Project IV: Hyperparameter Tuning via Bayesian Global Optimization

Samarth Rawal, Devan Pratt, Sumit Rawat, Wil Gibbs

Group #21

Introduction

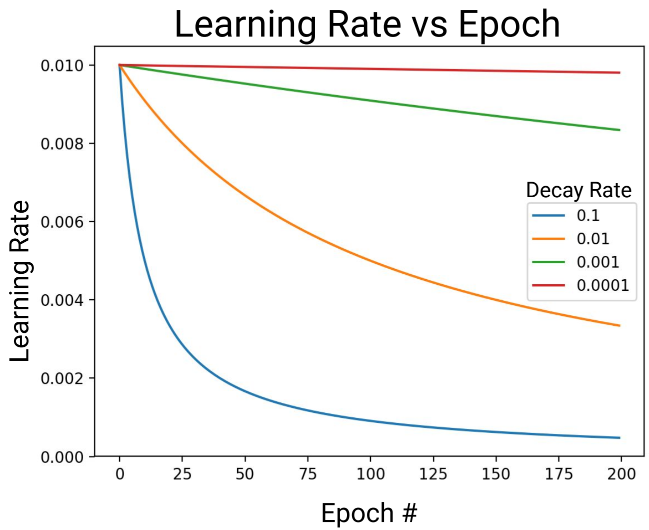
- Given the complexity of deep neural networks, it is important to efficiently arrive at well-performing parameters
- With a lot of layers in a convolutional neural network, it becomes infeasible to search the entire space
- We will investigate the efficiency of using Gaussian Processes to select the best Hyperparameters, compared to using Grid Search, empirically on the Street View House Numbers dataset [5]

Starting Learn Rate: η_0

- Learning rate helps to control how much the parameters are updated
- Can be within the 0.0-1.0 range
  - Typically lies between 0.01-0.1
- Very important to get right as values which are too low or too high waste a lot of time

Learning Rate Decay: δ

- Learning Rate Decay allows the Learning Rate (η_0) to settle into a minima
- Learning Rate Decay falls between 1.0 and 0.0 (exclusive)
  - Optimal Learning Rate Decays are model specific
- Learning Rate Decay can decrease training time
  - Larger values may miss important minima



Learning Rate vs Epoch Graph from: Brownlee, Jason. "Understand the Impact of Learning Rate on Neural Network Performance" Machine Learning Mastery, 3 Oct. 2019

Mini-Batch Size: B

- Mini-batch size(B) defines the number of training examples that are processed through the network in one pass
- Mini-batch size can be as low as one sample and as high as the size of the training dataset (given enough system memory)
  - Usually, a batch size of 32-1024 is chosen to approximate the gradient
- Smaller mini-batches introduce more stochasticity/noise into the training, while large mini-batches help converge faster

Dropout Parameters: p_1 and p_2

- Dropout is a form of regularization empirically demonstrated to improve the generalizability of neural networks [3]
- (Possibly) good dropout values to start experimenting with: 0.5 for hidden layers and 0.2 in input layers [3]
- Gridsearch Recommendations: range[0, 1.0) at increments of 0.1 [4]

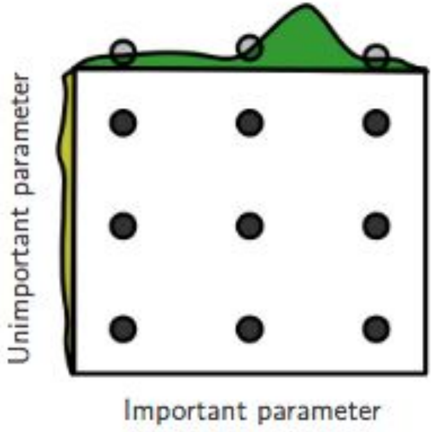
Bayesian Global Optimization

- *(Baseline)* Basic Grid Search has predefined search space -- not influenced by performance of previous search (full exploration)
- *(Conceptually)* BGO differs from this by “deciding” which hyperparameters to tweak based on prior results (good balance of exploration & exploitation)
- *(Technically)* The hyper parameters become the data to train on and new sets of parameters are chosen from the previous argmin of the validation model
- Each iteration improves the posterior distribution and helps the algorithm to determine the next parameter search space to be explored
- The past observations help the algorithm to find the optimal hyperparameter values in minimum number of iterations

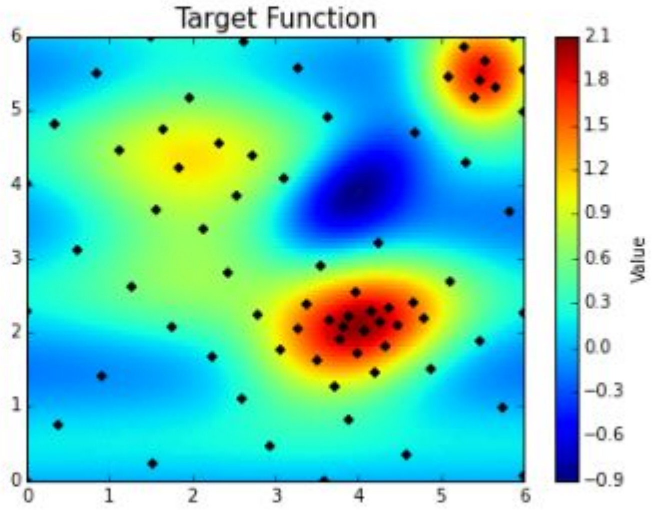
Group #21

Bayesian Global Optimization vs Grid Search

Grid Search



Bayesian Global Optimization



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

https://github.com/fmfn/BayesianOptimization

BGO vs GS: Results

- **Bayesian Global Optimization**
- Parameters: define the bounds + (random) exploration & (Bayesian) exploitation
  - `param_bounds = {`
  - `'lr_0': (1e-6, 1e-1),`
  - `'lr_decay': (1e-10, 0),`
  - `'B': (0.1, 3.2),`
  - `'p1': (0.1, 0.8),`
  - `'p2': (0.1, 0.8)`
  - `}`
  - `init_points=30 # explore`
  - `n_iter=70 # exploit`
- After 100 iterations:
  - `Val Loss: 1.134`
  - `lr_0: 0.03144, lr_decay: 0.3636, B: 64, p1: 0.3543`
  - `, p2: 0.5538`
  - `iteration: 21`

- **Grid Search** at static, predefined intervals
- Parameters: (10752 total!)
  - `lr_0 = [1e-1, 1e-3, 1e-5, 1e-6]`
  - `lr_decay = [0, 1e-1, 1e-3, 1e-4, 1e-6, 1e-8, 1e-10]`
  - `B = [32, 64, 128, 256, 512, 1024][::-1]`
  - `p1 = [0.1*i for i in range(1,9)]`
  - `p2 = [0.1*i for i in range(1,9)]`
- 1 set of Hyperparameters: ~1 min to train*
- After 800 iterations:
  - `Val Loss: 1.457`
  - `lr_0: 0.1, lr_decay: 0.1, B: 128, p1: 0.4, p2: 0.1`
  - `iteration: 241`

Note: These are preliminary results; we are currently running a more comprehensive Grid Search for the final report

* All experiments (and times mentioned) performed on Google Colaboratory w/ 1 Tesla K80 GPU

Conclusions

- Bayesian Global Optimization can spend computations and time more efficiently by better determining which hyperparameter regions are "worth" exploring further
- Grid Search does not place consideration into whether certain hyperparameter regions are worth spending more time exploring than others, and thus wastes a lot of computations and time
- Compared to Grid Search, BGO is a faster and more efficient method of optimizing hyperparameters, AND can find better parameters Grid Search may have otherwise missed

Software and References

Software

- Keras: https://keras.io/
- BayesianOptimization: https://github.com/fmfn/BayesianOptimization

References

1. Brownlee, J. (2019, October 3). Understand the Impact of Learning Rate on Neural Network Performance. Retrieved December 2, 2019, from https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/#:~:targetText=The amount that the weights,range between 0.0 and 1.0.
2. Bergstra, James, and Yoshua Bengio. "Random search for hyper-parameter optimization." *Journal of Machine Learning Research* 13.Feb (2012): 281-305.
3. Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *The journal of machine learning research* 15.1 (2014): 1929-1958.
4. Brownlee, Jason. “A Gentle Introduction to Dropout for Regularizing Deep Neural Networks.” Machine Learning Mastery, 6 Aug. 2019, machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/.
5. Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, “Reading digits in natural images with unsupervised feature learning,” 2011. [Online]. Available: http://ufldl.stanford.edu/housenumbers
6. J. Snoek, H. Larochelle, and R. P. Adams, “Practical bayesian optimization of machine learning algorithms in Advances in neural information processing systems”, 2012, pp. 2951–2959.
7. K. Weinberger,“Bayesian global optimization,”http://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote15.html, 2018.