Hyperparameter tuning and parallels with simulation optimisation

Luke Rhodes-Leader

LAI Reading Group

Department of Management Science
Lancaster University Management School

Simulation Optimisation Formulation

$$\min\ f(\mathbf{x})$$
$$\mathbf{x} \in C$$

where $f(\mathbf{x})$ is the performance measure of interest, and C is the feasible set or region for the d-dimensional solution variable $\mathbf{x}$.

What makes this a *simulation optimisation* (SO) problem is the need to estimate the solution performance measure $f(\mathbf{x})$ using a simulation-based estimator $\widehat{f}(\mathbf{x}; T, n, \mathbf{U})$. May also have to estimate whether some constraints are satisfied.

Often we are interested in the expected value of a random variable:

$$f(\mathbf{x}) = \mathbb{E}[Y(\mathbf{x})] \qquad \Rightarrow \qquad \widehat{f}(\mathbf{x}; T, n, \mathbf{U}) = \frac{1}{n}\sum_{j=1}^{n} Y_j(\mathbf{x}; T, \mathbf{U}_j)$$

Experiment design for SO

Our estimator is:

$$\widehat{f}(\mathbf{x}; T, n, \mathbf{U})$$

- $\mathbf{x}$ defines the solution, and may be categorical, discrete-valued or continuous-valued
- T, n are the run length and number of replications (only one may be relevant)
- $\mathbf{U}$ are the underlying (pseudo) random numbers, which in simulation can be *assigned*

In SO there is always the fundamental trade-off between exploration and exploitation.

Types of Simulation Optimisation

Small number of discrete solutions

Deciding between several different set-ups for a factory or hospital to maximize throughput

Continuous decision variables

Optimizing the statistical thresholds in clinical trial design

Discrete and integer-ordered decision variables

Optimizing the number of call centre staff on duty to minimize costs subject to constraints on response times

Errors in SO

Whatever the SO algorithm is, it will terminate having simulated $K < \infty$ solutions. The errors are...

1. *The optimal solution is never simulated,* meaning
$$\mathbf{x}^* \notin \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K\}.$$
This is the case in most optimisation problems.
2. *The best solution that was simulated is not selected,* meaning
$$\hat{\mathbf{x}}^* \neq \mathbf{x}_B = \text{argmin}_{i=1,2,\ldots,K} f(\mathbf{x}_i).$$
3. *The estimated objective function value of the selected solution is not very precise,*
$$\left| \widehat{f}(\hat{\mathbf{x}}^*) - f(\hat{\mathbf{x}}^*) \right| \text{ is large.}$$

Fitting ML models

The general machine learning model is to take some set of predictors/covariates, z and use a model g to make a prediction of some form:

$$y = g(z; \theta, \mathbf{x})$$

If we have data $\mathbf{Y}$, we can use these to learn the optimal parameters, θ.

Typically this is done using a **loss function** $L(\theta; \mathbf{Y}, \mathbf{x})$ with:

$$\hat{\theta} \approx \underset{\theta \in \Theta}{\operatorname{argmin}}\ L(\theta; \mathbf{Y}, \mathbf{x})$$

A common method for this is **Stochastic Gradient Descent** (SGD) (which shares a lot of similarities with SO).

Hyperparameters

$$y = g(z; \theta, \mathbf{x}), \qquad \hat{\theta} \approx \underset{\theta \in \Theta}{\operatorname{argmin}}\ L(\theta; \mathbf{Y}, \mathbf{x})$$

$\mathbf{x}$ is a set of 'hyperparameters' that are picked in advance. They influence the model behaviour/performance, and also ability to solve the optimisation problem.

Examples include:

- maximum tree depth in decision trees and random forests
- regularization parameters
- learning rates and batch sizes for SGD

Hyperparameter tuning

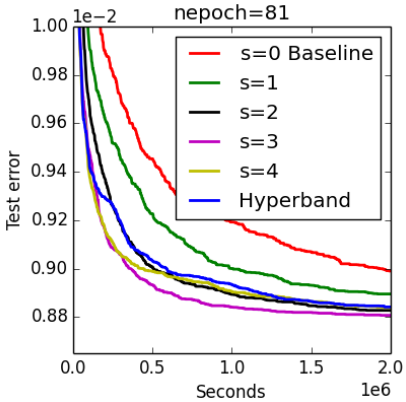


Figure 3 from Li et al. (2018)

Hyperparameter tuning is a process of determining which values of $\mathbf{x}$ that lead to the best model. This can make a difference to the performance We have a different optimisation problem:

$$\mathbf{x}^* = \underset{\mathbf{x}\in\mathcal{X}}{\operatorname{argmin}}\ L(\mathbf{x}, r) = \underset{\mathbf{x}\in\mathcal{X}}{\operatorname{argmin}}\ L(\hat{\theta}(r); \mathbf{Y}, \mathbf{x})$$

r is the amount of resource used for fitting the parameters $\hat{\theta}(r)$

My first attempt

- Fitting Random Forests to some data for predicting food hygiene ratings of restaurants
- A different seed means a different RF model results.
- In cross validation - the (random) data split also makes a difference
- So this struck me as a simulation optimisation problem
- I ended up using a basic Bayesian Optimisation algorithm, and doing many 'replications' to reduce the error
- This was naïve - assumed homoscedastic variance, did not properly consider the nature of the cross-validation procedure, etc...

More thought through attempts

- Parameters can be manually tuned to improve performance
- Bayesian Optimisation approaches are also fairly common, e.g. Snoek et al. (2012)
- These involve fitting a Gaussian Process (GP) model to the observations of the loss function and sequentially selecting the next best place to evaluate to find the optimal
- Issues arise in high dimensions, in potentially heteroscedastic noise and very non-smooth functions
- While basic, random search is also a popular alternative

Hyperband

- Hyperband (Li et al., 2018) is a parameter tuning algorithm based on random search
- It is an adaptation of the Successive Halving algorithm (Jamieson and Talwalkar, 2016) which:
  1. Samples n configurations, $\mathbf{x}$, from configuration space $\mathcal{X}$
  2. Uses some resource to evaluate the loss function at each configuration
  3. Keeps the best half of the solutions, doubles the resource, go back to step 2.
- With finite resource budget, there is a trade off between number of solutions explored and solution quality
- Hyperband repeats this step with different settings of this trade off
- "Resource" could be the amount of data given (in features or data points), the number of iterations of the optimisation process for fitting the model, the amount of time given to fit the model, etc...

Hyperband algorithm

Input: R, η (default $\eta = 3$)
Initialization: $s_{\max} = \lfloor \log_\eta(R) \rfloor$, $B = (s_{\max} + 1)R$
for $s \in \{s_{\max}, s_{\max} - 1, ..., 0\}$ **do**
    $n = \lceil B\eta^s / R(s+1) \rceil$, $r = R\eta^{-s}$
    begin successive halving algorithm
    $\mathcal{T} \leftarrow$ sample of n hyperparameter configurations from $\mathcal{X}$
    for $i \in \{0, ..., s\}$ **do**
        $n_i = \lfloor n\eta^{-i} \rfloor$ ▷ configurations in this round
        $r_i = r\eta^i$ ▷ resources per configuration
        $\mathcal{L} = \{L(\mathbf{x}, r_i) : \mathbf{x} \in \mathcal{T}\}$ ▷ Loss function evaluations
        $\mathcal{T} \leftarrow$ best $\lfloor n/\eta \rfloor$ configurations by $\mathcal{L}$
    end for
end for
return configuration with smallest $L(\cdot, \cdot)$ seen so far

More of Hyperband

- Fixed size version discussed here, but you could also continue the loop indefinitely for asymptotic analysis.
- Analysis of the algorithm is based on a Non-stochastic Infinite-Armed Bandit (NIAB)
- Analysis seems to consider that either
  - Loss function evaluation $L(\mathbf{x}, r)$ is deterministic; or
  - Loss function evaluation $L(\mathbf{x}, r)$ is unbiased

Multi-fidelity Simulation Optimisation

- Using fewer resources to evaluate the fit is like using a lower-fidelity model to evaluate the objective function.
- In SO, this could be queueing models instead of a DES, or just using fewer replications
- Using fewer resources will increase the error in the objective function evaluation:
  - Variance: fewer 'replications' means the estimator error due to stochastic noise is larger - this is what SO is designed for
  - Bias: a cheaper model will probably be wrong and any noisy observations you get from this will give you an incorrect value - this is where explicit information on MFSO is useful.

Hyperband inner loop

- Here we compare a fixed number, K, of configurations
- For convenience, let assume $\mathbf{x}_K$ is the best
- If this was just a noisy problem, this is a **Ranking and Selection** problem.
- This is essentially a bandit problem where the aim is to learn the best arm - not worrying about cumulative reward
- Two viewpoints on this:
  1. fixed precision - we want to meet a probabilistic guarantee of selecting the best/good solution with the minimum amount of effort
  2. fixed budget - we have a finite amount of computation and want to maximise the probability of selecting the best/good solution or minimise the expected opportunity cost
- Version 2 is more applicable here
- We focus on Probability of Correct Selection (PCS)

Simulation Budget Allocation Procedures

Aim

Allocate more simulation time to designs which are critical for identifying which of the set is the best.

In practice this means

- Fewer replications for designs which are clearly poor performers.
- More replications for designs which are close to the best value.
- More replications for designs which exhibit a high degree of uncertainty.

Two main alternatives exist: **OCBA** (optimal computing budget allocation) and **EVI** (expected value of information).

OCBA (Chen et al., 1997)

One can provide a lower bound for the PCS using a Bonferroni inequality.

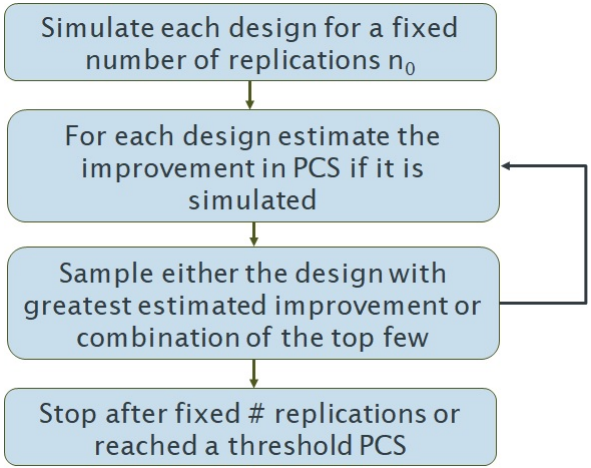
OCBA allocates the budget in an **asymptotically** (i.e. assuming an infinite budget) **'optimal'** way (based on the lower bound of PCS), assuming the observations are normally distributed:

$$\frac{r_i}{r_j} = \left(\frac{\sigma_i/d_{iK}}{\sigma_j/d_{jK}}\right)^2 \qquad \forall i,j \in \{1,...,K-1\}, \qquad r_K = \sigma_K \sqrt{\sum_{i=1}^{K-1} \frac{r_i^2}{\sigma_i^2}}$$

Here, $d_{iK} = L(\mathbf{x}_i) - L(\mathbf{x}_K)$.

As most of these values are unknown, allocation is done iteratively, with updated estimates of each quantity after each round.

OCBA (Chen et al., 1997)



Bringing in the bias

- Consider an evaluation with little resource - this is **biased** and possibly noisy.
- Pure OCBA is not designed for this.
- It is a low-fidelity observation of the model performance - with no indication of what the bias is.
- So good configurations could look bad.
- Throwing out solutions based on this is not necessarily a good move.
- Xu et al. (2016) and Cao et al. (2021) develop methods that prioritise solutions that look good after the first round, but don't abandon the others either.

Multi-Fidelity Optimization with Ordinal Transformation and Optimal Sampling (MOT^{2}OS)

- Key idea of Xu et al. (2016) is an ordinal transformation of the solution space using a low-fidelity model
- Solutions can then be grouped - equally in this case
- Resource allocated between the groups using OCBA (mean and variance are based on whole group)
- Within groups, resources allocated randomly

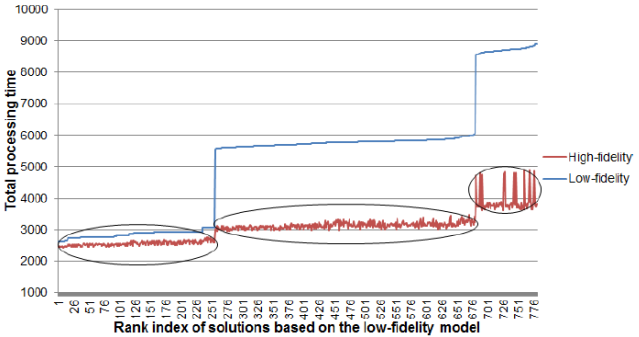


Figure 3 from Xu et al. (2016) - 10 groups are used

Further Improvement

- Cao et al. (2021) went one further and removed the equal allocation to groups
- Solutions are clustered, then put into groups
- Remainder of method is the same
- Clustering gives more priority to the best solutions, without completely removing the poorer solutions

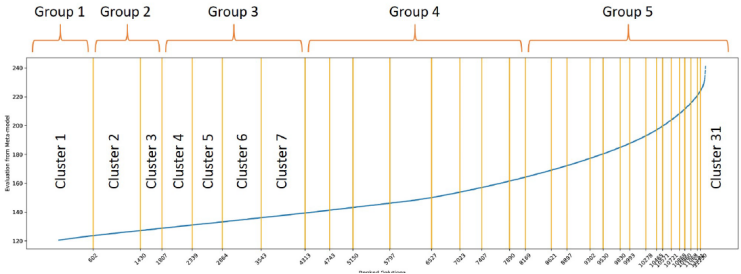


Figure 3 from Cao et al. (2021)

Hyperband Outer Loop

- Samples solutions from the solution space
- No obvious way of updating the sampling distribution - pure random search
- But assuming some level of continuity - looking in areas that have performed well so far seems sensible
- **Adaptive Random Search** tackles this problem (though typically in discrete SO problems, see Hong et al. (2014))
- The idea is to give more weight to areas that already look good, whilst still allowing anything in the space to be sampled
- The distribution over $\mathcal{X}$ needs to be fairly easy to sample from

ARS - an example

A slightly speculative example to demonstrate the how this might work

1. At the beginning, sample configurations, $\mathcal{T}_1$ from $F_{0,\mathcal{X}}$ - uniform across $\mathcal{X}$, set $i = 1$
2. Evaluate go through the inner loop of Hyperband, with estimation set $\mathcal{T}_i$
3. Create mixture multivariate normal centred on $\mathcal{T}_{\text{all}} = \mathcal{T}_{\text{all}} \cup \mathcal{T}_i$

$$F_{i,\mathcal{X}} \propto \sum_{\mathbf{x} \in \mathcal{T}_{\text{all}}} s(L(\mathbf{x}, r(\mathbf{x}))\mathcal{N}(\mathbf{x}, \Sigma)$$

   where $s(\cdot)$ is a non-negative strictly decreasing function
4. Sample new set of configurations from

$$\bar{F}_{i,\mathcal{X}} = (1 - \lambda)F_{i,\mathcal{X}} + \lambda F_{0,\mathcal{X}}$$

   Return to step 2.

Multi-fidelity Bayesian Optimisation Methods

- We could try to look at the bias in another way
- One method for doing this is to fit a model (e.g. a GP) to the error between each fidelity level, $\delta_r(\mathbf{x})$, in an auto-regressive structure:

$$L_1(\mathbf{x}) = \delta_1(\mathbf{x})$$
$$L_r(\mathbf{x}) = L_{r-1}(\mathbf{x}) + \delta_r(\mathbf{x})$$

  this is based on an initial idea for model calibration from Kennedy and O'Hagan (2002)
- The MFSKO algorithm Huang et al. (2006) used this alongside an augmented expected improvement
- To fit the error - you need observations of the error at all fidelity levels - so you need to apply full resource to at least some levels.

Multi-output GP models

- We could create a set of correlated GPs - perhaps for each fidelity
- The GP kernel now acts across both $\mathcal{X}$ and the fidelity
- Most models here are separable:

$$k((\mathbf{x}, r), (\mathbf{x}', r')) = k_{\mathcal{X}}(\mathbf{x}, \mathbf{x}') \times k_R(r, r')$$

- Fidelity could be discrete - in which case $k_R(r, r')$ is a covariance matrix - or it can be continuous
- This tries to capture the relationship across the fidelities - and helps when making decisions at the acquisition functions stage which chooses both $\mathbf{x}$ and r
- These are hard to fit - particularly for noisy, heteroscedastic observations and not much data - ask Harry Newton

Conclusions

- Hyperparameter tuning is the process of selecting non-data based parameters
- It can make a difference to algorithm performance
- We can view the hyperparameter tuning problem in a multi-fidelity simulation optimisation framework
- SO is learning a lot from ML with many new methods being firmly based on this idea
- Maybe - and it could be only a small amount - the simulation community has something to offer ML as well

Any questions?

Luke Rhodes-Leader

l.rhodes-leader@lancaster.ac.uk

Department of Management Science

Lancaster University Management School

References I

Cao, Y., Currie, C., Onggo, B. S., Higgins, M., 2021. Simulation optimization for a digital twin using a multi-fidelity framework. In: Kim, S., Feng, B., Smith, K., Masoud, S., Zheng, Z., Szabo, C., Loper, M. (Eds.), Proceedings of the 2021 Conference on Winter Simulation. IEEE, Piscataway, NJ, USA, pp. 1–12.

Chen, H.-C., Dai, L., Chen, C.-H., Yücesan, E., 1997. New development of optimal computing budget allocation for discrete event simulation. In: Andradóttir, S., Healy, K. J., Withers, D. H., Nelson, B. L. (Eds.), Proceedings of the 1997 Conference on Winter Simulation. IEEE, Piscataway, NJ, USA, p. 334–341.

Hong, L. J., Nelson, B. L., Xu, J., 2014. Discrete Optimization via Simulation. In: Fu, M. C. (Ed.), Handbook of Simulation Optimization. Vol. 216 of International Series in Operations Research and Management Science. Springer, New York, Ch. 2, pp. 9–44.

References II

Huang, D., Allen, T. T., Notz, W. I., Miller, R. A., 2006. Sequential Kriging Optimization using Multiple-fidelity Evaluations. Structural and Multidisciplinary Optimization 32 (5), 369–382.

Jamieson, K., Talwalkar, A., 09–11 May 2016. Non-stochastic best arm identification and hyperparameter optimization. In: Gretton, A., Robert, C. C. (Eds.), Proceedings of the 19th International Conference on Artificial Intelligence and Statistics. Vol. 51 of Proceedings of Machine Learning Research. PMLR, Cadiz, Spain, pp. 240–248.
URL https://proceedings.mlr.press/v51/jamieson16.html

Kennedy, M. C., O'Hagan, A., 01 2002. Bayesian calibration of computer models. Journal of the Royal Statistical Society Series B: Statistical Methodology 63 (3), 425–464.
URL https://doi.org/10.1111/1467-9868.00294

References III

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., Talwalkar, A., 2018. Hyperband: A novel bandit-based approach to hyperparameter optimization. Journal of Machine Learning Research 18 (185), 1–52.
URL http://jmlr.org/papers/v18/16-558.html

Snoek, J., Larochelle, H., Adams, R. P., 2012. Practical bayesian optimization of machine learning algorithms. In: Pereira, F., Burges, C., Bottou, L., Weinberger, K. (Eds.), Advances in Neural Information Processing Systems. Vol. 25. Curran Associates, Inc.
URL https://proceedings.neurips.cc/paper_files/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf

Xu, J., Zhang, S., Huang, E., Chen, C.-h., Lee, L. H., Celik, N., 2016. MO^{2}TOS: Multi-Fidelity Optimization with Ordinal Transformation and Optimal Sampling. Asia-Pacific Journal of Operational Research 33 (3), 1650017:1–26.