geoconvention
Virtual Event
September 13 - 15
2021

Bayesian Hyper-Parameter Optimization: Neural Networks, TensorFlow, Facies Prediction Example

Ryan A. Mardani
Data Energy

Summary

Hyper-parameters in machine learning are adjustable by user to better fit the observed data. Finding the optimum values for these parameters is a challenging task. Random search and grid search are time-consuming and sometimes impossible for high-dimensional search spaces. In this work, we used the Bayesian optimization approach, which is developed by scikit-learn as 'skopt', to determine neural network main parameters for facies class prediction. The optimum parameters are: Learning rate: 0.0035, Number of dense layers: 6, Number of nodes for each layer: 327, Activation function: 'relu'.

Introduction

The purpose of this work is to optimize the neural network model hyper-parameters to estimate facies classes from well logs. In machine learning, model parameters can be divided into two main categories:

1- Trainable parameters: such as weights in neural networks learned by training algorithms and the user does not interfere in the process.
2- Hyper-parameters: users can set them before training operation such as learning rate or the number of dense layers in the model. Selecting the best hyper-parameters can be a tedious task if you try it by hand and it is almost impossible to find the best ones if you are dealing with more than two parameters.

One way is to divide each parameter into a valid evenly range and then simply ask the computer to loop for the total possible combinations of parameters and calculate the results. The method is called Grid Search. Although it is done by machine, it will be a time-consuming and computationally expensive process. Suppose you have 3 hyper-parameters with 10 possible values in each. In this approach, you will run 10^3 neural network models (even with reasonable training datasets size, this task is huge).

Another way is a random search approach. In fact, instead of using organized parameter searching, it will go through a random combination of parameters and look for the optimized ones. You may estimate that chance of success decreases to zero for larger hyper-parameter tunings.

Facies classes are the key factors in building a reservoir facies model, which can be used to generate a conceptional training image for multiple-point geostatistics (Khani et al., 2017, 2018a, 2018b).

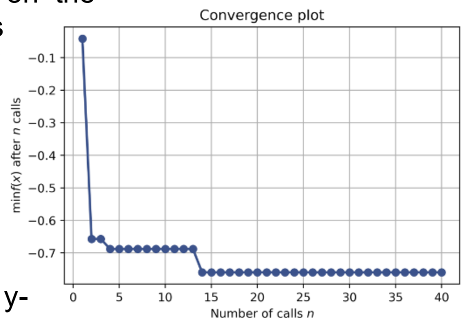
Methodology

Scikit-Optimize, skopt, which we will use here to the facies estimation task, is a simple and efficient library to minimize expensive noisy black-box functions. Bayesian optimization constructs another model of search-space for parameters. Gaussian Process is one kind of these models.

This generates an estimate of how model performance varies with hyper-parameter changes. To access the python code of this work please visit my GitHub (Ryan Mardani, 2020).
The dataset for this study comes from Hugoton and Panoma Fields in North America which was used in class exercise in The University of Kansas (Dubois et. al, 2007). It consists of log data of nine wells and facies groups.

In this work, we will predict facies from well logs using deep learning in Tensorflow. There are several hyper-parameters that we may adjust for deep learning. I will try to find out the optimized parameters for: 1- Learning rate, 2-Number of dense layers, 3-Number of nodes for each layer, and -4 activation function: 'relu' or sigmoid.

Using Tensorflow library, we should first define a model function which accept all hyper-parameters. Then train and evaluate model. The function aims to create and train a network with given hyper-parameters and then evaluate model performance with the validation dataset. It returns fitness value, negative classification accuracy on the dataset. It is negative because skopt performs minimization rather than maximization. We already checked the default hyper-parameter performance. Now we can examine Bayesian optimization from scikit-optimize library. Here we use 40 runs for fitness function, though it is an expensive operation and needs to be used carefully with datasets. Using *plot_convergence* function of skopt, we may see the optimization progress and the best fitness value found on y-axis.

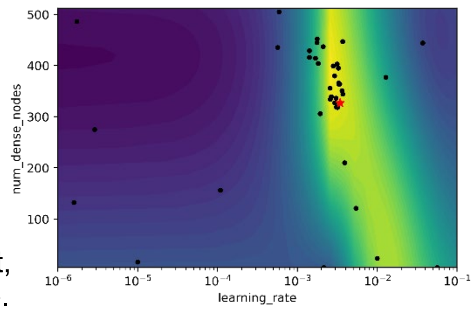


Optimal Hyper-Parameters

Using the *search_result* function, we can see the best hyper-parameter that Bayesian-optimizer generated. These parameters are determined as: Learning rate: 0.0035, Number of dense layers: 6, Number of nodes for each layer: 327, activation function: 'relu'.

Plots

First, let's look at 2D plot of two optimized parameters. Here we made landscape-plot of estimated fitness values for learning rate and number of nodes in each layer.



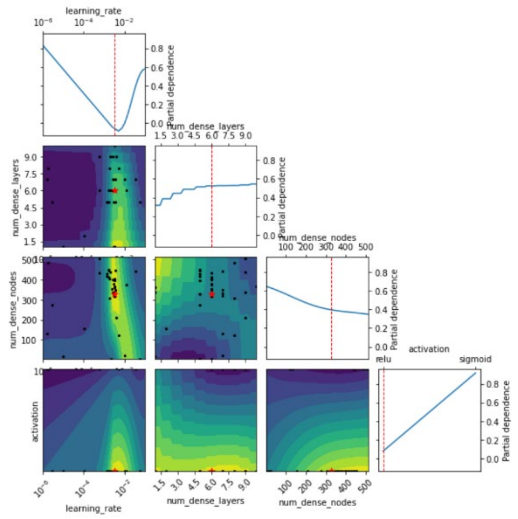
The Bayesian optimizer builds a surrogate model of search space and searches inside this dimension rather than real search-space, that is why it is faster. In the plot, the yellow regions are better and blue regions are worse. Black dots are the optimizer's sampling location and the red star is the best parameter found.

Some points:

1- The surrogate model can be inaccurate because it is built from only 40 samples of calls to the fitness function

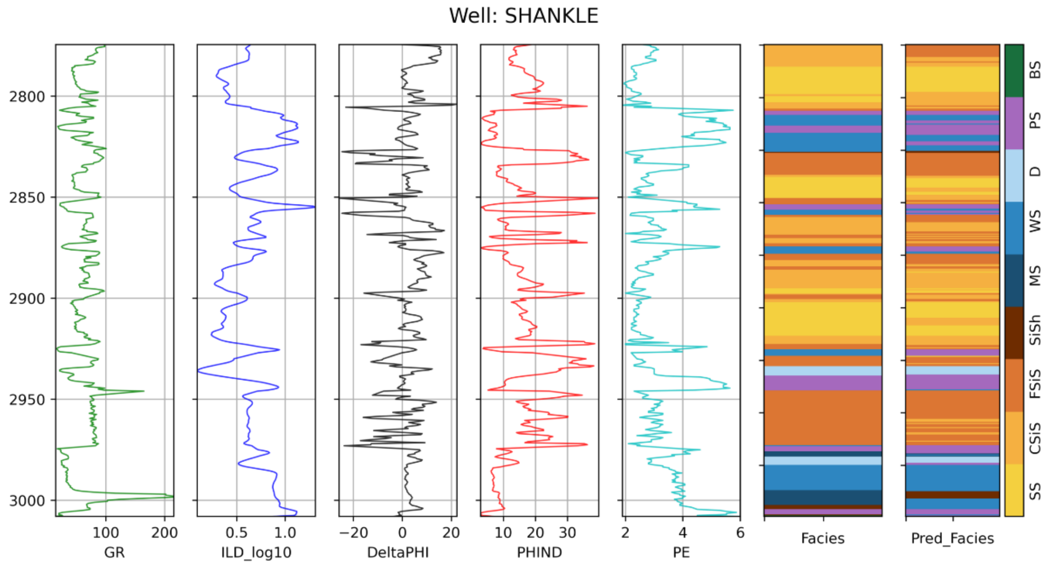
2- The plot may change in each time of optimization re-run because of random noise and training process in NN
3- This is 2D plot, while we optimized 4 parameters and could be imagined 4 dimensions.

In these plots, we can see how the optimization happened. The Bayesian approach tries to fit model parameters with prior info at the points with a higher density of sampling. Gathering all four parameters into a scikit-optimization approach will introduce the best results in this run if the learning rate is about 0.003, the number of dense layers 6, the number of nodes in each layer about 327, and activation function is 'relu'.



Evaluate the model with optimized hyper-parameters with blind data

Now we can make a model with optimized parameters to see the prediction visually in the blind data which is not exposed to modeling process.



We always expect that Machine Learning models will predict with blind data by less accuracy than training process if dataset is small or features are not big enough to cover all complexity of data dimensions. The accuracy for this prediction in almost 63%, though for classification problems, we should consider other evaluation metrics such as precision and recall.

Conclusions

In this work, we optimized hyper-parameters using a Bayesian approach with a scikit-learn library called skopt. This approach is superior to a random search and grid search, especially in complex datasets. Using this method, we can get rid of the hand-tuning of hyper-parameters for the neural networks, although in each run, you will face new parameters.

References

- Dubois, M. K., G. C. Bohling, and S. Chakrabarti, 2007, Comparison of four approaches to a rock facies classification problem, *Computers & Geosciences*, 33 (5), 599-617
- Khani, H., Hamdi, H., Nghiem, L., Chen, Z., & Costa Sousa, M. (2017, June 12). An Improved Regional Segmentation for Probability Perturbation Method. https://doi.org/10.3997/2214-4609.201701023
- Khani, H., Hamdi, H., Nghiem, L., Chen, Z., & Sousa, M. C. (2018a, March 13). Geologically Consistent History Matching of SAGD Process Using Probability Perturbation Method. SPE Canada Heavy Oil Technical Conference. https://doi.org/10.2118/189770-MS
- Khani, H., Hamdi, H., Nghiem, L., Chen, Z., & Sousa, M. C. (2018b). The Ability of Multiple-Point Geostatistics for Modelling Complex Fracture Networks in Tight and Shale Reservoirs. ECMOR XVI-16th European Conference on the Mathematics of Oil Recovery, 2018(1), 1–15.
- Matt Hall, 2016, Facies Classification, https://github.com/seg/2016-ml-contest/blob/master/Facies_classification.ipynb
- Ryan Mardani, 2020, https://github.com/mardani72/Hyper-Parameter_optimization/blob/master/Hyper_Param_Facies_tf_final.ipynb
- Havass-Labs, 2019, https://github.com/Hvass-Labs/TensorFlow-Tutorials