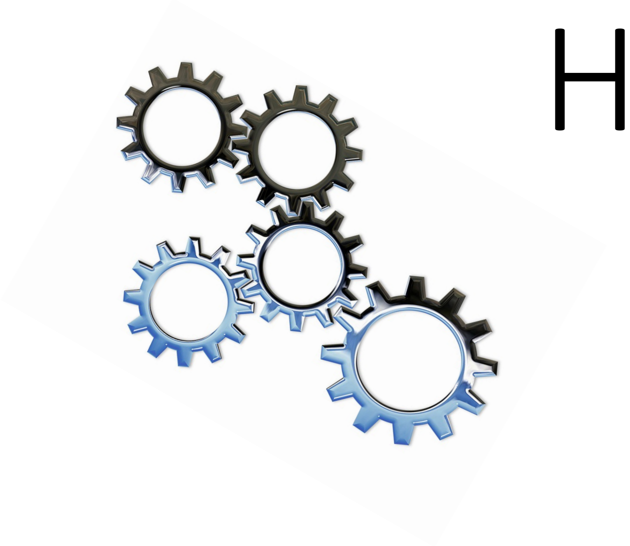


Hyperparameter Optimization

Luke Newman

Introduction

Hyperparameter optimization is a critical step in a data science project.

Once you have a shortlist of models you will want to fine-tune them.

Let’s discuss three techniques that are used to find optimal hyperparameters.

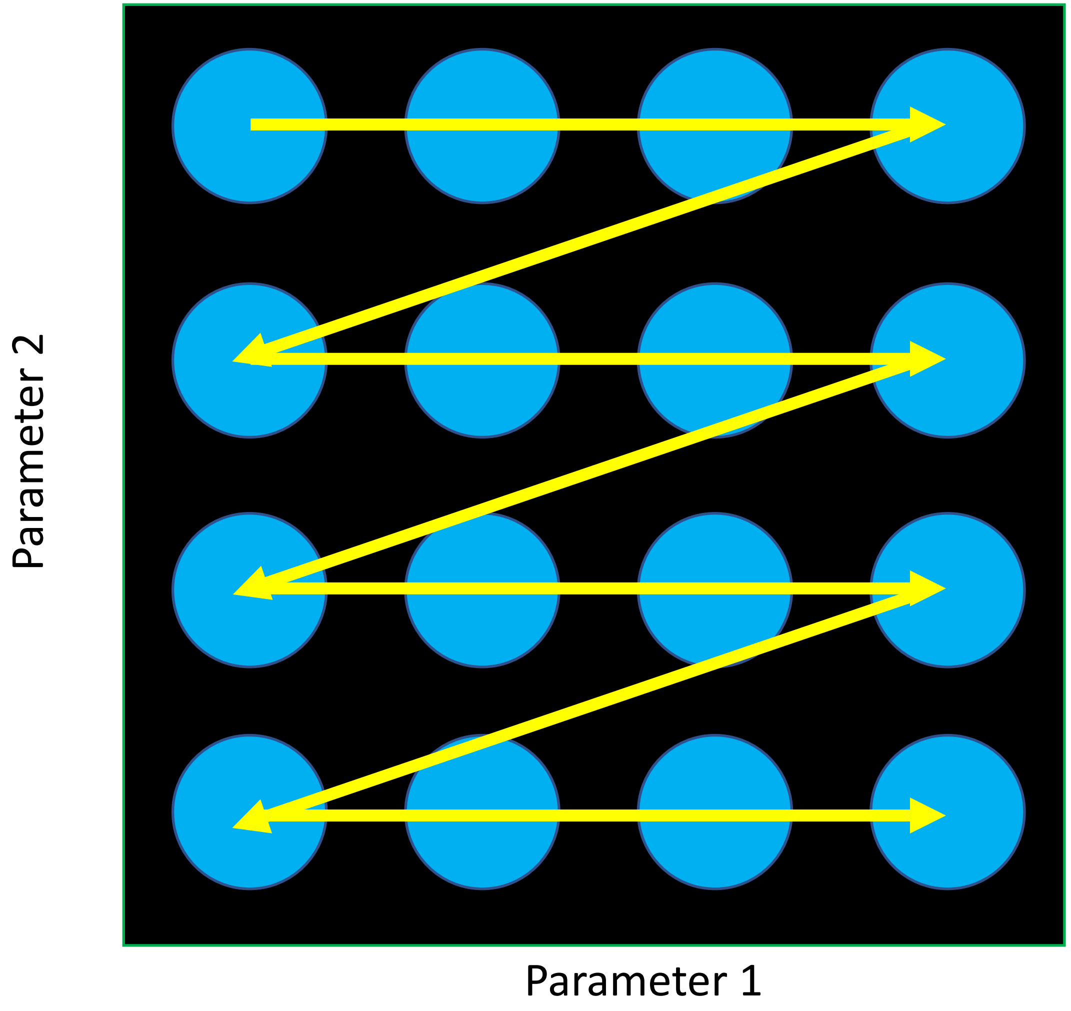
Randomized Search   Grid Search   Bayesian Optimization

Grid Search

Scikit_Learn's GridSearchCV

Tell it which hyperparameters you want it to experiment with and what values to try out.

It will use cross-validation to evaluate all possible combinations of hyperparameters.



Grid Search

Pros

Good when you are exploring relatively few combinations.

Finds optimal values within search space.

Cons

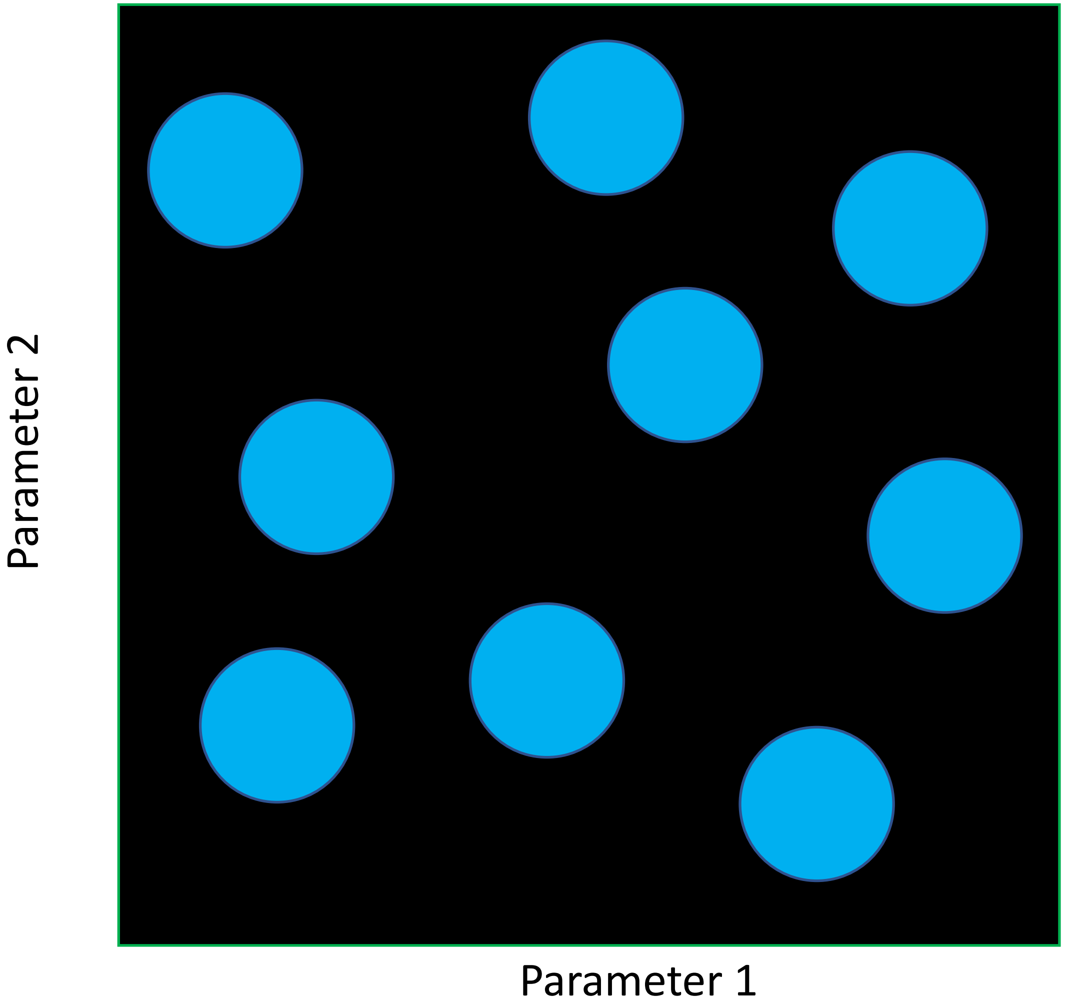
Computationally expensive when search space is large.

Hard to find optimal values in small search space.

Randomized Search

Scikit_Learn's RandomizedSearchCV

Evaluates a given number of random combinations by selecting a random value for each hyperparameter at every iteration.



Randomized Search

Pros

Will find a close to optimal solution within search space.

More control over the computing budget by specifying number of iterations.

Cons

Computationally expensive when training is slow (e.g., for more complex problems with larger datasets).

Will only explore a tiny portion of the hyperparameter space.

Bayesian Optimization

When dealing with complex problems on larger datasets, computational cost tends to be high.

Technique to explore a search space much more efficiently than randomly.

At a high level, Bayesian Optimization's core idea is:

- when a region of the hyperparameter space turns out to be good,
  it should be explored more.

Bayesian Optimization

Pros

Suitable for complex problems with large datasets.

Outperforms grid search and randomized search in most scenarios.

Cons

Increasing dimensions of search space requires more iterations to find optimal values.

Conclusions

If dealing with fairly simple problems randomized search is preferred.

For complex problems Bayesian Optimization is preferred.

Optimizing Random Forest Classifier on MNIST dataset

Metric: Accuracy	5-fold Cross-validation	Test Set
Grid Search	94.59%	94.86%
Randomized Search	96.20%	96.62%
Bayesian Optimization	97.00%	97.21%

Additional Resources

https://scikit-optimize.github.io/stable/

https://github.com/lukenew2/hyper_parameter_optimization

https://towardsdatascience.com/a-conceptual-explanation-of-bayesian-model-based-hyperparameter-optimization-for-machine-learning-b8172278050f