Technical Milestone Report

Devang Agrawal, Queens College

January 14, 2016

Abstract

Bayesian Optimization is used for global optimization of functions which are typically expensive to evaluate. Despite the many successes of Bayesian Optimization in low dimensions, scaling it to high dimensions has proven to be notoriously hard. Existing literature on this topic make very restrictive assumptions. In this project we aim to develop a novel high dimensional Bayesian Optimization framework by using neural networks to find a low dimensional (nonlinear) manifold to describe the function. The manifold can then be used for Bayesian Optimization.

1 Introduction

In many applications we are tasked with the zeroth order optimization of an expensive to evaluate function f. Some examples are hyper parameter tuning in expensive machine learning algorithms, experiment design, optimizing control strategies in complex systems, and scientific simulation based studies. Bayesian optimization finds the optimum of f is found by using as few queries as possible by managing exploration and exploitation[11]. Scaling Bayesian Optimization to high dimensions for practical problems has been very challenging. Existing methods solve the problem only under very restrictive assumptions[14, 7]. The aim of this project is to develop a novel framework for high dimensional Bayesian Optimization which can be used for a much more expressive and richer class of functions, than existing methods.

Neural networks have previously been used for dimensionality reduction and representational learning[5]. This project aims to use neural networks to learn a low dimensional representation of the data. This representation can then be used to perform Bayesian Optimization.

2 Background and Related work

2.1 Neural Networks

Neural Networks use multiple non-linear transforms to map from an input to an output. This gives them excellent feature extraction properties and they define the current state of the art in object recognition and in natural language processing[2]. Neural networks have also been used to learn low-dimensional codes from high-dimensional data[5].

2.2 Bayesian Neural Networks

Bayesian methods have often been applied to neural networks[10]. The goal of Bayesian neural networks is to uncover the full posterior distribution over the network weights in order to capture uncertainty, to act as a regularizer, and to provide a framework for model comparison. The full posterior is often intractable for most forms of neural networks, however it can be possible to use full or approximate Bayesian inference for small pieces of the overall architecture[6].

2.3 Bayesian Optimization

Bayesian Optimization(BO) is a well-established strategy for global optimization of noisy expensive black-box functions[11]. Bayesian optimization relies on the construction of a probabilistic model that defines a

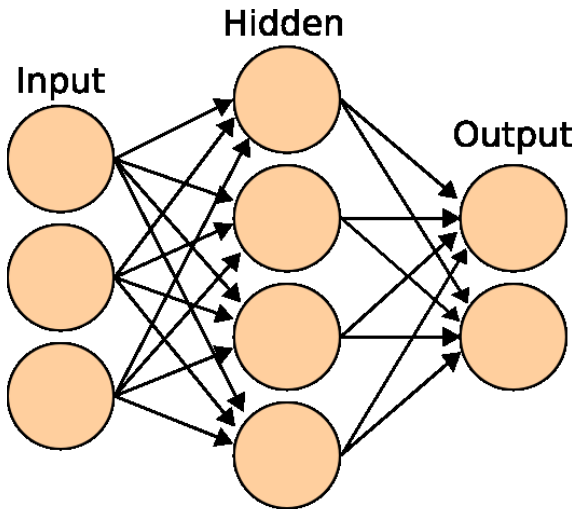


Figure 1: Illustration of a Neural Network with one hidden layer

distribution over objective functions from the input space to the objective of interest. The promise of every new experiment is quantified using an acquisition function, which, applied to the posterior mean and variance, expresses a trade-off between exploration and exploitation. Bayesian optimisation proceeds by performing a proxy optimization over this acquisition to determine the next input to evaluate.

2.4 High Dimensional Bayesian Optimization

Bayesian Optimization(BO) has been successfully applied to many applications such as tuning hyperparameters in learning algorithms[12], robotics[9], and object tracking[4]. However all such successes have been primarily limited to low (typically < 10) dimensions[14]. However scaling BO to high dimensions for practical problems has been challenging. Currently only two methods exist for doing BO in high dimensions under strong assumptions, which severely limit their applicability to practical problems.

Wang et al. 2013[14] perform regular BO on a low dimensional subspace. The assumption is that most dimensions do not change the objective function significantly and can be ignored. Their algorithm projects the high dimensional space down to a random low-dimensional space and performs BO there. Figure 2 illustrates this idea by using a D=2 dimensional function where only d=1 dimension is important. This method works very well only when its assumptions hold, however that makes it unsuitable for general high dimensional BO.

Kandasamy et al. 2013 [7] perform high dimensional BO by treating the function f as an additive function of mutually exclusive lower dimensionality components. This method also shows excellent performance only when its strong assumptions hold.

$$f(x) = f^{(1)}(x^{(1)}) + f^{(2)}(x^{(2)}) + \ldots + f^{(M)}(x^{(M)})$$

3 Proposed Method

The project aims to develop a general framework for performing BO in high dimensions. For this a hybrid network is used, we add a Gaussian Process(GP) or a Bayesian Linear Regressor(BLR) to the last hidden layer of a deep neural network. The neural network reduces the dimensionality of the data by finding a low-dimensional non-linear manifold to represent it.

to the GP or BLR. It can also potentially pick out interesting features from the data. Figure 3b shows an illustration of the hybrid network. To train the hybrid network, we first train the Neural Network from X to Y using back-propagation. Then we fix the network and train a GP or a BLR from the last hidden layer of the network (H) to Y2. The GP/BLR gives provides a probabilistic model of the target function, the uncertainty estimates from it can allow us to manage exploration and exploitation for Bayesian Optimization.

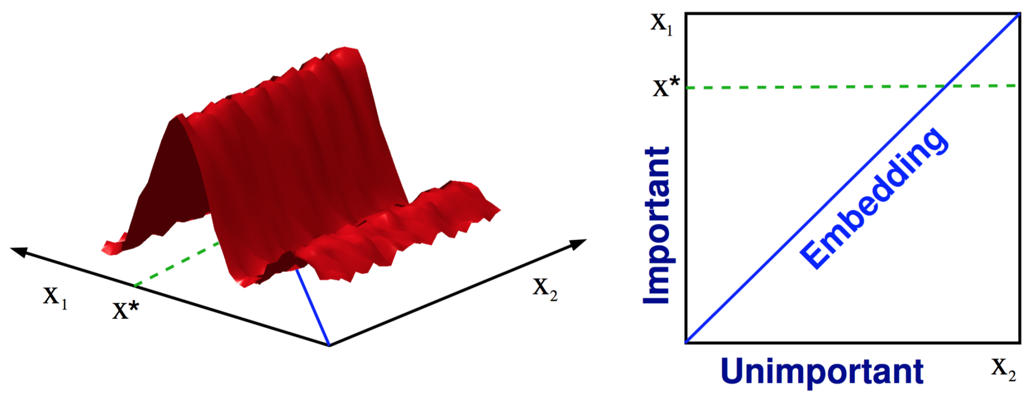
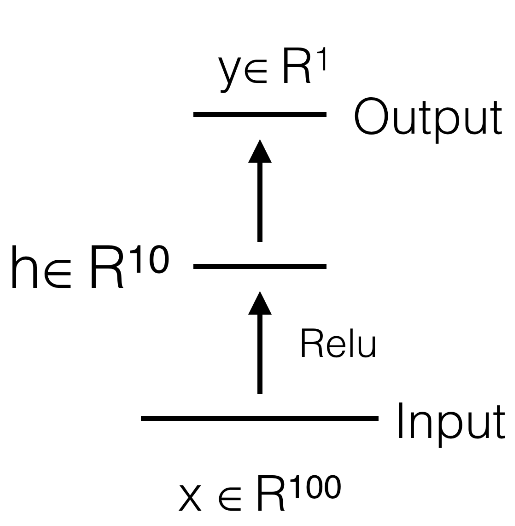
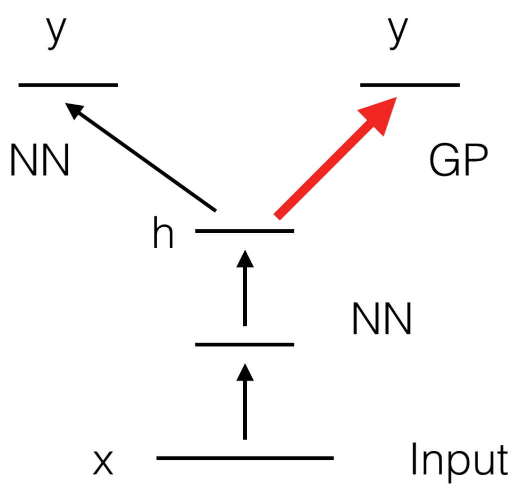


Figure 2: This function in D=2 dimensions only has d=1 effective dimension: the vertical axis indicated with the word important on the right hand side figure. Hence, the 1-dimensional embedding includes the 2-dimensional function's optimizer. It is more efficient to search for the optimum along the 1-dimensional random embedding than in the original 2-dimensional space. [14]



(a) Simple Neural Network used in experiments



(b) An illustration of the hybrid network

Figure 3: Neural Network Architectures used

3.1 Current Work

Theano[3][1] is an open-source numerical computation library for python. It performs symbolic differentiation and can easily run on GPUs allowing fast training of deep neural networks. It is extensively used by the Neural Network communities and has extensive documentation available. All experiments for this project were written in Theano.

As Bayesian Optimization aims to optimize expensive functions with minimum number of function evaluations, only a small number of data-points are available to train the neural network. This poses a challenge for Neural networks as they are generally used in scenarios with a large number of data-points. The performance of neural networks with very limited data-points is investigated in this report.

A synthetic dataset was created using a fixed neural network with randomly initialized weights. The neural network uses ReLU non linearity[2]. It takes a 100 dimensional input and gives a one dimensional output, it has one hidden layer with 10 hidden units in it. The architecture is illustrated in Figure 3a. Input data points $\mathbf{X} \in R^{100}$,$X = [x_1 \ldots, x_{100}]$ where $x_i \sim unif[-1,1] \forall i \in [1,100]$, the dataset was then generated by applying the neural networks to them to get the output points.

A neural network with the same architecture as Figure 3a was then trained on the synthetic data set. Figure 4a shows the evolution of test error with epochs for different number of training points, all the errors converge after 1000 epochs. All subsequent experiments used 1000 epochs. To better understand the performance of the neural networks with very small nuber of data-points, only a portio of the training data set is used to train the networks. In Figure 4c we plot the error versus the size of the portion of the data-set that was used to perform the training, separate curves are obtained by varying the amount of L2 regularization of the weights of the neural networks. A reference error was found by using simple mean prediction on the test data set, this was then added to the figures. As can be seen from the figure, with aggressive L2 regularization($\tilde{}$ 0.01) , the test error with 200 data-points is 0.31, which is significantly better than the reference error (0.73).

The effect of the width of the hidden layer on the performance of the neural networks was explored. Figure 4c shows that the curves obtained for different values of `hiddenWidth` (width of hidden layer) are very similar. The curves for `hiddenWidth=20 and hiddenWidth=10` are almost identical. The width of the hidden layer is chosen to be 10 for the rest of the experiments.

When training models on very small datasets, overfitting is the primary concern and hence regularization becomes very important. Apart from the L2 regularization of network weights, we also investigate random projections as a method of regularization. The 100 dimensional input space is projected down to a 50 dimensional subspace using a fixed, random linear transform. The learning is then done in the 50 dimensional sub-space. This approach did not show very good results as can be see in Figure 4d. One possible explanation for such performance is the complexity of the synthetic dataset which might have strong dependencies on all the input dimensions and hence can not tolerate an arbitrary random projection. More tests are proposed with different datasets to better understand the efficacy of random projections.

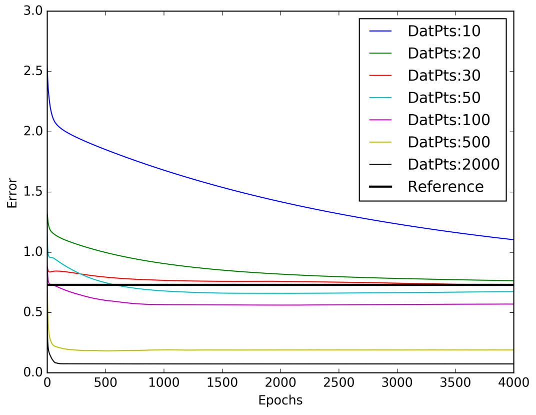
3.2 Future Work

For the ultimate success of the project in developing a novel high-dimensional Bayesian optimization framework, the following work needs to be completed.

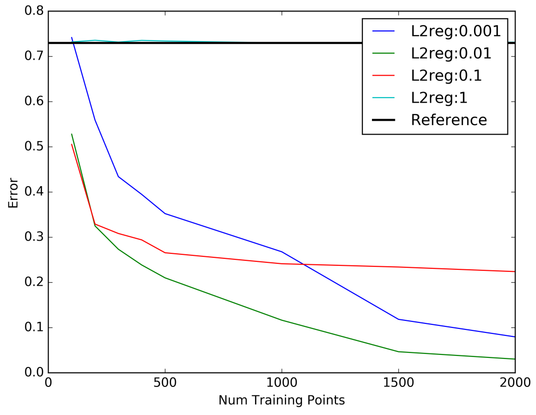
The neural networks ability to model the function and to reduce its dimensionality needs to be further improved. We have observed reasonable results in our current work, but experiments with various neural network architectures and training methods needs to be explored to further improve our performance. We believe that using deeper neural network architectures might help improve our performance, however we need to tread cautiously since deeper networks can substantially increase the risk of overfitting on small datasets such as ours. Different optimization strategies such as Stochastic Gradient Descent with momentum[2], ADA-DELTA[15] , ADAM[8] etc. might be able to give better performance in training the networks.

The performance of the neural network on a "real world" data set also needs to be evaluated. Real-world datasets can often be "well behaved" which can make the task of modeling them much easier than the highly non-linear and arbitrary synthetic data set used in current experiments.

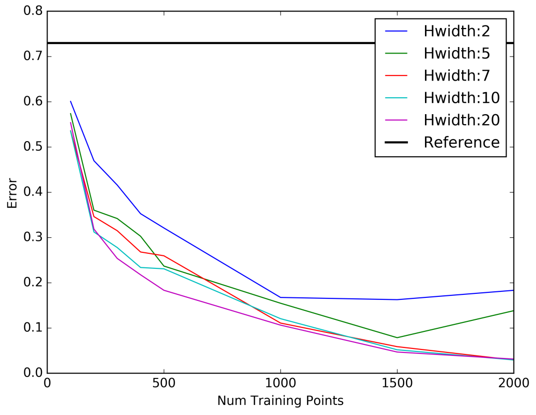
Once the neural network performance is acceptable, the hybrid network and the Bayesian optimization needs to be implemented. The model then needs to be benchmarked on some popular high dimensional



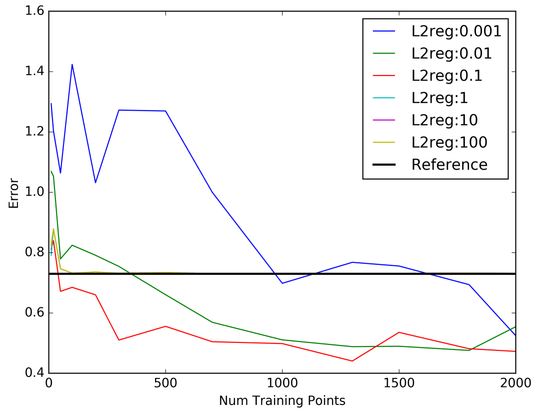
(a) Evolution of test error with epochs, for different number of training data points



(b) Error for different number of Training points for different amounts of L2 regularization



(c) Error for different number of Training points for different widths of the hidden layer



(d) The input space was first linearly projected down to a 50 dimensional sub-space

Figure 4: Investigating Neural Networks on Small Datasets

functions so that its performance can be compared to some existing high dimensional BO algorithms such as REMBO[14] and Add-GP-UCB[7].

4 Conclusion

In the work so far, very simple neural network based models have shown promising performance on a very complex synthetic dataset. Future work will try to improve this performance by using more complex networks. This performance will then be validated on “real-world” datasets. A Gaussian process will then be implemented, with the last hidden layer of the Neural Network as the input, to model the target function. The uncertainty estimates from the Gaussian process should enable us to do Bayesian Optimization on the target function.

References

[1] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[2] Yoshua Bengio, Ian J. Goodfellow, and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2015.

[3] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: A cpu and gpu math compiler in python. In *Proc. 9th Python in Science Conf*, pages 1–7, 2010.

[4] Misha Denil, Loris Bazzani, Hugo Larochelle, and Nando de Freitas. Learning where to attend with deep architectures for image tracking. *Neural computation*, 24(8):2151–2184, 2012.

[5] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[6] Geoffrey E Hinton and Ruslan R Salakhutdinov. Using deep belief nets to learn covariance kernels for gaussian processes. In *Advances in neural information processing systems*, pages 1249–1256, 2008.

[7] Kirthevasan Kandasamy, Jeff Schneider, and Barnabas Poczos. High dimensional bayesian optimisation and bandits via additive models. 03 2015.

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

[9] Daniel J Lizotte, Tao Wang, Michael H Bowling, and Dale Schuurmans. Automatic gait optimization with gaussian process regression. In *IJCAI*, volume 7, pages 944–949, 2007.

[10] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

[11] Jonas Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of bayesian methods for seeking the extremum. *Towards Global Optimization*, 2(117-129):2, 1978.

[12] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[13] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Md. Mostofa Ali Patwary, Prabhat, and Ryan P. Adams. Scalable bayesian optimization using deep neural networks. 02 2015.

[14] Ziyu Wang, Masrour Zoghi, Frank Hutter, David Matheson, and Nando de Freitas. Bayesian optimization in a billion dimensions via random embeddings. *arXiv preprint arXiv:1301.1942*, 2013.

[15] Matthew D Zeiler. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.