Practical Implementation of Gaussian Process Based Optimization in Convolutional Neural Network

Amey Athale[1] Ankush Tale[1] Uday PB[1] Shubham Shah[1]

[1]Arizona State University

{aathale, avtale, spshah23, uprativa}@asu.edu

Abstract

A machine learning model tries to find the right function that describes the given input data. Usually with a vanilla model we get a base level of accuracy, and in order to improve the accuracy we try to define hyper-parameters and optimize them that fine-tune the model. This allows the model to reach the global minimum on loss function in a more efficient way. There different approaches of optimizing the hyper-parameters. In this paper we compare two such approaches viz. Gaussian process based optimization and Grid search. The work [1] intends to demonstrate that hyper-parameter tuning using Gaussian Processes helps get better and quicker results when compared to Grid Search.

1. Introduction

Almost every machine learning model has some additional parameters which can be utilized to fine tune its performance. In this paper we discuss the effect of learning rate, learning decay, mini batch size, and dropout rates on every layer of a convolutional neural network. Our ideal goal is to obtain a set of these hyper-parameters which minimze the loss funtion in an efficient manner. This efficiency can be measured using different metrics like accuracy, precision, recall and f1 score. Finding a right set of parameters can be either done by a systematic search of the hyper parameters in a definite search space which is implemented by a brute-force method or by using Gaussian processes. Recently various strategies have been used for tuning the parameters of the machine learning algorithms. Bergstra et al [2] has demonstrated that grid search strategies are inferior to random search, and suggested the use of Gaussian process.

[1]Code is available at https://github.com/ankushtale/Bayesian_Global_Optimization_in_CNN

1.1. Grid Search

Grid search is one of the most fundamental methods to perform hyper-parameter optimization. It is essentially an exhaustive searching method through a manually specified subset of hyper-parameter space of an algorithm. This approach suffers from polynomial time complexity since we are iterating over each set of hyper-parameter values.

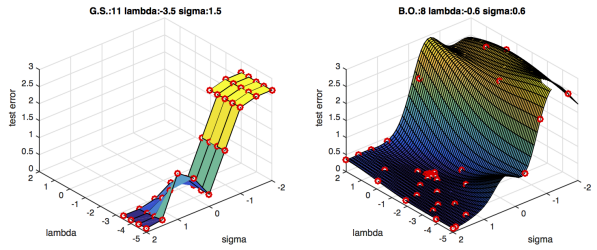

Figure 1. Comparison of grid search and Gaussian optimization approaches for 40 hyper-parameter value sets. [2]

1.2. Gaussian Process

Weinberger [2] discussed in his lecture that to learn the parameters of a classifier, generally 2 approaches are taken. First is the Maximum Likelihood, where we calculated the likelihood($p(D|w)$) of the probability density function and found the parameters which maximized it. He also discussed the method of MAP(Maximum A Posterior), where we turn things around and calculate $p(w|D)$, in which we assume the prior is also a Gaussian Distribution. Now, both of them required us to know the distribution of w. Instead, for a test point, if we wanted to calculate $p(y|x, D)$, we can simply use $p(y|x, w)$ and $p(w|D)$ and then marginalize it over w. This means, we can actually get the distribution of the prediction without knowing the distribution of w. From the above two approaches, we know $p(y|x, w)$ and $p(w|D)$ both are Gaussians, hence the resulting $p(y|x, D)$ must also be a Gaussian distribution.

Essentially:

$$p(y|x, D) \sim N(\mu, \sigma) \tag{1}$$

where in Eq 1, y is the loss function for the model, x is the set of hyper-parameters and D is the hyper-parameter search space. Experimentally, Weinberger [2] with the help of figure [2] explains how Gaussian process based optimization reaches the optimum hyper-parameter values with significantly lesser search points as compared to grid search.

2. Related work

Gaussian Process for Machine Learning An introduction to the basis of Gaussian processes and several examples of the processes are given in [3] where M. Seeger et al. [3] mentioned that Gaussian processes are primarily used in situations where the postulated relationship are usually represented by an ensemble of functions. They further extended that for any Gaussian process, there are two views that can be considered, namely: the process and the weight space view. The weight space view allows us to relate GP models to parametric linear models. From section 2.4 of [3], it can be inferred that although the weight space view relates non-parametric GP models with parametric linear models directly, there are differences that justify a slightly indirect relation. Gaussian processes are intuitively complex which further has led to lesser research when relatively compared to other techniques. However, there have been quite significant contributions to this field which are listed in the following sections.

Practical Bayesian Optimization of Machine Learning Algorithms Snoek et al. [1] presents methods for performing Bayesian optimization for hyper-parameter selection of general machine learning algorithms. The paper introduced a fully Bayesian treatment for Expectation Improvement, and algorithms for dealing with variable time regimes and running experiments in parallel. The effectiveness of it's approaches were demonstrated on three challenging recently published problems spanning different areas of machine learning.

3. Method

Tuning the parameters can be quite a difficult task as it is expected to find a solution from infinitely spread search space. Grid search particularly follows a brute force approach for a randomly selected set of values provided externally which may or not may not cover or summarize the whole search space. the ultimate aim is to build a non-parametric machine learning technique which works on top of the machine learning techniques that contain the specified hyper-parameters to be optimized.

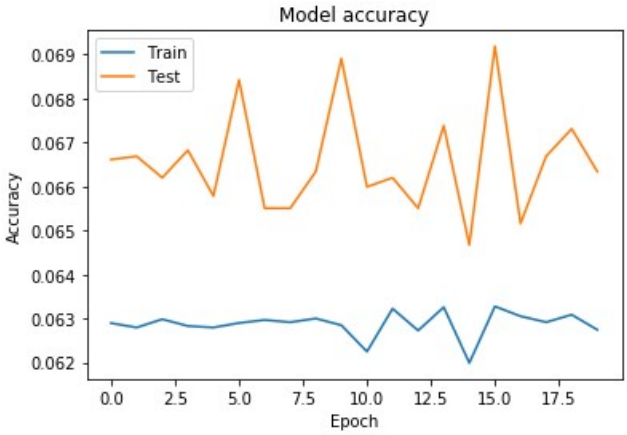

Figure 2. Accuracy over 20 epochs

In this section the technique which works on top of the Machine Learning techniques that contain the specified hyper-parameters is discussed.

3.1. Grid Search

The traditional way of performing hyper-parameter optimization has been grid search, or a parameter sweep, which is simply an exhaustive searching through a manually specified subset of the hyper-parameter space of a learning algorithm.

Since the parameter space of a machine learner may include real-valued or unbounded value spaces for certain parameters, manually set bounds and discretization may be necessary before applying grid search.

Grid search suffers from the curse of dimensionality as the time complexity is but is often parallel because the hyperparameter settings it evaluates are typically independent of each other.

4. Experiments

We implemented Grid search with the following values:

1. Starting Learning Rate η_0 = $[0.001, 0.01, 0.05, 0.1]$
2. Decay δ = $[0.0000001, 0.000001, 0.0001]$
3. Batch Size B= $[256, 512, 1024]$
4. Dropout p_1 = $[0.1, 0.25, 0.5]$
5. Dropout p_2 = $[0.1, 0.25, 0.5]$

The learning rate η would be

$$\eta = \frac{\eta_0}{(1 + \delta * t)} \tag{2}$$

In Eq 2, t is the epoch number. For the given hyper-parameter space Table [1] shows the values for grid search experiment which resulted in better accuracy.

4.1. Datasets

SVHN It is a popular real-world image dataset proposed by [4] which includes original, variable-resolution, color house-number images with character level bounding boxes as mentioned in [4]. It has two available formats, one with character level bounding and the other with images similar to MNIST dataset with 32-by-32 images.

4.2. Experimental Setup

Convolutional Neural Network Architecture: A Convolutional Neural Network with 3 convolution layers followed by 3 fully connected layers was utilized. More precisely, the first layer consisted of 32 filters of size 5 x 5 with zero padding and stride 1. Consequently, the second layer had 64 filters of size 3 x 3 with zero padding and stride 1. And the third convolution layer had 128 filters of size 3 x 3 with zero padding and stride 1. ReLU activation was applied to layers except the last layer which had softmax activation.

Training: The entire architecture for the preliminary implementation of grid search was implemented as per Prof Hemanth's requirements. Additionally, for better understanding of how neural networks work and what each layers do, a couple of pooling layers were also added. Firstly, we used Xavier uniform weight initializer to maximize the chances of reaching global minima. For error, we used the categorical cross-entropy loss function and as per what was discussed in class, we maximized our probability of getting maximum accuracy by using the Adam Optimizer.

Results: The above given architecture, when tested against 20% of data, gave an unsatisfactory accuracy of around 19.5% as seen in Table [1]. As this kind of result was not expected, we tested the same data against a standard architecture, namely LeNet-5, which gave a satisfactory accuracy of approximately 90%. After experimenting with weight and bais initializations along with other optimizers, it was evident that not only hyper-parameters but additional pre-processing steps would be essential for increasing the accuracy. The training and validation accuracy over 20 epochs also didn't show any significant increase in trend as seen in Figure [2]. This might be a result of the model getting stuck in local minima. Further experiments are essential to confirm these hypothesis.

5. Conclusion

After experimenting with a limited set of hyper-parameters values it was evident that the gird search method has higher space and time complexity and that there is a need for better parameter optimization method.

LR	Batch Size	Decay	DR1	DR2	Accuracy
0.001	256	1e-07	0.1	0.1	0.1958
0.001	256	1e-06	0.25	0.25	0.1958
0.001	512	1e-06	0.25	0.1	0.1593
0.001	512	1e-04	0.25	0.1	0.1107
0.001	256	1e-07	0.25	0.1	0.0969

Table 1. Various vales of accuracy based on Grid Search approach for hyper parameter tuning. Abbreviations LR - Learning Rate, DR1 - Dropout Rate P1, DR2 - Dropout Rate P2

6. Division of Work

Ankush Tale: I worked with Amey on writing the code for CNN training, documenting it on GitHub and experimenting with different models, random hyper-parameters contributed. I also proof-read the report and contributed to Experimental Setup section.

Uday PB: Uday took lead in drafting the report. He also studied a few Gaussian process implementations from other reference papers and wrote majority of the report.

Shubham Shah: Shubham was the main point of contact for Gaussian Processes based optimizations. He extensively studied the references and wrote the Abstract and Introduction section of the report.

Amey Athale: Amey and I paired in writing the code and he implemented the Grid Search method. He also studied the references and contributed in Related Work section.

7. Self-Peer evaluation

Ankush - XX	Uday - XX	Shubham - XX	Amey - XX

References

[1] H. L. J. Snoek and R. P. Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, pages 2951–2959, 2012. 2

[2] K.Weinberger. Bayesian global optimization. *http://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote15.html*, 2018. 1, 2

[3] M. Seeger. Gaussian process for machine learning. pages 3–10, 2004. 2

[4] A. C. A. B. B. W. Y. Netzer, T. Wang and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. *Available: http://ufldl.stanford.edu/housenumbers/*, 2011. 3