Bayesian Optimization for Hyperparameter Tuning in the Fully Connected Layers of VGG16

Søren Winkel Holm Oskar Wiese Anders Henriksen
Anne Agathe Pedersen

February 27, 2020

Abstract

Optimization of neural network hyperparameters is a costly process usually pervaded by uncertainty. This report combines Bayesian optimization with Gaussian processes to find optimal hyperparameters for the VGG16 image classification network when training on the CIFAR10 dataset.

1 Introduction

Humans are slow and faulty. This means that work costs heaps of money and mistakes happen often. To help mitigate this, machine learning can learn the task at hand in order to optimize the performance at a much lower cost. While effective for many applications, machine learning has one glaring issue that requires skill and tonnes of computing power to overcome; hyperparameter optimization. Optimizing the hyperparameters usually devolves into random guessing, though new methods are on the rise like Bayesian optimization, which greatly reduces the effort involved in finding the parameters with use of an acquisition function and probabilistic model.

In this paper, Bayesian optimization with Gaussian processes as probabilistic model and expected improvement, upper confidence bound and probability of improvement as acquisition function will be used to find the hyperparameters of the VGG16 classifier network when training to classify on the 10 classes of the CIFAR10 dataset, optimizing for the validation accuracy.

2 Methods

Gaussian process and Bayesian optimization...

Three different aquisition functions + random...

Gaussian Process

Acquisition Functions

In the following section three different acquisition functions will be presented. All of these acquisition functions are implemented by using the GPyOpt library.

Probability of Improvement
Probability of improvement evaluates *f*, the objective function, at the point most likely to improve the minimum value. In this project, the objective function returns the negative validation accuracy. Probability of improvement then evaluates at the point most likely to obtain a higher accuracy, since the the function is given the negative accuracy. Mathematically PI can be expressed as,

$$\mathrm{PI}(\mathbf{x}) = \mathrm{P}(f(\mathbf{x}) \geq f(\mathbf{x}^+) + \xi) = \Phi\left(\frac{\mu(\mathbf{x}) - f(\mathbf{x}^+) - \xi}{\sigma(\mathbf{x})}\right)$$

where ξ is a hyperparameter which can be set by the user. The hyperparameter is a trade off between exploration and exploitation. Φ is the cumulative distribution function of the Gaussian distribution.

Expected Improvement
The expectorated improvement tries to quantify the improvement instead of the probability of improving as Probability of Improvement does. It is possible to quantify the average value of improvement if we sample the objective function at x. The acquisition function then computes the expected value of the improvement function at a certain x. Mathematically EI can be expressed as,

$$\mathrm{EI}(\mathbf{x}) = \begin{cases} \left(\mu(\mathbf{x}) - f\left(\mathbf{x}^+\right)\right)\Phi(Z) + \sigma(\mathbf{x})\phi(Z) & \text{if } \sigma(\mathbf{x}) > 0 \\ 0 & \text{if } \sigma(\mathbf{x}) < 0 \end{cases}$$

Here the variable $Z = \frac{\mu(\mathbf{x}) - f(\mathbf{x}^+) - \xi}{\sigma(\mathbf{x})}$

Gaussian Process Upper Confidence Bound
The idea behind the GP-UCB is to directly use the mean and variance from the Gaussian distribution. These two parameters are used as the exploration / exploitation trade off. The function UCB (Ubber Confidence Bound) can be formulated as the following,

$$UCB(x) = \mu(x) + \kappa\sigma(x)$$

Where κ is a user defined variable used to control the trade off between exploration and exploitation.

3 Results

Comparison of aquisition functions and random...

	Hidden units	Dropout probability	Activation function	Validation loss
EI	1300	0.3748	Sigmoid	0.6635
MPI	3500	0.5982	Sigmoid	0.654
LCB	2900	0.5835	Sigmoid	0.651

Results and plots...

4 Discussion

Which one is better...

5 References

`https://gpyopt.readthedocs.io/en/latest/index.html`

6 Appendix

Hidden units	Dropout probability	Activation function	Validation loss
3500	0.4424	Sigmoid	0.6415
600	0.0536	ReLU	0.6135
800	0.0243	Sigmoid	0.624
900	0.5360	Tanh	0.621
900	0.6065	Tanh	0.6215
3500	0.2876	Sigmoid	0.639
3500	0.4812	Sigmoid	0.649
3500	0.5982	Sigmoid	0.654
3500	0.7073	Sigmoid	0.635
3500	0.6367	Sigmoid	0.6455
3500	0.7209	Sigmoid	0.6425
3500	0.8136	Sigmoid	0.6045

Hidden units	Dropout probability	Activation function	Validation loss
3300	0.2075	ReLU	0.6190
2900	0.4818	Sigmoid	0.6355
1700	0.3773	ReLU6	0.5955
3400	0.5452	Sigmoid	0.6275
1800	0.4797	ReLU	0.6130
2900	0.4818	Sigmoid	0.6340
2900	0.4891	Sigmoid	0.6420
2900	0.4947	Sigmoid	0.6365
2900	0.5835	Sigmoid	0.6510
2900	0.6168	Sigmoid	0.6365
2900	1.0000	Sigmoid	0.0965
2900	0.5604	Sigmoid	0.6350

Hidden units	Dropout probability	Activation function	Validation loss
2900	0.2120	Sigmoid	0.6535
1300	0.1657	ReLU	0.6465
200	0.9318	Tanh	0.3350
2200	0.7223	ReLU6	0.4905
1200	0.0595	ReLU6	0.6355
2900	1.0000	Sigmoid	0.1350
2900	0.1885	Sigmoid	0.6545
1300	0.5317	Sigmoid	0.6650
1300	1.0000	Sigmoid	0.0990
1300	0.3748	Sigmoid	0.6635
1200	0.4330	ReLU6	0.6145
1200	0.9814	ReLU6	0.0970