec. 7.2. Estimating f-divergences using density ratios has been well studied (Kanamori et al., 2011; Nguyen et al., 2010). In our implementations, we choose the KL-divergence (i.e. setting $f(r) = -\log(r)$) to instantiate f-divergences. We compare the performance of CKLIEP with KLIEP and true values using synthetic Gaussian data, where KLIEP has access to samples of all original distributions at all time. The sample size of each distribution in these experiments is 50,000.

We first simulate the scenario of a single original distribution which is a 64-D Gaussian distribution $p(x) = \mathcal{N}(\mu_0, \sigma_0^2 I)$, where $\mu_0 = 0, \sigma_0 = 1$. At each time step, we shift the distribution by a constant change on its mean and variance: $q_t(x) = \mathcal{N}(\mu_t, \sigma_t^2 I), \mu_t = \mu_0 + \Delta\mu * k, \sigma_t = \sigma_0 - \Delta\sigma * k, \Delta\mu = \Delta\sigma = 0.02$, k is the number of time steps within one estimation interval. We set the total number of time steps to 20. We estimate $p(x)/q_t(x)$ by applying CKLIEP with two different time intervals: (1). CKLIEP-d1 is to estimate $p(x)/q_t(x)$ at each time step, i.e. $k = 1$; (2). CKLIEP-d4 is to estimate $p(x)/q_t(x)$ at every four time steps, i.e. $k = 4$. We compare the Mean Absolute Error (MAE) of log ratios ($\mathscr{L}_{\text{MAE}} = \frac{1}{N}\sum_{n=1}^{N} |\log r^*(x_n) - \log \hat{r}(x_n)|$) estimated by CKLIEP

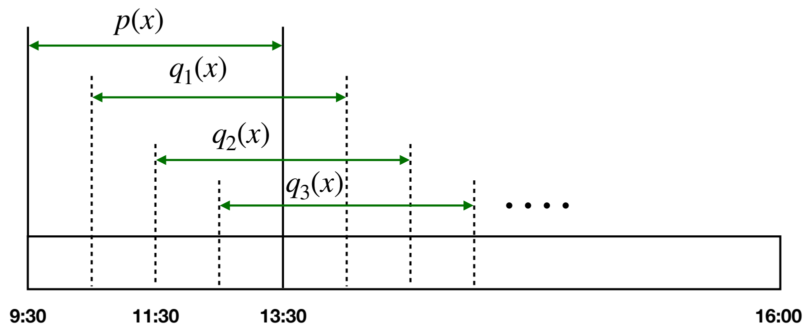


Figure 7.4: Sliding time windows of sampling the stock transaction data. We estimate $p(x)/q_t(x)$ every half an hour and sampling the transaction data from a two-hour window.

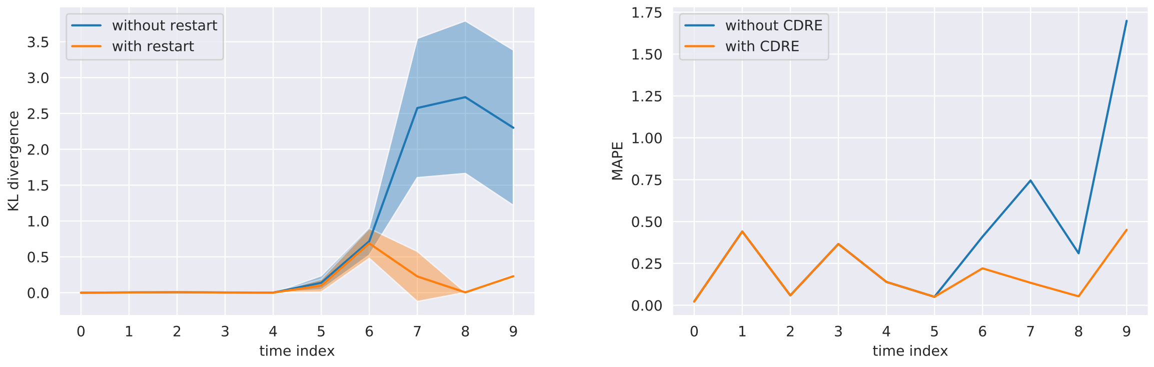
and KLIEP in Fig. 7.3a. We also compare the estimated KL-divergence with the true value in Fig. 7.3b. According to Theorem 7.5, the difference between $q_{t-1}(x)$ and $q_t(x)$ plays an important role in the estimation convergence, which explains why CKLIEP-d4 gets worse performance than CKLIEP-d1 since $q_{t-1}(x)$ and $q_t(x)$ are less similar in the case of CKLIEP-d4. KLIEP can be viewed as a special case of CKLIEP when $q_{t-1}(x) = p(x)$, so the difference between $p(x)$ and $q_t(x)$ is critical to its convergence as well. We can see that in Figs. 7.3a and 7.3b KLIEP has become much worse at the last two steps due to the two distributions are too far away from each other and thus causes serious difficulties in its convergence with a fixed sample size.

We also simulate the scenario of multiple original distributions by 64-D Gaussian data: $p_\tau(x) = \mathcal{N}(\mu_\tau, \sigma_\tau^2 I)$, $\tau \in \{1,2,\ldots,t\}, \mu_\tau = 2\tau, \sigma_\tau = 1$, in which cases we add a new original distribution ($p_t(x)$) at each time step. We shift each joined original distribution ($p_\tau(x), \forall \tau < t$) by a constant change as the same as the single pair scenario and set $k = 1, \Delta\mu = \Delta\sigma = 0.01$. In Fig. 7.3c, we compare the averaged KL-divergences ($\bar{\mathcal{D}} = \frac{1}{t}\sum_{\tau=1}^{t} \mathcal{D}_{KL}(q_{\tau,t} \| p_\tau)$) estimated by CKLIEP and KLIEP with the true value. CKLIEP outperforms KLIEP when the dynamic distributions getting farther away from the original distributions, which aligns with the scenario of a single original distribution.

7.3.3 Monitoring real stock data for a regression model

We demonstrate the effectiveness of CDRE in practice by real stock data. The dataset consists of one-day transactions of the Microsoft stock [1]. It includes the transaction time, price, volume and direction (initiated by selling or buying). We augment each transaction by concatenating it with its five previous transactions (excluding timestamps) and then treat it as an i.i.d. data sample. We draw samples from a two-hour time window of the data and slide the window with a 30-minute step size. We first train a Gaussian process regression model to predict the price of a transaction by samples from the initial two-hour time window, then we apply CKLIEP to monitor

[1]The data can be downloaded from `https://lobsterdata.com/info/DataSamples.php`, which is sample data for free. We only used the one level data.



(a) Monitored KL divergence (b) MAPE of the regression model

Figure 7.5: Monitoring stock data shift by CDRE for updating a regression model of transaction price prediction. (a) shows KL divergence between the training set of the regression model and samples from the latest two hours monitored by CDRE. The blue line is without restart during the progress of CDRE (the regression model has not been updated), the orange line is with restart (the regression model has been retrained by latest samples when the KL divergence larger than a threshold $\delta = 0.1$), the shaded area is plotted by the standard deviation of 5 runs. (b) shows the MAPE of the regression model without and with update by using CDRE.

the data shift at each time step (Fig. 7.4). When the KL divergence between the training set ($p(x)$) and the data from the latest two-hour window ($q_t(x)$) is larger than a threshold δ ($\delta = 0.5$ in our experiment), we retrain the regression model by samples from the latest two hours. We also restart the progress of CKLIEP as replacing the original distribution $p(x)$ by $q_t(x)$ when we retrain the regression model at time t.

We use the last 500 transactions of each two-hour window as the test set of each time step, which are excluded from the training set. And the training sample size is 6000 for each distribution. We evaluate the performance of the regression model by Mean Absolute Percentage Error ($\mathscr{L}_{MAPE} = \frac{1}{N}\sum_{n=1}^{N} 100 \times |y_n - \hat{y}_n|/y_n$) and provide the experiment results in Fig. 7.5. Fig. 7.5b shows that using CDRE to monitor the training data shift can enable retraining the regression model when it is necessary. It effectively prevents large degradation of the performance. The restart strategy of CDRE also helps with reducing the estimation variance in latter steps as shown in Fig. 7.5a because the new original distribution is much closer to the latest dynamic distribution after the restart.

7.4 Evaluating generative models in continual learning

In this section we demonstrate evaluating generative models in continual learning by CDRE, which can be viewed as an application of tracing multiple original distributions by estimating the f-divergences between them and their dynamic distributions. We first discuss the difference between f-divergences and other measures for generative models in static learning in sec. 7.4.1, and further show the effectiveness of f-divergences in continual learning in sec. 7.4.2.

In Fig. 3.5 we demonstrated a simplified scenario of generative models in continual learning: the model needs to learn multiple generative tasks sequentially and perform on all seen tasks. Specifically, the training dataset of task t consists of real samples of task t and samples of task $1 \cdots t-1$ generated by the previous model. The task index can be treated as the time index, the data distribution of the task τ can be viewed as a new original distribution $p_\tau(x)$ that added at time τ, and its corresponding dynamic distribution $q_{\tau,t}(x)$ is the sample distribution generated by the model after trained on task t. The goal of generative models in continual learning is to make $q_{\tau,t}(x)$ as close to $p_\tau(x)$ as possible. In the sense of measuring the difference between $q_{\tau,t}(x)$ and $p_\tau(x)$, we can evaluate a generative model in continual learning by estimating the averaged f-divergences over all learned tasks:

$$\bar{\mathscr{D}}_t = \frac{1}{t}\sum_{\tau=1}^{t} \mathbb{E}_{q_{\tau,t}}[f(r_{\tau,t}(x))] \approx \frac{1}{t}\sum_{\tau=1}^{t}\sum_{n=1}^{N_\tau} f(r_{\tau,t}(x_n))$$

7.4.1 Related measures for generative models in static learning

Since the existing measures for generative models applied in continual learning are usual measures applied in static learning, we discuss the difference between several existing measures and f-divergences in static learning first.

We here briefly review the definition of Fréchet Inception Distance (FID) (Heusel et al., 2017), Kernel Inception Distance (KID) (Bińkowski et al., 2018), and Precision and Recall for Distributions (PRD) (Sajjadi et al., 2018), which are popular measures for generative models. FID fits a Gaussian distribution to the representations computed by original data or generated data, and then computes the Fréchet distance (also known as the Wasserstein-2 distance) between the two Gaussian distributions:

$$FID = ||\mu_r - \mu_g||^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}). \tag{7.37}$$

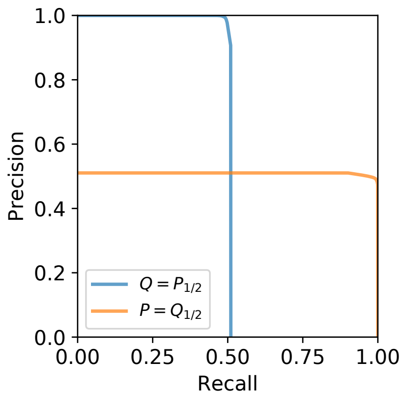
KID is the squared Maximum Mean Discrepancy (MMD) between representations. Bińkowski et al. (2018) suggested a 3rd-degree polynomial kernel $k(x,y) = (\frac{1}{d}x^T y + 1)^3$ which can compare three moments of two distributions:

$$KID = \frac{1}{m(m-1)}\sum_{i\neq j}^{m} k(x_i,x_j) + \frac{1}{n(n-1)}\sum_{i\neq j}^{n} k(\hat{x}_i,\hat{x}_j) - \frac{2}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} k(x_i,\hat{x}_j). \tag{7.38}$$

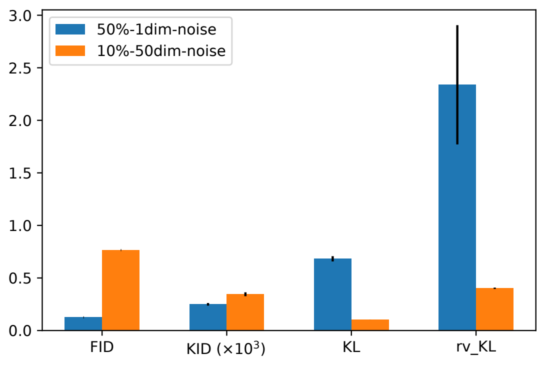
PRD decomposes two distributions P and Q as follows, where μ, ν_P, ν_Q are three distributions:

$$P = \beta\mu + (1-\beta)\nu_P, \quad Q = \alpha\mu + (1-\alpha)\nu_Q, \quad \alpha,\beta \in (0,1]. \tag{7.39}$$

The component ν_P denotes the part of P that is "missed" by Q, ν_Q denotes the noise part of Q. The set of attainable pairs of precision and recall of a distribution Q w.r.t. a distribution P consists of all (α,β) satisfying Eq. (7.39) and the tuple (0, 0).



(a) PRD of the first experiment in Tab. 7.1



(b) The experiments with different types of noise injected into MNIST (error bars from 5 runs).

Figure 7.6: Visualized results of experiments on MNIST dataset.

In the absence of prior work on evaluating generative models by f-divergences, we provide experimental results for demonstrating differences between f-divergences, FID, KID, and PRD. Through these experiments, we show that f-divergences can be alternative measures of generative models and one may obtain richer criteria by them.

We first present several experiments on MNIST dataset and demonstrate the experiment results through two popular members of f-divergences: KL-divergence and reverse KL-divergence. In the first experiment we have two cases: (i) the compared distribution P contains half of the classes of MNIST, and the evaluated distribution Q includes all classes of MNIST; (ii) the reverse of (i). We obtain density ratios with KLIEP and then estimate f-divergences by the ratios. The results are shown in Tab. 7.1, with PRD curves displayed in Fig. 7.6a. Since the objective of KLIEP is not symmetric, the estimated KL divergences are not symmetric when switching the two sets of samples. As we can see, $D_{KL}(P||Q)$ prefers Q with larger recall (case (i)) and $D_{KL}(Q||P)$ prefers the inverse case. Neither FID nor KID are able to discriminate between these two scenarios.

Table 7.1: Results of the first experiment on MNIST. $P = Q_{1/2}$ implies case (i) P contains half of the classes of MNIST, and the Q includes all classes of MNIST, $Q = P_{1/2}$ implies case (ii) Q contains half of the classes of MNIST, and the P includes all classes of MNIST. Standard deviations are from 5 runs.

	FID	KID	$D_{KL}(P\|\|Q)$	$D_{KL}(Q\|\|P)$
$P = Q_{1/2}$	50.39 ± 0.00	2.04 ± 0.01	0.67 ± 0.00	3.78 ± 1.22
$Q = P_{1/2}$	50.39 ± 0.00	2.03 ± 0.02	2.49 ± 0.30	2.38 ± 1.78

In the second experiment, we further show that f-divergences may provide different opinions with FID and KID in certain circumstances because FID and KID are based on Integral Probability Metrics (IPM) (Sriperumbudur et al., 2012) which focus on parts with the most probability mass whereas f-divergences are based on density ratios which may give more attention on parts with less probability mass (due to the ratio of two small values can be very large). To show this, we simulate two sets of noisy samples by injecting two different types of noise into MNIST data and evaluate them as the model samples on the pixel feature space (which is 784 dimensions). Regarding the first type of noise, we randomly choose 50% samples and 1 dimension to be corrupted (set the pixel value to 0.5); for the second one, we randomly choose 10% samples and 50 dimensions to be corrupted. The results are shown in Fig. 7.6b, in which KL-divergence and reverse KL-divergence disagree with FID and KID regarding which set of samples is better than the other. The results illustrate measures based on density ratios could be more sensitive to subtle differences than measures based on IPM.

In order to show that f-divergences can work with high dimensional data as the same as FID and KID, we also conducted an experiment with samples of StyleGAN[2] trained by FFHQ dataset (Karras et al., 2019). We estimate f-divergences on the inception feature space with 2048 dimensions as well. The sample size is 50,000 and we compare the model samples with real samples. We see that the f-divergences give reasonable results with small variance (Tab. 7.2), indicating it is capable of evaluating with high dimensional data.

Table 7.2: Evaluating StyleGAN on FFHQ dataset using f-divergences

	KL	rv_KL	JS	Hellinger
StyleGAN	2.47 ± 0.02	3.29 ± 0.18	0.86 ± 0.01	1.04 ± 0.02
Real samples	$0.02 \pm 9.1e-4$	$0.02 \pm 9.3e-4$	$0.01 \pm 4.5e-4$	$0.01 \pm 4.6e-4$

7.4.2 Experimental results in continual learning

Now we provide experimental results of evaluating generative models in continual learning. In our experiments, the evaluated GANs include WGAN (Arjovsky et al., 2017), WGAN-GP (Gulrajani et al., 2017), and two members of f-GANs (Nowozin et al., 2016): f-GAN-rvKL and f-GAN-JS, which instantiate the f-GAN by reverse KL and Jesen-Shannon divergences respectively. All GANs are tested as conditional GANs (Mirza & Osindero, 2014) using task indices as conditioners, and one task includes a single class of the data. We trained the GANs on Fashion-MNIST (Xiao et al., 2017). The sample size used in CDRE is 6,000 for each distribution. We evaluate these GANs in continual learning by a few members of f-divergences, and compare the results with common measures FID, KID as used in static learning.

We deployed two feature generators in the experiments: 1) A classifier pre-trained on real samples of all classes, which is a Convolutional Neural Network (CNN) and the extracted features are the activations of the last hidden layer (similar with inception feature); 2) A CVAE trained

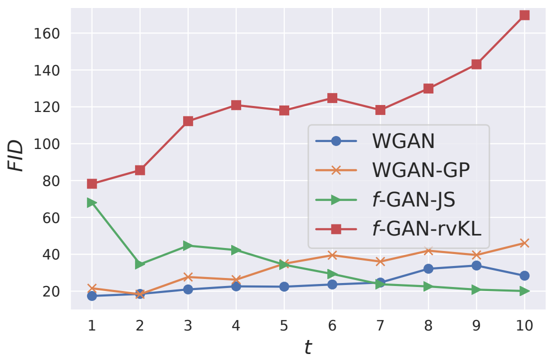
[2]`https://github.com/NVlabs/stylegan`

along with the procedure of continual learning, and the features are the output of the encoder. The number of features is 64 for both classifier and CVAE. We deployed the pre-trained classifier as the feature generator for FID,KID, and deployed the CVAE as the feature generator for CKLIEP in all experiments. Fig. 7.7 compares the evaluations on Fashion-MNIST for the GANs by FID, KID, and four members of f-divergences (which are estimated by CKLIEP): KL, reverse KL, Jensen-Shannon, Hellinger. All these measures are the lower the better. We also display randomly chosen samples generated by those GANs in Fig. 7.9 for a better understanding of the evaluations.

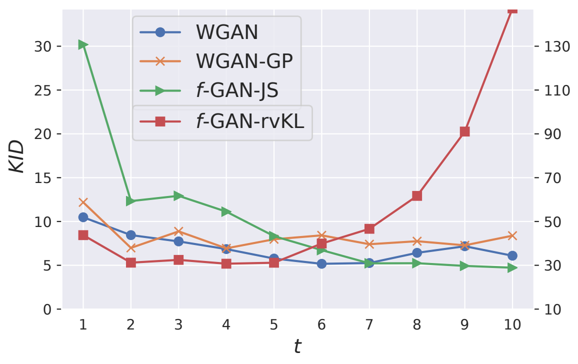
In general, all measurements give similar evaluations. For example, f-GAN-rvKL has the worst performance on all measures during the whole task sequence, whereas WGAN and WGAN-GP have similar performance on all measures. One main disagreement is that f-GAN-JS shows a decreasing trend on FID and KID but shows a slightly increasing trend on members of f-divergences. According to displayed samples of f-GAN-JS (Fig. 7.9c), there is no notable improvement observed while f-GAN-JS learning more tasks, the evaluations given by FID and KID seem doubtful in this case. In principle, f-divergences may exhibit different preferences because the mass with lower density could contribute more to them than to the other measures as we demonstrated in Fig. 7.6b. To verify if this is the reason of the disagreement between f-divergences and FID, KID, we split the generated samples of an evaluated distribution into two parts and each part contains approximately 50% of the total mass (which is approximated by the histogram). We set one part including bins with higher probability densities and the other part with lower ones, then we plot PRD curves of both parts in Fig. 7.8 for all GANs. We have done such analysis on the last two tasks since the disagreement among the measures is obvious at this stage. As shown in Figs. 7.8a and 7.8b, the PRD of the part with lower-density of the WGAN is better than f-GAN-JS in terms of Recall at both tasks, which could be the reason why f-divergences do not prefer f-GAN-JS over WGAN like FID and KID as the part with lower density could contribute more to them. Overall, these experimental results demonstrate that f-divergences estimated by CKLIEP can provide meaningful evaluations for generative models in continual learning, which could be an alternative when other measures are not applicable or incapable of telling a difference.

7.5 Summary

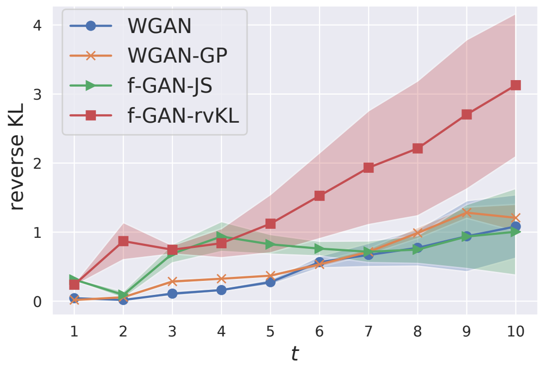
In this chapter we proposed a novel method CDRE for estimating density ratios in an iterative fashion, which does not require storing historical samples and can obtain better estimation than standard DRE. We demonstrate the efficacy of our method in a range of online applications and as a new measurement of generative models in continual learning. The experiments showed that it is able to accurately approximate density ratios by testing with synthetic Gaussian distributions. We also showed that it can be an effective method in practice as well by testing with



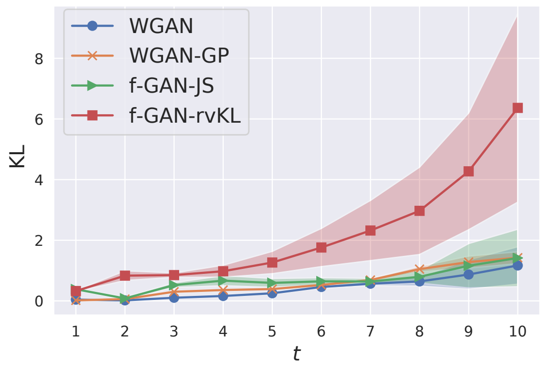
(a) FID



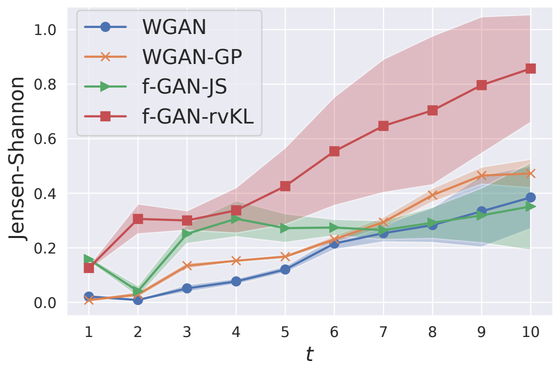
(b) KID



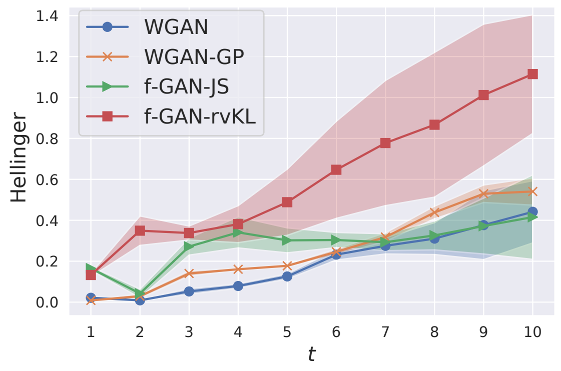
(c) CKLIEP (reverse KL)



(d) CKLIEP (KL)



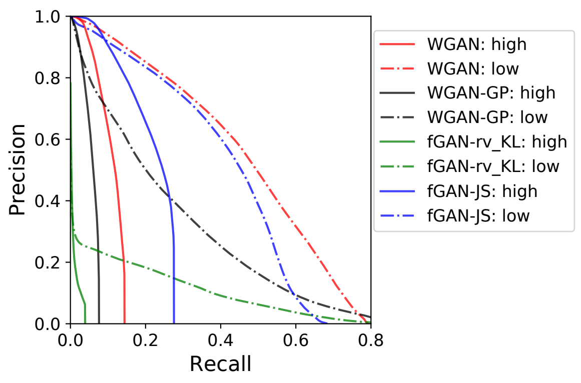
(e) CKLIEP (JS)



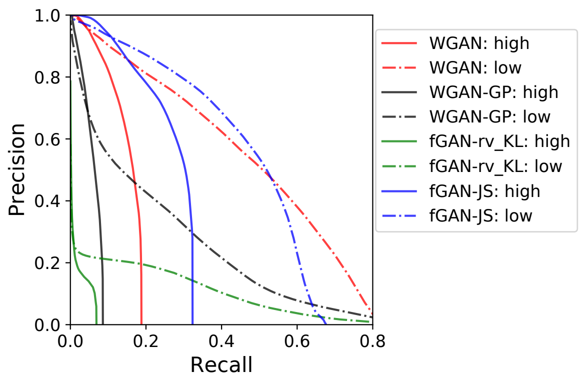
(f) CKLIEP (Hellinger)

Figure 7.7: Evaluating GANs in continual learning on Fashion-MNIST. The shaded area are plotted by standard deviation of 10 runs. The x-axis is task index and y-axis is the specified measurement as in each sub-caption. The y-axis in the right side of Fig. 7.7b is the y-axis of the red line (f-GAN-rvKL), which is in a much larger scale than others. All the measures are the lower the better.

real stock transaction data. Besides the online applications, we demonstrated that CDRE can provide additional insights for model selection in continual learning. In addition, we proposed a simple approach CVAE for feature generation in continual learning when a pre-trained classifier is not available. Our experiments showed that CDRE combined with CVAE can work well on high-dimensional data with low fidelity, which is a rather difficult scenario for density ratio



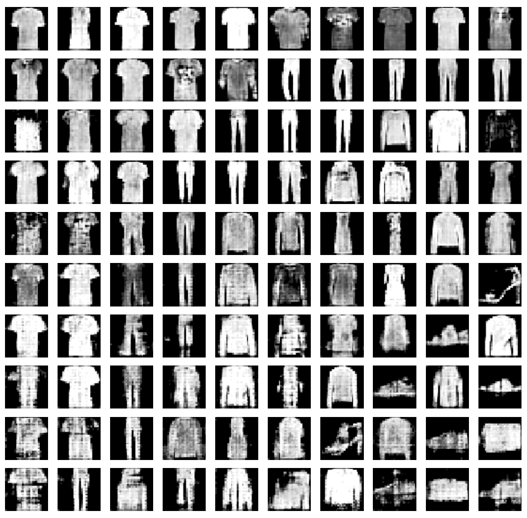
(a) PRD curves of two parts of samples at task 9

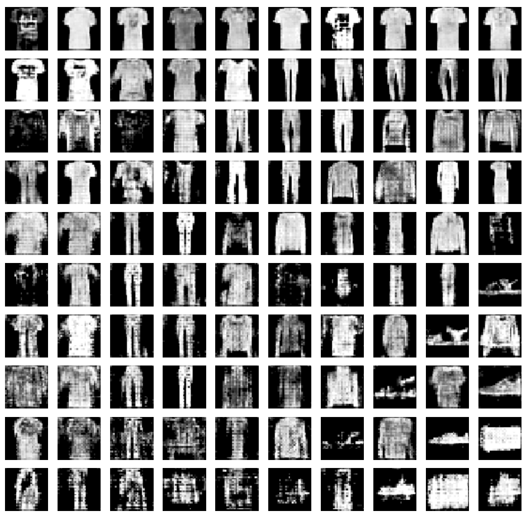


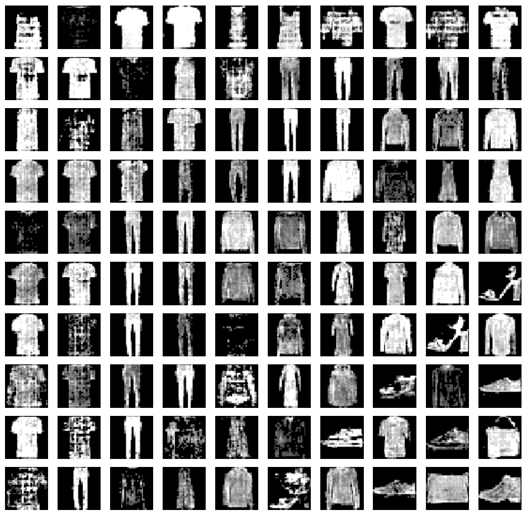
(b) PRD curves of two parts of samples at task 10

Figure 7.8: PRD curves evaluated at the last two tasks for all GANs, which is plotted by two parts of generated samples and each part contains 50% density mass, one part includes samples with higher probability densities and the other part includes samples with lower probability densities. The PRD of the part with lower-density of samples generated by WGAN (red dotted lines) is better than f-GAN-JS (blue dotted lines) at both tasks, which could be the reason why f-divergences do not prefer f-GAN-JS over WGAN like FID and KID.

estimation.


(a) WGAN


(b) WGAN-GP


(c) f-GAN-JS

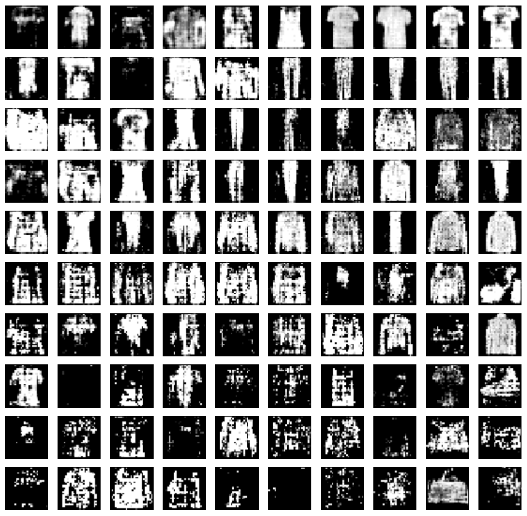

(d) f-GAN-rvKL

Figure 7.9: Fashion-MNIST samples generated by several GANs in continual learning. In each sub-figure, each row displays images generated by the model at each task, the order is from the top to bottom (task 1 to 10). The generated samples are from all learned classes at task i. The displayed samples are uniformly randomly chosen from generated samples of each class.

Chapter 8

Conclusions and Future Work

8.1 Summary and Conclusions

In this thesis we introduced three main categories of solutions for continual learning, including regularization-based, architecture-based, and replay-based solutions. Among all these categories, replay-based methods are the most practical ones as they can work without task identifiers during training and testing time. In other words, they are feasible to single-headed models under online training without the awareness of task boundaries. In contrast, regularization-based and architecture-based methods are not applicable in such scenarios. The regularization-based methods can work without task identifiers during testing time but they need to know task boundaries during training for the consolidation of parameters' information of learned tasks, for example, update the prior of parameters to the posterior learned in the previous task in VCL. And architecture-based methods are the most demanding ones which often require task identifiers during training and testing time for the consolidation of task-shared components and selection of task-specific components. In addition, architecture-based methods are often computationally expensive because model compression and/or expansion are usually involved during the training progress. Due to these reasons, we explored our ideas outside the scope of architecture-based methods in this thesis.

We first made attempts to improve Bayesian continual learning based on Variational Continual Learning (VCL) by a regularization-based and a replay-based method. The regularization-based method Gaussian Natural Gradient (GNG) updates parameters of a Bayesian Neural Network (BNN) by natural gradients with Adam optimizer which gives a more efficient trajectory of moving towards the optimum of the new task while keeping close to the optimum of the previous task. The replay-based method Stein Gradient-based Episodic Memories (SGEM) composes the episodic memory by Stein Variational Gradient Descent (SVGD) along with the posterior approximation which can retain information of the posteriors in the memorized samples. Both methods have shown notable improvements on VCL, especially in the Permuted MNIST tasks. We consider they are less effective on split tasks because split tasks are less homogeneous and the optimum of each task is farther to each other. Since both methods aim to preserve the performance on old tasks by preserving more information of the previous posteriors, they may have less flexibility to search in a broader space where the optimum of multiple tasks lies when the tasks are less similar. It particularly degrades the performance on single-headed models due to the output layer tend to change mostly in split tasks. This is also a general issue of other regularization-based methods which makes us lean to replay-based methods in our following work.

Besides the practicality, some of the replay-based methods also have shown greater efficiency than regularization-based methods. For instance, the simple Experience Replay can outperform EWC, OEWC with a small episodic memory (Chaudhry et al., 2019b, Sec. 6.5). The memory size is smaller than storing the Fisher information of previous parameters, and the computational overhead is negligible comparing with computing the importance of parameters (e.g. Fisher information). The two fundamental problems of replay-based approaches are: (*i*) how to make the best use of episodic memories; and (*ii*) how to construct most representative episodic memories. Regarding the first problem, a line of work has made attempts to utilize the gradients produced by samples in the episodic memory as we introduced in Sec. 3.3.3. In general, those methods try to reduce the diversity of gradients over different tasks. Along this line, we further demonstrated the diversity of gradients strongly correlates to the discriminativeness of representations and made theoretical analysis through a linear model.

These findings also connect Deep Metric Learning (DML) to continual learning which opens a door to more options for continual learning. Moreover, we identified a weak spot in existing approaches of DML when applying them to continual learning. The essential idea of DML is to maximize margins between different classes and minimize distances/similarities between samples within a class. Maximizing margins shares the same interest with continual learning because it can reduce the diversity of gradients, however, continual learning would prefer less compact within-class representations as it may lose important information for learning future tasks by the compression. According to these findings, we proposed a simple yet efficient auxiliary objective Discriminative Representation Loss (DRL) for classification tasks in continual learning which have shown outstanding performance on both permuted and split benchmarks in the most difficult online setting with single-headed models. In addition, it has much lower computational cost comparing with methods that require computing similarity of gradients.

Our findings also shed light on the second problem: how to construct most representative episodic memories? A straightforward idea is that it would be better when the memorized samples can preserve more variance of the data distribution. In most of our experiments, the memory with randomly chosen samples outperforms those selected by gradient diversity because the memory size is limited and hence random samples contain more diversity as we demonstrated in Fig. 5.1. However, the variance is only one factor to represent a data distribution, we also need to consider how to maintain the most representative information of each task, and how to extract general information that would be useful for future tasks.

In addition to the approaches mentioned above we have also proposed two methods, β^3-Item Response Theory (β^3-IRT) model and Continual Density Ratio Estimation (CDRE), for evaluating classifiers and generative models in continual learning, respectively. Both can deal with a situation where the existing measurements are not applicable for model selection in continual learning. The β^3-IRT model infers the latent ability of a classifier which is adaptively weighted by instance-wise difficulties. The difficulty of each data instance is decided by the average response over all classifiers. Hence, it can be viewed as the relative difficulty given by a certain group of classifiers and the ability is also the relative ability among this group. As a Bayesian probabilistic model, β^3-IRT can be easily adapted to continual learning by applying the framework of VCL, which makes it feasible for evaluating classifiers in continual learning as well. When the average accuracy cannot tell the difference between two classifiers, the ability may be able to provide further insights when the two classifiers have different performance on different levels of instance-wise difficulty. In addition to infer the overall ability of classifiers simply by prediction performance, it may be beneficial to extend β^3-IRT by assuming a hierarchy of multiple sub-abilities which could include the evaluation on memory cost, computing time, RAM occupation, etc..

On the other hand, generative models have a different training process in continual learning as they can generate replay samples and play the role of episodic memories by themselves. In this sense, the quality of the generated samples is critical to prevent forgetting. The existing measurements of generative models are in essence measuring the difference between the original data distribution and the generated data distribution, which requires true samples from the original distribution. However, it may be impossible in the setting of continual learning. To this end, we proposed CDRE for evaluating generative models in continual learning which can be used to approximate f-divergences between the original and generated data distributions without storing samples of the original distribution. More importantly, CDRE is able to give more accurate approximation than standard DRE when the two distributions are less similar. We introduced an instantiation of CDRE by KLIEP which is derived from the form of KL-divergence. When estimating divergences which are not based on log-ratios it may be preferable to try some other form of ratio estimators (*i.e.* replacing KLIEP's formulation). For instance, it may be preferable to use Least-Squares Importance Fitting (LSIF) (Kanamori et al., 2009) when estimating Pearson χ^2 divergence, since a small deviation in log-ratio can result in large squared errors. Also, since LSIF itself is based on Pearson χ^2 divergence, it appears to be a more natural choice. On the other hand, it has been discussed by Mohamed & Lakshminarayanan (2016) that the discriminator of some type of GANs can be viewed as a density ratio estimator. For instance, it is feasible to apply the formulation of the discriminators of f-GAN (Nowozin et al., 2016) to the basic estimator in CDRE. It is also possible to estimate the Bregman divergence by ratio estimation (Uehara et al., 2016). These variants could give more options when applying CDRE to different types of data or scenarios.

8.2 Future Work

Through our experimental results, we see that most regularization-based and replay-based methods of continual learning suffer from the dilemma of the stability and plasticity of a model. According to the theoretical analysis in sec. 3.5, the memory is a key to obtain better performance over all tasks. A better memory that can recover the optimal solution of old tasks more easily while learning a new task. In existing work the episodic memory is mostly kept static for past tasks, it could be more sophisticated if we can evolve the memorized information after learning more tasks, like humans usually aggregate knowledge by developing hierarchical concepts and representations. For example, it may be worth trying to extract and aggregate attention-like representations with a hierarchical structure since the attention mechanism (Vaswani et al., 2017) have shown significant advantages in language models in which aggregating similar concepts is also important. We believe the construction of a dynamic and hierarchical episodic memory is an essential part of continual learning and it could be an important direction for future work.

In the current literature of continual learning, most benchmarks are from Class-IL scenarios and existing work pay more attention to the problem of Class-IL as well. However, Domain-IL scenarios are also common scenarios in practice. For instance, a very large dataset can be divided into several sub-sets and a model can be trained upon these sub-sets sequentially to avoid expensive computational cost. Another common example could be a model needs to be continuously trained on training sets that are collected continuously and these training sets include different domain instances in the same group of classes. These scenarios in continual learning are similar with the backward covariate shift scenario as introduced in Sec. 7.3. Applying CDRE to estimate the importance weights on samples of old tasks and combining the re-weighted loss on old tasks with the loss on the new task could be a solution to such cases. We consider this as a future work that is also worthy to try.

Architecture-based methods can obtain better performance regarding the trade-off between stability and plasticity but with a high cost in terms of computation and practicality. Practicality is mainly affected by task-specific components which require task identifiers during testing time. Combining episodic memories may be able to remove this cost by inferring the task identifiers through memorized information. One question is how much more information can be preserved by task-specific components than by episodic memories? If it is not significant, we can consider to expand the task-shared components when it is necessary and eliminate the need of task-specific components by episodic memories, which may combine the strength of architecture-based and replay-based methods without losing the efficiency and practicality. We believe this is another promising direction of future work for continual learning.

Overall, we believe further exploring replay-based methods and combining the strength of it with other types of methods is a promising direction for efficient continual learning.

APPENDIX A

APPENDIX

We provide the key code snippets of each proposed methods in this chapter. The full code repositories are public in github and we will share the links as well. We implemented all methods in Python based on Tensorflow 1.x.

A.1 Code for GNG and SVGD

The below code snippet of GNG shows how to apply natural gradients to variables of a Gaussian distribution, and variables of the Gaussian distribution must be defined as $\mu, \log\sigma$. This function is embedded in the process of VI and can be found in hsvi.py in the repository of Hierarchical Stochastic Variational Inference (HSVI) (`https://github.com/yc14600/hsvi`)

```
def natural_gradients_gaussian(self, loss, scope):
    trans_parm = self.trans_parm[scope]
    grads_and_vars = []
    for qz_vars in six.itervalues(trans_parm):
        g = tf.gradients(loss, qz_vars)
        if g[0] is not None:
            g[0] *= tf.exp(2.*qz_vars[1])
            grads_and_vars.append((g[0], qz_vars[0]))
        if g[1] is not None:
            g[1] *= 0.5
            grads_and_vars.append((g[1], qz_vars[1]))
    return grads_and_vars
```

The following function shows how to generate SGEM for a BNN in continual learning, which is provided in coreset_util.py in the repository `https://github.com/yc14600/utils`. The class SVGD() is provided in the repository of HSVI.

```
def gen_stein_coreset(init, core_y_data, qW, qB, n_samples, ac_fn, \
    conv_W=None, LR=False, noise_std=0.001, sess=None):
    stein_core_x = tf.get_variable('stein_cx', \
            initializer=init.astype(np.float32), dtype=tf.float32)
    print('gen␣stein␣coreset')
    if LR: # for linear regression
        stein_core_y = Normal(loc=tf.matmul(stein_core_x, qW)+qB, \
                    scale=noise_std)
    elif conv_W is not None:
        ## to do: change to general function ##
        h = forward_cifar_model(stein_core_x, conv_W)
        stein_core_y = forward_nets(qW, qB, h, ac_fn=ac_fn, \
        bayes=True, num_samples=10)[-1]
    else:
        stein_core_y = forward_nets(qW, qB, stein_core_x, ac_fn=ac_fn, \
        bayes=True, num_samples=10)[-1]
    lnp = tf.reduce_mean(stein_core_y.log_prob(core_y_data), axis=0)
    dlnp = tf.gradients(lnp, stein_core_x)
    svgd = SVGD()
    core_sgrad = svgd.gradients(stein_core_x, dlnp[0])
    return stein_core_x, stein_core_y, core_sgrad
```

Both GNG and SGEM are tested on VCL, by setting vi_type = "KLqp_analytic_GNG" and coreset_type = "stein" in VCL_test.py, respectively. The source code of VCL with these extensions is provided in the repository `https://github.com/yc14600/cl_models/tree/master/cl_models/discriminative/vcl`.

A.2 Code for DRL

The following function is the key implementation of DRL which is provided in the repository `https://github.com/yc14600/discriminative-representation-loss/tree/main`.

```
def config_loss(self, x, y, var_list, H, discriminant=True, \
                    likelihood=True, compact_center=False, *args, **kargs):

    loss, ll, reg, dis = 0., 0., 0., 0.
    if likelihood: # add cross entropy loss
```

```
        ll = tf.reduce_mean(
                tf.nn.softmax_cross_entropy_with_logits_v2(
                    logits=H[-1],labels=y))
        loss += ll

    if discriminant:    # add DRL loss
        yids = tf.matmul(y, tf.transpose(y))
        N = self.B
        mask = tf.eye(N)
        for h in H[:]:
            if len(h.shape) > 2:
                h = tf.reshape(h,[N,-1])
            sim = tf.matmul(h,tf.transpose(h))
            if not self.lamb0:  # use both L_bt and L_wi
                dis += tf.reduce_mean(sim*(1.-yids)
                           +self.alpha*sim*(yids-mask))
            else:   # use only L_wi
                dis += tf.reduce_mean(self.alpha*sim*(yids-mask))
        loss += self.lambda_dis * dis

    if self.reg:  # add L1 or L2 regularization loss
        print('add␣regularization␣loss')
        reg = tf.losses.get_regularization_loss()
        loss += self.lambda_reg *reg
    return loss,ll,reg,dis
```

A.3 Code for β^3-IRT

Below is the definition of β^3-IRT model and the full source code can be found in the repository `https://github.com/yc14600/beta3_IRT`.

```
class Beta_IRT:
    def __init__(self,M,C,theta_prior,delta_prior,a_prior):
        self.M = M
        self.C = C
        self.theta_prior = theta_prior # prior of ability
        self.delta_prior = delta_prior # prior of difficulty
        self.a_prior = a_prior  # prior of discrimination
        if isinstance(a_prior,ed.RandomVariable):
```

```
        # variational posterior of discrimination
        self.qa = Normal(loc=tf.Variable(tf.ones([M])),
                scale=tf.nn.softplus(tf.Variable(tf.ones([M])*.5)),
                name='qa')
    else:
        self.qa = a_prior

    with tf.variable_scope('local'):
        # variational posterior of ability
        if isinstance(self.theta_prior,RandomVariable):
            self.qtheta = TransformedDistribution(
                        distribution=Normal(
                          loc=tf.Variable(tf.random_normal([C])),
                          scale=tf.nn.softplus(
                                tf.Variable(tf.random_normal([C])))),
                        bijector=ds.bijectors.Sigmoid(),
                        sample_shape=[M],name='qtheta')
        else:
            self.qtheta = self.theta_prior
        # variational posterior of difficulty
        self.qdelta = TransformedDistribution(
                    distribution=Normal(
                      loc=tf.Variable(tf.random_normal([M])),
                      scale=tf.nn.softplus(
                          tf.Variable(tf.random_normal([M])))),
                    bijector=ds.bijectors.Sigmoid(),
                    sample_shape=[C],name='qdelta')

    alpha = (tf.transpose(self.qtheta)/self.qdelta)**self.qa

    beta = ((1. - tf.transpose(self.qtheta))
                                /(1. - self.qdelta))**self.qa

    # observed variable
    self.x = Beta(tf.transpose(alpha),tf.transpose(beta))
```

A.4 Code for CDRE

The following code snippets show how to define a log-linear density ratio estimator and how to adapt it to a continual density ratio estimator. The full source code is in the repository `https://github.com/yc14600/cdre`.

```
class LogLinear_Estimator(Estimator):
    def define_estimator_vars(self,scope,net_shape,conv,
                                ac_fn,batch_norm,reg):
        with tf.variable_scope(scope):
            self.W, self.B, self.nu_H = self.define_base_fc(
                self.nu_ph,net_shape, conv=conv,ac_fn=ac_fn,
                batch_norm=batch_norm, training=self.is_training,
                reuse=False,reg=reg)
            _, __, self.de_H = self.define_base_fc(
                self.de_ph,net_shape,conv=conv,ac_fn=ac_fn,
                batch_norm=batch_norm,training=self.is_training,
                reuse=True,reg=reg)
            self.nu_H[-1] = tf.clip_by_value(self.nu_H[-1],-60.,60.)
            self.de_H[-1] = tf.clip_by_value(self.de_H[-1],-60.,60.)

            # the output of \phi() for samples from numerator and
            # denominator distribution, respectively
            self.nu_r, self.de_r = self.nu_H[-1],self.de_H[-1]
        return

    def log_ratio(self,sess,x_nu,x_de,
                  nu_r=None,de_r=None,coef=None,
                  correction=1.,*args,**kargs):
        nu_r = self.nu_r if nu_r is None else nu_r
        de_r = self.de_r if de_r is None else de_r
        coef = self.coef if coef is None else coef
        if coef is None:
            log_r = nu_r - tf.log(tf.reduce_mean(tf.exp(de_r)))
                      - tf.log(correction)
        else:
            log_r = tf.matmul(nu_r,coef) -
                  tf.log(tf.reduce_mean(tf.exp(tf.matmul(de_r,coef))))
        feed_dict={self.nu_ph:x_nu,self.de_ph:x_de}
        if self.batch_norm:
            feed_dict.update({self.is_training:False})
        return sess.run(log_r,feed_dict)

    def ratio(self,sess,x_nu,x_de,nu_r=None,de_r=None,
                coef=None,correction=1.,*args,**kargs):
        log_r = self.log_ratio(sess,x_nu,x_de,nu_r,de_r,
                            coef,correction=correction)
        return np.exp(log_r)

class Continual_LogLinear_Estimator(Continual_Estimator):
    def update_estimator(self,sess,increase_constr=False,
                        nu_samples=None,de_samples=None,
                        restart=False):
        if not restart:
            self.prev_nu_r,self.prev_de_r =
                            self.save_prev_estimator(sess)
            self.estimator.nu_r = self.estimator.nu_H[-1] -
                                            self.prev_nu_r
            self.estimator.de_r = self.estimator.de_H[-1] -
                                            self.prev_de_r
            if increase_constr:
                self.lambda_constr += self.lambda_constr
            print('lambda_c',self.lambda_constr)
            if self.lambda_constr == 0:
                self.update_correction(nu_samples,de_samples,sess)
        else:
            self.prev_nu_r,self.prev_de_r = None,None
            self.estimator.nu_r = self.estimator.nu_H[-1]
            self.estimator.de_r = self.estimator.de_H[-1]
        self.update_train(self.estimator,restart=restart)
        return
```

BIBLIOGRAPHY

Cvpr 2020 continual learning in computer vision competition: Approaches, results, current challenges and future directions. *Artificial Intelligence*, 303:103635, 2022. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2021.103635. URL `https://www.sciencedirect.com/science/article/pii/S0004370221001867`.

Adel, T., Zhao, H., and Turner, R. E. Continual Learning with Adaptive Weights (CLAW). In *International Conference on Learning Representations*, 2020.

Ahn, H., Cha, S., Lee, D., and Moon, T. Uncertainty-based Continual Learning with Adaptive Regularization. In *Advances in Neural Information Processing Systems*, pp. 4392–4402, 2019.

Aljundi, R., Belilovsky, E., Tuytelaars, T., Charlin, L., Caccia, M., Lin, M., and Page-Caccia, L. Online Continual Learning with Maximal Interfered Retrieval. In *Advances in Neural Information Processing Systems*, pp. 11849–11860, 2019a.

Aljundi, R., Lin, M., Goujaud, B., and Bengio, Y. Gradient-based Sample Selection for Online Continual Learning. In *Advances in Neural Information Processing Systems*, pp. 11816–11825, 2019b.

Ans, B. and Rousset, S. Avoiding Catastrophic Forgetting by Coupling Two Reverberating Neural Networks. *Comptes Rendus de l'Académie des Sciences-Series III-Sciences de la Vie*, 320(12): 989–997, 1997.

Ans, B. and Rousset, S. Neural Networks with A Self-refreshing Memory: Knowledge Transfer in Sequential Learning Tasks without Catastrophic Forgetting. *Connection science*, 12(1):1–19, 2000.

Arjovsky, M., Chintala, S., and Bottou, L. Wasserstein Generative Adversarial Networks. In *International Conference on Machine Learning*, pp. 214–223, 2017.

Bińkowski, M., Sutherland, D. J., Arbel, M., and Gretton, A. Demystifying MMD GANs. In *International Conference on Learning Representations (ICLR)*, 2018.

Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer, 2006.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017. doi: 10.1080/01621459.2017.1285773. URL `https://doi.org/10.1080/01621459.2017.1285773`.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. Weight Uncertainty in Neural Network. In *International Conference on Machine Learning*, pp. 1613–1622, 2015.

Bonnabel, S. Stochastic Gradient Descent on Riemannian Manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013.

Bouchikhi, I., Ferrari, A., Richard, C., Bourrier, A., and Bernot, M. Non-parametric Online Change-point Detection with Kernel LMS by Relative Density Ratio Estimation. In *2018 IEEE Statistical Signal Processing Workshop (SSP)*, pp. 538–542. IEEE, 2018.

Breiman, L. Random Forests. *Machine learning*, 45(1):5–32, 2001.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. *Classification and Regression Trees*. CRC press, 1984.

Buzzega, P., Boschini, M., Porrello, A., Abati, D., and Calderara, S. Dark Experience for General Continual Learning: A Strong, Simple Baseline. In *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, 2020.

Campbell, T. and Broderick, T. Bayesian Coreset Construction via Greedy Iterative Geodesic Ascent. *arXiv preprint arXiv:1802.01737*, 2018.

Chaudhry, A., Dokania, P. K., Ajanthan, T., and Torr, P. H. Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 532–547, 2018.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. Efficient Lifelong Learning with A-GEM. In *International Conference on Learning Representations*, 2019a. URL `https://openreview.net/forum?id=Hkf2_sC5FX`.

Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H., and Ranzato, M. On Tiny Episodic Memories in Continual Learning. *arXiv preprint arXiv:1902.10486*, 2019b.

Chaudhry, A., Khan, N., Dokania, P., and Torr, P. Continual Learning in Low-rank Orthogonal Subspaces. *Advances in Neural Information Processing Systems*, 33, 2020.

Chen, Y., Diethe, T., and Lawrence, N. Facilitating Bayesian Continual Learning by Natural Gradients and Stein Gradients. *Continual Learning Workshop of 32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, 2018.

Chen, Y., Silva Filho, T., Prudencio, R. B., Diethe, T., and Flach, P. β^3-IRT: A New Item Response Model and its Applications. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1013–1021. PMLR, 2019.

Chen, Y., Diethe, T., and Flach, P. Discriminative Representation Loss (DRL): A More Efficient Approach Than Gradient Re-projection in continual learning. *arXiv preprint arXiv:2006.11234*, 2020.

Chen, Y., Liu, S., Diethe, T., and Flach, P. Continual Density Ratio Estimation in an Online Setting. *arXiv preprint arXiv:2103.05276*, 2021.

Chrysakis, A. and Moens, M.-F. Online Continual Learning from Imbalanced Data. In *International Conference on Machine Learning*, pp. 1952–1961. PMLR, 2020.

Delange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Deng, J., Guo, J., Xue, N., and Zafeiriou, S. Arcface: Additive Angular Margin Loss for Deep Face Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4690–4699, 2019.

Ebrahimi, S., Elhoseiny, M., Darrell, T., and Rohrbach, M. Uncertainty-Guided Continual Learning with Bayesian Neural Networks. In *International Conference on Learning Representations*, 2020.

Embretson, S. and Reise, S. *Item Response Theory for Psychologists*. Taylor & Francis, 2013. ISBN 9781135681470. URL `https://books.google.com.br/books?id=9Xm0AAAAQBAJ`.

Farajtabar, M., Azizan, N., Mott, A., and Li, A. Orthogonal Gradient Descent for Continual Learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 3762–3773. PMLR, 2020.

Figurnov, M., Mohamed, S., and Mnih, A. Implicit Reparameterization Gradients. *Advances in Neural Information Processing Systems*, 31:441–452, 2018.

Frénay, B. and Verleysen, M. Classification in the Presence of Label Noise: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869, 2014.

Friedman, J., Hastie, T., Tibshirani, R., et al. *The Elements of Statistical Learning*, volume 1. Springer series in statistics New York, 2001.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. C. Improved Training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, pp. 5767–5777, 2017.

Guo, Y., Liu, M., Yang, T., and Rosing, T. Improved Schemes for Episodic Memory-based Lifelong Learning. *Advances in Neural Information Processing Systems*, 33, 2020.

Hastie, T., Rosset, S., Zhu, J., and Zou, H. Multi-Class Adaboost. *Statistics and its Interface*, 2(3): 349–360, 2009.

He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems*, pp. 6626–6637, 2017.

Hinton, G., Vinyals, O., and Dean, J. Distilling the Knowledge in A Neural Network. *arXiv preprint arXiv:1503.02531*, 2015.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. Stochastic Variational Inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Honkela, A., Tornio, M., Raiko, T., and Karhunen, J. Natural conjugate gradient in variational inference. In *International Conference on Neural Information Processing*, pp. 305–314. Springer, 2007.

Huggins, J., Campbell, T., and Broderick, T. Coresets for Scalable Bayesian Logistic Regression. In *Advances in Neural Information Processing Systems*, pp. 4080–4088, 2016.

Jang, E., Gu, S., and Poole, B. Categorical Reparametrization with Gumble-Softmax. In *International Conference on Learning Representations (ICLR 2017)*. OpenReview. net, 2017.

Jung, S., Ahn, H., Cha, S., and Moon, T. Continual Learning with Node-Importance based Adaptive Group Sparse Regularization. *arXiv preprint arXiv:2003.13726*, 2020.

Kanamori, T., Hido, S., and Sugiyama, M. Efficient Direct Density Ratio Estimation for Non-stationarity Adaptation and Outlier Detection. In *Advances in neural information processing systems*, pp. 809–816, 2009.

Kanamori, T., Suzuki, T., and Sugiyama, M. f-Divergence Estimation and Two-Sample Homogeneity Test Under Semiparametric Density-Ratio Models. *IEEE Transactions on Information Theory*, 58(2):708–720, 2011.

Karras, T., Laine, S., and Aila, T. A Style-based Generator Architecture for Generative Adversarial Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4401–4410, 2019.

Kawahara, Y. and Sugiyama, M. Change-point Detection in Time-series Data by Direct Density-Ratio Estimation. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pp. 389–400. SIAM, 2009.

Kaya, M. and Bilge, H. Ş. Deep Metric Learning: A Survey. *Symmetry*, 11(9):1066, 2019.

Khan, M., Nielsen, D., Tangkaratt, V., Lin, W., Gal, Y., and Srivastava, A. Fast and Scalable Bayesian Deep Learning by Weight-perturbation in Adam. In *International Conference on Machine Learning*, pp. 2611–2620. PMLR, 2018.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pp. 10215–10224, 2018.

Kingma, D. P. and Welling, M. Auto-Encoding Variational Bayes. In *Proc. of the 2nd Int. Conf. on Learning Representations (ICLR)*, 2013.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming Catastrophic Forgetting in Neural Networks. *Proceedings of the National Academy of Sciences*, pp. 201611835, 2017.

Knoblauch, J., Husain, H., and Diethe, T. Optimal Continual Learning has Perfect Memory and is NP-hard. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Krizhevsky, A., Hinton, G., et al. Learning Multiple Layers of Features from Tiny Images. Technical report, Citeseer, 2009.

Kschischang, F. R., Frey, B. J., and Loeliger, H.-A. Factor Graphs and the Sum-Product Algorithm. *IEEE Transactions on Information Theory*, 47(2):498–519, 2001.

Kucukelbir, A., Tran, D., Ranganath, R., Gelman, A., and Blei, D. M. Automatic Differentiation Variational Inference. *Journal of Machine Learning Research*, 18(14):1–45, 2017.

Le, Y. and Yang, X. Tiny Imagenet Visual Recognition Challenge. *CS 231N*, 7, 2015.

LeCun, Y., Cortes, C., and Burges, C. J. MNIST Handwritten Digit Database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2, 2010.

Lesort, T., Caselles-Dupré, H., Garcia-Ortiz, M., Stoian, A., and Filliat, D. Generative Models from the perspective of Continual Learning. *arXiv preprint arXiv:1812.09111*, 2018.

Li, J., Li, F., and Todorovic, S. Efficient Riemannian Optimization on the Stiefel Manifold via the Cayley Transform. In *International Conference on Learning Representations*, 2020.

Li, X., Zhou, Y., Wu, T., Socher, R., and Xiong, C. Learn to Grow: A Continual Structure Learning Framework for Overcoming Catastrophic Forgetting. In *International Conference on Machine Learning*, pp. 3925–3934, 2019.

Li, Z. and Hoiem, D. Learning without Forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Liu, J., Bai, Y., Jiang, G., Chen, T., and Wang, H. Understanding Why Neural Networks Generalize Well Through GSNR of Parameters. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=HyevIJStwH`.

Liu, Q. and Wang, D. Stein Variational Gradient Descent: A General Purpose Bayesian Inference Algorithm. In *Advances In Neural Information Processing Systems*, pp. 2378–2386, 2016.

Liu, Q., Lee, J., and Jordan, M. A Kernelized Stein Discrepancy for Goodness-of-fit Tests. In Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 276–284, New York, New York, USA, 20–22 Jun 2016. PMLR.

Liu, S., Yamada, M., Collier, N., and Sugiyama, M. Change-point Detection in Time-series Data by Relative Density-Ratio Estimation. *Neural Networks*, 43:72–83, 2013.

Lomonaco, V. and Maltoni, D. CORe50: a New Dataset and Benchmark for Continuous Object Recognition. In *Proceedings of the 1st Annual Conference on Robot Learning (CoRL)*, volume 78, pp. 17–26, 2017. URL `http://proceedings.mlr.press/v78/lomonaco17a.html`.

Lomonaco, V., Pellegrini, L., Rodriguez, P., Caccia, M., She, Q., Chen, Y., Jodelet, Q., Wang, R., Mai, Z., Vazquez, D., et al. CVPR 2020 Continual Learning in Computer Vision Competition: Approaches, Results, Current Challenges and Future Directions. *arXiv preprint arXiv:2009.09929*, 2020.

Lopez-Paz, D. and Ranzato, M. Gradient Episodic Memory for Continual Learning. In *Advances in Neural Information Processing Systems*, pp. 6467–6476, 2017.

Louizos, C., Ullrich, K., and Welling, M. Bayesian Compression for Deep Learning. In *Advances in Neural Information Processing Systems*, pp. 3288–3298, 2017.

MacKay, D. J. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

Mallya, A., Davis, D., and Lazebnik, S. Piggyback: Adapting a Single Network to Multiple Tasks by Learning to Mask Weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 67–82, 2018.

Martínez-Plumed, F., Prudêncio, R. B., Martínez-Usó, A., and Hernández-Orallo, J. Making Sense of Item Response Theory in Machine Learning. In *European Conference on Artificial Intelligence, ECAI*, pp. 1140–1148, 2016.

McAllester, D. and Stratos, K. Formal Limitations on the Measurement of Mutual Information. In *International Conference on Artificial Intelligence and Statistics*, pp. 875–884, 2020.

McCloskey, M. and Cohen, N. J. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Mika, S., Ratsch, G., Weston, J., Scholkopf, B., and Mullers, K.-R. Fisher Discriminant Analysis with Kernels. In *Neural networks for signal processing IX: Proceedings of the 1999 IEEE signal processing society workshop (cat. no. 98th8468)*, pp. 41–48. Ieee, 1999.

Mirza, M. and Osindero, S. Conditional Generative Adversarial Nets. *arXiv preprint arXiv:1411.1784*, 2014.

Mohamed, S. and Lakshminarayanan, B. Learning in Implicit Generative Models. *arXiv preprint arXiv:1610.03483*, 2016.

Nguyen, C. V., Li, Y., Bui, T. D., and Turner, R. E. Variational Continual Learning. In *International Conference on Learning Representations*, 2018.

Nguyen, X., Wainwright, M. J., and Jordan, M. I. Estimating Divergence Functionals and the Likelihood Ratio by Convex Risk Minimization. *IEEE Transactions on Information Theory*, 56 (11):5847–5861, 2010.

Nichol, A., Achiam, J., and Schulman, J. On First-Order Meta-Learning Algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

Nishihara, R., Minka, T., and Tarlow, D. Detecting Parameter Symmetries in Probabilistic Models. *arXiv preprint arXiv:1312.5386*, 2013.

Noel, Y. and Dauvier, B. A Beta Item Response Model for Continuous Bounded Responses. *Applied Psychological Measurement*, 31(1):47–73, 2007.

Nowozin, S., Cseke, B., and Tomioka, R. f-GAN: Training Generative Neural Samplers Using Variational Divergence Minimization. In *Advances in Neural Information Processing Systems*, pp. 271–279, 2016.

Paisley, J., Blei, D. M., and Jordan, M. I. Variational Bayesian Inference with Stochastic Search. In *Proceedings of the 29th International Conference on Machine Learning*, pp. 1363–1370. Omnipress, 2012.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. Continual Lifelong Learning with Neural Networks: A Review. *Neural Networks*, 2019.

Pascanu, R. and Bengio, Y. Revisiting Natural Gradient for Deep Networks. In *International Conference on Learning Representations 2014 (Conference Track)*, April 2014. URL `http://arxiv.org/abs/1301.3584`.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Pinheiro, J. C. and Bates, D. M. Unconstrained Parametrizations for Variance-Covariance Matrices. *Statistics and Computing*, 6(3):289–296, 1996.

Prabhu, A., Torr, P. H., and Dokania, P. K. Gdumb: A simple approach that questions our progress in continual learning. In *Proceedings of the European Conference on Computer Vision*, 2020.

Ranganath, R., Gerrish, S., and Blei, D. M. Black Box Variational Inference. In *AISTATS*, pp. 814–822, 2014.

Rao, D., Visin, F., Rusu, A., Pascanu, R., Teh, Y. W., and Hadsell, R. Continual Unsupervised Representation Learning. In *Advances in Neural Information Processing Systems*, pp. 7647–7657, 2019.

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. iCARL: Incremental Classifier and Representation Learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 2001–2010, 2017.

Requeima, J., Gordon, J., Bronskill, J., Nowozin, S., and Turner, R. E. Fast and flexible multi-task classification using conditional neural adaptive processes. *Advances in Neural Information Processing Systems*, 32:7959–7970, 2019.

Rezende, D. and Mohamed, S. Variational Inference with Normalizing Flows. In *International Conference on Machine Learning*, pp. 1530–1538, 2015.

Rhodes, B., Xu, K., and Gutmann, M. U. Telescoping Density-Ratio Estimation. *arXiv preprint arXiv:2006.12204*, 2020.

Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., , and Tesauro, G. Learning to Learn without Forgetting By Maximizing Transfer and Minimizing Interference. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=B1gTShAct7`.

Ritter, H., Botev, A., and Barber, D. Online structured laplace approximations for overcoming catastrophic forgetting. In *Advances in Neural Information Processing Systems*, pp. 3738–3748, 2018.

Roth, K., Milbich, T., Sinha, S., Gupta, P., Ommer, B., and Cohen, J. P. Revisiting Training Strategies and Generalization Performance in Deep Metric Learning. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Ruder, S. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Sajjadi, M. S., Bachem, O., Lucic, M., Bousquet, O., and Gelly, S. Assessing Generative Models via Precision and Recall. In *Advances in Neural Information Processing Systems*, pp. 5228–5237, 2018.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. Improved Techniques for Training GANs. In *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

Schroff, F., Kalenichenko, D., and Philbin, J. Facenet: A Unified Embedding for Face Recognition and Clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

Schwarz, J., Luketina, J., Czarnecki, W. M., Grabska-Barwinska, A., Teh, Y. W., Pascanu, R., and Hadsell, R. Progress & Compress: A Scalable Framework for Continual Learning. *arXiv preprint arXiv:1805.06370*, 2018.

Serra, J., Suris, D., Miron, M., and Karatzoglou, A. Overcoming Catastrophic Forgetting with Hard Attention to the Task. In *International Conference on Machine Learning*, pp. 4548–4557, 2018.

Shalev-Shwartz, S. et al. Online Learning and Online Convex Optimization. *Foundations and trends in Machine Learning*, 4(2):107–194, 2011.

Shalev-Shwartz, S. et al. Online Learning and Online Convex Optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

Shimodaira, H. Improving Predictive Inference Under Covariate Shift by Weighting the Log-Likelihood Function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

Shin, H., Lee, J. K., Kim, J., and Kim, J. Continual Learning with Deep Generative Replay. In *Advances in Neural Information Processing Systems*, pp. 2990–2999, 2017.

Sluban, B. and Lavrač, N. Relating Ensemble Diversity and Performance: A Study in Class Noise Detection. *Neurocomputing*, 160:120–131, 2015.

Sriperumbudur, B. K., Fukumizu, K., Gretton, A., Schölkopf, B., Lanckriet, G. R., et al. On the Empirical Estimation of Integral Probability Metrics. *Electronic Journal of Statistics*, 6: 1550–1599, 2012.

Stojanov, P., Gong, M., Carbonell, J., and Zhang, K. Low-Dimensional Density Ratio Estimation for Covariate Shift Correction. In Chaudhuri, K. and Sugiyama, M. (eds.), *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pp. 3449–3458. PMLR, 16–18 Apr 2019. URL `http://proceedings.mlr.press/v89/stojanov19a.html`.

Sugiyama, M., Suzuki, T., Nakajima, S., Kashima, H., von Bünau, P., and Kawanabe, M. Direct Importance Estimation for Covariate Shift Adaptation. *Annals of the Institute of Statistical Mathematics*, 60(4):699–746, 2008.

Sugiyama, M., Suzuki, T., and Kanamori, T. *Density Ratio Estimation in Machine Learning*. Cambridge University Press, 2012.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

Tomczak, J. and Welling, M. VAE with a VampPrior. In *International Conference on Artificial Intelligence and Statistics*, pp. 1214–1223, 2018.

Tran, D., Kucukelbir, A., Dieng, A. B., Rudolph, M., Liang, D., and Blei, D. M. Edward: A library for probabilistic modeling, inference, and criticism. *arXiv preprint arXiv:1610.09787*, 2016.

Tseran, H., Khan, M. E., Harada, T., and Bui, T. D. Natural Variational Continual Learning. *Continual Learning Workshop of 32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, 2018.

Uehara, M., Sato, I., Suzuki, M., Nakayama, K., and Matsuo, Y. Generative Adversarial Nets from a Density Ratio Estimation Perspective. *arXiv preprint arXiv:1610.02920*, 2016.

van de Ven, G. M. and Tolias, A. S. Three Scenarios for Continual Learning, 2019.

van den Oord, A., Vinyals, O., et al. Neural Discrete Representation Learning. In *Advances in Neural Information Processing Systems*, pp. 6306–6315, 2017.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is All You Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 6000–6010, 2017.

Vitter, J. S. Random Sampling with a Reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1):37–57, 1985.

Wang, H. and Yeung, D.-Y. A Survey on Bayesian Deep Learning. *ACM Computing Surveys (CSUR)*, 53(5):1–37, 2020.

Wang, J., Zhou, F., Wen, S., Liu, X., and Lin, Y. Deep Metric Learning with Angular Loss. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2593–2601, 2017.

Wang, X., Han, X., Huang, W., Dong, D., and Scott, M. R. Multi-Similarity Loss with General Pair Weighting for Deep Metric Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5022–5030, 2019.

Weinberger, K. Q., Blitzer, J., and Saul, L. K. Distance Metric Learning for Large Margin Nearest Neighbor Classification. In *Advances in Neural Information Processing Systems*, pp. 1473–1480, 2006.

Wen, Y., Tran, D., and Ba, J. BatchEnsemble: an Alternative Approach to Efficient Ensemble and Lifelong Learning. In *International Conference on Learning Representations*, 2020.

Winn, J. and Bishop, C. M. Variational Message Passing. *Journal of Machine Learning Research*, 6(Apr):661–694, 2005.

Wu, C., Herranz, L., Liu, X., wang, y., van de Weijer, J., and Raducanu, B. Memory Replay GANs: Learning to Generate New Categories without Forgetting. In *Advances in Neural Information Processing Systems 31*, pp. 5962–5972. 2018.

Wu, C.-Y., Manmatha, R., Smola, A. J., and Krahenbuhl, P. Sampling Matters in Deep Embedding Learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2840–2848, 2017.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms, 2017.

Yoon, J., Yang, E., Lee, J., and Hwang, S. J. Lifelong Learning with Dynamically Expandable Networks. In *International Conference on Learning Representations*, 2018.

Yoon, J., Kim, S., Yang, E., and Hwang, S. J. Scalable and Order-robust Continual Learning with Additive Parameter Decomposition. In *International Conference on Learning Representations*, 2020.

Zenke, F., Poole, B., and Ganguli, S. Continual Learning Through Synaptic Intelligence. In *International Conference on Machine Learning*, pp. 3987–3995, 2017.

Zhang, C., Bütepage, J., Kjellström, H., and Mandt, S. Advances in Variational Inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):2008–2026, 2019. doi: 10.1109/TPAMI.2018.2889774.

Zhang, H., Wu, C., Zhang, Z., Zhu, Y., Zhang, Z., Lin, H., Sun, Y., He, T., Muller, J., Manmatha, R., Li, M., and Smola, A. ResNeSt: Split-Attention Networks. *arXiv preprint arXiv:2004.08955*, 2020a.

Zhang, J., Zhang, J., Ghosh, S., Li, D., Zhu, J., Zhang, H., and Wang, Y. Regularize, Expand and Compress: Nonexpansive Continual Learning. In *The IEEE Winter Conference on Applications of Computer Vision*, pp. 854–862, 2020b.