UNDERGRADUATE RESEARCH OPPORTUNITY PROGRAM
(UROP) PROJECT REPORT

ACTIVE LEARNING ON BAYESIAN NEURAL NETWORKS

April 10, 2018

Author: Lam Chi Thanh
Supervisor: Professor Bryan Low Kian Hsiang
Supervisor: Nguyen Quoc Phong

School of Computing
Department of Computer Science

Abstract

Recent advances in deep learning focus on using big data to train and regularize huge models with lots of parameters. These deep learning models often fall short when it does not have enough data. On the other hand, in active learning, we aim to minimize the amount of labeled data we need to train a model. Many active learning acquisition functions depend on model uncertainty, which is often not captured in deep learning models. We want to study information-based active learning application to Bayesian neural networks (Bayesian learning models can capture uncertainty). Our goal is to show that by being Bayesian and using active learning, we can train our model using less labeled data.

Subject Descriptors:

I.2.5 Programming Languages and Software
I.2.6 Learning

Keywords:

Active learning, Bayesian neural networks

Implementation Software and Hardware:

Tensorflow v1.4, Edward v1.3, Keras v2.0, macOS 10.13, Ubuntu 16.04

Acknowledgement

I would like to express my special thanks of gratitude to professor Bryan Low, for welcoming me to his research group, guiding me along the way, and giving me sincere advices with his knowledge and experience. I would like to express my special thanks of gratitude to Nguyen Quoc Phong, for answering lots of my questions and patiently helping me with technical details. I learned many new things during the research duration, and I am really thankful to my supervisors and this research opportunity.

Contents

1. Introduction

In applications where unlabeled data is numerous, but it is expensive to manually label them, it is desirable to formulate a learning algorithm that can initially learn on a (very) little amount of labeled data, then it chooses by itself which data point it would like the user to label next. This is often referred to as active learning. In active learning, the base model is trained on a small amount of data, then it uses an acquisition function to decide the next data points (in a significantly larger pool of unlabeled data) for an human expert to label.

Active learning has been used in various statistical and machine learning models and shown empirically to reduce the amount of labeled data needed to learn a concept [2] [3] [4].

We would like to apply Bayesian method to neural networks and convolutional neural networks, together with active learning, then evaluate its effectiveness on various datasets. In order words, we hope to show that by being Bayesian and use active learning, we can train model with high accuracy with significantly less data.

2. Bayesian Neural Network

A Bayesian neural network is a neural network with a prior distribution $p(\omega)$ defined over its weights. As we observe the data D, the Bayesian neural network is trained by perform Bayes rule, updating its prior to obtain the posterior $p(\omega|D)$:

$$p(\omega|D) = \frac{p(D|\omega)p(\omega)}{p(D)} \tag{1}$$

Prediction. To make prediction, we can compute the predictive distribution given an input x^* as follow:

$$p(y^*|x^*, D) = \int p(y^*|x^*, \omega)p(\omega|D)d\omega \tag{2}$$

Recent advance in deep learning, most notably convolutional neural networks (CNN), which are designed to capture spatial information of the input, can obtain state-of-the-art result in object recognition [5]. Bayesian CNN is an Bayesian approach to CNN , which aim to reduce overfitting and to capture uncertainty [6].

In practice, the posterior $p(\omega|D)$ is often computationally intractable. We can define conjugate prior for the likelihood we use, or use numerical methods to approximate the posterior. Two major techniques to approximate the posterior are Variational Inference and Markov chain Monte Carlo.

2.1. Variational Bayesian methods

In variational inference, we use a parameterized distribution $q_\theta(\omega)$, often referred to as variational distribution, in a tractable family to approximate the posterior, meaning that it is computationally cheaper to compute the predictive distribution above using $q_\theta(\omega)$.

$$p(y^*|x^*, D) = \int p(y^*|x^*, \omega)q_\theta(\omega)d\omega \tag{3}$$

In variational inference, we choose the distribution $q_\theta(\omega)$ by minimizing the Kullback-Leibler divergence (which we will refer to as KL divergence from now on) between $q_\theta(\omega)$ and the posterior $p(\omega|D)$.

$$KL(q_\theta(\omega), p(\omega|D)) = \int q_\theta(\omega) \log \frac{q_\theta(\omega)}{p(\omega|D)} \tag{4}$$

This inference problem has now become an optimization problem. The KL divergence can be rewritten as:

$$\begin{aligned} KL(q_\theta(\omega), p(\omega|D)) &= \int q_\theta(\omega)(\log q_\theta(\omega) - \log p(\omega|D)) \\ &= \mathbb{E}_{q_\theta(\omega)}[\log q_\theta(\omega) - \log p(\omega|D)] \\ &= \mathbb{E}_{q_\theta(\omega)}[\log q_\theta(\omega) - (\log p(D,\omega) - \log p(D))] \\ &= \mathbb{E}_{q_\theta(\omega)}[\log q_\theta(\omega) - \log p(D,\omega)] + \log p(D) \end{aligned}$$

$$\begin{aligned} \log p(D) &= KL(q_\theta(\omega), p(\omega|D)) - \mathbb{E}_{q_\theta(\omega)}[\log q_\theta(\omega) - \log p(D,\omega)] \\ &= KL(q_\theta(\omega), p(\omega|D)) + \mathbb{L}(Q) \end{aligned}$$

Since the log evidence $\log p(D)$ is constant with respect to $q_\theta(\omega)$, thus maximizing $\mathbb{L}(Q)$ is equivalent to minimizing the KL divergence. The $\mathbb{L}(Q)$ objective function is commonly referred to as the negative variational free energy.

2.2. Markov chain Monte Carlo methods

Monte Carlo methods can be used to approximate difficult integrations arise in Bayesian learning for neural network, one example is the integration required to make prediction (refer to equation 2). Unlike variational Bayes methods, Markov chain Monte Carlo methods make no assumptions about the form of the posterior distribution. Therefore, at least in theory, it gives a better approximation to variational inference. However, its main disadvantage it may require a very long time to converge to the desired distribution. Popular algorithms are Gibbs sampling, Metropolis-Hasting and Hamiltonian Monte Carlo.

2.3. Dropout as Bayesian Approximation [1]

Dropout can be used to perform approximate Bayesian inference in deep convolutional neural networks with many parameters [1]. We train a model with dropout before weight layers. And we perform dropout at test time, running several stochastic forward passes to sample from the approximate posterior. More formally, the dropout distribution on every weight is define as follow:

$$q_\theta(\omega_i) = pN(m_i, s^2 I_k) + (1-p)N(0, s^2 I_k) \tag{5}$$

Here, p is the dropout keep probability, m_i and ω_i is the value of the weights in the i^{th} row before and after dropout respectively. We use the dropout distribution $q_\theta(\omega)$ to approximate the true posterior, by minimizing the KL divergence between $q_\theta(\omega)$ and $p(\omega|D)$ (equation 4).

It has been shown that the first term in KL divergence can be approximated by $-\sum_{n=1}^{N} \int q(\omega) \log p(y_n|x_n, \omega) d\omega$. Each term in the sum can by approximated using Monte Carlo integration. The second term in ELBO objective function can be approximated by $\sum_{i=1}^{L} \frac{p_i l^2}{2} ||M_i||_2^2 + \frac{l^2}{2} ||m_i||_2^2$, which is equivalent to the L_2 regularization on the weights and biases in the neural networks.

To make predictions in a Bayesian CNN with dropout, we use dropout at test time, run several stochastic forward passes and average the result.

3. Active Learning

3.1. Pool-based active learning

In pool-based active learning, we have a pool of unlabeled data points. It is assumed that all data points are independently and identically distributed by some underlying distribution. We start with an initial model which is trained on a initial labeled training set. The size of this training set is often significantly smaller than the size of the pool. Then the model, our active learner, select a set of k unlabeled data points S from the pool, ask an external human expert to label S, add S to the training set and remove S from the pool. We then retrain our model from scratch using the new training set.

The main difference between an active learner and a regular passive learner is the querying component: How does our model choose the next data points? We will refer to this querying component as our active learning acquisition function from now on.

3.2. Active learning acquisition functions

In this section, we explore various information-based acquisition functions. We denote D to be our current training set, and x be a single data point in our pool set.

Maximizing Entropy [7]

$$\mathbb{H}[y|x, D] = -\sum_c p(y = c|x, D) \log p(y = c|x, D) \tag{6}$$

Entropy, introduced by Shannon [7], is used to measure the disorder or uncertainty of an event. Entropy of a data point x is specified in equation 6. We choose data points with highest predictive entropy (highest uncertainty). This acquisition function only applies for classification problem (output is discrete). In the case of continuous, we use differential entropy instead of entropy.

Maximizing Mean Variance [8]

$$\sigma(x) = \frac{1}{C} \sum_c (\mathbb{E}_{p(\omega|D)}[p(y = c|x, \omega)^2] - \mathbb{E}_{p(\omega|D)}[p(y = c|x, \omega)]^2) \tag{7}$$

Predictive variance of a data point x is defined in equation 7. We choose data points with highest predictive variance.

Maximizing expected reduction in posterior entropy (BALD) [9]

$$\mathbb{H}[\omega|D] - \mathbb{E}_{y\sim p(y|x,D)}[\mathbb{H}[\omega|y,x,D]] \tag{8}$$

The expected reduction in posterior entropy for a data point x is defined in equation 8. Following Houlsby et al [10] we can rewrite the above as follow:

$$\mathbb{H}[y|x,D] - \mathbb{E}_{\omega\sim p(\omega|D)}[\mathbb{H}[y|x,\omega]] \tag{9}$$

The equation above can be interpreted as choosing data point x for which the model is marginally most uncertain about y (high $\mathbb{H}[y|x,D]$), but individual setting of the parameters are confident (low $\mathbb{E}_{\omega\sim p(\omega|D)}[\mathbb{H}[y|x,\omega]]$) [10]. This can be interpreted as choosing data point x that the posterior disagree about the outcome the most. Hence, we will refer to this active learning objective as Bayesian Active Learning by Disagreement (BALD) [10].

Random Acquisition

Choose the next data point to label uniformly at random. We will use Random Acquisition function as a baseline to assess other functions' performance.

Note. In regression problem, it is common to define a Gaussian likelihood $p(y|x,\omega) \sim N(f^\omega(x), \sigma^2)$, where $f^\omega(x)$ is the output of the neural network and σ^2 is a constant noise. Under this setting, the two acquisition functions **maximizing mean variance** and **maximizing expected reduction in posterior entropy (BALD)** are equivalent.

4. Proposed acquisition function: Layer BALD

Motivation Bishop discussed in his book [11] about Local Variational methods, in which we try to approximate a subset of parameters in the model, instead of all parameters. Therefore, we would like to construct an active learning acquisition function that can help us greedily train the network layer by layer.

Extending on the work of Houlsby et al [10], we propose a new acquisition function using mutual information between prediction distribution and neural network layer posterior.

Let ω_i be the set of parameters (weights and biases) of layer i and let $\omega = \{\omega_i\}_{i\in[1,L]}$ be the set of all parameters in our model. We calculate

the mutual information between predictive distribution and layer i posterior, conditioning on all other layer posterior.

$$\mathbb{I}[y, \omega_i | x, D] = \mathbb{H}[y|x, D, \{\omega_j\}_{j \neq i}] - \mathbb{E}_{\omega_i \sim p(\omega_i|D, \{\omega_j\}_{j \neq i})}[\mathbb{H}[y|x, \omega]] \tag{10}$$

For each data point x, we compute the mutual information with respect to every layer in the network, we then choose the layer l with highest mutual information (highest expected reduction in posterior entropy), to perform inference on. We use conditional inference, updating only the posterior of layer l while fixing other layers' posterior.

$$q_\theta(\omega_l | \{\omega_i\}_{i \neq l}) q_\theta(\{\omega_i\}_{i \neq l}) \approx p(\omega|D) \tag{11}$$

Please refer to appendix B for experiment results with greedy layer-wise training with acquisition function layer-BALD.

5. Experimental results

Boston Housing Dataset

We will use Hernndez-Lobato et al. [12]'s neural network architecture for this dataset: Bayesian neural network with one hidden layer of 50 hidden units. We define Gaussian prior over its weights. We also use Gaussian likelihood with fixed noise variance since this is a regression problem. We will use Variational Inference, using a Gaussian distribution to approximate the posterior.

Active learning. We create a training set with 20 data points, a test set with 100 data points, and the pool set with 386 data points. We then run active learning 10 iterations. For each iteration, we add 2 data points from the pool set to our training set using a acquisition function, then initialize a new model and train from scratch using the new training set. This experiment is run 10 times and we measure the average mean square error.

Acquisition function	Mean Square Error
Maximum Predictive Variance	67.64429675
Random	78.94710625

Boston Housing Dataset Accuracy

With 40 data points in our training set (9.6% size of the original training set), we achieve an average MSE of 68 (correspond to RMSE 8). We then increase the number of hidden units in the neural network to 50. After 10 iterations with a total of 40 data points, we are able to achieve a MSE of roughly 30 (correspond to RMSE 5).

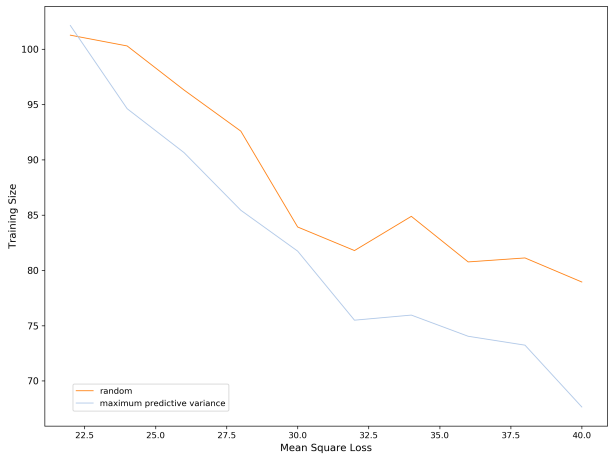

Figure 1: Mean square error of models with different training data sizes

MNIST dataset

LeCun et al. [13] proposed a neural network architecture with 2 hidden layers, 300 hidden units with mean square error loss function, which achieve 4.7% error rate. We use the same neural network as Lecun, and further define a Gaussian prior over all weights in the network, creating a Bayesian neural network. We use Variational Inference with Gaussian variational distribution to approximate the posterior.

Active learning. We create a training set of 100 data points, a test set of 10,000 data points, and a pool set of 59,900 data points. We then run active learning 10 iterations. For each iteration, we add 50 data points from the pool set to our training set using a acquisition function, then initialize a new model and train from scratch using the new training set.

With 600 data points, we can achieve an average accuracy of $87\% - 88\%$. We then double the active learning batch size from 50 data points to 100 data points. We found that after 10 iterations, with total 1100 data points in the training set, we obtain the accuracy as follow.

With only 1,100 data points in our training set (1.8% size of the original training set), the best acquisition function are able to obtain 89.74% accuracy (corresponding to 10.26% error).

Acquisition function	Accuracy
Random	83.56%
Maximum Entropy	85.80%
Maximum Mean Variance	86.85%
Maximum BALD	88.34%

MNIST Dataset Accuracy with 10 active learning iterations - batch size of 50

Acquisition function	Accuracy
Random	84.12%
Maximum Entropy	88.88%
Maximum Mean Variance	89.12%
Maximum BALD	89.74%

MNIST Dataset Accuracy with 10 active learning iterations - batch size of 100

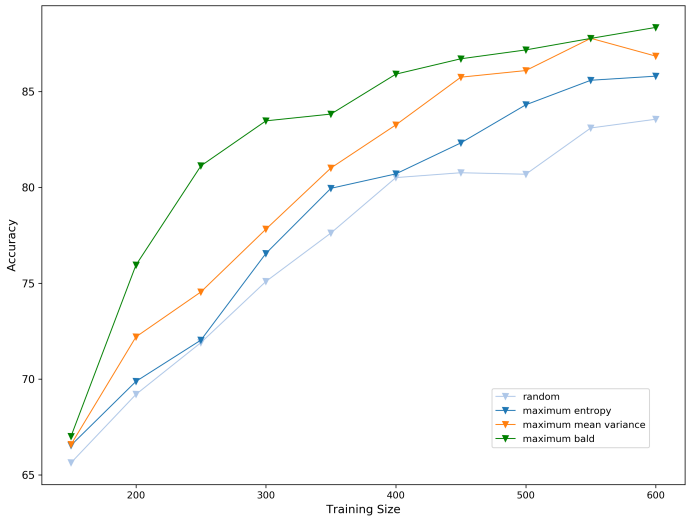

Figure 2: MNIST Dataset Accuracy with 10 active learning iterations - batch size of 50

We run another experiment with the same active learning setting, but this time with a Bayesian convolutional neural network. We use Keras's architecture of CNN on MNIST dataset: convolution$(32, 3, 3)$- relu-convolution$(64, 3, 3)$-relu-max pooling$(2, 2)$-dropout(0.25)-dense-relu-dropout(0.5)-dense-softmax. We use dropout to approximate Bayesian inference [1]. Result after 10 active learning iterations and batch size of 50, with a total of 600 data points.

Acquisition function	Accuracy
Random	87.89%
Maximum Entropy	93.96%
Maximum Mean Variance	94.24%
Maximum BALD	92.89%

MNIST Dataset Accuracy with 10 active learning iterations - batch size of 50

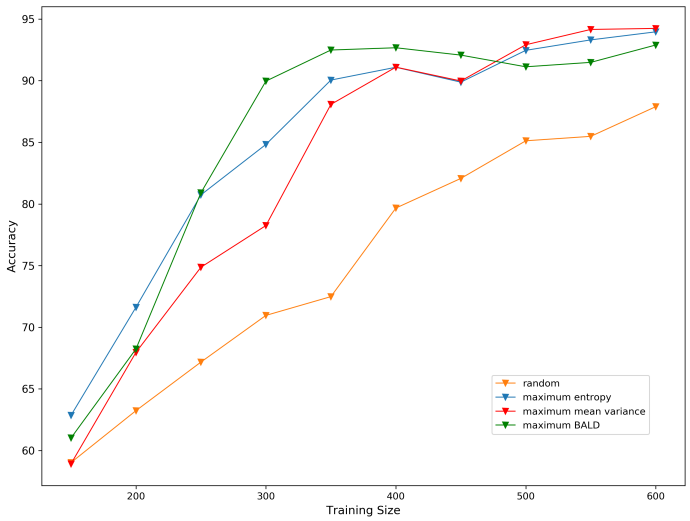

Figure 3: MNIST Dataset Accuracy with 10 active learning iterations - batch size of 50

We achieve an accuracy of 93% to 94% with active learning and 600 data points (1% size of the original training set). We then double the active learning batch size from 50 to 100 data points. After 10 iterations, with a

total of 1100 data points (1.8% size of the original training set), we achieve the result as follow.

Acquisition function	Accuracy
Random	94.84%
Maximum Entropy	96.34%
Maximum Mean Variance	96.89%
Maximum BALD	97.25%

MNIST Dataset Accuracy with 10 active learning iterations - batch size of 100

6. Conclusion

We evaluate the effect of being Bayesian and the use of active learning on neural network and convolutional neural networks. Our empirical results on the above datasets have shown that Bayesian approach with active learning allow the models to achieve high accuracy when trained with very little data, comparable to non-Bayesian model trained on the full original training set in some cases.

We want to highlight that all of our active learning acquisition functions make no implicit assumption about the model, in other words they are model-independent. This makes them not as powerful as some model-dependent acquisition functions (for e.g. Simple Margin, MaxMin Margin query methods for Support Vector Machine [2]). On the other hand, this also make them applicable to most machine learning model (not restricted to only neural network).

Appendices

A. Other experiments

We present other methods and experiments we have tried but did not achieve good results.

Deep BNN with Gaussian prior on MNIST

In feedforward neural network, Gaussian prior over weights are very popular in practice. In this experiment, we train our neural network using the whole training set of 60,000 data points (without active learning). We learned that neural network with two hidden layer often perform well. However, when

Architecture	Accuracy
$768 \times 64, 64 \times 10$	94.84%
$768 \times 128, 128 \times 10$	96.34%
$768 \times 300, 300 \times 10$	96.89%
$768 \times 700, 700 \times 10$	97.25%

MNIST Dataset Accuracy with 10 active learning iterations - batch size of 100

we work on neural network with three or more layers, the accuracy drops significantly. This phenomenon is also observed and discussed by Neal [14].

Greedy Layer-wise training with BNN on MNIST

Hinton et al. [15] and Bengio et al. [16] proposed two greedy layer-wise training algorithms on deep belief net and on deep neural network. Inspired by the idea, we hope to find a way to do layer-wise inference on a Bayesian neural network. Our hope is that doing variational inference on a subset of the parameters can be much faster compared to do inference on the whole models.

We test this idea on a Bayesian neural network with two hidden layers. We then implement active learning using our proposed acquisition function layer-BALD. In every iteration, we actively select the data and the layer with highest expected reduction in posterior entropy, then we fix other layer's distribution and do inference only on the chosen layer.

Our empirical results shows that this method is much faster compare to training on the whole network. However, it gives poor accuracy 64% with a total of 600 data points.

References

[1] Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: Representing model uncertainty in deep learning., International Conference on Machine Learn- ing (2016).

[2] S. Tong, D. Koller, Support vector machine active learning with applications to text classification, Journal of Machine Learning Research (2001).

[3] S. Tong, E. Chang, Support vector machine active learning for image retrieval, Proceedings of the ninth ACM international conference on Multimedia (2001) 107–118.

[4] R. I. Y.Gal, Z.Ghahramani, Deep bayesian active learning with image data (2016).

[5] S. I. Krizhevsky, A., G. E. Hinton, Imagenet classification with deep convolutional neural networks (2012).

[6] G. Z. Gal, Y., Bayesian convolutional neural networks with bernoulli approximate variational inference (2016).

[7] C. E. Shannon, A mathematical theory of communication, Bell System Technical Journal (1948).

[8] B. V. C. R. Kendall, A., Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding (2015).

[9] D. J. C. Mackay, Information-based objective functions for active data selection, Neural Computation 4 (1992) 589–603.

[10] H. F. G. Z. Houlsby, N., M. Lengyel, Bayesian active learning for classification and preference learning (2011).

[11] C. M. Bishop, Pattern Recognition And Machine Learning, Springer, 2006.

[12] A. R. P. Hernndez-Lobato, J. M., Probabilistic backpropagation for scalable learning of bayesian neural networks (2015).

[13] J. L. B. L. B. A. C. C. D. J. D. H. G. I. M. U. S. E. S. P. Lecun, Y., V. Vapnik, Comparison of learning algorithms for handwritten digit recognition (1998).

[14] R. M. Neal, Bayesian learning for neural networks (1995).

[15] O. S. Hinton, G. E., Y. W. Teh, A fast learning algorithm for deep belief nets, Neural computation (2006).

[16] L. P. P. D. Bengio, Y., H. Larochelle, Greedy layer-wise training of deep networks (2007).