Hyperparameter Tuning Using Gaussian Processes In Convolutional Neural Network

Aditya Deotale, Anik Pait, Gokulan Vikash Babu, Amogh Bhabal
adeotale@asu.edu, apait@asu.edu, gvikashb@asu.edu, abhabal@asu.edu
Arizona State University
Tempe, AZ-85281

Abstract

Convolutional neural networks (CNN) have gained popularity in the field of image recognition and classification. One of the challenges associated with neural networks or different machine learning models is to find the set of hyper-parameters that will improve the accuracy of the model and give the best results. Different techniques have been used in the past to find the set of optimal hyper-parameters. The paper compares grid search and Gaussian Process search to find the optimal hyperparameters for our image recognition CNN model that classifies cropped color digits from the SVHN dataset [6].

1. Introduction

This study focuses on using Gaussian processes to find optimal hyper-parameters for a CNN model. The project will be focussing on building a CNN model for the image classification of Street View House Number (SVHN) dataset [6] that will classify the cropped color digits. It is similar to the MNIST [5] dataset but takes into account the real world problems associated with the images. The project will focus on optimizing the following 5 hyper-parameters:

- Starting learning rate: η_0
- Decay: δ where $\eta = \frac{\eta_0}{(1+\delta * t)}$
- Mini-Batch size: *B*
- Dropout parameter: *p1* for the first fully connected layer
- Dropout parameter: *p2* for the second fully connected layer

The goal of this project is to find a configuration of the above mentioned hyper-parameters using Gaussian process that will give us the best accuracy on the validation set. We will also find the optimal configuration of hyper-parameters using grid search and demonstrate that the process of finding the optimal configuration of hyper-parameters is more time efficient in Gaussian Process than in the grid search. Gaussian processes has been proved to be efficient in terms of sensitivity and calibration in experiments. It is similar to Bayesian optimization from reaching priors to posterior probability. Different regions of parameter space may have different number for execution time, hence using expected improvement time per second is a preferable measure to improve the wallclock time. The right choice of gaussian priors and acquisition function plays a vital role in expressing the assumptions about optimized functions.

2. Related Work

Grid search is an iterative algorithm and important for hyperparameter tuning but why it is important? According to paper [8], Hyperparameter tuning gives better performances in Super learner than an untuned model. A super learner is an ensemble learning which constitutes of different machine learning algorithms coming together to make a prediction function and contains contributions of each of the algorithm in the final prediction. To demonstrate the importance of tuning the hyperparameters, a case study was held by a group of students interested in the healthcare industry, containing a super learning analysis on the Predictive modelling. The aim was to predict on whether the anti depressant drugs were being preferred by the primary physicians on other indications than the depression. Two Super learners were used with five different machine learning algorithms: Neural networks, Support Vector Machine, Decision Tree, Random forests and LASSO Regression with one with tuned hyperparameters of the individual algorithms and the other with using the default hyperparameters values (using the Superlearner R package). The data was divided into training set, validation set and the testing set(independent) and testing was done with different model parameters using cross validation. The features were normalised before training and grid search was used to come up with different hyperparamaters values and find the most optimal value with the highest testing accuracy. Brier score and Concordance Statistic were used with 95% confidence interval to measure the performance of the super learner on the binary classification. The tuned super learner performance was compared to the performance of another study where a multivariate logistic regression model, carefully constructed/tuned, was used on the same data. The findings were interesting to the fact that the Brier score for a tuned super learner was 0.322 (with a 32% reduction in mean squared error) compared to random classification with the default values with a score of 0.309.Although there was no much difference between the logistic regression model and the tuned super learner but there were hyperparameters values whose tuned values coincided with default hyperparameter values. But it was found that models with those configurations didnt perform well and the models with different tuned hyperparameters gave better efficiency.

Artificial Neural networks have achieved state-of-the-art results in many tasks as they do not require manually engineered features. However, they do need a set of hyperparameters that directly affect the model's performance. Hyperparameters are usually chosen by manual, grid or random search. These techniques are time consuming and require expensive computations. Bayesian optimization based Gaussian Process(GP) surpasses these techniques by following a systematic way to identify the hyperparameters. In GP approach, the model's performance is extracted as a sample from a GP, for each hyperparameter combination. The paper [3] focuses on deriving the optimal or near-optimal hyperparameters for Artificial Neural Networks model that classifies dialog acts as per the utterance. The model takes as input word vectors $w_{1:l} \in \mathbb{R}^m$ and convolutes it with a filter matrix $v_{1:h} \in \mathbb{R}^m$ to get a scalar feature,

$$c_t = tanh(\sum_{i=1}^{h} v_i^T w_{t+i-1} + b_f)$$

where $b_f \in \mathbb{R}$ is a bias term. This convolution is performed with n different filters to get a vector $c_t \in \mathbb{R}^n$. The convolution process is repeated to generate $t - h + 1$ convolutions. This is supplied to the max pooling layer to generate utterances. The utterances u_i are then fed to a feed forward NN to classify them to *k* classes, where each class is a dialog act. This ANN model uses 5 hyperparameters - the filter size *h*, the number of filters *n*, the dropout rate *p* and history sizes *d1,d2*. A set of hyperparameter combinations, say , are mapped to F1-score(on test set) like $f : \chi \rightarrow \mathbb{R}$. The paper assumes a prior distribution on the function $x \in \chi$, which allows to consider a probabilistic model for *f* using all the previous evaluations. The model for *f* is then used to compute the optimal hyperparameter combination to evaluate next.

Several computer vision algorithms rely on the hyperparameters for the best performance. Generally, these hyperparameters are manually tuned while evaluating the algorithms' performance measures on a specific dataset. Determining a model's performance based on it's algorithm and the hyperparameters, for different problem domains with their datasets is a difficult task. [2] proposed a state of the art approach to support automated tuning of hyper-parameters, that was tested on computer vision problems such as object detection and face identification task. The goal of this approach was to replace the manual tuning process with an unbiased automated process of optimizing the algorithm. The approach first involves specifying the hyperparameters within the search space using a "null distribution specification language". The null prior distribution for the search problem will be randomly sampled to obtain a configuration of parameters. A loss function is calculated for this configuration with the image dataset, and this information will be stored in a database as a history of loss functions associated with the configuration of parameters. A "Hyperparameter Optimization Algorithm" takes in the null prior distribution and history of loss functions as input, and provides suggestions on which configuration of parameters to try next. The loss function for this configuration is calculated, and depending upon whether the loss function is minimized or not, the process is either stopped and the configurations are selected, or they are recorded in the database, and are used by the Hyperparameter Optimization Algorithm to suggest a new configuration. The goal is to find a configuration of parameters with the minimum loss function, that would bring the algorithm to its highest potential. The Hyperparameter Optimization Algorithms tested were the Tree-structured Parzen Estimators and Random Search.

3. Methods

There are two main methods to tune the hyperparameters of the model to which our paper focuses:

3.1. Grid Search

Grid Search is an iterative process of searching for the optimal hyperparameter values by taking different combinations of multiple hyperparameters similar to brute force search, and come up with the most optimal hyperparameter value that gives the highest accuracy on the testing dataset or the validation dataset. In case of a convolutional neural network, there are multiple hyperparameters such as number of neurons per layer, number of hidden layers, padding factor and pooling factor, etc. Grid search has some benefits and as a result has been quite popular and widely used for finding optimal hyperparameters. It is very simple to implement without any technical overhead and very reliable in lower dimensional spaces [1]. Grid search trials are conducted by combining every possible value of the hyper-parameters. So the number of trials in a gird search is product of number of hyper-parameters. This results in curse of dimensionality because the number of trials grows exponentially with the number of hyper-parameters.

3.2. Gaussian Process Search

A GP $f(x)$ is defined as a collection of random variables and is specified by its mean function $m(x)$ and covariance function $\kappa(x, x^{'})$, also called as kernel.

$$m(x) = [f(x)]$$

$$\kappa(x, x^{'}) = [(f(x) - m(x))(f(x^{'}) - m(x^{''}))]$$

In our case, $f(x)$ is the testing accuracy on the test set evaluated for the CNN model using the given hyperparameter combinations, which is a 5-dimensional vector consisting of starting learning rate η_0, decay δ where $\eta = \frac{\eta_0}{(1+\delta * t)}$, mini-Batch size *B*, dropout parameter *p1* for the first fully connected layer, dropout parameter *p2* for the second fully connected layer.

$$\text{Let } X = (x_1, x_2, ..., x_p),$$

$$\mathbf{f} = (f(x_1), f(x_2), ..., f(x_p))$$

$$\text{and } X^* = (x_{p+1}, x_{p+2}, ..., x_s),$$

$$\mathbf{f}^* = (f(x_{p+1}), f(x_{p+2}), ..., f(x_s))$$

be the training inputs and outputs, and test inputs and outputs, respectively. Here, **f** is known and **f*** is unknown. The objective is to find the distribution of **f*** given X, X^* and **f**, in order to select the $x^* \in X^*$ that is most likely to give the highest testing accuracy on the validation set. The joint distribution of **f** and **f*** according to the prior is,

$$\begin{bmatrix} f \\ f^* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m \\ m^* \end{bmatrix}, \begin{bmatrix} K(X,X) & K(X,X^*) \\ K(X^*,X) & K(X^*,X^*) \end{bmatrix} \right)$$

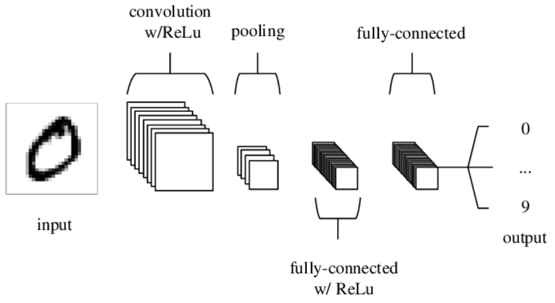

Figure 1. CNN Architecture of the model [7].

f* is derived by conditioning the joint Gaussian prior on the observations X^*, X and $\mathbf{f} \sim \mathcal{N}(\mu, \Sigma)$. Thus, the posterior prediction probability is given by,

$$p(\mathbf{f}^* | X^*, X, f) = \mathcal{N}(\mathbf{f}^* | \mu^*, \Sigma^*)$$

where,

$$\mu^* = m - K(X^*, X)K(X, X)^{-1}(f - m), \quad (1)$$

$$\Sigma^* = K(X^*, X^*) - K(X^*, X)K(X, X)^{-1}K(X, X^*) \quad (2)$$

The GP search algorithm uses (1) and (2) to compute μ and Σ that is used for posterior prediction of hyperparameter combinations. In *Figure 2*, the x-axis specifies the number of hyperparameter combinations for which F1-score was computed, and the y-axis specifies the best F1-score observed by the different techniques [3].

4. Experiments

In this project, we plan to create a CNN model with the hyper-parameters mentioned in *table 1*. The architecture of our CNN would look similar to the architecture described in *Figure 1*.

Hyperparameter	Values
Number of convolutional layer	3
Number of connected layer	3
Number of filters	32, 64, 128
Size of filter	5x5, 3x3, 3x3
Padding	0
Stride	1
Dimensions of fully connected layers	1024, 1024, 10
Activation function (non-final layer)	ReLU
Activation function (final layer)	Softmax

Table 1: Candidate values for CNN parameters and each hyperparameter. Since η_0, δ, B, p_1, p_2 can take different values, there are numerous possible hyperparameter combinations.

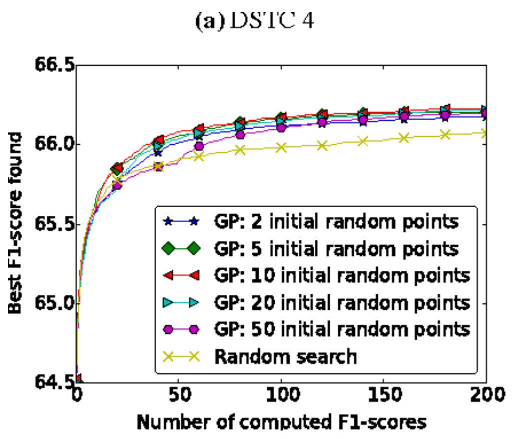

Figure 2. Performance of GP search and random search for hyperparameter optimization on DSTC 4 [4].

5. Division of Work

For this phase of the project, we divided the research papers to read among ourselves. Anik and Amogh did study on the Grid search and Hyperparameter tuning and its importance. Aditya and Gokulan studied the Gaussian process optimization to include the required information in the paper. For the implementation purpose, Anik and Amogh are looking on the iterative process of Grid search and implementing it to find the optimal CNN hperparameters. Aditya and Gokulan are looking on how Gaussian process can be used to find the same. Then we will compare the results on both the approaches. We are looking to use tensorflow in Python for the same.

6. Conclusion

Hence, it can be concluded from the above sections that tuning of hyperparameters to get the optimal performance is important. From the papers, we can derive certain conclusions. Paper [8] talks about the comparisons between a tuned model and an untuned model on how the hyperparamaters are sensitive to the model's performance and with the help of Brier measure, it demonstrates the same. Paper [3] talks about how Gaussian processes can be used to derive the posterior predictions of optimal hyperparameters. It demonstrates the same by classifying utterances based on actions. Paper [2] talks about automating the process of optimizing the hyperparameters by sampling the null prior distribution in the search space to get the configuration of hyperparameters that minimises the loss function. All these papers present the idea of tuning the hyperparameters to get better efficiency in results by following the approach as mentioned in experiment section, in Gaussian Proccess Search.

References

[1] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[2] James Bergstra, Daniel Yamins, and David Daniel Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. 2013.

[3] Franck Dernoncourt and Ji Young Lee. Optimizing neural network hyperparameters with gaussian processes for dialog act classification. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 406–413. IEEE, 2016.

[4] Seokhwan Kim, Luis Fernando D'Haro, Rafael E. Banchs, Jason D. Williams, and Matthew Henderson. The fourth dialog state tracking challenge. pages 435–449, 2017.

[5] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[6] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[7] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *ArXiv e-prints*, 11 2015.

[8] Jenna Wong, Travis Manderson, Michal Abrahamowicz, David Buckeridge, and Robyn Tamblyn. Can hyperparameter tuning improve the performance of a super learner? 2019.