

Random Search for Hyper-Parameter Optimization

James Bergstra and Yoshua Bengio

February 10, 2011

Many machine learning algorithms have *hyper-parameters* - flags, values, and other configuration information that guides the algorithm. Sometimes this configuration applies to the space of functions that the learning algorithm searches (e.g. the number of nearest neighbours to use in KNN). Sometimes this configuration applies to the way in which the search is conducted (e.g. the step size in stochastic gradient descent). For better or for worse, it is common practice to judge a learning algorithm by its best-case-scenario performance. Researchers are expected to maximize the performance of their algorithm by optimizing over hyper-parameter values by e.g. cross-validating using data withheld from the training set. Despite decades of research into global optimization (e.g. [8, 4, 9, 10]) and the publishing of several hyper-parameter optimization algorithms (e.g. [7, 1, 3]), it would seem that most machine learning researchers still prefer to carry out this optimization by hand, and by grid search (e.g. [6, 5, 2]).

Here, we argue that in theory and experiment grid search (i.e. lattice-based brute force search) should almost never be used. Instead, quasi-random or even pseudo-random experiment designs (random experiments) should be preferred. Random experiments are just as easily parallelized as grid search, just as simple to design, and more reliable. Looking forward, we would like to investigate sequential hyper-parameter optimization algorithms and we hope that random search will serve as a credible baseline.

Does random search work better? We did an experiment (Fig. 1) similar to [5] using random search instead of grid search. We op-

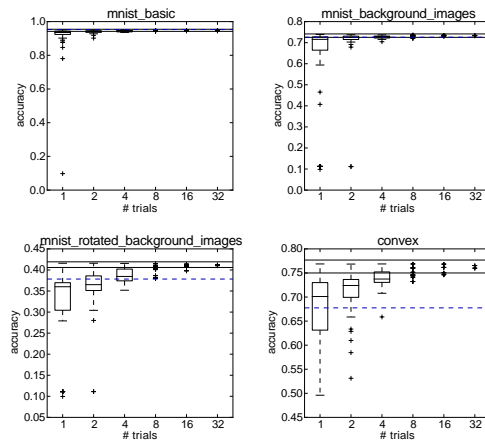


Figure 1: Horizontal axis: number of trials in an experiment. Vertical axis: expected test-set accuracy of the best model from an experiment of the given size. Dashed blue line is accuracy reported in [5] based on grids averaging 100 trials.

timized neural network performance for several tasks by varying six hyper-parameters: weight initialization strategy, initialization range, number of hidden units, type of nonlinearity, learning rate, and annealing schedule. We drew 32 assignments for these hyper-parameters from the same sort of distribution that the original lattice was designed to cover, and found among our 32 trials better results than were found by the 100 trials (on average) that explored the original lattice. Just 8 random trials were typically enough to out-perform the original results based on 100 gridded trials.

Why does random search work better?

The answer has to do with *low effective dimension*. A function has low effective dimension (LED) to the degree that it can be approxi-

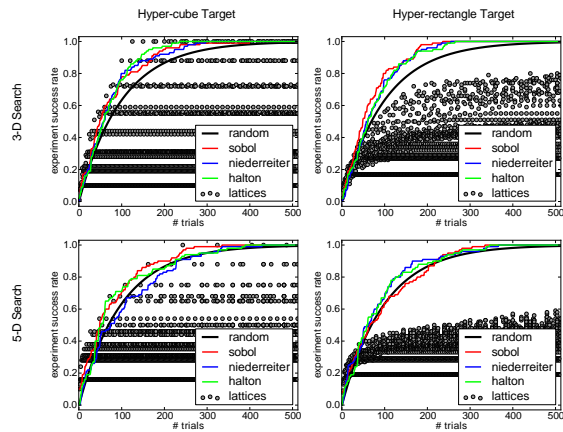


Figure 2: Probability of success in a simulated task of finding a small (volume 0.01%) multidimensional interval (needle) in a unit hyper-cube (haystack). In a low number of dimensions (3) and with a regularly-shaped needle (all dimensions equally long) a well-chosen lattice makes grid search relatively effective. In more dimensions (5) and with an irregularly-shaped needle (same volume, but not all dimensions equally long) grid search is relatively ineffective relative to random and quasi-random trials.

ated by a function of a subset of its arguments. Figure 2 illustrates that experiments based on random or low-discrepancy sequences deal much better with LED functions than those based on lattices. To see why, consider that if the needle is sometimes tiny in some dimension and sometimes tiny in another, then a lattice must be fine in *both* dimensions to reliably find the needle. Random sets, and better quality low-discrepancy sets are more robust in this setting.

In other experiments (not shown in this abstract) we use feature-selection algorithms to establish that indeed, for several datasets and for several model families, the function that maps from the hyper-parameter configuration space to generalization performance has a relatively low effective dimension. This result explains why random search is generally better than grid search for hyper-parameter optimization.

References

- [1] I. Czogiel, K. Luebke, and C. Weihs. Response surface methodology for optimizing hyper parameters. Technical report, Universität Dortmund Fachbereich Statistik, September 2005.
- [2] Geoffrey Hinton. A practical guide to training restricted boltzmann machines. Technical Report 2010003, University of Toronto, 2010. version 1.
- [3] Frank Hutter. *Automated Configuration of Algorithms for Solving Hard Computational Problems*. PhD thesis, University of British Columbia, 2009.
- [4] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.
- [5] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In Zoubin Ghahramani, editor, *Proceedings of the 24th International Conference on Machine Learning (ICML’07)*, pages 473–480. ACM, 2007.
- [6] Yann LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural Networks, Tricks of the Trade*, Lecture Notes in Computer Science LNCS 1524. Springer Verlag, 1998.
- [7] Alexander Nareyek. Choosing search heuristics by non-stationary reinforcement learning. *Applied Optimization*, 86:523–544, 2003.
- [8] J.A. Nelder and R. Mead. A simplex method for function minimization. *Comput. J.*, 7:308313, 1965.
- [9] M.J.D. Powell. A direct search optimization method that models the objective and constraint functions by linear interpolation. *Advances in Optimization and Numerical Analysis*, page 5167, 1994.
- [10] Thomas Weise. *Global Optimization Algorithms - Theory and Application*. Self-Published, second edition, 2009. Online available at <http://www.it-weise.de/>.