ThompsonBALD: A New Approach to Bayesian Batch Active Learning for Deep Learning via Thompson Sampling

Jaeik Jeon[1]

MSc Computational Statistics and Machine Learning

Supervisor: Dr. Brooks Paige

September 2020

[1]**Disclaimer:** This report is submitted as part requirement for the MSc Computational Statistics and Machine Learning at UCL. It is substantially the result of my own work except where explicitly indicated in the text. The report may be freely copied and distributed provided the source is explicitly acknowledged.

Abstract

Deep learning has been successfully applied to many pattern recognition tasks aided by a very large labeled dataset. When the cost of acquiring labels is high, however, the deployment of deep learning is restrictive. To ease this problem, probabilistic active learning methods can be used for actively choosing data to be labeled from an unlabeled dataset. In this work, we introduce a novel Bayesian batch active learning approach, called ThompsonBALD. ThompsonBALD approximates the mutual information between a point and model parameters, and is used as an acquisition function for querying a batch of unlabeled data through Thompson sampling. While naive greedy selection based on the mutual information results correlated query, ThompsonBALD efficiently samples diverse batch. We demonstrate ThompsonBALD achieves competing performance compared to other recently invented methods with significantly reduced computational time.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Deep convolutional neural networks (CNNs) have been firmly established as state of the art approaches in many areas of machine learning researches such as image classification. Recent advances and universal successes in deep learning require massive amounts of data to make an inference for the large number of parameters the model has. The fact that deep neural networks are not saturated often also accelerates the constant desire to collect much data. In many cases, however, the data should be labeled by domain experts who have time and cost constraints.

The data-driven models such as CNNs often assume that the data is provided from uncontrollable sources. However, training the model can be made more efficient by actively selecting training data. There might be two possible scenarios. The first scenario is to decide where to look next in order to maximize information gain. This is often the case in scientific experiments when data measurements are expensive or slow (Gramacy et al., 2015; Gu et al., 2018; Oakley, 1999). The second scenario is that there already exists a part of dataset (a.k.a *unlabeled* dataset), and a subset of data points from the dataset is chosen by a model to ask the user for a label. For example, there are many unlabeled MRI scans and a medical specialist is commissioned to label them. Considering the cost, we aim to reduce the number of data that needs to be labeled. In this work, we focus on the second scenario for image classification problems. This framework is often referred to as active learning (MacKay, 1992; Cohn et al., 1996).

Active learning has been applied successfully in fields such as computational chemistry (Warmuth et al., 2002), text classification (Tong and Koller, 2001; Zhu et al., 2003), relation extraction Bloodgood (2018) and manufacturing (Tong, 2001). Despite the empirical successes of active learning, these methods used in these paper are difficult to be leveraged by deep learning. This is because (Tong, 2001) focuses on low dimensional data and (Warmuth et al., 2002; Zhu et al., 2003; Bloodgood, 2018) rely on kernel through support vector machines (SVMs) or Gaussian processes to cope with high dimensional data. There exist other approaches using an uncertainty based approach, which is relatively easy to apply to CNNs, such as Joshi et al. (2009); Brinker (2003), yet Sener and Savarese (2017) argues that they are not effective for CNNs.[1]

1.2 Related Works

Various papers recently have attempted to solve the "Deep active learning problem". Gal et al. (2017) rely on the Bayesian way of achieving model uncertainty, assessing the utility of candidate data points via *acquisition functions*. One example is the BALD acquisition function (Houlsby et al., 2011), measuring the mutual information between response and model parameters. Tools in Bayesian neural networks such as MC dropout (Gal and Ghahramani, 2016), Bayes by backprop (Blundell et al., 2015) and stochastic Gradient Langevin Dynamics (SGLD) (Welling and Teh, 2011) have enabled BALD to be used in deep learning by approximating a posterior distribution over the weights.

The acquisition functions return a single most informative point from an unlabeled dataset. To reduce computational time, a batch of points may need to be selected. In Gal et al. (2017), top B points are selected sorted by BALD score. This has been shown to outperform random acquisition for MNIST, yet it does not necessarily scale to a larger and more complex dataset because the collected points are not *jointly* informative.

Several techniques for deep learning that are robust to batch selection have been proposed. Kirsch et al. (2019) propose BatchBALD as an extension of BALD. This takes information

[1] This is due to batch sampling. We will discuss this in section 3.2.2

overlap of multiple points into account, so that they are jointly informative. Sener and Savarese (2017) approaches the batch active learning as a core-set selection by solving the K-center problem. Pinsler et al. (2019) construct a batch that best approximates the complete log posterior inspired by Bayesian coresets as sparse subset approximation (Campbell and Broderick, 2019). In this work, we focus on the Bayesian approach of modeling uncertainty, as in Pinsler et al. (2019); Kirsch et al. (2019).

1.3 Contribution

As the recent deep Bayesian active learning algorithms heavily rely on approximate inference, they are sensitive to the quality of the model parameter uncertainty. Despite this fact, they have been evaluated with just a single inference method in each of their papers' experiments. Most importantly, the most recent work of Kirsch et al. (2019) and Paisley et al. (2012) are not cross-evaluated since they are published simultaneously. In this paper, we provide fair experiments of them with multiple approximate inference methods on various datasets, so that practitioners can identify their pros and cons in various settings.

Besides comparative experiments, we also introduce a novel Bayesian batch active learning algorithm, ThompsonBALD, that mitigates the issues mentioned above. The key idea of ThompsonBALD is to recast the active learning framework to a reinforcement learning problem. We show that the BALD acquisition function can be re-formulated as the conditional expectation of the reward. Depending on the formulation of BALD, we propose two versions of ThompsonBALD that quantify the reward differently. We leverage the ThompsonBALD reward via Thompson sampling (Thompson, 1933) to cope with exploration-exploitation trade-offs. Our empirical analysis demonstrates the competing performance with Gal et al. (2017); Pinsler et al. (2019); Kirsch et al. (2019) with the significantly low computational cost.

1.4 Dissertation outline

This dissertation structured as follows. In chapter 2, we review recent advances in Bayesian neural networks and associated scalable approximate inference procedures. In chapter 3, we present various acquisition functions in active learning and their usage for estimating the utility

of candidate data points. We discuss the limitations of greedy selection through these functions and review recently proposed Bayesian batch active learning algorithms Gal et al. (2017); Kirsch et al. (2019); Pinsler et al. (2019). In chapter 4, we cross-evaluate with two variational inference methods: MC dropout and neural linear on MNIST, Fashion MNIST and their repeated version. In chapter 5, we propose a new active learning algorithm, ThompsonBALD, faster than other baseline algorithms as well as competing performance. In chapter 6, we evaluate our algorithm not only as in the same setting in chapter 4, but also in additional datasets and criterion. In chapter 7, we draw our conclusion with future research directions.

Chapter 2

Bayesian Neural Network

This chapter aims to provide theoretical grounds of Bayesian neural network that is necessary for the deep Bayesian batch active learning in chapter 3, and structured as follows. In section 2.1, we describe principles of Bayesian modeling of uncertainty. In section 2.2, we discuss approximate inference methods that could be applied to neural networks. In section 2.3 and 2.4, we present two variational inference techniques such as Bayes by backprop and MC dropout.

2.1 Bayesian Modelling

Given training inputs $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subseteq X$ and the corresponding outputs $\{\mathbf{y}_1, \ldots \mathbf{y}_N\} \subseteq Y$, the regression (or classification) task is to discover the dependence of a variable $\mathbf{y} \in Y$ on an input variable $\mathbf{x} \in X$ with the assumption that the dependence is controlled by some function $\mathbf{f}^{\boldsymbol{\omega}} : X \to Y$, parametrized by some $\boldsymbol{\omega} \in \Omega$. In Bayesian framework, we would like to reduce uncertainty on the model parameter $\boldsymbol{\omega}$ so that the relationship $\mathbf{y} = \mathbf{f}^{\boldsymbol{\omega}}(\mathbf{x})$ is likely to generate the outputs $\mathbf{Y}$.

Before we observe any data, uncertainty on the model parameters is expressed as some probabilistic model $p(\boldsymbol{\omega})$, i.e. *prior distribution*, representing our initial beliefs on the dependence of $\mathbf{x}$ and $\mathbf{y}$. We then update our belief on the parameter once some data $(\mathbf{x}, \mathbf{y})$ is observed by combining the prior belief with a likelihood distribution $p(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega})$. This combining process is done by invoking the Bayes' theorem,

$$p(\boldsymbol{\omega}|\mathbf{x}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega}) p(\boldsymbol{\omega})}{p(\mathbf{y}|\mathbf{x})}, \tag{2.1}$$

This is the posterior distribution over the parameter $\boldsymbol{\omega}$, representing a distribution of our beliefs

on the model parameter rather than just point estimates. Using the posterior distribution on the model parameter, prediction on an output for a new input data $\mathbf{x}^*$ is done by integrating

$$p(\mathbf{y}^*|\mathbf{x}^*,\mathbf{x},\mathbf{y}) = \int_{\boldsymbol{\omega}\in\Omega} p(\mathbf{y}^*|\mathbf{x}^*,\boldsymbol{\omega})p(\boldsymbol{\omega}|\mathbf{x},\mathbf{y}), \tag{2.2}$$

i.e. all the possible parameter's configurations are averaged out.
The most important part (*which also makes Bayesian inference difficult*) is the normalizing factor (or Bayes factor or marginal likelihood) in (2.1):

$$p(\mathbf{y}|\mathbf{x}) = \int_{\boldsymbol{\omega}\in\Omega} p(\mathbf{y}|\mathbf{x},\boldsymbol{\omega})p(\boldsymbol{\omega}) \tag{2.3}$$

This marginalization over the model parameter $\boldsymbol{\omega}$ can analytically be derived for simple models using conjugate prior, yet this is analytically intractable especially for deep neural architectures that we will mainly focus on. This also implies that the inference steps both in (2.1) and (2.2) is intractable as well. Hence, from now on, we will discuss ways to make an inference through approximation.

2.2 Approximate Inference

As the normalizing factor involved in the posterior distribution is intractable due to the integration, the recent advances in Bayesian modeling (especially Bayesian neural network) highly rely on approximate inference methods. Several methods have been suggested using variational inference and Markov chain Monte Carlo (MCMC).

MCMC is a sampling-based technique that generates a Markov chain that consists of a finite number of random samples from the posterior distribution. The application of MCMC to the Bayesian neural network has its root from Neal (1995)'s thesis which uses Hamiltonian Monte Carlo (HMC). HMC takes its advantage of gradient and momentum information, however, since it requires to acquire that information from the entire dataset and also involves a Metropolis accept-reject step, it is impractical for large dataset (Neal et al., 2011; Riquelme et al., 2018). In this context, Welling and Teh (2011) proposes Stochastic Gradient Langevin Dynamics (SGLD) that makes scalable to a large dataset by using *the stochastic estimates* of the gradient of the likelihood similar to Graves (2011), as well as the Langevin method which is a simplification

of Hamiltonian dynamics (the simplification is done through the use of single leapfrog step).

Variational inference is another popular method in Bayesian modelling that aims to approximate the posterior distribution by another simple distribution, e.g. variational distribution (Jordan et al., 1999). Then inference of posterior $p(\boldsymbol{\omega}|\mathbf{x},\mathbf{y})$ is naturally reformulated as an optimization problem that reduces the distance (with some abuse of words) between variational distribution and the true posterior. Formally, we use the Kullback–Leibler (KL) divergence (Kullback and Leibler, 1951) as a measure of *similarity* between the variational distribution $q_{\boldsymbol{\theta}}(\boldsymbol{\omega})$ parametrized by $\boldsymbol{\theta}$ and the true posterior and minimize it with respect to $\boldsymbol{\theta}$:

$$\mathrm{KL}[q_{\boldsymbol{\theta}}(\boldsymbol{\omega})||p(\boldsymbol{\omega}|\mathbf{x},\mathbf{y})] = \int_{\boldsymbol{\omega}\in\Omega} q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \log \frac{q_{\boldsymbol{\theta}}(\boldsymbol{\omega})}{p(\boldsymbol{\omega}|\mathbf{x},\mathbf{y})} \geq 0 \tag{2.4}$$

Note that the KL divergence is not a proper metric because it is not a symmetric function, and thus this is a measure of similarity, not a measure of distance. Prior to discuss ways to the appropriate optimization problem, we remind that the predictive distribution $p(y^*|\mathbf{x}^*,\mathbf{x},\mathbf{y})$ can be approximated by replacing the posterior distribution with $q^*_{\boldsymbol{\theta}}(\boldsymbol{\omega})$, and further be approximated with the Monte Carlo (MC) integration:

$$p(\mathbf{y}^*|\mathbf{x}^*,\mathbf{x},\mathbf{y}) = \int_{\boldsymbol{\omega}\in\Omega} p(\mathbf{y}^*|\mathbf{x}^*,\boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathbf{x},\mathbf{y}) \tag{2.5}$$

$$\approx \int_{\boldsymbol{\omega}\in\Omega} p(\mathbf{y}^*|\mathbf{x}^*,\boldsymbol{\omega}) q^*_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \xrightarrow[T\to\infty]{} \frac{1}{T}\sum_{t=1}^{T} p(\mathbf{y}^*|\mathbf{x},\hat{\boldsymbol{\omega}}_t) := \hat{q}_{\boldsymbol{\theta}}(\mathbf{y}^*|\mathbf{x}^*) \tag{2.6}$$

with $\hat{\boldsymbol{\omega}}_t \sim q_{\boldsymbol{\theta}}(\boldsymbol{\omega})$ and T is the number of MC samples.

Our objective is to minimize the KL divergence of $q_{\boldsymbol{\theta}}(\boldsymbol{\omega})$ to $p(\boldsymbol{\omega}|\mathbf{x},\mathbf{y})$. This is equivalent to maximizing the variational lower bound[1] (e.g. evidence lower bound (ELBO) (Neal and Hinton, 1998)) with respect to the variational parameter $\boldsymbol{\theta}$, where the bound is achieved by using the Jensen's inequality:

$$\text{log evidence} = \log p(\mathbf{y}|\mathbf{x}) \geq \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{\omega})}[\log p(\mathbf{y}|\mathbf{x},\boldsymbol{\omega})] - \mathrm{KL}[q_{\boldsymbol{\theta}}(\boldsymbol{\omega})||p(\boldsymbol{\omega})] := L_{\mathrm{VI}}(\boldsymbol{\theta}) \tag{2.7}$$

[1]This is due the following equality: log evidence=$\log p(\mathbf{y}|\mathbf{x}) = \mathrm{KL}[q^*_{\boldsymbol{\theta}}(\boldsymbol{\omega})||p(\boldsymbol{\omega}|\mathbf{x},\mathbf{y})] + L_{\mathrm{VI}}(\boldsymbol{\theta})]$. Then, the log evidence and ELBO gets equivalent when the divergence gets zero.

In order to optimize this objective, the first step would be to differentiate the variational lower bound L_{VI} with respect to $\boldsymbol{\theta}$. However, its exact gradient is infeasible. The usual approach is done by Paisley et al. (2012), Monte Carlo unbiased gradient estimator, using a stochastic approximation of the gradients:

$$\nabla_{\boldsymbol{\theta}} \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{\omega})}[f(\boldsymbol{\omega})] = \mathbb{E}_{q_{\boldsymbol{\theta}}(\boldsymbol{\omega})}[f(\boldsymbol{\omega}) \nabla_{\boldsymbol{\theta}} \log q_{\boldsymbol{\theta}}(\boldsymbol{\omega})] \approx \frac{1}{T} \sum_{t=1}^{T} f(\hat{\boldsymbol{\omega}}_t) \nabla_{\boldsymbol{\theta}} \log q_{\boldsymbol{\theta}}(\hat{\boldsymbol{\omega}}_t), \tag{2.8}$$

where the identity $\nabla_{\boldsymbol{\theta}} q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) = q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \nabla_{\boldsymbol{\theta}} \log q_{\boldsymbol{\theta}}(\boldsymbol{\omega})$ is used, and T is the number of MC samples from $\hat{\boldsymbol{\omega}}_t \sim q_{\boldsymbol{\theta}}(\boldsymbol{\omega})$. However, this turns out to have too high variance to use in practice (Paisley et al., 2012; Kingma and Welling, 2013). This further makes us difficult to train the deep neural network because stochastic gradient descent (SGD) is often employed for optimization, where the speed of convergence depends on its variance (Robbins and Monro, 1951).

2.3 The (Local) Reparametrization Trick

Reparametrization Trick In this context, Kingma and Welling (2013) proposes a generic Stochastic Gradient Variational Bayes (SGVB) estimator by reparametrizing the random parameter $\boldsymbol{\omega} \sim q_{\boldsymbol{\theta}}(\boldsymbol{\omega})$ as $\boldsymbol{\omega} = g(\boldsymbol{\theta}, \boldsymbol{\varepsilon})$, where $g(\cdot)$ is a differentiable function and $\boldsymbol{\varepsilon} \sim p(\boldsymbol{\varepsilon})$ is a random noise variable. With this reparametrization, an unbiased differentiable minibatch MC estimates of the variational lower bound can be achieved as

$$\hat{L}_{VI}(\boldsymbol{\theta}) = \frac{N}{M} \sum_{m=i}^{M} \underbrace{\log p(y_m | \mathbf{x}_m, \boldsymbol{\omega} = g(\boldsymbol{\theta}, \boldsymbol{\varepsilon}))}_{L_m} - \text{KL}[q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) || p(\boldsymbol{\omega})], \tag{2.9}$$

where $(\mathbf{x}_m, y_m)_{m=1}^{M}$ is a minibatch of data with M random points with N observations in a dataset (Kingma and Welling, 2013). Note that we assume that the KL divergence term can be computed analytically.[2] As shown by Kingma et al. (2015), the total contribution to $\text{Var}\left(\frac{N}{M} \sum_{m=i}^{M} L_m\right)$ by the $\text{Cov}(L_m, L_l)$ does not decrease with the minibatch size M,[3] which implies the variance may not decrease with M if the covariance is large. We will soon see how we achieve $\text{Cov}(L_m, L_l) = 0$ and cope with corresponding inevitable computational inefficiency.

[2] For example, the KL divergence between two Gaussian can be obtained in closed form. If not, this also can be approximated with MC samples for richer prior/posterior combinations, as in Blundell et al. (2015).

[3] $\text{Var}\left(\frac{N}{M} \sum_{m=1}^{M} L_m\right) = \frac{N^2}{M^2} \left(\sum_{i=1}^{M} \text{Var}(L_m) + 2 \sum_{m=1}^{M} \sum_{l=m+1}^{M} \text{Cov}(L_m, L_l)\right) = \frac{1}{M} \text{Var}(L_m) + \frac{M-1}{M} \text{Cov}(L_m, L_l)$ (Kingma et al., 2015)

Bayes by Backprop Aided by this reparametrization trick, Blundell et al. (2015) introduces *Bayes by Backprop* which is a backpropagation-compatible algorithm for learning probability distributions over the weights of a neural network. Blundell et al. (2015) uses a Gaussian distribution for each individual weights for variational posterior and implements mean field approximation to $q_{\boldsymbol{\theta}}(\boldsymbol{\omega})$ which fully factorises over the weights, as in Hinton and Van Camp (1993):

$$q_{\boldsymbol{\theta}}(\boldsymbol{\omega}) = \prod_{i=1}^{L} q_{\boldsymbol{\theta}}(W_i) = \prod_{i=1}^{L} \prod_{j=1}^{K_i} \prod_{k=1}^{K_{i+1}} q_{\mu_{ijk}\sigma_{ijk}}(w_{ijk}) = \prod_{ijk} N(w_{ijk}; \mu_{ijk}, \sigma_{ijk}^2), \tag{2.10}$$

where L is number of hidden layers, K_i is number of features in each layer. Blundell et al. (2015) reparametrize each weights in the layers by: $w_{ijk} = \mu_{ijk} + \sigma_{ijk}\varepsilon_{ijk}$, $\varepsilon_{ijk} \sim N(0,1)$, and further parametrize the standard deviation through a Softplus function $\sigma_{ijk} = \log(1+\exp(\rho_{ijk}))$ so that it is always non-negative. In conclusion, the weights of each fully connected layer is reparametrized as

$$w_{ijk} = g(\boldsymbol{\theta}, \boldsymbol{\varepsilon}) = \mu_{ijk} + \log(1+\exp(\rho_{ijk}))\varepsilon_{ijk} \tag{2.11}$$

with the variational posterior parameters $\boldsymbol{\theta} = (\boldsymbol{\mu}, \boldsymbol{\rho})$. The number of the parameters are doubled for the same network size which may require more time to converge. The reparametrized ELBO can now be written as (Blundell et al., 2015):

$$L_{\text{VI}}(\boldsymbol{\theta}) = \mathbb{E}_{p(\boldsymbol{\varepsilon})}\left[\log p(\mathbf{y}|\mathbf{x}, \boldsymbol{\omega}) - \log \frac{q_{\boldsymbol{\theta}}(\boldsymbol{\omega})}{p(\boldsymbol{\omega})}\right] \tag{2.12}$$

Combining this with the minibatch MC esitmates of the ELBO in (2.9), each step of optimizing the objective is followed by: 1) sample $\varepsilon_{ijk} \sim N(0,1)$ 2) calculate w_{ijk} according to (2.11) 3) calculate the gradient with respect to $\boldsymbol{\theta}$ 4) update the variational parameters. As a result, by sampling separate weight matrix W for each training example in the minibatch, $\text{Cov}(L_l, L_m)$ is enforced to be 0. (Kingma et al., 2015)

Local Reparametrization Trick Kingma et al. (2015) proposes a new parameterization trick computationally more efficient than sampling separate weight matrices W_i for each example in the minibatch, but also results in a lower variance gradient estimator. This is done by translating

the global uncertainty in the weights into a form of local uncertainty.

For simplicity, consider a simple fully connected neural network with K neurons in a hidden layer. The input to the neural network is a $M \times K$ input feature matrix A. The input matrix is multiplied by a $K \times K$ weight matrix W to get the pre-activation neurons $B = AW$. The posterior on W is approximated with a fully factorized Gaussian: $q_{\boldsymbol{\theta}}(w_{ij}) = N(\mu_{ij}, \sigma_{ij}^2)$, parameterized as $w_{ij} = \mu_{ij} + \sigma_{ij}\varepsilon_{ij}, \varepsilon_{ij} \sim N(0,1)$. As before, by sampling a separate weight matrix W for each training example in the minibatch, $\text{Cov}(L_m, L_l) = 0$ is enforced, yet this is impractical (Kingma et al., 2015). This is because $M \times K \times K$ random numbers need to be sampled in each minibatch.

Kingma et al. (2015) leverages the fact that with a input A and Gaussian distributions over weights W, the resulting posterior distribution conditioned on A for pre-activation neurons B is also Gaussian:

$$q_{\boldsymbol{\theta}}(b_{mj}|A) = N(\gamma_{mj}, \delta_{mj}), \text{ with } \gamma_{mj} = \sum_{i=1}^{K} a_{mi}\mu_{ij}, \text{ and } \delta_{mj} = \sum_{i=1}^{K} a_{mi}^2\sigma_{ij}^2 \tag{2.13}$$

Sampling directly from B significantly reduces the computational cost, requiring only $M \times 1000$ numbers, using $b_{mj} = \gamma_{mj} + \sqrt{\delta_{mj}}\xi_{mj}$ with $\xi_{mj} \sim N(0,1)$. Kingma et al. (2015) also shows both theoretically and empirically that the gradient estimator with this local reparametrization has a lower variance with faster and more stable training process. Lastly, it is important to note that by implementing the local remparmetrization trick, the KL divergence is now intractable since the pre-activation is sampled instead of weights. Therefore, a Gaussian prior should be used for the tractable divergence.[4]

2.4 Monte Carlo Dropout

Gal and Ghahramani (2015b,a) recast the dropout training in deep neural networks as approximate Bernoulli variational inference in Bayesian NNs. The dropout (Hinton et al., 2012; Srivastava et al., 2014) is a stochastic regularisation technique, enabling the deep neural networks to avoid over-fitting in practice. For simplicity, again, we review the dropout model for the case

[4]Derivation for KL divergence between two Gaussians can be found: https://stats.stackexchange.com/questions/60680/kl-divergence-between-two-multivariate-gaussians

of single hidden layer neural network. We assume the following network: $\hat{\mathbf{y}} = \sigma(\mathbf{x}M_1 + \mathbf{b})M_2$. $\mathbf{x}$ is an input with size $1 \times K_1$, the size of weight matrices M_1, M_2 is $K_1 \times K_2$ and $K_2 \times K_3$, $\sigma(\cdot)$ is some non-linearity, and $\mathbf{b}$ is a bias term with size $1 \times K_2$.

Dropout is applied by sampling two vectors $\mathbf{z}_1, \mathbf{z}_2$ from a Bernoulli distribution with some parameter $p_i \in [0,1]$ of size $1 \times K_1$ and $1 \times K_2$ respectively. Then the output of the network is given by

$$\hat{\mathbf{y}} = (\sigma((\mathbf{x} \circ \mathbf{z}_1)M_1 + \mathbf{b}) \circ \mathbf{z}_2)M_2 = \sigma(\mathbf{x}(\text{diag}(\mathbf{z}_1)M_1) + \mathbf{b})(\text{diag}(\mathbf{z}_2)M_2) \tag{2.14}$$

Therefore, we see that some of the network's weights are turned off by sampling $\mathbf{z}_i$ during training. The same binary values are used for back-propagation. Gal and Ghahramani (2015b) relates the variational inference in the Bayesian neural network by defining the variational distribution $q(W_i)$ for every layer i as:

$$W_i = \text{diag}([z_{ij}]_{j=1}^{K_i})M_i,\ z_{ij} \sim \text{Bernoulli}(p_i),\ i = 1,2,\ j = 1,\ldots,K_i, \tag{2.15}$$

where M_i are variational parameters $\boldsymbol{\theta}$ to be optimized via back-propagation. Again, the integral in ELBO is approximated with MC samples, $\hat{\boldsymbol{\omega}} \sim q(\boldsymbol{\omega})$, where sampling is performed via dropout on layer i with weights (M_1, M_2). The KL term $\text{KL}[q(\boldsymbol{\omega})||p(\boldsymbol{\omega})]$ is approximated by $\lambda \sum_{i=1}^{2} ||W_i||_2^2 + ||\mathbf{b}_i||_2^2$ with some weight decay λ (Gal and Ghahramani, 2015b), which results the dropout minimization objective as:

$$L_{VI}^{\text{dropout}} = -\frac{1}{N}\sum_{n=1}^{N} \log p(\mathbf{y}_n|\mathbf{x}_n, \hat{\boldsymbol{\omega}}_t) + \lambda \sum_{i=1}^{2} ||W_i||_2^2 + ||\mathbf{b}_i||_2^2 \tag{2.16}$$

Finally, we approximate the predictive posterior using MC integration, as in (2.6), by performing dropout to sample from the variational distribution.[5]

[5] When dropout is used for a stochastic regularisation technique, dropout is performed only during training and the dropout is removed.

Chapter 3

Bayesian Active Learning

In this chapter, we study the theory of Bayesian active learning for deep learning. The structure is as follows. In section 3.1, we review various acquisition functions that estimate the utility of candidate data points. In section 3.2, we focus on BALD, showing its limitation when it comes to batch selection. In section 3.3 and 3.4, we discuss two most recent state of the art Bayesian batch active learning algorithms: BatchBALD and ACS-FW. Eventually, we will see that both BatchBALD and ACS-FW have a close relationship with BALD.

3.1 Active Learning Foundations

Active learning (MacKay, 1992; Cohn et al., 1996), which is also known as "optimal experimental design" in the statistics literature (Chaloner and Verdinelli, 1995), is a well established technique for attaining data efficiency by intelligently selecting data to train model.

In the active learning framework, the key hypothesis is a model or system learns from a small subset of data so that it reduces both labeling and computational cost, aiming the deployment of machine learning system economical, and ultimately queries *optimal* data points. The model iteratively queries data with interaction with *oracle* (often human annotator), where the oracle asks to seek the most informative data.

The active learning framework consists of the current labeled training set $D_0 = \{\mathbf{x}_n, y_n\}_{n=1}^N$, the unlabeled dataset $D^{\text{Pool}} = \{\mathbf{x}_m\}_{m=1}^M$, and a Bayesian discriminative model $p(y|\mathbf{x}, \boldsymbol{\omega})$ parametrized by some model parameter $\boldsymbol{\omega} \sim p(\boldsymbol{\omega}|D_0)$. Note that Bayesian inference is made

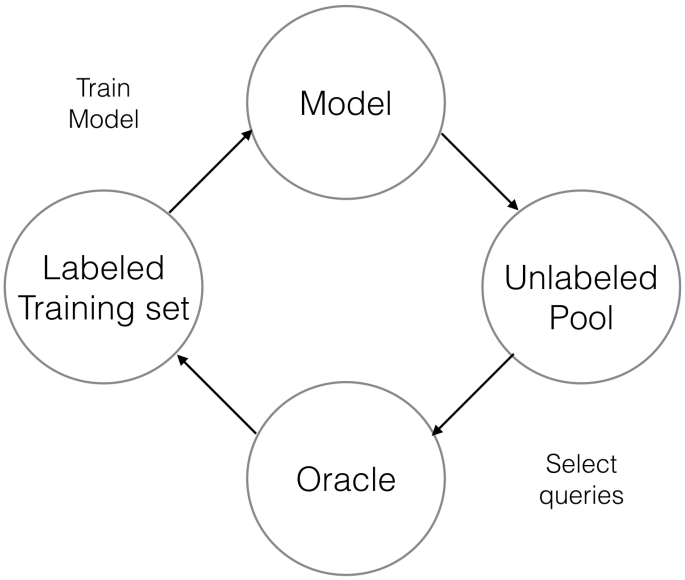


Figure 3.1: The active learning cycle. A model is initially learned on a small number of points in the labeled training set D_0. Queries are then selectively drawn from the unlabeled pool and the learner requests labels to the oracle. The labeled instances are then combined with the existed labeled training set and the model parameters are updated on the newly updated dataset.

over the parameter $\boldsymbol{\omega}$ from the training set D_0 in order to obtain the posterior. We assume that from the unlabeled data set D^{Pool} the oracle is able to provide us correct labels $\{y_m\}_{m=1}^{M}$ for the corresponding data $\mathbf{x} \in D^{\text{Pool}}$. After the selected data $\mathbf{x}^*$ moves from D^{Pool} to D_0, the model parameter $\boldsymbol{\omega}$ is updated accordingly. This framework is referred to as *pool based sampling* (Lewis and Gale, 1994)

A natural question at this point would then be - *"How to evaluate the informativeness of points in unlabelled pool?"* There have been several heuristics on the way to query using uncertainty of probabilistic models. Formally, a point $\mathbf{x}^*$ is chosen for which maximizes an acquisition function:

$$\mathbf{x}^* = \underset{\mathbf{x} \in D^{\text{Pool}}}{\arg\max}\, \alpha(\mathbf{x}, p(\boldsymbol{\omega}|D_0)) \tag{3.1}$$

We will now introduce several acquisition functions:

1. Maximize the variation ratios (Freeman, 1965)

$$\alpha_{\text{var-ratios}}(\mathbf{x}, p(\boldsymbol{\omega}|D_0)) \coloneqq 1 - \max_y p(y|\mathbf{x}, D_0) \tag{3.2}$$

This measures confidence of prediction of model, yet this does not take the remaining label distribution's label into account. In order to consider more information about probable labels, Scheffer et al. (2001) proposes:

2. Margin sampling

$$\alpha_{margin-sampling}(\mathbf{x}, p(\boldsymbol{\omega}|D_0)) \coloneqq -(p(\hat{y}_1|\mathbf{x}, D_0) - p(\hat{y}_2|\mathbf{x}, D_0)), \tag{3.3}$$

where $\hat{y}_1$ and $\hat{y}_2$ are the first and second most probable class label model prediction. Intuitively this exploits multi-class uncertainty so that it reduces the ambiguity of a point close to margin. However, it is still hard to discriminate between points close to margins if number of classes we have is large, due to the ignorance of information for remaining labels. In this context, the most popular strategy would be using information theoretic measure which is:

3. Shannon's entropy (Shannon, 1948)

$$\alpha_{max-entropy} \coloneqq -\sum_c p(y = c|\mathbf{x}, D_0) \log p(y = c|\mathbf{x}, D_0) \tag{3.4}$$

$$\coloneqq \mathbb{H}[y|\mathbf{x}, D_0], \tag{3.5}$$

which measures how much information is contained in the predictive distribution. In other words, it is a measure of uncertainty which attain its maximum value when the predictive probability for all classes are identical. As a result, selecting a point $\mathbf{x}^*$ that has maximum entropy is equivalent to selecting the point for which model is the most uncertain for its prediction.

This query strategy framework is often called as *uncertainty sampling* (Lewis and Gale, 1994). Seung et al. (1992) proposes another query strategy framework called *query-by-committee* (QBC), in which a committee of models $C = \{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_J\}$is trained on the current labeled

dataset D_0, and a point $\mathbf{x}^*$ is chosen according to *the principle of maximal disagreement*. This disagreement among a committee is a valid estimate of the uncertainty (Seung et al., 1992), which requires a committee of models and the measure of disagreement among the committee. We will now introduce and review two query approaches that can be interpreted through the QBC framework.

3.2 Bayesian Active Learning by Disagreement (BALD)

3.2.1 BALD construction

Bayesian Active Learning by Disagreement (BALD) proposed by Houlsby et al. (2011) is an information theoretic acquisition function which expresses information gain as a mutual information between predictions and model posterior. BALD acquisition function $a_{BALD}(\mathbf{x}_i, p(\boldsymbol{\omega}|D_0))$ is defined as:

$$\mathbb{I}[y_i, \boldsymbol{\omega}|\mathbf{x}_i, D_0] = \sum_{y_i} \int_{\boldsymbol{\omega}} p(y_i, \boldsymbol{\omega}|\mathbf{x}_i, D_0) \log \frac{p(y_i, \boldsymbol{\omega}|\mathbf{x}_i, D_0)}{p(y_i|\mathbf{x}_i, D_0)p(\boldsymbol{\omega}|\mathbf{x}_i, D_0)} \tag{3.6}$$

$$= \mathbb{H}[\boldsymbol{\omega}|D_0] - E_{p(y_i|\mathbf{x}_i, D_0)}[\mathbb{H}[\boldsymbol{\omega}|y_i, \mathbf{x}_i, D_0]] \tag{3.7}$$

$$= \mathbb{H}[y_i|\mathbf{x}_i, D_0] - \mathbb{E}_{p(\boldsymbol{\omega}|D_0)}[\mathbb{H}[y_i|\boldsymbol{\omega}, \mathbf{x}_i]] \tag{3.8}$$

Depending on how joint distribution of y_i and $\boldsymbol{\omega}$ in mutual information is arranged and conditioned, the objective can be expanded and interpreted in two ways.

First, we focus on the equation (3.7) which consists of two terms: predictive entropy (Lindley, 1956; Bernardo, 1979; Cover, 1999) which measures the uncertainty about the posterior of parameters and expected posterior entropy. As a new point is observed, we hope the updated posterior entropy is decreased. Hence, by choosing a data point $\mathbf{x}_i$ that maximises the difference between the two, we expect the decrease in expected uncertainty of posterior model parameters is maximized. Although this greedy sequential decision making policy has near-optimal performance (Golovin and Krause, 2011), the form of (3.7) is intractable, especially in Bayesian neural architectures.

Thanks to the symmetry of mutual information, the form of equation (3.8) is tractable by hav-

ing entropy in output space, and plus gives us another intuition. Note that this is maximized when $\mathbb{H}[y_i|\mathbf{x}_i, D_0]$ is high and $E_{p(\boldsymbol{\omega}|D_0)}[\mathbb{H}[y_i|\boldsymbol{\omega}, \mathbf{x}_i]]$ is low. That is, we want a data $\mathbf{x}_i$ that has the highest marginal uncertainty about y_i, but the mean of prediction uncertainty produced by individual model parameters is low. As a result, we measure information gain through disagreement between the predicted outcomes under the posterior parameters, and this naturally leads to i) construct a committee of models by sampling from the model posterior $p(\boldsymbol{\omega}|D_0)$ and ii) a measure of disagreement among committee members.

The tractable form of BALD is achieved by approximating it by variational distribution $q^*_\theta(\boldsymbol{\omega})$ (Gal et al., 2017). We first approximate the predictive entropy as follows:

$$\begin{aligned}
\mathbb{H}[y_i|\mathbf{x}_i, D_0] &= -\sum_c p(y=c|\mathbf{x}_i, D_0) \log p(y_i=c|\mathbf{x}_i, D_0) \\
&= -\sum_c \int_{\boldsymbol{\omega}} p(y_i=c|\mathbf{x}_i, \boldsymbol{\omega}) p(\boldsymbol{\omega}|D_0) \log \left(\int_{\boldsymbol{\omega}} p(y=c|\mathbf{x}_i, \boldsymbol{\omega}) p(\boldsymbol{\omega}|D_0) \right) \\
&\approx -\sum_c \int_{\boldsymbol{\omega}} p(y_i=c|\mathbf{x}_i, \boldsymbol{\omega}) q^*_\theta(\boldsymbol{\omega}) \log \left(\int_{\boldsymbol{\omega}} p(y_i=c|\mathbf{x}_i, \boldsymbol{\omega}) q^*_\theta(\boldsymbol{\omega}) \right) \\
&\approx -\sum_c \left(\frac{1}{T} \sum_{t=1}^{T} p(y_i=c|\mathbf{x}_i, \hat{\boldsymbol{\omega}}_t) \right) \log \left(\frac{1}{T} \sum_{t=1}^{T} p(y_i=c|\mathbf{x}_i, \hat{\boldsymbol{\omega}}_t) \right) \\
&:= \hat{\mathbb{H}}[y_i|\mathbf{x}_i, D_0],
\end{aligned} \tag{3.9}$$

with $\hat{\boldsymbol{\omega}}_t \sim q^*_{\boldsymbol{\theta}}(\boldsymbol{\omega})$ for $t = 1, \ldots, T$ is a MC samples from T number of stochastic forward passes through the model. Similarly,

$$\begin{aligned}
\mathbb{E}_{p(\boldsymbol{\omega}|D_0)}[\mathbb{H}[y_i|\boldsymbol{\omega}, \mathbf{x}_i]] &= -\mathbb{E}_{p(\boldsymbol{\omega}|D_0)} \left[\sum_c p(y_i=c|\mathbf{x}, \boldsymbol{\omega}) \log p(y_i=c|\mathbf{x}, \boldsymbol{\omega}) \right] \\
&\approx -\mathbb{E}_{q^*_{\boldsymbol{\theta}}(\boldsymbol{\omega})} \left[\sum_c p(y_i=c|\mathbf{x}, \boldsymbol{\omega}) \log p(y_i=c|\mathbf{x}, \boldsymbol{\omega}) \right] \\
&\approx -\frac{1}{T} \sum_{c,t} p(y_i=c|\mathbf{x}_i, \hat{\boldsymbol{\omega}}_t) \log p(y_i=c|\mathbf{x}_i, \hat{\boldsymbol{\omega}}_t)
\end{aligned} \tag{3.10}$$

Finally, combining (3.9) and (3.10) gives us a computationally tractable estimator $\hat{\mathbb{I}}_{BALD}[y_i, \boldsymbol{\omega}|\mathbf{x}_i, D_0]$.

(a) BALD **(b)** BatchBALD **(c)** ACS-FW

Figure 3.2: t-SNE projection (Maaten and Hinton, 2008) of the acquired batch from different active learning methods. Neural linear model, which we will define in chapter 4, is used on MNIST dataset. Given 88% test accuracy, 75 points are acquired, represented as black dots.

3.2.2 BALD in Batch Active Learning Setting

So far, we reviewed an acquisition function that the active learner uses the uncertainty of the probabilistic model and the principle of maximal disagreement. In particular, these strategies decide which *single* data point is most informative and thus to be labeled. However, retraining the model every time after querying a single data point is infeasible, especially for deep neural networks with millions of parameters. Instead, we choose a *batch* of points to train a model.[1]

Naive approach of extending BALD to batch setting would be to acquire a batch of data points sorting by top B BALD scores, as in Gal et al. (2017). That is, a naive batch BALD score is a summation of individual BALD scores i.e.

$$a_{Naive-BatchBALD}(\{\mathbf{x}_1,\ldots,\mathbf{x}_B\}, p(\boldsymbol{\omega}|D_0)) = \sum_{i=1}^{B} \mathbb{I}[y_i, \boldsymbol{\omega}|\mathbf{x}_i, D_0], \tag{3.11}$$

then we greedily select a batch of points from this score.

The problem of this approach is that some of the selected examples may be similar and close each other, and thus they are not *jointly* informative. In other words, this naive batch construction with the BALD score may lead to highly correlated queries (Sener and Savarese, 2017;

[1] This is often refer to as *batch mode active learning* (Hoi et al., 2006a,b; Guo and Schuurmans, 2008)

Pinsler et al., 2019; Kirsch et al., 2019). Furthermore, Sener and Savarese (2017) empirically shows that classical active learning algorithms are not effective when they are applied to the convolutional neural networks (CNNs) (Rumelhart et al., 1985; LeCun et al., 1989) in batch active learning settings due to correlation from batch sampling. In Figure 3.2, we visualise the acquired batch by t-SNE projection (Maaten and Hinton, 2008). In t-SNE space, it is clear to see that BALD is prone to correlated queries, whereas BatchBALD and ACS-FW (which will discuss in section 3.3 and 3.4) avoid correlated queries.

In fact, Kirsch et al. (2019) shows that selecting a batch of B points according to

$$\{\mathbf{x}_1^*, \dots \mathbf{x}_B^*\} \coloneqq \underset{\{\mathbf{x}_1,\dots,\mathbf{x}_B\}\subseteq D_{\text{Pool}}}{\arg\max} a_{Naive-BatchBALD}(\{\mathbf{x}_1,\dots,\mathbf{x}_B\}, p(\boldsymbol{\omega}|D_0)) \tag{3.12}$$

over estimates the joint mutual information between *batch* of points and model posteriors due to double-counting the information overlap. This result is building upon the work from Yeung (1991) which proves the following:

$$\mathbb{H}(x,y) = \mu^*(x \cup y),\ \mathbb{I}(x,y) = \mu^*(x \cap y),\ \mathbb{E}_{p(y)}[\mathbb{H}(x|y)] = \mu^*(x \setminus y) \tag{3.13}$$

with information measure μ^*. In this perspective, $a_{Naive-BatchBALD}$ is a sum of individual information intersections intersections $\sum_b^B \mu^*(y_b \cap \boldsymbol{\omega})$, and thus the overlapped information $\{\mu^*(y_i \cap y_j \cap \boldsymbol{\omega})\}_{i\neq j}^B$ is counted multiple times.

Active learning in the batch setting requires careful consideration such that all the points should be 1) jointly informative, 2) dissimilar to the other selected points, and 3) contains unique information (Hoi et al., 2006a,b). There have been various batch active learning methods taking information in a large number of unlabeled instances into account (Li and Guo, 2013; Guo and Schuurmans, 2008; Zhdanov, 2019; Azimi et al., 2012; Sener and Savarese, 2017; Kirsch et al., 2019; Pinsler et al., 2019). Among them, we specifically consider probabilistic methods that scalable to large neural networks: BatchBALD and Sparse Subset Approximation.

3.3 BatchBALD

3.3.1 BatchBALD construction

In order to avoid the information overlap, Kirsch et al. (2019) proposes an extension of BALD acquisition function that approximates mutual information between a joint of multiple data points and model posteriors. BatchBALD acqusition function $a_{BatchBALD}(\{\mathbf{x}_{1:B}\}, p(\boldsymbol{\omega}|D_0))$ is defined similarly to (3.8) but treat a batch of points $\{\mathbf{x}_1, \ldots, \mathbf{x}_B\}$ as a joint random variable $\mathbf{x}_{1:B}$, e.g.

$$\mathbb{I}(y_{1:B}, \boldsymbol{\omega}|\mathbf{x}_{1:B}, D_0) = \mathbb{H}(y_{1:B}|\mathbf{x}_{1:B}, D_0) - \mathbb{E}_{p(\boldsymbol{\omega}|D_0)}\mathbb{H}(y_{1:B}|\mathbf{x}_{1:B}, \boldsymbol{\omega}, D_0). \tag{3.14}$$

Given $\boldsymbol{\omega}$, y_is are independent and thus the conditional joint entropy is decomposed and further can be approximated with variational posterior distribution $q^*_{\boldsymbol{\theta}}(\boldsymbol{\omega})$[2]:

$$\begin{aligned}
\mathbb{E}_{p(\boldsymbol{\omega})}\mathbb{H}[y_{1:B}|\boldsymbol{\omega}] &= \sum_{i=1}^{B} \mathbb{E}_{p(\boldsymbol{\omega})}[\mathbb{H}(y_i|\boldsymbol{\omega})] && (3.15)\\
&= -\sum_{i=1}^{B} \mathbb{E}_{p(\boldsymbol{\omega})}\left[\sum_c p(y_i = c|\boldsymbol{\omega}) \log p(y_i = c|\boldsymbol{\omega})\right] && (3.16)\\
&\approx -\sum_{i=1}^{B} \mathbb{E}_{q^*_{\boldsymbol{\theta}}(\boldsymbol{\omega})}\left[\sum_c p(y_i = c|\boldsymbol{\omega}) \log p(y_i = c|\boldsymbol{\omega})\right] && (3.17)\\
&\approx -\frac{1}{T}\sum_{i=1}^{B}\sum_{c,t} p(y_i = c|\hat{\boldsymbol{\omega}}_t) \log p(y_i = c|\hat{\boldsymbol{\omega}}_t) && (3.18)
\end{aligned}$$

with $\hat{\boldsymbol{\omega}}_t \sim q^*_{\boldsymbol{\theta}}(\boldsymbol{\omega})$. When it comes to computing the joint entropy, the marginal y_is deos not factorises. The equality $p(y_{1:B}) = \mathbb{E}_{p(\boldsymbol{\omega}|D_0)}[p(y_{1:B}|\boldsymbol{\omega}]$ is employed so that the joint entropy is achieved by averaging over all possible configurations $\hat{y}_{1:B}$ of $y_{1:B}$ with respect to MC samples $\hat{\boldsymbol{\omega}} \sim p(\boldsymbol{\omega})$:

$$\begin{aligned}
\mathbb{H}[y_{1:B}] &= \mathbb{E}_{p(y_{1:B})}[-\log p(y_{1:B})] && (3.19)\\
&= \mathbb{E}_{p(\boldsymbol{\omega})}\left[\mathbb{E}_{p(y_{1:B}|\boldsymbol{\omega})}[-\log \mathbb{E}_{p(\boldsymbol{\omega})}[p(y_{1:B}|\boldsymbol{\omega})]]\right] && (3.20)\\
&\approx -\sum_{\hat{y}_{1:B}} \left(\frac{1}{T}\sum_t p(\hat{y}_{1:B}|\hat{\boldsymbol{\omega}}_t)\right) \log\left(\frac{1}{T}\sum_t p(\hat{y}_{1:B}|\hat{\boldsymbol{\omega}}_t)\right). && (3.21)
\end{aligned}$$

[2] $\mathbf{x}_{1:B}$ and D_0 is left out for brevity

We now have a tractable form of mutual information between $y_{1:B}$ and $\boldsymbol{\omega}$. However, finding a batch $(\mathbf{x}_{1:B}, y_{1:B})$ maximizing the BatchBALD score is intractable due to the combinatorial explosion; it requires considering all possible subsets of the pool set. Kirsch et al. (2019) proposes myopic greedy algorithm, which is proven as a $1-\frac{1}{e}$-approximate (Krause et al., 2008; Nemhauser et al., 1978) by showing that BatchBALD acquisition function is submodular.

Algorithm 1 Greedy BatchBALD (Kirsch et al., 2019)

Given acquisition size B, unlabeled dataset D^{Pool}, model posterior $p(\boldsymbol{\omega}|D_0)$
$A_0 \leftarrow \emptyset$
for $b \in 1, \dots, B$ **do**
  foreach $\mathbf{x} \in D^{\text{Pool}} \backslash A_{b-1}$ **do** $s_{\mathbf{x}} \leftarrow a_{\text{BatchBALD}}(A_{b-1} \cup \{\mathbf{x}\}, p(\boldsymbol{\omega}|D_0))$
  $x_b \leftarrow \arg\max_{\mathbf{x} \in D^{\text{Pool}} \backslash A_{b-1}} s_{\mathbf{x}}$
  $A_b \leftarrow A_{b-1} \cup \{\mathbf{x}_b\}$
end for
return acquisition batch $\{\mathbf{x}_1, \dots, \mathbf{x}_b\}$

As implemented by Kirsch et al. (2019), the benefit of this greedy algorithm is a batch matrix multiplication, benefited by the factorization $p(y_{1:b}|\boldsymbol{\omega}) = p(y_{1:b-1}|\boldsymbol{\omega})p(y_b|\boldsymbol{\omega})$. Both $p(y_{1:b-1}|\boldsymbol{\omega})$ and $p(y_b|\boldsymbol{\omega})$ can be represented as matrix $\hat{P}_{1:b-1}$ of shape $c^{b-1} \times T$ and $\hat{P}_b$ of shape $c \times T$, where each rows contain all possible configurations of $y_{1:b}$ and each columns contain different MC samples. c is number of classes. By noting that $\hat{P}_{1:b} = \hat{P}_{1:b-1}\hat{P}_b^T$, $p(\hat{y}_{1:b}|\hat{\boldsymbol{\omega}})$ can be recursively computed. Hence, the greedy BatchBALD algorithm has the time complexity $O(c^B \cdot |D^{\text{Pool}}|^B \cdot k)$

Since the batch matrix multiplication is still expensive even when the batch size is more than 5, Kirsch et al. (2019) perform importance sampling for estimating BatchBALD. Specifically, Kirsch et al. (2019) assumes that we have m samples $\hat{y}_{1:n-1}$ from the possible configurations $y_{1:n-1}$, and approximate the joint entropy as:

$$\mathbb{H}[y_{1:B}] = -\frac{1}{m} \sum_{\hat{y}_{1:b-1}}^{m} \sum_{\hat{y}_b} \frac{\hat{P}_{1:b-1}\hat{P}_b^T}{\hat{P}_{1:b-1}\mathbb{1}_{T\times 1}} \log\left(\frac{1}{T}\hat{P}_{1:b-1}\hat{P}_b^T.\right) \tag{3.22}$$

Now, $\hat{P}_{1:b-1}$ has a shape $m \times T$ where each rows contain sampled configurations and each columns contain different MC samples. Through the sampling, the time complexity is now reduced to $O(Bc \cdot \min\{c^B, m\} \cdot |D^{\text{Pool}}| \cdot T)$. In practice, the exact entropy will be computed for

the first 4 data points and the sampling will be performed afterwards, as in Kirsch et al. (2019).

3.3.2 Relationship between BALD and BatchBALD

With the information measure μ^* previously defined in equation (3.13), recall that BALD computes $\sum_b^B \mu^*(y_b \cap \boldsymbol{\omega})$. Kirsch et al. (2019) shows that BatchBALD computes $\mu^*(\cup_i y_i \cap \boldsymbol{\omega})$, which gives us intuition that $a_{BatchBALD} \leq a_{BALD}$.[3] From this, we see that BatchBALD takes information overlap into account unlike BALD. Furthermore, Kirsch et al. (2019) shows that the performance of the greedy BatchBALD algorithm 1 is upper bounded by BALD with acquisition size 1 by showing the following relationship:

$$a_{BALD}(\{\mathbf{x}\}, p(\boldsymbol{\omega}|D_0 \cup \{(\mathbf{x}_1, \tilde{y}_1), \ldots, (\mathbf{x}_{b-1}, \tilde{y}_{b-1})\}) \leq a_{BatchBALD}(A_{b-1} \cup \{\mathbf{x}\}, p(\boldsymbol{\omega}|D_0)), \tag{3.23}$$

where $\tilde{y}_1, \ldots, \tilde{y}_{b-1}$ are the true labels of $\mathbf{x}_1, \ldots, \mathbf{x}_{b-1}$. The right-hand side is the BatchBALD score at b^{th} step of the greedy algorithm in 1, and the left-hand side is the BALD score where the mutual information is computed with the true labels of the chosen data points. Then, according to Kirsch et al. (2019), as the uncertainty on $\boldsymbol{\omega}$ is reduced by the expanded training set, the mutual information decreases, and thus information overlap between y and $\boldsymbol{\omega}$ decreases. Therefore, BALD with batch size 1 is an upper-bound for BatchBALD performance. This Kirsch et al. (2019)'s analogy based on information-measure theory tells us that the performance of BALD and BatchBALD has the following ranking: naive BALD $<$ BatchBALD $<$ BALD with batch size 1.

3.4 Bayesian Batch Active Learning as Sparse Subset Approximation

Recall that, from (3.7), the BALD objective was to maximize *the decrease in expected posterior entropy*, i.e.

$$a_{BALD}(\mathbf{x}_i, p(\boldsymbol{\omega}|D_0)) = \mathbb{H}[\boldsymbol{\omega}|D_0] - E_{p(y_i|\mathbf{x}_i, D_0)}[\mathbb{H}[\boldsymbol{\omega}|y_i, \mathbf{x}_i, D_0]], \tag{3.24}$$

and uncertainty sampling through BALD acquisition function is a myopic greedy approximation by iterative choice a single point. The more principle way to minimize the uncertainty of

[3] The proof is given in (Kirsch et al., 2019).

the model on labels would be to minimize the uncertainty about the parameters directly using the entropy (MacKay, 1992). That is, we may query a batch of points D^* that minimizes the *expected* posterior entropy, i.e.

$$D^* = \underset{D \subseteq D^{\text{Pool}}}{\arg\min}\, \mathbb{E}_{Y^{\text{Pool}} \sim p(Y^{\text{Pool}}|X^{\text{Pool}}, D_0)}[\mathbb{H}[\boldsymbol{\omega}|D_0, D^{\text{Pool}}]], \tag{3.25}$$

where $X^{\text{Pool}} = \{\mathbf{x}_i\}_{i=1}^N$ and $Y^{\text{Pool}} = \{\mathbf{y}_i\}_{i=1}^N$ is a set of N number of points and labels in the pool set respectively. However, in general, solution to this problem is hard to attain since it is NP-hard in general. Instead of consider all possible subsets $D \subseteq D^{\text{Pool}}$ to minimize the expected predictive posterior entropy, Pinsler et al. (2019) proposes a novel approach that finds a subset D' such that the updated log posterior $\log p(\boldsymbol{\theta}|D_0 \cup D')$ best approximates the complete data log posterior $\log p(\boldsymbol{\theta}|D_0 \cup D^{\text{Pool}})$. The framework that *best approximates* the complete log posterior is inspired from Campbell and Broderick (2019)'s work which reformulates the Bayesian coreset (Huggins et al., 2016) as sparse subset approximation. In this subsection, we will introduce the background of Bayesian coresets (Huggins et al., 2016) and its reformulation to the Hilbert coreset Campbell and Broderick (2019) with its induced norm on the log-likelihood function space. Finally, we conclude this subsection by explaining how they can be applied to the batch active learning framework Pinsler et al. (2019).

3.4.1 Bayesian Coresets

Bayesian coreset (Huggins et al., 2016) aims to construct an *weighted* subset of the original dataset whose log likelihood best approximates the full log likelihood. By noting that the full log likelihood $L(\boldsymbol{\theta})$ can be written as a sum of individuals, i.e. $L(\boldsymbol{\theta}) = \sum_{i=1}^N L_i(\boldsymbol{\theta}) = \sum_{i=1}^N \log p(y_i|\mathbf{x}, \boldsymbol{\theta})$ given the parameter $\boldsymbol{\theta} \in \Theta$, the weighted log likelihood $L(\mathbf{w}, \boldsymbol{\theta}) := \sum_{b=1}^B w_b L_b(\boldsymbol{\theta})$ with weights $\mathbf{w} := (w_b)_{b=1}^B$ aims to satisfy

$$|L(\mathbf{w}, \boldsymbol{\theta}) - L(\boldsymbol{\theta})| \leq \varepsilon |L(\boldsymbol{\theta})|,\ \forall \boldsymbol{\theta} \in \Theta,\ B \ll N \tag{3.26}$$

Huggins et al. (2016) proposes an algorithm that constructs a Bayesian coreset satisfies (3.26) with high probability with some random weights. First begin with computing the *sensitivity* σ_n

of each data point,

$$\sigma_i := \sup_{\boldsymbol{\theta}\in\Theta} \left| \frac{L_i(\boldsymbol{\theta})}{L(\boldsymbol{\theta})} \right| \tag{3.27}$$

with $\sigma = \sum_{i=1}^{N} \sigma_i$, we will subsample the dataset by randomly generates the weights $W_b = \frac{\sigma}{\sigma_b}\frac{M_b}{B}$. M_b is sampled from the B number of independent draws with the probability $\frac{\sigma_i}{\sigma}$, i.e.

$$(M_1, M_2, \ldots, M_N) \sim \text{Multi}\left(B, \left(\frac{\sigma_i}{\sigma}\right)_{i=1}^{N}\right). \tag{3.28}$$

With the Bayesian coreset constructed by this algorithm (i.e. $\tilde{D} = (w_b, \mathbf{x}_b, y_b)_{b=1}^{B}$), we can expect $L(W, \boldsymbol{\theta})$ converges to $L(\boldsymbol{\theta})$ as B grows from the fact that

$$\mathbb{E}[L(\mathbf{w}, \boldsymbol{\theta})] = \sum_{b=1}^{B} \mathbb{E}[w_b] L_b(\boldsymbol{\theta}) = L(\boldsymbol{\theta}). \tag{3.29}$$

3.4.2 Hilbert Bayesian Coresets as Sparse Vector Sum Approximation

Campbell and Broderick (2019) reformulates this Bayesian coreset construction as sparse vector sum approximation. This reformulation views $L_i : \Theta \to \mathbb{R}$ and $L = \sum_{i=1}^{N} L_i$ as a vector in a vector space of functions and reconstructs the problem of the best coreset size of M to a convex optimization with binary constraints. That is, our aim is to minimize the approximation error subject to a constraint on the number of nonzero entries in $\mathbf{w}$:

$$\min_{\mathbf{w}\in\mathbb{R}^N} ||L(\mathbf{w}) - L||^2 \quad \text{such that} \quad \mathbf{w} \geq 0, \ \sum_{i=1}^{N} \mathbb{1}[w_i > 0] \leq B \tag{3.30}$$

Then by solving the above convex optimization problem, we will find the weights $\mathbf{w}$ such that the resulting approximated posterior $L(\mathbf{w}, \boldsymbol{\theta})$ that will be *as close as* possible to the full posterior $L(\boldsymbol{\theta})$. For the notion of *closeness*, one can use the bounded uniform norm weighted by the full log-likelihood $L(\boldsymbol{\theta})$ (e.g. $||L_i|| := \sup_{\boldsymbol{\theta}\in\Theta} \left|\frac{L_i(\boldsymbol{\theta})}{L(\boldsymbol{\theta})}\right|$). However, the exact computation of similarity under this norm is generally intractable. Even though its upper bound can be used instead (Huggins et al., 2016), the lack of the *directionality* (since uniform norm does not correspond to an inner-product) limits its performance in a fundamental way (Campbell and Broderick, 2019).

In this context, (Campbell and Broderick, 2019) proposes a way to construct the coreset in a

Hilbert space induced by an inner product $\langle L_i, L_j \rangle$ using a norm corresponding to an inner product, $||\cdot||$. The notion of the directionality is important for the optimization procedure since it enables us to take the similarity between selected points account. This similarity measure is used to choose the most aligned vector $L_i(\boldsymbol{\theta})$ along with the direction of the greatest improvement. In particular, we choose the coreset points based on the residual error in the coreset approximation vector, $L - L(\mathbf{w})$.

In order to introduce directionality to the Hilbert Bayesian coreset, the cardinality constraint in (3.30) is relaxed to a polytope constraint (Campbell and Broderick, 2019):

$$\min_{\mathbf{w} \in \mathbb{R}_N} (\mathbf{w}-1)^T K (\mathbf{w}-1) \;\; \text{such that } \mathbf{w} \geq 0, \sum_{i=1}^{N} \sigma_i w_i = \sigma \tag{3.31}$$

where $K \in \mathbb{R}^{B \times B}$ is a kernel matrix with $K_{i,j} = \langle L_i, L_j \rangle$ and we used the fact that $||L - L(\mathbf{w})||^2 = (\mathbf{w}-\mathbf{1})^T (\mathbf{w}-\mathbf{1})$. Each vertices of the polytope have a single nonzero component. Furthermore, the polytope contains the optimal point $\mathbf{w} = \mathbf{1} = (1, 1, \ldots, 1)^T$ with the cost $L(\mathbf{1}) = L$ and has vertices $\{\frac{\sigma}{\sigma_i}\mathbf{1}_i\}_{i=1}^N$. Then the refined convex optimization problem in (3.31) can be solved via the Frank-Wolfe algorithm (Frank et al., 1956), which consists of three steps: given the k^{th} iterate with $\mathbf{x}_k$ for the convex optimization problem $\min_{\mathbf{x} \in D} f(\mathbf{x})$ 1) find $\mathbf{s}_k = \arg\min_{\mathbf{s} \in D} \nabla f(\mathbf{x}_k)\mathbf{s}$ for a search direction $d_k = s_k - x_k$; 2) find a step size $\gamma_k = \arg\min_{\gamma \in [0,1]} f(\mathbf{x}_k + \gamma d_k)$; 3) update $\mathbf{x}_{k+1} = \mathbf{x}_k + \gamma_k d_k$. Note that in our case, we optimize the objective in (3.31) by searching over all vertices of the polytope, and hence the weights $\mathbf{w}$ are updated along some vertex, say f^{th}, of the polytope; in other words f^{th} coordinate unit vector. In specific, the Frank-Wolfe direction is determined by the most aligned vector L_f with the residual error $L - L(\mathbf{w})$. Formally, this can be written as:

$$f = \arg\max_{n \in N} \left\langle L - L(\mathbf{w}), \frac{1}{\sigma_n} L_n \right\rangle^{4} = \arg\max_{n \in N} \frac{1}{\sigma_n} \sum_{m=1}^{N} (1 - w_m) \langle L_m, L_n \rangle.^{5} \tag{3.32}$$

After we find some vertex f, the Frank-Wolfe direction is given as $d = \frac{\sigma}{\sigma_f}\mathbf{1}_f - \mathbf{w}$, Then the line search is performed, i.e. $\mathbf{w} = \mathbf{w} + \gamma d_t$ for some $\gamma \in [0, 1]$

[4] This form is used in practice for the random projection. The detailed procedure will be discussed in 3.4.4

[5] We will not use this form in practice. This form is useful when we have access for closed form inner product $\langle L_i, L_j \rangle$

Algorithm 2 FW: Hilbert coresets via Frank–Wolfe (Campbell and Broderick, 2019)

Given $(L_n)_{n=1}^N, M, \langle \cdot, \cdot, \rangle$
$\sigma_n \leftarrow \sqrt{\langle L_n, L_n \rangle}, \forall n$
$\sigma \leftarrow \sum_n \sigma_n$
$\mathbf{w} \leftarrow 0$
for $t \in 1, \ldots, B$ **do**
  $f \leftarrow \arg\max_{n \in N} \left\langle L - L(\mathbf{w}), \frac{1}{\sigma_n} L_n \right\rangle$
  $\gamma \leftarrow \frac{\left[\left\langle \frac{\sigma}{\sigma_f} L_f - L(\mathbf{w}), L - L(\mathbf{w}) \right\rangle\right]}{\left[\left\langle \frac{\sigma}{\sigma_f} L_f - L(\mathbf{w}, \frac{\sigma}{\sigma_f} L_f - L(\mathbf{w}) \right\rangle\right]}$
  $\mathbf{w} \leftarrow (1-\gamma)\mathbf{w} + \gamma \frac{\sigma}{\sigma_f} \mathbf{1}_f$
end for
return $\mathbf{w}$

Note that the algorithm 2 for Bayesian posterior approximation assumes an arbitrary Hilbert norm, but also the notion of directionality highly relies on the choice of an inner product. Campbell and Broderick (2019) suggests that these inner products should have two desirable properties: it should be i) a good indicator of posterior discrepancy and ii) efficiently computable or approximable. Two candidates of possible inner products that satisfy desiderata for such a norm are suggested in Campbell and Broderick (2019). One is based on the Fisher information distance (Johnson and Barron, 2004) and the other is a weighted L^2 norm. But also approximations of the inner products through *random features* (Rahimi and Recht, 2008) are covered for the case when the gradients of the log-likelihood, $\nabla L(\boldsymbol{\omega})$, is difficult to compute for the fisher information distance. As deep neural networks do not have tractable log-likelihood gradients, we will use the weighted L^2 norm aided by random features.

Before discussing inner products that induce a Hilbert space with associated norm $||\cdot||$, we will first discuss how to construct Hilbert Coreset via Frank-Wolfe optimization (i.e., approximate the complete data posterior of the model parameters) *for the active learning framework*. Moreover, this approach will also be extended to more complex models such as deep Bayesian neural networks and associated variational inference procedure through random features.

3.4.3 Computing the expected log posterior

Section 3.4.1 and 3.4.2 assumes that we are given N number of points $\{\mathbf{x}_i,\}_{i=1}^N$, and all of them are labeled with $\{y_i\}_{i=1}^N$. For this reason, we could easily compute a likelihood $p(y_i|\boldsymbol{\omega},\mathbf{x}_i)$ for each observations, where our model is parameterized by $\boldsymbol{\omega} \in \Theta$ with some prior $p(\boldsymbol{\omega})$. However, in the active learning framework, recall that only the labels in the training set D_0 is given and the points in the pool dataset are assumed to be unlabeled. This motivates us to take the expectation of the log-likelihood with respect to the current predictive posterior distribution.

Our aim is to approximate the complete data posterior of the model parameters, and we want to select a subset or batch of points $D' = \left(\{\mathbf{x}'_i\}_{i=1}^B, \{y'_i\}_{i=1}^B\right)$ such that the log posterior $\log p(\boldsymbol{\omega}|D_0 \cup D^{\text{Pool}}) \subseteq D^{\text{Pool}}$ best approximates the full log posterior $\log p(\boldsymbol{\omega}|D_0 \cup D^{\text{Pool}})$ where $D^{\text{Pool}} = \left(\{\mathbf{x}_i\}_{i=1}^N, \{y_i\}_{i=1}^N\right)$. Since the true label $\{y_i\}_{i=1}^N$ in the pool data set D^{Pool} is assumed to be unknown, we essentially take expectation with respect to the predictive posterior distribution distribution $p(y_1,\dots,y_N|\mathbf{x}_1,\dots,\mathbf{x}_N,D_0) = p(y_1|\mathbf{x}_1,D_0)\dots p(y_N|\mathbf{x}_N,D_0)$, where we assume conditional independence of the outputs given the inputs. The expected full data log posterior is then given and can be expanded by

$$\mathbb{E}_{y_1,\dots,y_N}\left[\log p\left(\boldsymbol{\omega}|D_0 \cup D_{unlabeled}\right)\right] = \mathbb{E}_{y_1,\dots,y_N}\left[\log \frac{p(\boldsymbol{\omega}|D_0)\prod_{i=1}^N p(\mathbf{x}_i,y_i|D_0,\boldsymbol{\omega})}{\prod_{i=1}^N p(\mathbf{x}_i,y_i|D_0)}\right] \tag{3.33}$$

$$\propto \sum_{i=1}^N \mathbb{E}_{y_i}\left[\log p(y_i|\mathbf{x}_i,\boldsymbol{\omega})\right] - \mathbb{E}_{y_i}\left[\log p(y_i|\mathbf{x}_i,D_0)\right] \tag{3.34}$$

$$:= \sum_{i=1}^N L_i(\boldsymbol{\omega}) := L \tag{3.35}$$

Note that $\log p(\boldsymbol{\omega}|D_0)$ term is dropped since this only depends on the D_0, which does not affect to find a batch that best approximates of $\sum_i L_i(\boldsymbol{\theta})$. $L_i(\boldsymbol{\omega})$ can be written in more explicitly as follows:

$$L_i(\boldsymbol{\omega}) = \sum_c p(y_i=c|\mathbf{x}_i,D_0)\log p(y_i=c|\mathbf{x}_i,\boldsymbol{\omega}) - \underbrace{\sum_c p(y_i=c|\mathbf{x}_i,D_0)\log p(y_i|\mathbf{x}_i,D_0)}_{\mathbb{H}[y_i|\mathbf{x}_i,D_0]} \tag{3.36}$$

Note that the latter term $\mathbb{H}[y_i|\mathbf{x}_i, D_0]$ is the marginal predictive entropy as in the BALD acquisition function in section 3.2.2. The first term $\mathbb{E}_{y_i}[\log P(y_i|\mathbf{x}_i, \boldsymbol{\omega})]$ shares some similarity to the BALD's $\mathbb{E}_{p(\boldsymbol{\omega}|D_0)}[\mathbb{H}[y|\mathbf{x}, \boldsymbol{\omega}]]$, but not necessarilly the same. In fact, the BALD quantity turn out to be proportianal to squared norm[6] of L_n for the Bayesian linear regression model (Pinsler et al., 2019), which we will discuss this in later section. Since predictive entropy for Bayesian neural networks are intractable, the estimator is approximated by using the variational distribution:

$$\mathbb{H}[y_i|\mathbf{x}_i, D_0] = -\sum_c \int_{\boldsymbol{\omega}} p(y_i = c|\mathbf{x}_i, \boldsymbol{\omega}) p(\omega|D_0) \cdot \log \int_{\boldsymbol{\omega}} p(y_i = c|\mathbf{x}, \boldsymbol{\omega}) p(\boldsymbol{\theta}|D_0) \tag{3.37}$$

$$\approx -\sum_c \int_{\boldsymbol{\omega}} p(y_i = c|\mathbf{x}_i, \boldsymbol{\omega}) q^*_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \cdot \log \int_{\boldsymbol{\omega}} p(y_i = c|\mathbf{x}, \boldsymbol{\omega}) q^*_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \tag{3.38}$$

$$\approx -\sum_c \left(\frac{1}{T} \sum_{t=1}^{T} \hat{p}_c^{t(i)} \right) \log \left(\frac{1}{T} \sum_{t=1}^{T} \hat{p}_c^{t(i)} \right), \tag{3.39}$$

where $\hat{p}_c^{t(i)} = p(y_i = c|\mathbf{x}_i, \hat{\boldsymbol{\omega}}_t)$ is a MC sample from the approximate posterior $\hat{\boldsymbol{\omega}}_t \sim q^*_{\boldsymbol{\theta}}(\boldsymbol{\omega})$ as dicussed in 3.2.1. Similarly we now compute and approximate the first term into the tractable form through variation distribution:

$$\sum_c p(y_i = c|\mathbf{x}_i, D_0) \log p(y_i|\mathbf{x}_i, \boldsymbol{\omega}) = \sum_c \left(\int_{\boldsymbol{\omega}} p(y = c|\mathbf{x}, \boldsymbol{\omega}) p(\boldsymbol{\omega}|\mathbf{x}_i, y_i) \right) \log p(y_i|\mathbf{x}_i, \boldsymbol{\omega}) \tag{3.40}$$

$$\approx \sum_c \left(\int_{\boldsymbol{\omega}} p(y = c|\mathbf{x}, \boldsymbol{\omega}) q^*_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \right) \log p(y_i|\mathbf{x}_i, \boldsymbol{\omega}) \tag{3.41}$$

$$\approx \sum_c \left(\frac{1}{T} \sum_{t=1}^{T} \hat{p}_c^{t(i)} \right) \log p(y_i|\mathbf{x}_i, \boldsymbol{\omega}). \tag{3.42}$$

Finally, we combine (3.42) and (3.39) to get

$$L_i(\boldsymbol{\omega}) = \sum_c \frac{1}{T} \sum_{t=1}^{T} \hat{p}_c^{t(i)} \log p(y_i|\mathbf{x}_i, \boldsymbol{\omega}) - \sum_c \left(\frac{1}{T} \sum_{t=1}^{T} \hat{p}_c^{t(i)} \right) \log \left(\frac{1}{T} \sum_{t=1}^{T} \hat{p}_c^{t(i)} \right) \tag{3.43}$$

To sum up, the expected full data log posterior $\mathbb{E}_{y_1,\ldots,y_N}[\log p(\boldsymbol{\omega}|D_0 \cup D^{\text{Pool}}]$ is modified to $L_i(\boldsymbol{\omega})$, so that approximating the complete data posterior of the model parameters is being tractable. Recall that $L_i : \Theta \to \mathbb{R}$ is viewed as a vector in the Hilbert space induced by some

[6] We haven't defined any inner products and their associated norms in the Hilbert space. We will introduce this in the subsequent section.

inner product $\langle L_i, L_j \rangle$ with the desirable points. In the Hilbert space, we can introduce a notion of directionality so that a similarity between points can be quantified. The Bayesian coreset construction with a Hilbert space norm can be obtained through the convex optimization problem introduced in (3.31) which has a polytope constraint. In the subsequent subsection, we will describe how to compute this inner product in a tractable manner with random projection, enables us to solve large scale active learning problem with non-linear probabilistic model.

3.4.4 Random Projection

The inner product $\langle L_i, L_j \rangle$ between infinite dimensional vectors $L_i, L_j : \Theta \to \mathbb{R}$ is often intractable for complex models. Even if we can access to closed-form expressions, $O(N^2)$ computation is required when computes the Frank-Wolfe direction (3.32), where N is a number of points in the pool set. Thus, the pool size is restricted to moderate size. In this context, Campbell and Broderick (2019); Pinsler et al. (2019) suggest to approximate the inner product through random projection (Rahimi and Recht, 2008). That is, we could simply project this infinite dimensional vector to the finite vector via:

$$\hat{L}_i = \frac{1}{\sqrt{J}} \left(L_i(\hat{\boldsymbol{\omega}}_1), \ldots, L_i(\hat{\boldsymbol{\omega}}_J) \right)^T, \quad \text{where} \quad \hat{\boldsymbol{\omega}}_1, \ldots, \hat{\boldsymbol{\omega}}_J \sim q^*_{\boldsymbol{\theta}}(\boldsymbol{\omega}). \tag{3.44}$$

With this projections, we can then approximate the weighted Euclidean inner product $\mathbb{E}_{q^*_{\boldsymbol{\theta}}(\boldsymbol{\omega})}[L_i(\boldsymbol{\omega}) L_j(\boldsymbol{\omega})]$ as simple dot products between vectors

$$\langle L_i(\boldsymbol{\omega}), L_j(\boldsymbol{\omega}) \rangle_{q^*_{\boldsymbol{\theta}}(\boldsymbol{\omega})} = \mathbb{E}_{q^*_{\boldsymbol{\theta}}(\boldsymbol{\omega})}[L_i(\boldsymbol{\omega}) L_j(\boldsymbol{\omega})] \approx \hat{L}_i^T \hat{L}_j. \tag{3.45}$$

With this formulation, $O(NJ)$ time is required for calculating $\left\langle \sum_{i=1}^N \hat{L}_i - \hat{L}(\mathbf{w}), \frac{1}{\sigma_i} \hat{L}_i \right\rangle$, and thus the computational burden for constructing a batch is significantly reduce.

Pinsler et al. (2019) suggests that the squared norm of $L_i(\boldsymbol{\omega})$, $\langle L_i(\boldsymbol{\omega}), L_i(\boldsymbol{\omega}) \rangle$ can be used as a greedy acquisition function since the norm of L_i, σ_i, is related to the magnitude of reduction for solving convex optimization problem in (3.30) through the notion of sensitivity. From this perspective, our approximated norm of L_i through random projection can be written as:

$$\alpha_{ACS-FW}(\mathbf{x}; D_0 \propto \sum_{j=1}^{J} \left(\mathbb{E}_y \left[\log p(y|\mathbf{x}, \hat{\boldsymbol{\omega}}_j) \right] - \mathbb{H}\left[y|\mathbf{x}, D_0 \right] \right)^2, \ \hat{\boldsymbol{\omega}}_1, \ldots, \hat{\boldsymbol{\omega}}_J \sim q^*_{\boldsymbol{\theta}}(\boldsymbol{\omega}) \tag{3.46}$$

Note that this looks noticeably similar to BALD acquisition function. The only difference is that $p(y|\mathbf{x}, \boldsymbol{\omega}_j)$ is not averaged out with respect to $\boldsymbol{\omega}_j$ inside the log of the first term. In fact, Pinsler et al. (2019) shows that the two quantities are proportional each other, and hence equivalent under a greedy maximize with specific inner product choice and model. See Appendix A for the detail.

Chapter 4

Comparative Investigations

In this chapter, we investigate and discuss the results obtained during an empirical comparison for the performance of each active learning strategies introduced in chapter 3.

We have reviewed recently published papers that develop novel active learning algorithms for deep neural architectures. Each paper reports that their result is state of the art, yet the performed experiments are limited and not cross evaluated to each other's setting. Pinsler et al. (2019) reports that the ACS-FW outperforms other active learning strategies such as BALD and non-probabilistic batch active learning approach such as K-center (Sener and Savarese, 2017) for the *neural linear*[1] with a ResNET18 (He et al., 2016). It uses a moderately large batch size; 3000 for Fashion MNIST (Xiao et al., 2017) and SVHN (Goodfellow et al., 2013), and 5000 for CIFAR10 (Krizhevsky et al., 2009). In contrast, Kirsch et al. (2019) reports that BatchBALD outperforms BALD in three datasets: MNIST (LeCun et al., 1998), RepeatedMNIST and EMNIST (Cohen et al., 2017) for the MC dropout network. However, the experiments are done with relatively small batch sizes: 10 and 40.

In this context, practitioners and domain experts who want to utilize these recently developed active learning tools in their area may be difficult to choose. There are two main factors that need to be considered. First is a type of approximate inference. Every uncertainty-based active learning algorithms are sensitive to this since we approximate the weight posterior by variational distribution. Another is acquisition batch size since it cause correlated batch size, and hence it

[1] We will explain neural linear architecture in the subsequent section.

is important to see how well the methods decorrelates the query. This section provides an in-depth comparison of each algorithm in an identical setting. Throughout the chapter, we attempt to answer the following questions:

- Which uncertainty approximation scheme is preferable for each method, MC dropout or Neural Linear? Are the results consistent for different datasets?
- How does the acquisition batch size affects active learning performance?
- Is there a single active learning algorithm that beats every other?

This chapter is structured as follows. In section 4.1, we describe and explain active learning environments with the specific network and relevant hyperparameters we use in experiments. In section 4.1.1 and 4.1.2, we compare BALD, BatchBALD and ACS-FW on MNIST and Fashion MNIST dataset with different batch sizes. In 4.1.3, we compare the algorithms on Repeated (F)MNIST dataset as in Kirsch et al. (2019). All the implementations are in Pytorch with GPU processing.

4.1 Experiments and Results

This section aims to provide rationale evaluations of recently developed active learning algorithms on the Bayesian neural network. As explained earlier, the performance of active learning algorithms highly depends on variational inference methods (Barbano et al., 2019) and acquisition batch size. Since the performance of active learning algorithms has been evaluated in different environments, we will cross-evaluate the two algorithms, BatchBALD and ACS-FW, in the same setting by varying *inference method* and *batch size*.

In our experiment, an active learning loop is repeated until our budget is exhausted. For each iteration in the loop, we train the model from scratch using labeled dataset and query new points using the chosen active learning strategy. The chosen points are labeled and concatenated with the previous training set. After that, we train the model again with the updated training set. This loop is described in algorithm 3.

We consider random acquisition as a baseline, which chooses a point uniformly at random from the pool. But also, we will consider BALD with batch size 1 as a baseline. As reported in Kirsch

Algorithm 3 Active Learning Loop

Given training (labeled) dataset D_0, pool dataset D^{Pool}, acquisition batch size B, budget
$D_0^{\text{To-be-labeled}} = \emptyset$
model prior $p(\boldsymbol{\omega})$
num_initial_label $\leftarrow |D_0|$
while $|D_0| <$ num_initial_label+budget **do**
  Train model parameter $p(\boldsymbol{\omega}|D_0)$
  $\{\mathbf{x}_1, \ldots, \mathbf{x}_B\} \leftarrow \alpha_{\text{acq-type}}(D^{\text{Pool}}, p(\boldsymbol{\omega}|D_0), B)$
  $D_0 \leftarrow D_0 \cup \{(\mathbf{x}_i, y_i\}_{i=1}^{B}$
  $D^{\text{Pool}} \leftarrow D^{\text{Pool}} \backslash \{(\mathbf{x}_i, y_i\}_{i=1}^{B}$
end while

et al. (2019) and discussed in section 3.3.2, BatchBALD approximates BALD with acquisition size 1. That is BALD with acquisition size 1 can be seen as an upper bound for BatchBALD performance. Hence, we will see how BatchBALD and ACS-FW perform as well as BALD with acquisition size 1.

For uncertainty propagated with MC dropout to the Bayesian CNN model, we use the same network as in Gal et al. (2017); Kirsch et al. (2019). This is convolution-relu-convolution-relu-max pooling-dropout-dense-relu-dropout-dense-softmax, with 32 convolution kernels, 4x4 kernel size, 2x2 pooling, dense layer with 128 units, and dropout probabilities 0.5. When we train this MC dropout model, we use the Adam optimizer (Kingma and Ba, 2014) with learning rate 0.001 and betas $0.9/0.999$. The network is trained for 150 epochs with minibatch size 128.

Pinsler et al. (2019) uses neural linear (Riquelme et al., 2018) model for quantifying uncertainty over weights, where the Bayesian inference is used only small part of network. ResNest18 is used as a feature extractor followed by one stochastic fully connected layer. The last layer is Bayes By Backpropagation layer with Mean field approximation with local reparameterization trick. As explained earlier, fully factorized Gaussian is used for variatioanl posetrior, $q_{\boldsymbol{\theta}}(w_{jk}) = N(w_{jk}; \mu_{jk}, \sigma_{jk}^2 = \log(1+\exp(\rho))^2)$. In this layer, the variatonal parameters are initialized by $\mu_{jk} \sim N(0, 0.05)$ and $\rho_{jk} \sim N(-4, 0.05)$. This initialization is from Pinsler et al. (2019). Reason for this small variance is to enforcing small variance in the gradient estimator (Barbano et al., 2019). Number of features in the last layer is assumed to be 256 and trained 250 epochs with minibatch size 128.

For both of the networks, 100 MC samples are used for estimating test accuracy, and 100 MC samples are used for estimating BALD and BatchBALD and $L_i(\boldsymbol{\theta})$ for the expected log posterior in equation (3.35). The number of projections is set to be 10. We used officially published code by both authors [2][3]

We follow the Kirsch et al. (2019)'s experimental setting which sets acquisition batch size relatively small: 5, 10 and 40. This is due to the fact that BatchBALD is hardly scalable to large batch size. Even though sampling configuration can be used by the equation (3.22), this is still infeasible to scaling batch size more than 50. Another critical issue on BatchBALD would be the quality of the estimation with a large batch size B. This means the quality of the estimator gets worse as the B increases. For this reason, we also stick into the small batch size assumption for comparison between BALD, BatchBALD, and ACS-FW. In practice, as in Kirsch et al. (2019), the joint entropies are computed exactly for the first 4 data points and use MC sampling after that. Accordingly, the comparison contains two different variational inference methods.

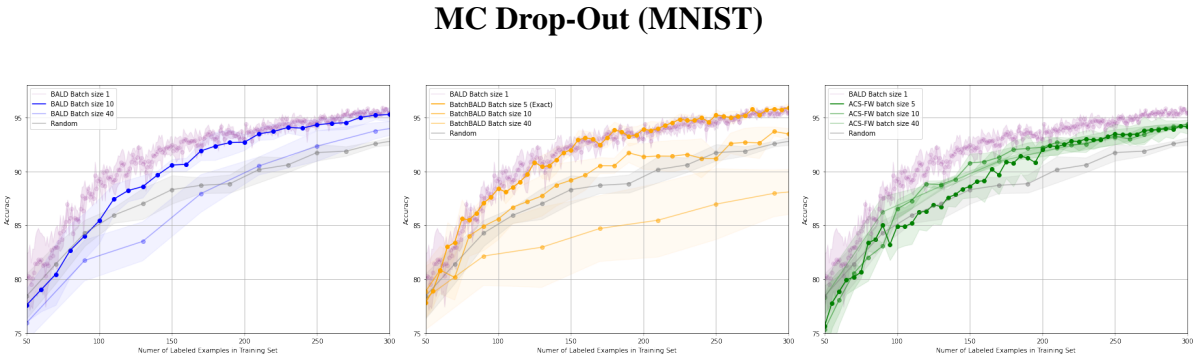
4.1.1 MNIST

We start by showing how every active learning techniques work ideally on MNIST dataset for both MC dropout and Neural linear model. The MC dropout model is trained with a random initial training set of 50 data points. Since Neural linear with ResNet 18 is relatively larger model size, it is trained with 200 initial data points. Lastly, for the MNIST dataset, our budget is assumed to be 250 points. We report and discuss the experiments in figure 4.1 and 4.2

4.1.2 Fashion MNIST

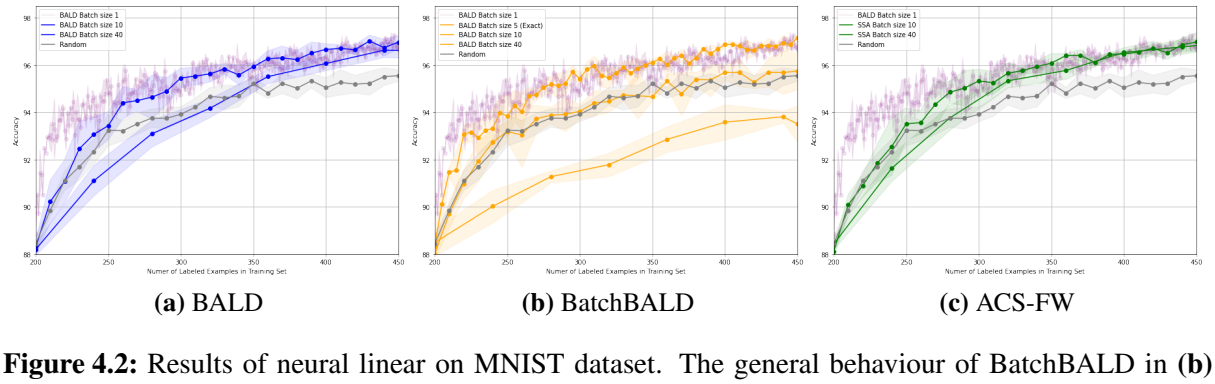
For the experiments with the MNIST dataset, it might be too easy to compare the three algorithms. Thus, we consider the more difficult Fashion MNIST dataset, which consists of 10 classes, comprising clothes and shoes. The training set consists of $60,000$ examples and a test set consists of $10,000$ examples. Each example is a 28x28 gray-scale image. For this dataset, the result we obtain in figure 4.3 and 4.4 provide us a different insight into the three methods.

[2] https://github.com/BlackHC/batchbald_redux
[3] https://github.com/rpinsler/active-bayesian-coresets



(a) BALD **(b)** BatchBALD **(c)** ACS-FW

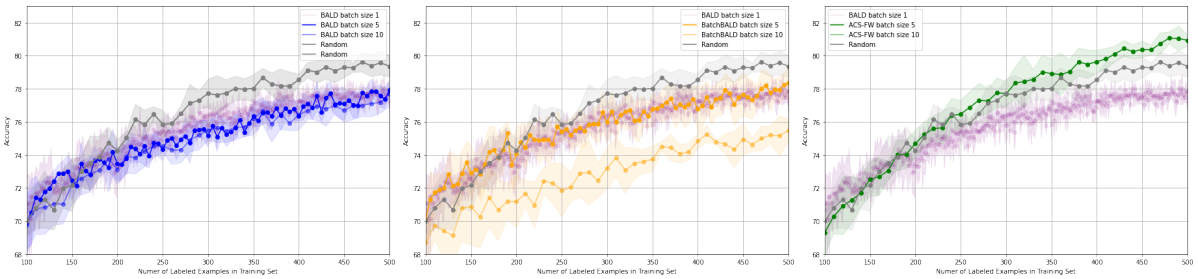
Figure 4.1: Results of MC dropout inference on MNIST dataset. **(a) BALD** There is a significant performance drop compared to batch size 1. Especially, BALD with batch size 40 performs worse than random. **(b) BatchBALD** BatchBALD defends well from the increase of batch size when the exact joint mutual information is achieved (i.e. batch size 5). When the importance sampling is performed, the level of performance drop is even larger than BALD. This is a somewhat surprising result and the different results with the experiment in Kirsch et al. (2019). We remind that we use the code officially published by the authors. **(c) ACS-FW** ACS-FW is able to maintain performance for all batch size variations, and hence least affected by correlations. However, this couldn't perform as well as BALD with batch size 1 with a significant margin for all variations.



(a) BALD **(b)** BatchBALD **(c)** ACS-FW

Figure 4.2: Results of neural linear on MNIST dataset. The general behaviour of BatchBALD in **(b)** is similar to the previous model. The difference comes from the performance of BALD and ACS-FW. BALD in **(a)** performs better than MC dropout inference. Specifically we clearly see that BALD with batch size 10 performs similarly to BatchBALD with batch size 5. Lastly, ACS-FW in **(c)** now performs as well as BALD with batch size 1 across all the variations, and is least affected by the batch size; every variation beats random.

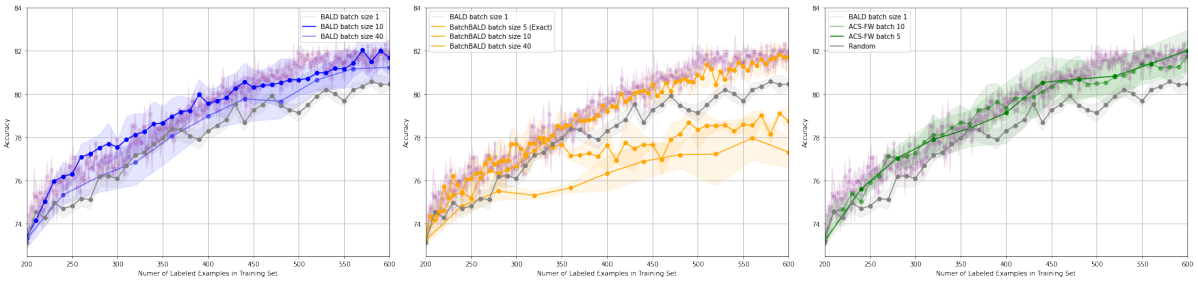
MC Drop-Out (Fashion MNIST)



(a) BALD **(b)** BatchBALD **(c)** ACS-FW

Figure 4.3: Result of MC dropout inference on Fashion MNIST and dataset. BALD and BatchBALD performed worse even compared to a random baseline. Both BALD and BatchBALD fails to avoid many duplicate data points. Since BALD with batch size 1 is upper bounded by random performance, all other methods based on BALD are worse than random. In contrast, ACS-FW solves this problem and outperforms random.

Neural Linear (Fashion MNIST)



(a) BALD **(b)** BatchBALD **(c)** ACS-FW

Figure 4.4: Result of neural linear inference on Fashion MNIST and dataset. Unlike the MC dropout model, all three active learning methods aided by neural linear outperforms random. This experiment shows that the quality of uncertainty of model parameters certainly matters for the active learning performance. There is almost no performance drop by BALD for increasing batch size in **(a)**. Even though Exact BatchBALD in **(b)** estimates perform identically to BALD with batch size 1, it fails with an increase of batch. Lastly, ACS-FW in **(c)** performs as good as BALD with batch size 1.

4.1.3 Repeated (Fashion) MNIST

As mentioned earlier, querying through the B best BALD score contains many similar data points that lead to poor performance. Therefore, in assessing active learning strategies, a critical issue is to evaluate whether the strategy is able to avoid redundant data points. In this context, Kirsch et al. (2019) replicates each data points in the MNIST training dataset two times in order

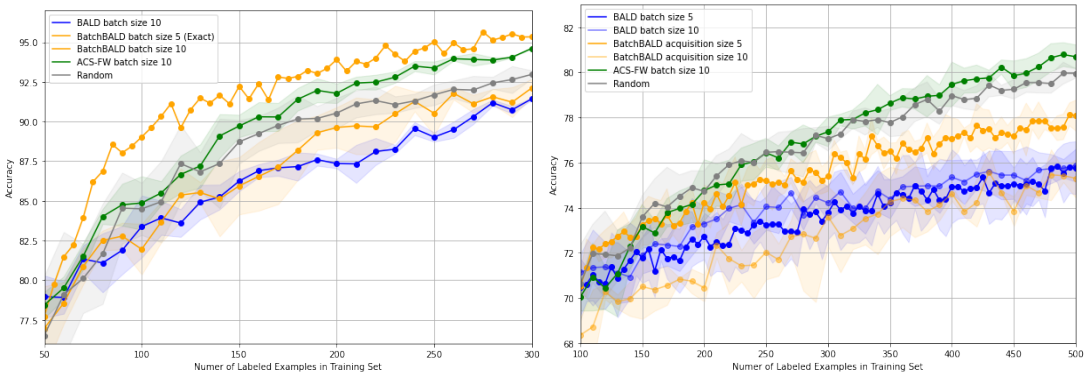
to show how this manifests in practice. Inspired by this, we assess the strategies in repeated MNIST (RMNIST) and repeated Fashion MNIST (RFMNIST), replicating each training point three times. Isotropic Gaussian noise with a standard deviation of 0.1 is added to each point. The initial RMNIST dataset for MC dropout and neural linear model is constructed with 50 and 100 initial points, and the initial RFMNIST dataset for MC dropout and Neural linear model is constructed with 200 initial points. We report and discuss the experiments in figure 4.5 and 4.6

To sum up, we clearly see that all algorithms perform better than the baseline with neural linear, while BALD and BatchBALD with MC dropout perform worse than random. That is, the uncertainty of model parameters are better-estimated with neural linear for solving active learning task. Regardless of inference methods, ACS-FW shows significant improvement compared to random when we consider the more difficult Fashion MNIST dataset. This shows the robustness of ACS-FW for inference type. Yet, the performance of ACS-FW is upper-bounded by BALD with batch size 1 for MNIST dataset.

Therefore, there is no single active learning algorithm that outperforms others in every setting (e.g. inference type, acquisition batch size, and dataset). Most importantly, each of the algorithms has its own weaknesses. BALD is easy to get correlated. BatchBALD is hardly scalable to a large batch setting. ACS-FW actually does not produce the desired number of samples during each iteration.[4] In the subsequent chapter, we will introduce simple and powerful alternatives to those previously introduced algorithms, overcoming these limitations.

[4]For this reason, the curves we reported in the figures is plotted with the quadratic interpolation. The actual performances for the actual acquired batches are reported in figure B.2.

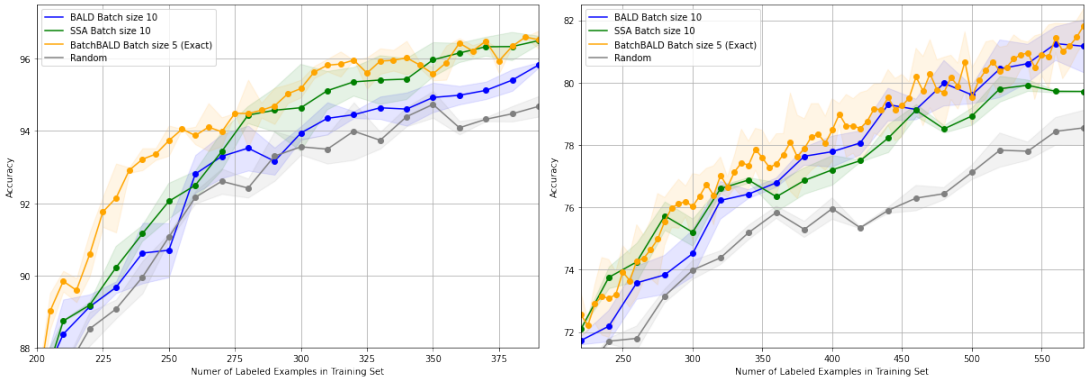
MC Drop-out



(a) Repeated MNIST **(b)** Repeated Fashion MNIST

Figure 4.5: Result of MC dropout inference on RMNIST and RFMNIST dataset. In this scenario, BALD performs poorly, worse than random across all active learning loop. Exact BatchBALD outperforms with the replication for the RMNIST dataset with a large margin, yet worse than random for the RFMNIST. In contrast, ACS-FW is able to cope with replication across both datasets, though its margin is not significant.

Neural linear



(a) Repeated MNIST **(b)** Repeated Fashion MNIST

Figure 4.6: Result of neural linear inference on RMNIST and RFMNIST dataset. All methods surprisingly outperform random baseline. Exact BatchBALD performs best in this setting for both datasets. Another surprising result we notice is the strong performance of BALD in RFMNIST dataset; BALD outperforms ACS-FW. This result shows neural linear is preferable for BALD and its variant, and ACS-FW is less affected by the uncertainty approximation scheme.

Chapter 5

BALD by Thompson Sampling

In section 3.2 and 3.3 we discussed the extension of BALD in batch setting. Gal et al. (2017) greedily selects top B BALD scores so that each individual points are highly informative. However, in chapter 4, we empirically show that they are not necessarily informative *jointly*, and thus curtail performance severely. BatchBALD (Kirsch et al., 2019) extends BALD to the joint mutual information between a batch of points $y_{1:B}$ and model parameters $\boldsymbol{\omega}$ which provide us less correlated queries by taking information overlap into account. However, it is difficult to scale BatchBALD to large batch sizes because enumerating all possible configurations is infeasible. ACS-FW (Pinsler et al., 2019) avoids correlated queries by constructing the Hilbert coreset via Frank-Wolfe optimization. Yet, this algorithm often returns a batch smaller than the batch size B that we desire.

A correlated query implies that the query contains many near replicas for each data point, which is equivalent to say that it does not actively *explore*. The lack of exploration caused by the greedy approach is a well-known shortcoming when it comes to the Bernoulli Bandit problem in the reinforcement learning community (Russo et al., 2017). For example, suppose that we can select three actions a_k for $k = 1, 2, 3$. Each time we select one action, some reward $r \in \mathbb{R}$ is returned, where the mean of the reward corresponding to each action is assumed to be unknown. Instead, we have an access to *beleifs* of these expected rewards, which can be expressed as a posterior distribution $p(R_k = r_k|\boldsymbol{\theta}, D)$, parametrized by $\boldsymbol{\theta}$ and D represents history of the rewards of each actions.

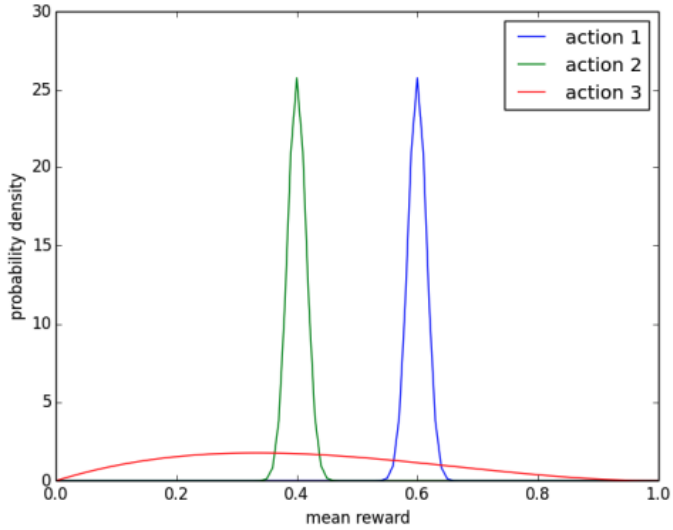


Figure 5.1: Probability density functions over mean rewards. (Russo et al., 2017)

Suppose we have a posterior rewards distributions for every three actions as illustrated in figure 5.1. Then, action 1 seems to be appropriate to take since its expected reward is the highest among the three. It is rationale to avoid action 2 since it is unlikely that $R_2|\boldsymbol{\theta},D > R_1|\boldsymbol{\theta},D$. Action 3 has the highest variance, so we are highly uncertain about the reward for this action. In other words, there is non-zero probability that the reward from action 3 is higher than from action 1, i.e. $R_3|\boldsymbol{\theta},D > R_1|\boldsymbol{\theta},D$. To see if this is really the case, we should take action 3. However, if we take action greedily, action 3 would be chosen. That is, the greedy selection fails to take the uncertainty to involve in action 3 into account, and hence it fails to *explore* and learns about that action.

Ideally, both exploration and exploitation should be addressed. The simplest approach to exploration is done through randomly perturbing greedy actions. This is called ε-greedy algorithm, taking the greedy action with probability $1-\varepsilon$ and otherwise selects an action randomly. Besides the ε-greedy dithering method, various algorithms such as Thompson sampling (Thompson, 1933), Upper Confidence Bound (UCB) (Lai and Robbins, 1985) and the Bayes-optimal approach of Gittins et al. (2011) have been proposed. Among these, we consider Thompson sampling, for which the empirical successes from a variety of fields have been reported in (Hernández-Lobato et al., 2017; Graepel et al., 2010; Chapelle and Li, 2011; May et al., 2012).

In this section, we propose ThompsonBALD, a novel Bayesian batch active learning approach that mitigates the exploration-exploitation trade-offs. Depending on how the BALD acquisition function is reformulated, there are two versions of such an approach that satisfy the principle of maximal disagreement. We will also consider another dithering attempt to BALD, Janz et al. (2017a), by using just 2 MC samples to estimate the BALD value.

5.1 Thompson Sampling

We first review Thompson sampling, recasting it to active learning framework. Suppose an agent takes a sequence of actions $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots \mathbf{x}^{(t)}, \dots$ from an action space $A \triangleq D^{\text{Pool}}$. Thus action space is finite in our case. If the agent takes some action $\mathbf{x}^{(t)} \in A$, the environment generates y_t and the reward r_t from some parametric distributions $p(y_t = y|\mathbf{x}^{(t)}, \boldsymbol{\omega})$ and $p(r_t = r|\mathbf{x}^{(t)}, \boldsymbol{\omega})$, respectively. The reward indicates the informativeness of the action $\mathbf{x}_t$ towards reducing uncertainty of $\boldsymbol{\omega}$.[1] We choose an action $\mathbf{x}_t$ that maximizes the conditional expectation of r_t, $\mathbb{E}[r_t|\mathbf{x}^{(t)}, \boldsymbol{\omega}]$, in order to maximize the information update on $\boldsymbol{\omega}$,[2]. We have a prior knowledge of $\boldsymbol{\omega}$ from the training dataset D_0, giving $p(\boldsymbol{\omega}|D_0)$. The information update is done through Bayes rule: $p(\boldsymbol{\omega}|\mathbf{X}, Y, D_0) \propto p(\boldsymbol{\omega}|D_0)p(Y|\mathbf{X}, \boldsymbol{\omega}, D_0)$, where $(\mathbf{X}, Y) = \left(\{\mathbf{x}_t\}_{t=1}^N, \{\mathbf{y}_t\}_{t=1}^N\right) := D_0$.

The greedy and Thompson sampling are different in a way that they deal with the model parameter $\boldsymbol{\omega}$. The greedy agent chooses the action which has the largest expected value, $\mathbb{E}_{\boldsymbol{\omega}|D_0}[\mathbb{E}[r_t|\mathbf{x}^{(t)}, \boldsymbol{\omega}]]$. In contrast, Thompson sampling chooses the action which has the largest value under a particular random realization $\mathbb{E}[r_t|\mathbf{x}^{(t)}, \hat{\boldsymbol{\omega}}]$ for some random sample $\hat{\boldsymbol{\omega}} \sim p(\boldsymbol{\omega}|D_0)$. This is equivalent to say that we choose an action $\mathbf{x}_t$ with its probability of being optimal. The general procedure of the Thompson sampling algorithm is outlined in the algorithm 4

A natural question at this stage would then be - "what is the conditional expectation of r_t in active learning framework?". We will now propose two expected reward formulation based on the BALD acquisition function, providing substantial performance improvements compared to BatchBALD and ACS-FW.

[1] The higher reward, the larger decrease in uncertainty of $\boldsymbol{\omega}$

[2] In general reinforcement learning, the goal is often to maximize cumulative reward $\sum_{i=1}^{t} r_i$

Algorithm 4 Thompson Sampling

Given $D_0 = \emptyset$
for t=1,..., T **do**
  Sample $\hat{\boldsymbol{\omega}} \sim p(\boldsymbol{\omega}|D_0)$
  $\mathbf{x}_t \leftarrow \arg\max_{\mathbf{x}} \mathbb{E}[r_t|\mathbf{x}^{(t)}, \boldsymbol{\omega}]$
  Take the action $\mathbf{x}^{(t)}$ and observe y_t
  $D_0 \leftarrow D_0 \cup (\mathbf{x}^{(t)}, y_t)$
end for

5.2 ThompsonBALD construction

We formulate the expected reward based on BALD. Recall that the BALD acquisition function $a_{BALD}(\mathbf{x}, p(\boldsymbol{\omega}|D_0))$ is given and can be rearranged as

$$\mathbb{I}[y, \boldsymbol{\omega}|\mathbf{x}, D_0] = \mathbb{H}[y|\mathbf{x}, D_0] - \mathbb{E}_{\boldsymbol{\omega}|D_0}[\mathbb{H}[y|\mathbf{x}, D_0, \boldsymbol{\omega}]] \tag{5.1}$$

$$= \mathbb{E}_{\boldsymbol{\omega}|D_0}[\mathbb{H}[y|\mathbf{x}, D_0] - \mathbb{H}[y|\mathbf{x}, D_0, \boldsymbol{\omega}]] \tag{5.2}$$

$$:= \mathbb{E}_{\boldsymbol{\omega}|D_0}[\mathbb{E}[r^{\text{BALD}}|\mathbf{x}, D_0, \boldsymbol{\omega}]] \tag{5.3}$$

As explained earlier, the greedy action is taken by $\arg\max_{\mathbf{x}} \mathbb{E}_{\boldsymbol{\omega}|D_0}[\mathbb{E}[r^{\text{BALD}}|\mathbf{x}, D_0, \boldsymbol{\omega}]]$. Therefore, in this perspective, the greedy selection of BALD as in Gal et al. (2017) enforces only exploitation by choosing the action that maximizes the expected reward.

When it comes to Thompson sampling, however, approximation procedure of the expected reward $\mathbb{E}[r^{\text{BALD}}|\mathbf{x}, D_0, \boldsymbol{\omega}]$ is not straightforward. The marginal predictive entropy $\mathbb{H}[y|\mathbf{x}, D_0]$ is intractable in practice, and thus approximated with MC samples. With a single sample $\hat{\boldsymbol{\omega}} \sim q^*_{\boldsymbol{\theta}}(\boldsymbol{\omega})$, it is approximated as $\sum_c p(y_i = c|\mathbf{x}_i, \hat{\boldsymbol{\omega}}) \log p(y_i = c|\mathbf{x}_i, \hat{\boldsymbol{\omega}})$, which is equal to the conditional predictive entropy $\mathbb{H}[y|\mathbf{x}, D_0, \boldsymbol{\omega}]$. Therefore, $\mathbb{E}[r^{\text{BALD}}|\mathbf{x}, D_0, \boldsymbol{\omega}]$ is always 0 in this setting.

In this context, we will consider the marginal predictive entropy $\mathbb{H}[y|\mathbf{x}, D_0]$ as an independent estimates to the conditional predictive entropy $\mathbb{H}[y|\mathbf{x}, D_0, \boldsymbol{\omega}]$. Then this is approximated with independent MC samples to the one with the Thompson sample used for computing $\mathbb{H}[y|\mathbf{x}, D_0, \boldsymbol{\omega}]$.

Then, the conditional expectation of reward is now given as:

$$\mathbb{E}[r^{\text{BALD}}|\mathbf{x}, D_0, \boldsymbol{\omega}] := \hat{\mathbb{H}}[y|\mathbf{x}, D_0] - \mathbb{H}[y|\mathbf{x}, D_0, \boldsymbol{\omega}] \tag{5.4}$$

Recall that BALD is a score based on a disagreement between marginal uncertainty and uncertainty quantified by the individual setting of $\boldsymbol{\omega}$ on model prediction. Therefore, Thompson sampling with this expected reward chooses an action $\mathbf{x}$ with its probability of maximizing this disagreement. The corresponding pseudocode is shown in 5.

Algorithm 5 ThompsonBALD

Given $q^*_{\boldsymbol{\theta}}(\boldsymbol{\omega})$, $\hat{\mathbb{H}}[y|\mathbf{x}, D_0]$, acquisition batch size B, $D^{\text{To-be-labeled}} = \emptyset$
for b=1,2,...,B **do**
  Sample $\hat{\boldsymbol{\omega}} \sim q^*_{\boldsymbol{\theta}}(\boldsymbol{\omega})$
  $\mathbf{x}_b^{\text{To-be-labeled}} = \text{argmax}_{\mathbf{x} \in D^{\text{Pool}} \setminus D^{\text{To-be-labeled}}} \hat{\mathbb{H}}[y|\mathbf{x}, D_0] - \mathbb{H}[y|\mathbf{x}, D_0, \hat{\boldsymbol{\omega}}]$
  $D^{\text{To-be-labeled}} = D^{\text{To-be-labeled}} \cup \{\mathbf{x}_b^{\text{To-be-labeled}}\}$
end for

5.3 ThompsonBALD Alternative

The BALD acquisition function is in the QBC framework that follows the principle of maximal disagreement, and the disagreement among the committee is measured by the *difference in predictive entropy*. In this section, we will show the disagreement measure of BALD can be interpreted as *mean difference in log probability* by rearranging the BALD function, and ultimately propose an alternative to the ThompsonBALD.

For this time, we start with the definition of mutual information between y and $\boldsymbol{\omega}$, and rearrange this as follows:

$$\mathbb{I}[y, \boldsymbol{\omega}|\mathbf{x}, D_0] = \sum_y \int_{\boldsymbol{\omega}} p(y, \boldsymbol{\omega}|\mathbf{x}, D_0) \log \frac{p(y, \boldsymbol{\omega}|\mathbf{x}, D_0)}{p(y|\mathbf{x}, D_0) p(\boldsymbol{\omega}|\mathbf{x}, D_0)} \tag{5.5}$$

$$= \mathbb{E}_{\boldsymbol{\omega}|D_0} \left[\sum_y \left(p(y|\mathbf{x}, D_0, \boldsymbol{\omega}) \log \frac{p(y|\mathbf{x}, \boldsymbol{\omega}, D_0)}{p(y|\mathbf{x}, D_0)} \right) \right] \tag{5.6}$$

$$:= \mathbb{E}_{\boldsymbol{\omega}|D_0} [\mathbb{E}[r^{\text{BALD-alternative}}|\mathbf{x}, D_0, \boldsymbol{\omega}]] \tag{5.7}$$

From equation (5.7), we now say that BALD score measures the mean log difference[3] between the conditional probability $p(y|\mathbf{x},\boldsymbol{\omega},D_0)$ and the marginal probability $p(y|\mathbf{x},D_0)$, where the mean is taken with respect to the conditional predictive probability $p(y|\mathbf{x},\boldsymbol{\omega},D_0)$. As explained earlier, this disagreement measure always returns 0. For this time, we will assume $p(y|\mathbf{x},D_0)$ to be independent from conditional predictive distribution in $r^{\text{BALD-alternative}}$, and therefore estimate it separately as $\mathbb{E}_{\boldsymbol{\omega}|D_0}[p(y|\mathbf{x},D_0,\boldsymbol{\omega})] \coloneqq \hat{p}^{y|\mathbf{x},D_0}$. Hence the posterior reward function is now written as

$$\mathbb{E}[r^{\text{BALD-alternative}}|\mathbf{x},D_0,\boldsymbol{\omega}] \coloneqq \mathbb{E}_{y|\mathbf{x},D_0,\boldsymbol{\omega}}\left[\log \frac{p(y|\mathbf{x},\boldsymbol{\omega},D_0)}{\hat{p}^{y|\mathbf{x},D_0}}\right]. \tag{5.8}$$

The detailed Thompson sampling algorithm for this objective is outlined in algorithm 6

Algorithm 6 ThompsonBALD-alternative

Given $q^*_{\boldsymbol{\theta}}(\boldsymbol{\omega})$, $\hat{p}^{y|\mathbf{x},D_0}$, acquisition batch size B, $D^{\text{To-be-labeled}} = \emptyset$
for b=1,2,...,B **do**
  Sample $\hat{\boldsymbol{\omega}} \sim q^*_{\boldsymbol{\theta}}(\boldsymbol{\omega})$
  $\mathbf{x}_b^{\text{To-be-labeled}} = \text{argmax}_{\mathbf{x}\in D^{\text{Pool}}\setminus D^{\text{To-be-labeled}}} \sum_y p(y|\mathbf{x},D_0,\hat{\boldsymbol{\omega}}) \log \frac{p(y|\mathbf{x},D_0,\hat{\boldsymbol{\omega}})}{\hat{p}^{y|\mathbf{x}}}$
  $D^{\text{To-be-labeled}} = D^{\text{To-be-labeled}} \cup \{\mathbf{x}_b^{\text{To-be-labeled}}\}$
end for

As a result, ThompsonBALD enforces both exploitation and exploration by taking variance into account produced by a single Monte Carlo sample from the posterior. Note that the exploration scheme is applied only to the uncertainty quantified by the individual setting of $\boldsymbol{\omega}$, and hence ThompsonBALD can select a point that has a low BALD score by the following scenario; mean of $\mathbb{H}[y|\mathbf{x},D_0,\hat{\boldsymbol{\omega}}]$ is small but its large variance leads to large disagreement with $\mathbb{H}[y,\mathbf{x},D_0]$.

Lastly, for both ThompsonBALD and ThompsonBALD-alternative, we would like to highlight its fast algorithm time complexity. It only requires the same complexity as with plain BALD: $O(c\cdot|D^{\text{Pool}}|\cdot T)$. This is a significantly smaller complexity compare to BatchBALD. Recall that the time complexity of the BatchBALD algorithm is $O(c^B\cdot|D^{\text{Pool}}|^B\cdot T)$. It still has the large complexity with Mont-Carlo sampling: $O(Bc\cdot\min\{c^B,m\}\cdot|D^{\text{Pool}}|\cdot T)$. Therefore, if ThompsonBALD shows competing results with BatchBALD, ThompsonBALD is definitely preferable.

[3] One can say that $\mathbb{E}[r^{\text{BALD-alternative}}(y|\mathbf{x},\boldsymbol{\omega})]$. is the KL divergence from $p(y|\mathbf{x},D_0)$ to $p(y|\mathbf{x},\boldsymbol{\omega},D_0)$, i.e. $\text{KL}[p(y|\mathbf{x},\boldsymbol{\omega},D_0)||p(y|\mathbf{x},D_0]$

5.4 Related Works

To the best of our knowledge, Janz et al. (2017a,b) are only two papers that attempt to inject noise in BALD acquisition function. In Janz et al. (2017a), BALD is used in the context of generating valid sequences from sequence-based deep models. A greedy selection of a point $\mathbf{x}_t$ given $\mathbf{x}_{t-1}$ generates a single most informative sequence. Therefore a collection of generated sequences are individually informative but not diverse, and hence Janz et al. (2017a) introduce noise in BALD. As explained earlier, the single MC sample always gives 0, and hence two MC sample is used. We will denote this method as NoisyBALD, and this is the first attempt to utilize NoisyBALD in a proper active learning framework. The corresponding pseudocode is shown in Algorithm 7.

Algorithm 7 NoisyBALD

Given $q^*_{\boldsymbol{\theta}}(\boldsymbol{\omega})$, acquisition batch size B, $D^{\text{To-be-labeled}} = \emptyset$
for b=1,2,...,B **do**
  Sample $\hat{\boldsymbol{\omega}}_1, \hat{\boldsymbol{\omega}}_2 \sim q^*_{\boldsymbol{\theta}}(\boldsymbol{\omega})$
  $\hat{\mathbb{H}}[y|\mathbf{x}, D_0] = \sum_c \frac{1}{2} \sum_{t=1}^{2} \hat{p}_c^t \log \hat{p}_c^t$
  $\hat{\mathbb{E}}_{p(\boldsymbol{\omega}|D_0)}[\mathbb{H}[y|\mathbf{x}, D_0, \boldsymbol{\omega}]] = \sum_c \left(\frac{1}{2} \sum_{t=1}^{2} \hat{p}_c^t\right) \log \left(\frac{1}{2} \sum_{t=1}^{2} \hat{p}_c^t\right)$
  $\mathbf{x}_b^{\text{To-be-labeled}} = \text{argmax}_{\mathbf{x} \in D^{\text{Pool}} \setminus D^{\text{To-be-labeled}}} \hat{\mathbb{H}}[y|\mathbf{x}, D_0] - \hat{\mathbb{E}}_{p(\boldsymbol{\omega}|D_0)}[\mathbb{H}[y|\mathbf{x}, D_0, \boldsymbol{\omega}]]$
  $D^{\text{To-be-labeled}} = D^{\text{To-be-labeled}} \cup \{\mathbf{x}_b^{\text{To-be-labeled}}\}$
end for

Janz et al. (2017b) introduces noise to BALD for the similar reason, introducing noise to generate diverse sequences, but sampling from a Boltzamann distribution:

$$p(\mathbf{x}|\boldsymbol{\theta}) = \frac{\exp(a_{BALD}(y, \boldsymbol{\omega})/\theta)}{\sum_{\mathbf{x} \in c} \exp(a_{BALD}(y, \boldsymbol{\omega})/\theta)}, \tag{5.9}$$

with temperature parameter θ and response y corresponding to $\mathbf{x}$. Note that the temperature parameter need to be tuned. In experiments, we will implement NoisyBALD in algorithm 7.

Chapter 6

ThompsonBALD Experiments

In chapter 3, we reviewed recently developed Bayesian active learning algorithms: BALD, BatchBALD, and ACS-FW. In chapter 4, we reported and discussed the empirical comparisons for the performance of them. From this experimental investigation, we concluded that there is no single active learning algorithm that outperforms others in every setting. In chapter 5, we proposed two novel Bayesian batch active learning approaches mitigating the exploration-exploitation trade-offs via Thompson sampling.

In this chapter, considering the previous empirical comparisons as baselines, we demonstrate the benefits of ThompsonBALDs and NoisyBALD. The experimental setting involves three variations: variational inference type, acquisition batch size and dataset. As before, Two variational inference methods are considered: MC dropout, and Neural linear. Please see chapter 4 for descriptions of model architecture, batch sizes, and relevant hyperparameters. In addition to all batch size configurations used in chapter 4, we further evaluate ThompsonBALD in larger batch settings, except BatchBALD due to the computational issue. Lastly besides (R)MNIST and (R)FMNIST, we also consider SVHN (Goodfellow et al., 2013) and CIFAR10(Krizhevsky et al., 2009) with the same batch size in Kirsch et al. (2019). Again, BatchBALD is not evaluated for this due to the computational burden. All the model architectures, active learning loops, and optimization hyperparameters are described in chapter 4.

6.1 Results

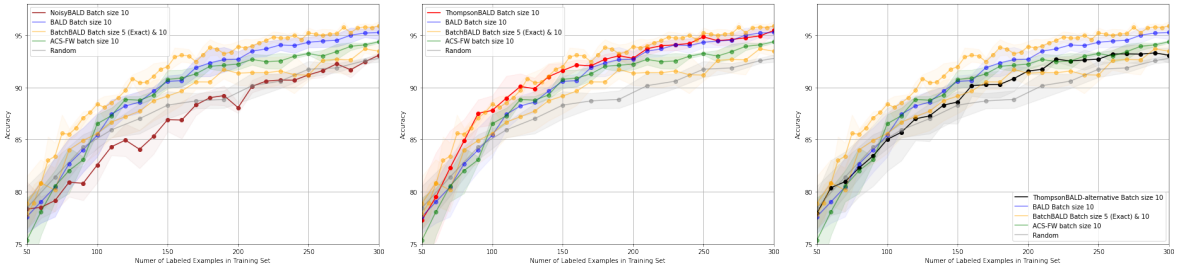
Long story short, in all cases, ThompsonBALD at least as well as its competitors. Especially ThompsonBALD with neural linear in figure 6.2b, 6.2e, 6.4b and 6.4e results in a small but important accuracy improvement. However, ThompsonBALD with MC dropout on MNIST still is not able to outperform the random baseline. We see that the fits of BALD to MC dropout uncertainty approximation is not good enough for solving complex datasets such as Fashion MNIST. We discuss details of other algorithms in figure 6.1 and 6.2.

As before, we conduct experiments on R(F)MNIST datasets. As the dataset contains redundant points, it gives us a better intuition in the exploration quality of each dithering method: the disagreement on predictive entropies/log-probabilities and NoisyBALD estimator. The result and analysis is described in figure 6.5 and 6.6.

For the additional experiment, we conduct experiments on complex datasets such as SVHN and cifar10 as in Kirsch et al. (2019). This experiment will be done with the same initial number of datasets and batch sizes as the paper did. In specific, for Fashion MNIST and svhn, the model is initially trained with 1000 points, request acquisition batch of size 3000 and a budget of 12,000. For cifar10, since the dataset is relatively harder, the model is initially trained with 3000 points, request acquisition batch of size 5000 and a budget of 20,000. As noted earlier, due to the extreme computational complexity of BatchBALD of large batch size, BatchBALD is not implemented in this experiment. This is reported in figure 6.7

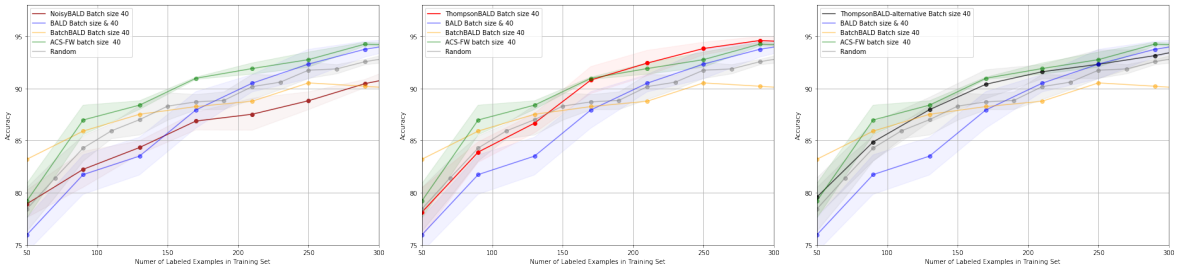
In conclusion, even with BALD with Thompson sampling, we see that there is no single active learning method outperforms others for both variational inference methods. Each algorithm have pros and cons. First, BatchBALD gives theoretical guarantees on the quality of their greedy algorithm relative to a full search for the best batch. However, it is hardly scalable to large batch size as it is infeasible to enumerate all possible configurations for more than 5 batches. Second, ACS-FW is scalable to approximate posterior methods (quality). However, it does not actually produce the desired number of samples during each iteration. Lastly, ThompsonBALD has the same time complexity with BALD and hence much faster than BatchBALD with competing performance. However, it does not have solid theoretical grounds for finding the best batch.

MC Dropout (MNIST); batch size 10



(a) NoisyBALD **(b)** ThompsonBALD **(c)** ThompsonBALD-alternative

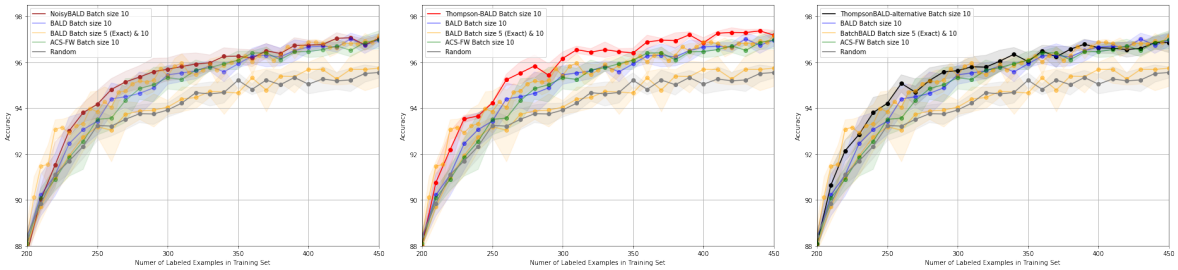
MC Dropout (MNIST); batch size 40



(d) NoisyBALD **(e)** ThompsonBALD **(f)** ThompsonBALD-alternative

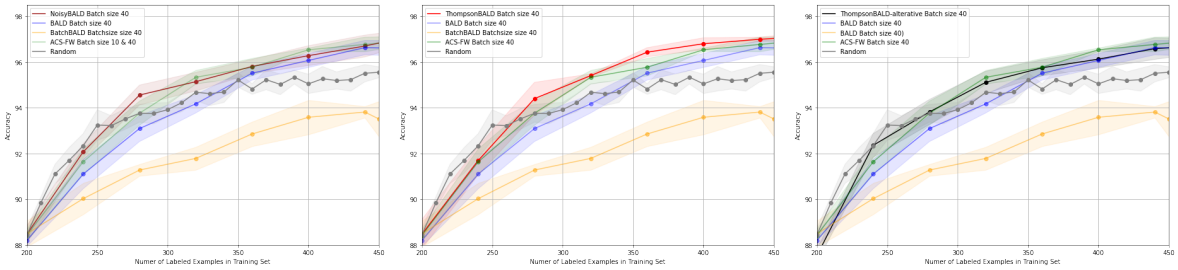
Figure 6.1: Only ThompsonBALD is able to perform as well as exact BatchBALD or ACS-FW. Yet, it is upper bounded by exact BatchBALD but not significant, and under-perform compared to ACS-FW for the first two active learning loop. NoisyBALD performs the worst, and ThompsonBALD-alternative outperforms BALD only with large batch size. We think that ThompsonBALD-alternative results from "being too noisy for small batch size, but being noisy appropriately for decorrelating the query", and NoisyBALD results from the former.

Neural Linear (MNIST); batch size 10



(a) NoisyBALD **(b)** ThompsonBALD **(c)** ThompsonBALD-alternative

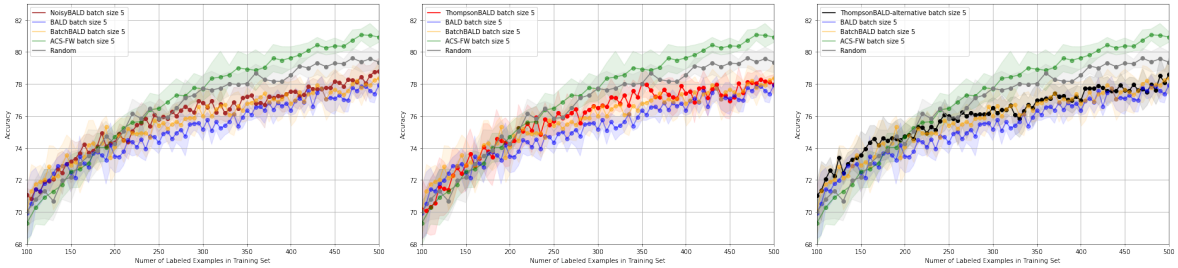
Neural Linear (MNIST); batch size 40



(d) NoisyBALD **(e)** ThompsonBALD **(f)** ThompsonBALD-alternative

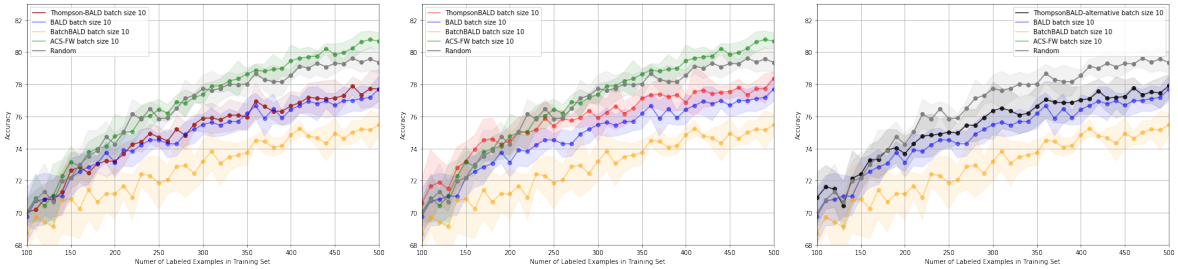
Figure 6.2: All of our newly introduced methods are able to perform as well as exact BatchBALD and ACS-FW. Especially, ThompsonBALD outperforms every method.

Fashion MNIST; MC Dropout; batch size 5



(a) NoisyBALD **(b)** ThompsonBALD **(c)** ThompsonBALD-alternative

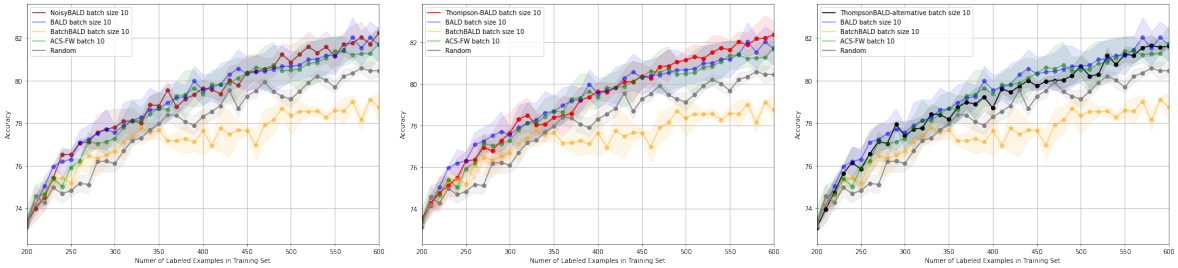
Fashion MNIST; MC Droout ; batch size 10



(d) NoisyBALD **(e)** ThompsonBALD **(f)** ThompsonBALD-alternative

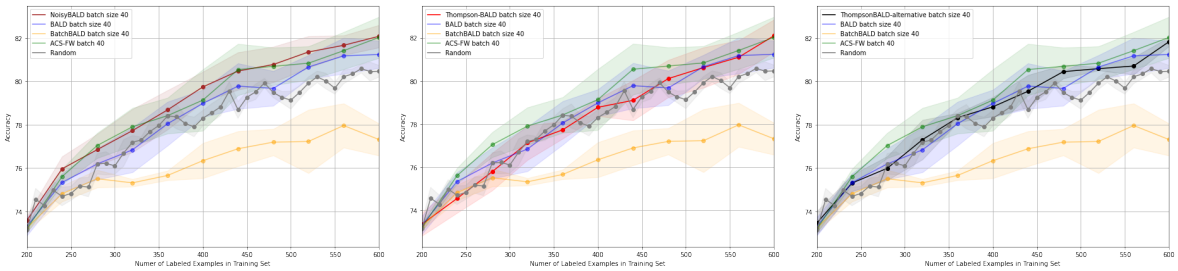
Figure 6.3: All BALD-variants methods are not able to beat the random baseline. This is not a surprising result because even BALD with batch size 1 couldn't beat this setting as in figure 4.3. Nevertheless, NoisyBALD and ThompsonBALD result in a small but important accuracy improvement compared to exact BatchBALD with both batch sizes.

Fashion MNIST; Neural Linear; batch size 10



(a) NoisyBALD **(b)** ThompsonBALD **(c)** ThompsonBALD-alternative

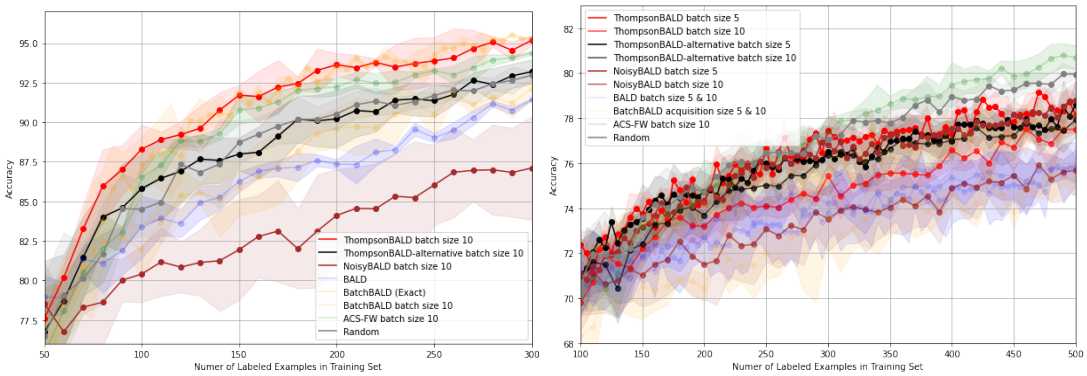
Fashion MNIST; Neural Linear; batch size 40



(d) NoisyBALD **(e)** ThompsonBALD **(f)** ThompsonBALD-alternative

Figure 6.4: All of our newly introduced methods are able to perform as well as exact BatchBALD and ACS-FW. In this case, NoisyBALD performed the best.

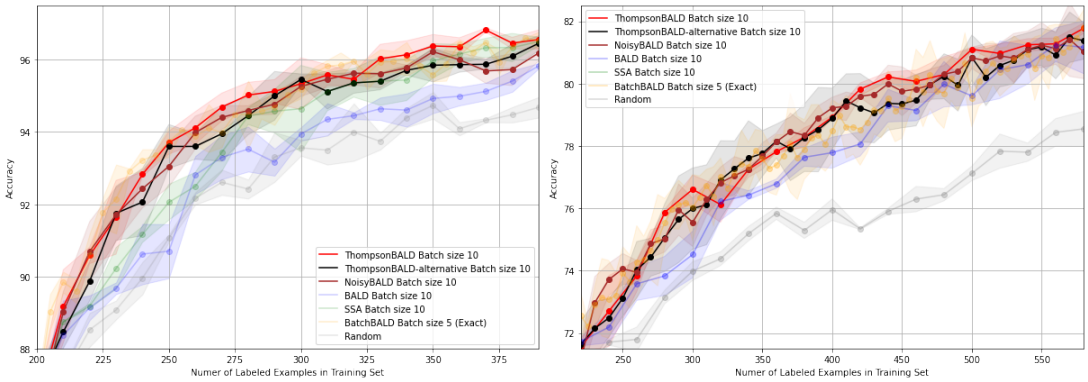
MC Dropout



(a) Repeated MNIST **(b)** Repeated Fashion MNIST

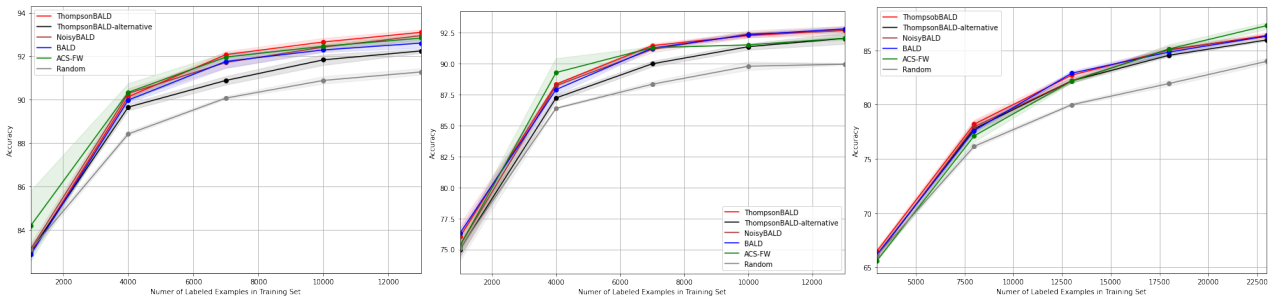
Figure 6.5: ThompsonBALD outperforms all BALD-variants methods on both datasets, and also ACS-FW on RMNIST. ThompsonBALD-alternative performs second best on RMNIST, and is least affected by the increase of batch size. One surprising thing is NoisyBALD performs the worst on both datasets with MC dropout.

Neural Linear



(a) Repeated MNIST **(b)** Repeated Fashion MNIST

Figure 6.6: With neural linear, our methods outperform all the baseline introduced in 3. Among them, we see that ThompsonBALD performs the best with a small margin.



(a) Fashion MNIST **(b)** SVHN **(c)** cifar 10

Figure 6.7: This is the same experiment to the one in Pinsler et al. (2019). We see that ThompsonBALD performs as well as or better than BALD for all three datasets, whereas ThompsonBALD-alternative performs worse. Interestingly, even though the noise is injected to NoisyBALD, there is no performance drop. ACS-FW performs better than others in cifar 10, but worse in svhn.

Chapter 7

Conclusion

We have introduced a novel Bayesian batch active learning algorithm, ThompsonBALD. ThompsonBALD reduces the required computational time compared to BatchBALD and shows equivalent or improved performance over BALD, BatchBALD and ACS-FW. Our algorithm, as well as the others, were evaluated in a fair and rationale setting so that we could identify each algorithms' pros and cons. Even though our algorithm does not outperform others in every setting, we expect ThompsonBALD can be a strong baseline as it is significantly fast and straightforward to implement.

7.1 Future Direction

The performance of the batch active learning algorithms depends on how well we enforce diversity among the different selected points. In this work, we proposed two ways of approximating the BALD acquisition function, maintaining the desirable QBC properties. Although ThompsonBALD provided a significant improvement, they can be limited to the informative power of BALD, as in figure 6.3. The interesting follow up work would be to understand why it works and when it fails. The lack of theoretical analysis is the reason that it is not very popular (Chapelle and Li, 2011). The recent theoretical analyses rely on asymptotic optimality convergence of regret (Chapelle and Li, 2011; Li et al., 2011; Russo et al., 2017). We leave the formulating of regret in the active learning framework for future research.

We propose another approach to constructing the diverse batch. We focus on the expected full data log posterior in equation (3.35). We argue that all the points in the pool dataset are dis-

tributed as determinantal point process, and the corresponding kernel matrix can be constructed using the $L_i(\boldsymbol{\omega})$. We believe that this matrix contains all the necessary information of informativeness and similarity measures for diverse batch sampling. We describe our initial attempt of the approach in Appendix D.

Appendix A

Analytic Form of the Weighted Fisher Inner Product for Linear Model

Pinsler et al. (2019) derived closed form term for the weighted Fisher inner product, $\mathbb{E}[\nabla_{\boldsymbol{\omega}} L^T \nabla_{\boldsymbol{\omega}} L_j]$ for simple scalar Bayesian linear regression. The model is given as:

$$y_i = \boldsymbol{\omega}^T \mathbf{x}_i + \varepsilon_i, \;\; \varepsilon_i \sim N(0, \sigma_0^2), \;\; \boldsymbol{\omega} \sim p(\boldsymbol{\omega}), \tag{A.1}$$

and the closed-form term for the weighted Fisher inner product and its norm are derived as

$$\langle L_i, L_j \rangle_{\hat{\pi},F} = \frac{\mathbf{x}_i^T \mathbf{x}_j}{\sigma_0^4} \mathbf{x}_i^T \Sigma_{\boldsymbol{\omega}} \mathbf{x}_j \implies \alpha_{ACS}(\mathbf{x}_i; D_0) \coloneqq \frac{\mathbf{x}_i^T \mathbf{x}_i}{\sigma_0^4} \mathbf{x}_i^T \Sigma_{\boldsymbol{\omega}} \mathbf{x}_i \tag{A.2}$$

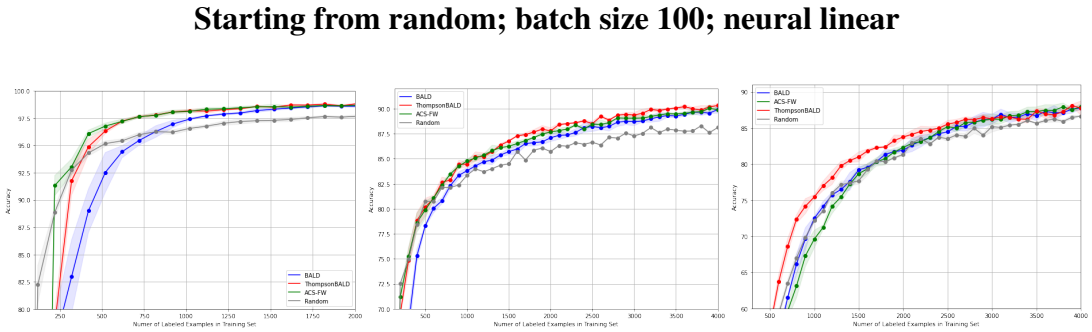
where $\Sigma_{\boldsymbol{\omega}} = \sigma_0^2 (X^T X + \sigma_0^2 I)^{-1}$. Furthermore, for this model, the BALD function is derived as

$$\alpha_{BALD}(\mathbf{x}_i; D_0) = \frac{1}{2} \log \left(\sigma_0^2 + \frac{\mathbf{x}_i^T \Sigma_{\boldsymbol{\omega}} \mathbf{x}_i}{\sigma_0^2} \right). \tag{A.3}$$

It can be clearly noticed that, by dropping $\mathbf{x}_i^T \mathbf{x}_i$ in $\alpha_{ACS}(\mathbf{x}_i, D_0)$, the two acquisition functions are proportional to each other, i.e. $\exp(2\alpha_{BALD}(\mathbf{x}_i; D_0)) \propto \alpha_{ACS}(\mathbf{x}_i; D_0)$. This is equivalent to say that if we greedily choose a batch points according to the magnitude of reduction (i.e. the sensitivity σ), then α_{ACS} and α_{BALD} acquisition function returns the same batch.

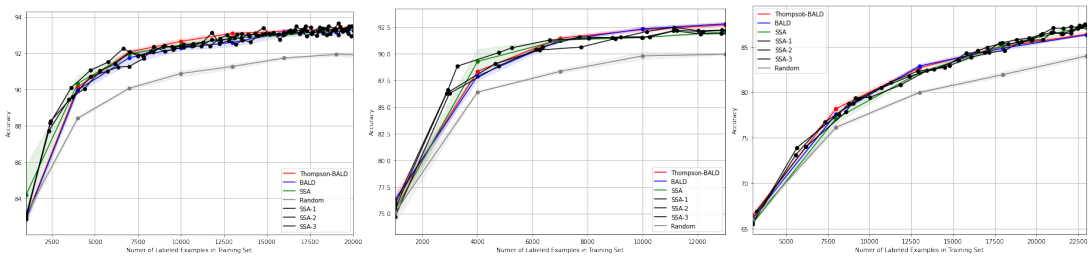
Appendix B

Additional Experiments



(a) MNIST **(b)** Fashion MNIST **(c)** SVHN

Figure B.1: In Kirsch et al. (2019), the experiment with EMNIST dataset is done with random initialized model. We do a similar experiment with more diverse dataset, but with the neural linear architecture. We start with random initialized model with 10% accuracy and update the model with 100 batch size.



(a) Fashion MNIST **(b)** SVHN **(c)** Cifar 10

Figure B.2: Recall that ACS-FW does not produce the desired number of samples in each batch iteration. This plot shows the actual performance of ACS-FW (SSA) by showing its accuracy with its exact batch size, not interpolate it. This is the same experiment with figure 6.7

Appendix C

Conditional Expectation of Rewards Visualization

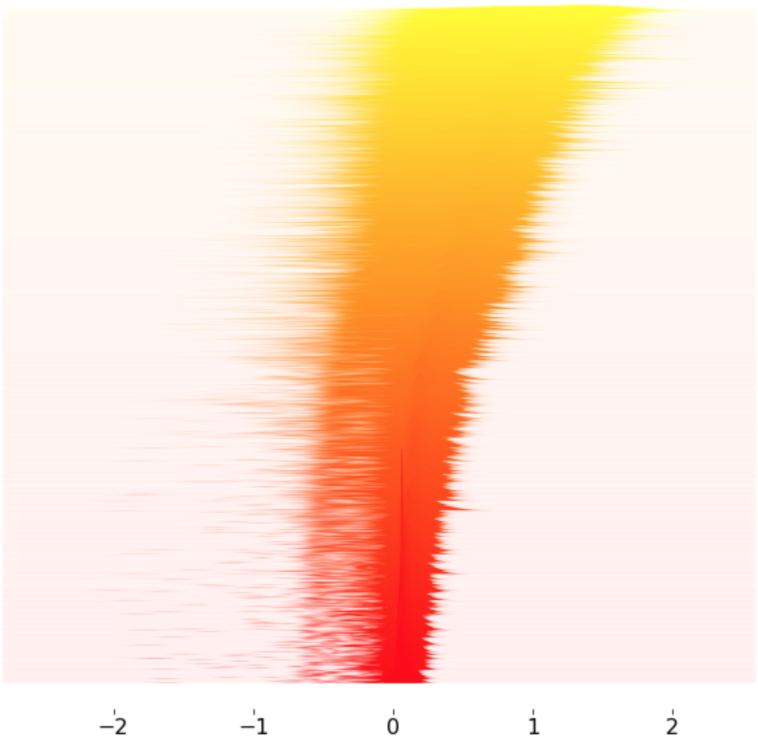


Figure C.1: Density plot of the conditional expectation of rewards for the MC dropout architecture on the MNIST dataset. 1000 data points are randomly sampled and queried based on the BALD score. x-axis represents the BALD score. 100 conditional expectation of rewards are samples for each data point. We see that high variance for high BALD score points and hence efficiently ease the problem of exploration-exploitation trade-offs through Thompson sampling.

Appendix D

Bayesian Batch Active Learning via Determinantal Point Process

Recall that the expected full data log posterior $\mathbb{E}_{y_1,\ldots y_N}[\log p(\boldsymbol{\omega}|D_0 \cup D^{\text{Pool}})]$ is modified to

$$L_i(\boldsymbol{\omega}) = \mathbb{E}_{y_i}\left[\log p(y_i|\mathbf{x}_i,\boldsymbol{\omega})\right] - \mathbb{E}_{y_i}\left[\log p(y_i|\mathbf{x}_i,D_0)\right], \tag{D.1}$$

where $L_i : \boldsymbol{\Theta} \to \mathbb{R}$ is a vector in the Hilbert space. With the weighted L^2 norm, we project this infinite dimensional vector to finite space through random projection:

$$\hat{L}_i = [L_i(\boldsymbol{\omega}_1),\ldots,L_J(\boldsymbol{\omega}_J)],\ \ \boldsymbol{\omega}_j \sim p(\boldsymbol{\omega}|D_0) \tag{D.2}$$

We will consider a determinantal point process (DPP) P on a ground set D^{Pool}, which is then simply a probability measure on $2^{|D^{\text{Pool}}|}$. We focus on L-ensembles defined by:

$$P(\mathbf{B} = B) = \frac{\det(L_B)}{\det(L+I)},\ \ \text{for } B \subseteq D^{\text{Pool}} \tag{D.3}$$

with

$$L = \begin{pmatrix} \ldots & \hat{L}_1^T & \ldots \\ \ldots & \hat{L}_2^T & \ldots \\ & \vdots & \\ \ldots & \hat{L}_N^T & \ldots \end{pmatrix} \begin{pmatrix} \vdots & \vdots & & \vdots \\ \hat{L}_1 & \hat{L}_2 & \ldots & \hat{L}_N \\ \vdots & \vdots & & \vdots \end{pmatrix} \approx \begin{pmatrix} \langle L_1,L_1\rangle & \langle L_1,L_2\rangle & \ldots & \langle L_1,L_N\rangle \\ \langle L_2,L_1\rangle & \langle L_2,L_2\rangle & \ldots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \langle L_N,L_1\rangle & \langle L_N,L_2\rangle & \ldots & \langle L_N,L_N\rangle \end{pmatrix} \tag{D.4}$$

Then we see that diagonal term is the sensitivity term σ^2 in the coreset construction, which can be seen as a greedy acquisition function, and the off-diagonal terms measure the cosine similarity.

In DPP, it is easy to condition a DPP on some event B that all of the elements in a set $\tilde{B}$ are observed. For $B \cap \tilde{B} = \emptyset$,

$$P_L(\mathbf{B} = B \cup \tilde{B} | \tilde{B} \subseteq \mathbf{B}) = \frac{\det(L_{B \cup \tilde{B}})}{\det(L + I_{\tilde{B}})}, \tag{D.5}$$

where $I_{\tilde{B}}$ is the matrix with ones in the diagonal entries indexed by elements of $D^{\text{Pool}} \backslash \tilde{B}$. This is in conjunction with BatchBALD in a sense that it gives a score of a point in the Pool set conditioned on all the additional data points that have already been picked and the training datset D_0.

The benefit of DPP is that it allows us to make a comparative study of *expressiveness* by splitting the model into the quality and diverse components. This is possible since our DPP kernel L is constructed as a gram matrix $L = B^T B$, where $B = \begin{pmatrix} \vdots & \vdots & & \vdots \\ \hat{L}_1 & \hat{L}_2 & \dots & \hat{L}_N \\ \vdots & \vdots & & \vdots \end{pmatrix}$. We will write each $\hat{L}_i$ in B as the product of a quality term $q_i \in \mathbb{R}^+$ and a vector or normalized diversity features $\boldsymbol{\phi}_i \in \mathbb{R}^J$, $||\boldsymbol{\phi}_i|| = 1$. Then the entries of the kernel is now

$$L_{ij} = q_i \boldsymbol{\phi}_i^T \boldsymbol{\phi}_j q_j. \tag{D.6}$$

Here q_i is a measure of a point $\mathbf{x}_i$ and $\boldsymbol{\phi}_i^T \boldsymbol{\phi}_j \in [-1, 1]$ is a measure of similarity between the points $\mathbf{x}_i$ and $\mathbf{x}_j$. We will denote this similarity as

$$S_{ij} = \boldsymbol{\phi}_i^T \boldsymbol{\phi}_j = \frac{L_{ij}}{\sqrt{L_{ii} L_{jj}}}. \tag{D.7}$$

The advantage of this decomposition is that it enables us to independently model quality and diversity and combine them into a unified model. That is we naturally consider the exploration-exploitation trade-offs by actively mixing the quality model q and the diversity model S so that

we achieve a balanced result.

In active learning, we need a cardinality constraint k, as we assume a budget of points to be labeled. We will condition a DPP on the cardinality of the random set $\mathbf{B}$. A k-DPP on a set $D^{\text{Pool}} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ is a distribution over all subsets $B \subseteq D^{\text{Pool}}$ with cardinality k. Then the k-DPP P_L^k gives probabilities

$$P_L^k(B) = \frac{\det(L_B)}{\sum_{|B'|=k} \det(L'_B)} = P_L^k(B) = \frac{\det(L_B)}{e_k(\lambda_1, \lambda_2, \ldots, \lambda_N)} \tag{D.8}$$

where $|B| = k$, $e_k(\lambda_1, \lambda_2, \ldots, \lambda_N) = \sum_{T \subseteq \{1,2,\ldots,N\}, |T|=k} \prod_{n \in T} \lambda_n$, and $\lambda_1, \ldots, \lambda_N$ are the eigenvalues of L. Then, one simple way to sample the exact k points is by combining algorithm 8 and algorithm 2 in Kulesza and Taskar (2012).

Despite this usefulness of DPPs, it is difficult to make an inference for DPP when N is large. This is because the standard inference algorithms rely on eigendecompositions and determinant computation. For our case, N is often above $50,000$ and hence may not even have the memory to write it down. Various Fast DPP sampling papers are published solving this issue (Chen et al., 2018; Li et al., 2016; Kang, 2013; Han et al., 2017). Dual form is another way to ease this problem. We hope this approach gives competing or better performance than ACS-FW, as well as fixing its bug (the algorithm does not produce the desired number of samples during each iteration).

Bibliography

Javad Azimi, Alan Fern, Xiaoli Zhang-Fern, Glencora Borradaile, and Brent Heeringa. Batch active learning via coordinated matching. *arXiv preprint arXiv:1206.6458*, 2012.

Riccardo Barbano, Jonathan Gordon, Robert Pinsler, and José Miguel Hernández-Lobato. Investigating inference in bayesian neural networks via active learning. 2019.

José M Bernardo. Expected information as expected utility. *the Annals of Statistics*, pages 686–690, 1979.

Michael Bloodgood. Support vector machine active learning algorithms with query-by-committee versus closest-to-hyperplane selection. In *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*, pages 148–155. IEEE, 2018.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Klaus Brinker. Incorporating diversity in active learning with support vector machines. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 59–66, 2003.

Trevor Campbell and Tamara Broderick. Automated scalable bayesian inference via hilbert coresets. *The Journal of Machine Learning Research*, 20(1):551–588, 2019.

Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical Science*, pages 273–304, 1995.

Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *Advances in neural information processing systems*, pages 2249–2257, 2011.

Laming Chen, Guoxin Zhang, and Eric Zhou. Fast greedy map inference for determinantal point process to improve recommendation diversity. In *Advances in Neural Information Processing Systems*, pages 5622–5633, 2018.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996.

Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

Marguerite Frank, Philip Wolfe, et al. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

Lintin C Freeman. Elementary applied statistics: for studies in behavioraal science/by linton c. freeman. 1965.

Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*, 2015a.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation. *arXiv preprint arXiv:1506.02157*, 2015b.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pages 1183–1192, 2017.

John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-armed bandit allocation indices*. John Wiley & Sons, 2011.

Daniel Golovin and Andreas Krause. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. *Journal of Artificial Intelligence Research*, 42: 427–486, 2011.

Ian J Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. *arXiv preprint arXiv:1312.6082*, 2013.

Thore Graepel, Joaquin Quinonero Candela, Thomas Borchert, and Ralf Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft's bing search engine. Omnipress, 2010.

Robert B Gramacy, Derek Bingham, James Paul Holloway, Michael J Grosskopf, Carolyn C Kuranz, Erica Rutter, Matt Trantham, R Paul Drake, et al. Calibrating a large computer experiment simulating radiative shock hydrodynamics. *The Annals of Applied Statistics*, 9 (3):1141–1168, 2015.

Alex Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356, 2011.

Mengyang Gu, Xiaojing Wang, James O Berger, et al. Robust gaussian stochastic process emulation. *The Annals of statistics*, 46(6A):3038–3066, 2018.

Yuhong Guo and Dale Schuurmans. Discriminative batch mode active learning. In *Advances in neural information processing systems*, pages 593–600, 2008.

Insu Han, Prabhanjan Kambadur, Kyoungsoo Park, and Jinwoo Shin. Faster greedy map inference for determinantal point processes. *arXiv preprint arXiv:1703.03389*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

José Miguel Hernández-Lobato, James Requeima, Edward O Pyzer-Knapp, and Alán Aspuru-Guzik. Parallel and distributed thompson sampling for large-scale accelerated exploration of chemical space. *arXiv preprint arXiv:1706.01825*, 2017.

Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, 1993.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Steven CH Hoi, Rong Jin, and Michael R Lyu. Large-scale text categorization by batch mode active learning. In *Proceedings of the 15th international conference on World Wide Web*, pages 633–642, 2006a.

Steven CH Hoi, Rong Jin, Jianke Zhu, and Michael R Lyu. Batch mode active learning and its application to medical image classification. In *Proceedings of the 23rd international conference on Machine learning*, pages 417–424, 2006b.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

Jonathan Huggins, Trevor Campbell, and Tamara Broderick. Coresets for scalable bayesian logistic regression. In *Advances in Neural Information Processing Systems*, pages 4080–4088, 2016.

David Janz, Jos van der Westhuizen, and José Miguel Hernández-Lobato. Actively learning what makes a discrete sequence valid. *arXiv preprint arXiv:1708.04465*, 2017a.

David Janz, Jos van der Westhuizen, Brooks Paige, Matt J Kusner, and José Miguel Hernández-Lobato. Learning a generative model for validity in complex discrete structures. *arXiv preprint arXiv:1712.01664*, 2017b.

Oliver Johnson and Andrew Barron. Fisher information inequalities and the central limit theorem. *Probability Theory and Related Fields*, 129(3):391–409, 2004.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Ajay J Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. Multi-class active learning for image classification. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2372–2379. IEEE, 2009.

Byungkon Kang. Fast determinantal point process sampling with application to clustering. In *Advances in Neural Information Processing Systems*, pages 2319–2327, 2013.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in neural information processing systems*, pages 2575–2583, 2015.

Andreas Kirsch, Joost van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. In *Advances in Neural Information Processing Systems*, pages 7026–7037, 2019.

Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(Feb):235–284, 2008.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *arXiv preprint arXiv:1207.6083*, 2012.

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

Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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

Chengtao Li, Stefanie Jegelka, and Suvrit Sra. Fast dpp sampling for nystr\" om with application to kernel methods. *arXiv preprint arXiv:1603.06052*, 2016.

Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 297–306, 2011.

Xin Li and Yuhong Guo. Adaptive active learning for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 859–866, 2013.

Dennis V Lindley. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, pages 986–1005, 1956.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

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

Benedict C May, Nathan Korda, Anthony Lee, and David S Leslie. Optimistic bayesian sampling in contextual-bandit problems. *The Journal of Machine Learning Research*, 13(1): 2069–2106, 2012.

Radford M Neal and Geoffrey E Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.

Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

RM Neal. Bayesian learning for neural networks [phd thesis]. *Toronto, Ontario, Canada: Department of Computer Science, University of Toronto*, 1995.

George L Nemhauser, Laurence A Wolsey, and Marshall L Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1): 265–294, 1978.

Jeremy Oakley. *Bayesian uncertainty analysis for complex computer codes*. PhD thesis, University of Sheffield, 1999.

John Paisley, David Blei, and Michael Jordan. Variational bayesian inference with stochastic search. *arXiv preprint arXiv:1206.6430*, 2012.

Robert Pinsler, Jonathan Gordon, Eric Nalisnick, and José Miguel Hernández-Lobato. Bayesian batch active learning as sparse subset approximation. In *Advances in Neural Information Processing Systems*, pages 6359–6370, 2019.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

Carlos Riquelme, George Tucker, and Jasper Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. *arXiv preprint arXiv:1802.09127*, 2018.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

Daniel Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. A tutorial on thompson sampling. *arXiv preprint arXiv:1707.02038*, 2017.

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

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

H Sebastian Seung, Manfred Opper, and Haim Sompolinsky. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294, 1992.

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

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

Simon Tong. *Active learning: theory and applications*, volume 1. Stanford University USA, 2001.

Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001.

Manfred KK Warmuth, Gunnar Rätsch, Michael Mathieson, Jun Liao, and Christian Lemmen. Active learning in the drug discovery process. In *Advances in Neural information processing systems*, pages 1449–1456, 2002.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Raymond W Yeung. A new outlook on shannon's information measures. *IEEE transactions on information theory*, 37(3):466–474, 1991.

Fedor Zhdanov. Diverse mini-batch active learning. *arXiv preprint arXiv:1901.05954*, 2019.

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