A Strategy for Ranking Optimization Methods using Multiple Criteria

Ian Dewancker IAN@SIGOPT.COM
Michael McCourt MIKE@SIGOPT.COM
Scott Clark SCOTT@SIGOPT.COM
Patrick Hayes PATRICK@SIGOPT.COM
Alexandra Johnson ALEXANDRA@SIGOPT.COM
George Ke L2KE@UWATERLOO.CA
SigOpt, San Francisco, CA 94108 USA

Abstract

An important component of a suitably automated machine learning process is the automation of the model selection which often contains some optimal selection of hyperparameters. The hyperparameter optimization process is often conducted with a black-box tool, but, because different tools may perform better in different circumstances, automating the machine learning workflow might involve choosing the appropriate optimization method for a given situation. This paper proposes a mechanism for comparing the performance of multiple optimization methods for multiple performance metrics across a range of optimization problems. Using nonparametric statistical tests to convert the metrics recorded for each problem into a partial ranking of optimization methods, results from each problem are then amalgamated through a voting mechanism to generate a final score for each optimization method. Mathematical analysis is provided to motivate decisions within this strategy, and sample results are provided to demonstrate the impact of certain ranking decisions.

Keywords: evaluation methods, Bayesian optimization, sequential model-based optimization, empirical analysis

1. Introduction

As machine learning tools continue to grow in importance and breadth of application, the need to automate these tools for non-expert use and interpretation has grown accordingly; research into this topic has been associated with the term AutoML (Feurer et al., 2015). Any suitably automated machine learning framework would include some automation of model and hyperparameter selection, and this hyperparameter search is generally carried out using a black-box, gradient-free optimization method (Thornton et al., 2013). Many such methods seek to take information about a function $f : \mathcal{X} \to \mathbb{R}$ and create a data-generated model in a Bayesian optimization framework (Brochu et al., 2010; Martinez-Cantin et al., 2007). In the AutoML setting, this f function likely represents a measurement of how well certain hyperparameters define a model that matches given data such as, e.g., a cross-validation residual, likelihood function or risk functional. Additionally, Bayesian optimization can be used to simultaneously tune multiple aspects of a machine learning model, such as data-preprocessing as well as model hyperparameters Coates et al. (2011).

Bayesian optimization can utilize a variety of models, including Gaussian processes (Snoek et al., 2012), random forests (Hutter et al., 2011a), and tree-structured Parzen estimators (Bergstra et al., 2011), each of which has its own strengths. Comparing the performance of multiple optimization methods is beneficial when identifying the best optimization method for model fitting in a given situation, which is an important part of an efficient and robust automated machine learning framework.

Eggensperger et al. (2013); Martinez-Cantin (2014); Eggensperger et al. (2015) are among the literature that has previously summarized the performance of Bayesian optimization methods. Some publications involve the use of potentially inappropriate statistical analysis, and most provides little guidance as to how performance on multiple functions f can be analyzed in chorus. Consequently, results from an optimization often read in the form of a confidence interval (derived from a small sample size) relevant to only a single function and without any means for broader interpretation.

To address these difficulties, Section 2 details a strategy for ranking performance between various optimization methods on multiple metrics and aggregating that performance across multiple functions. By allowing for multiple metrics, optimizers can be studied in more detail, e.g., considering both the quality of the solution and the speed with which it is attained. We demonstrate our evaluation strategy using an open source suite of benchmark functions available at McCourt (2016). Section 3 outlines our results and provides some interpretation.

2. Evaluating Optimization Performance

Many metrics exist for describing the quality of an optimizer, and each application values them differently. In the context of an AutoML problem, the goal of the optimizer is to find the optimal model design or hyperparameters, and the quality of an optimizer might be judged on the proximity of the solution to the optimal design or the speed with which that solution is found (thus facilitating more model experimentation/learning).

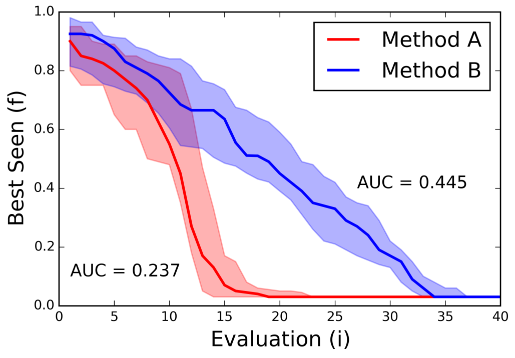

Figure 1: Hypothetical optimization methods A and B both achieve the same best found value of 0.03 at the end of 40 evaluations. Method A, however, finds the optimum in fewer evaluations, thus the lower AUC value.

Here, we focus on only these two metrics, accuracy and speed, which is sufficient to demonstrate the hierarchical nature of our ranking strategy; others could be included if desired. These metrics are derived from the best seen traces, $f_{\text{best}}[i]$, which record the best seen objective value after i objective function evaluations. Other thoughts regarding the design and incorporation of performance metrics are presented in Section 2.2.

The accuracy measurement is referred to as the **Best Found** metric, $f_{\text{best}}[T]$, and records the optimal f value observed after the total T evaluations. To measure the speed of improvement, we supplement the best found metric with the **Area Under Curve** metric, $\frac{1}{T}\sum_{i=1}^{T}(f_{\text{best}}[i] - f_{\text{LB}})$. A specified lower bound f_{LB} on the function ensures the AUC is always positive. The name AUC reflects the physical interpretation of the metric as an approximate integral of the best seen traces. In the context of an AutoML problem, the **Best Found** metric represents the quality of the resulting AutoML system, whereas the **Area Under Curve** metric represents the cost of tuning this optimal AutoML system.

A sample of $f_{\text{best}}[i]$ for two different optimization methods tested 30 times is shown in Figure 1. The shaded region represents the inter-quartile range and reminds us that these optimization methods are inherently stochastic and that any results should be interpreted statistically.

2.1. Performance Ranking and Aggregation

We proceed by using our metrics to establish a partial ranking (allowing for ties) of how multiple optimization methods perform on a given function. For this section we choose to more highly value better model fit (good **Best Found** values) and reserve comparisons of optimization speed (good **Area Under Curve** values) for to distinguish between methods producing the same fit. Specifically, we combine the two metrics in a hierarchical fashion:

- First, we use pairwise Mann-Whitney U tests (discussed in the supplemental content) at a chosen α significance on the **Best Found** results to determine a partial ranking based only on that statistic,
- Any tied results from that step are then subject to additional partial ranking using the same test on the **Area Under Curve** metric,
- Ties remaining after ranking attempts using both metrics are left as ties.

This ordering could be reversed to more strongly emphasize speed over accuracy.

This ranking process would be carried out on each function in a specified set which, in effect, allows each function to "cast a ballot" listing the optimization methods in order of performance on that function. We allow for ties throughout this process, thus favoring no stated preference rather than a weak preference, in recognition of the high level of randomness in the hyperparameter optimization. These ballots are then aggregated using a Borda ranking system (Dwork et al., 2001); the tables in Section 3 are an example. This approach generalizes to using additional metrics, provided they are applied in a specified order of importance.

The appropriate α signficance can be determined by considering the "family-wise error" α_F, the combined probability of any type I errors during the total set of pairwise statistical tests on each metric, given that each test has α probability of a type I error (Demšar, 2006). These tests are not independent (the same samples are used for multiple tests,) but even if they were, the probability of at least a single type I error is bounded by

$$\alpha_F \leq 1 - (1 - \alpha)^{\binom{m}{2}},$$

where m is the number of algorithms undergoing pairwise comparison. In Section 3 we use $m = 7$ algorithms, thus our desired $\alpha_F = .01$ can be achieved with $\alpha \approx .0005$.

2.1.1. EXAMPLE RANKING AND AGGREGATION

Suppose we use METHOD A, METHOD B, METHOD C and METHOD D to optimize a single function 30 times and record the **Best Found** and **Area Under Curve** values for each optimization. The Mann-Whitney U tests on the best found value may yield the statistically significant results on the left, which can be reverse sorted in order of number of *losses* to observe the partial ranking, and resulting Borda values, on the right:

$$\left.\begin{array}{l}\text{METHOD A} < \text{METHOD D},\\ \text{METHOD A} < \text{METHOD C},\\ \text{METHOD B} < \text{METHOD D}.\end{array}\right\} \qquad \begin{array}{ccccc}(\text{METHOD A}, \text{METHOD B}) & < & \text{METHOD C} & < & \text{METHOD D}.\\ 2 & & 1 & & 0\end{array}$$

Note that sorting by losses is just a simple way to isolate the worst performers and certainly not the only mechanism of creating a ranking (Cook et al., 2007), e.g., one could rank by number of wins. Any group of ties, such as (METHOD A, METHOD B) in this example, is refined by studying the area under curve statistical test; if that test stated that METHOD A had a smaller AUC value than METHOD B, that information would be present in the final ranking

$$\underset{3}{\text{METHOD A}} < \underset{2}{\text{METHOD B}} < \underset{1}{\text{METHOD C}} < \underset{0}{\text{METHOD D}}.$$

If, on the other hand, the area under curve test was statistically insignificant, the original ranking and associated Borda values would be used.

If the test set under consideration contained 6 functions, $f_{1:6}$, and each reported the rankings below, the proposed aggregation strategies would generate the total rankings found in Table 1.

$$\begin{aligned} f_1 &: \quad \text{A} < \text{B} < \text{C} < \text{D}, \\ f_2 &: \quad (\text{A, B}) < (\text{C, D}), \\ f_3 &: \quad \text{C} < \text{A} < (\text{B, D}), \\ f_4 &: \quad \text{D} < (\text{A, C}) < \text{B}, \\ f_5 &: \quad (\text{A, B, C, D}) \text{ all tied}, \\ f_6 &: \quad \text{B} < (\text{A, C, D}), \end{aligned}$$

Table 1: Sample aggregation of results

ALGORITHM	BORDA	FIRSTS	TOP THREE
METHOD A	8	3	6
METHOD B	7	3	6
METHOD C	5	2	6
METHOD D	3	2	5

Tables of this form appear in Section 3, where the TOP THREE criterion is more insightful than in this example; to account for ties, all algorithms at the top of a function's "ballot" receive credit in the FIRSTS column, and similarly for the TOP THREE column.

2.1.2. STATISTICAL CONSIDERATIONS

Some previous empirical analyses of optimization methods prefer parametric statistics using the central limit theorem (Eggensperger et al., 2013; Bergstra et al., 2014) to the nonparametric statistics we employ (Hutter et al., 2011a). Potential nonnormality of samples of $f_{\text{best}}[T]$ makes parametric statistics on small sample sizes a treacherous endeavor.

As an example, consider optimization using the simple optimization method random search (Bergstra and Bengio, 2012); each suggestion $\mathbf{x}_i$ is chosen i.i.d. from $X \sim \text{Unif}(\mathcal{X})$. Observed function values $y_i = f(\mathbf{x}_i)$ are therefore i.i.d. realizations of some random variable Y, whose distribution is determined by f and $\mathcal{X}$. After T random samples, the smallest observation becomes the best found value, thus $f_{\text{best}}[T] = \min\{y_1, \ldots, y_T\}$.

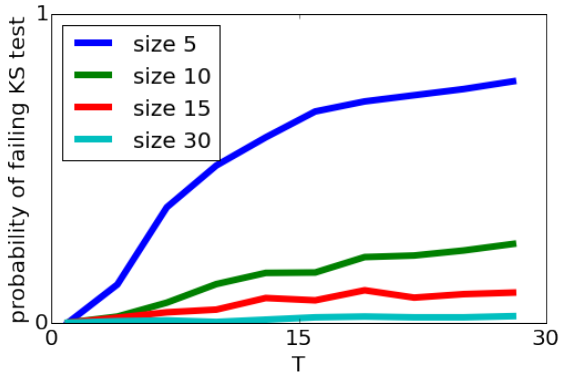

Figure 2: Demonstration that an increasingly accurate random search (increasing T) yields results which look less normal (more likely to fail a Kolmogorov-Smirnov test) and thus demand nonparametric statistical analysis.

The minimum of these i.i.d. values is the first order statistic $Y_{(1)} \equiv f_{\text{best}}[T]$, whose cumulative distribution function can be determined through the CDF of the random observations,

$$\begin{aligned} F_{Y_{(1)}}(y) = P(Y_{(1)} < y) &= 1 - P(\min\{Y_1, \ldots, Y_T\} > y) \\ &= 1 - P(Y_1 > y) \cdots P(Y_T > y) = 1 - \left[1 - (F_Y(y))\right]^T . \end{aligned}$$

A similar quantity phrased in a different context is presented in Bergstra and Bengio (2012).

The viability of a t test for studying samples of $Y_{(T)}$ is dependent on how closely the distribution of sample means matches a normal. The central limit theorem guarantees this for sufficiently large samples, but, because $F_{Y_{(1)}}(y)$ is certainly not a normal CDF, it is unlikely that small sample means are sufficiently normal. Additionally, as T increases the distribution becomes increasingly skewed positive because $Y_{(1)}$ has a minimum value, $f(\mathbf{x}_{\text{opt}})$.

The impact of this on an example is displayed graphically in Figure 2. We minimized the 1D function $f(x) = |x|$ on $x \in [-1, 1]$ with random search over T function evaluations. Sample means of size $\{5, 10, 15, 30\}$ were taken from that distribution and tested with a Kolmogorov-Smirnov test for normality; each KS test used 500 samples and significance .05. We ran 800 KS tests for each T value and graphed the probability of rejecting the null hypothesis – the belief that sample means are normally distributed and t tests are appropriate. This result can be verified analytically by studying the appropriate Berry-Esseen inequality (Korolev and Shevtsova, 2010). Furthermore, this situation is likely exacerbated by the use of a "better" optimization method because it will more quickly approach $\mathbf{x}_{\text{opt}}$ and likely be more positively skewed.

2.2. Optimizer Performance as Multicriteria Optimization

In some industrial settings it may be the case that the domain expert (e.g., data scientist or financial analyst), wants to use an AutoML tool without possessing any insight as to the tool's internal specifications or design methodology. Indeed, this is the presumed goal of an AutoML tool. These users likely still have a preference between the speed of the training process and the accuracy of the resulting model (and thus a preference as to the desired optimization tool), but they may not have the means to express that preference in explicit numerical terms, i.e., a function.

We can phrase this tradeoff between competing metrics as a multicriteria optimization problem (Ehrgott, 2006; Jahn, 2009); in the context of these two metrics we have introduced, users simultaneously want to maximize the performance while minimizing the training time. Unfortunately, because an improvement in one of these objectives likely causes worse performance in the other, there likely is no unique solution to this problem. Some judgment must be made between all the *Pareto efficient* results as to which is most desired, which is especially difficult for users without machine learning expertise.

That balance between competing objectives is often resolved with some sort of scalarization, where all of the objectives are combined into a scalar function which can then be optimized; however, doing so requires knowledge of the relative significance of each component which is probably implicitly, but not explicitly, known by the user. Our proposed solution to this multicriteria problem falls along lexicographic lines (Fishburn, 1975; Harzheim, 2005): we suggest that a user state their preference between available criteria (accuracy before speed in the example above with more discussed in the next section) and we use those preferences, up to statistical ties, to rank the available optimizers.

This method loses some of the subtlety in potential tradeoffs (e.g., a small but statistically significant gain in best found value would trump a massive gain in area under curve) but allows for minimal information from the end user. In some circumstances, especially those with users primarily focused on their application, this reduced interaction may be preferable.

2.2.1. ALTERNATE METRICS

Our metrics are by no means the only criteria by which optimization algorithms can, or should, be judged. **Best Found** measures proximity to the optimal function value $f(\mathbf{x}_{\text{opt}})$, but not proximity to the optimal vector $\mathbf{x}_{\text{opt}}$ which could be more insightful in some AutoML-pertinent circumstances. The metrics could account for the probabilistic nature of the problem; for example, the probability of $f_{\text{best}}[T]$ being more than 10% from the optimal value (Dolan and Moré, 2002). Knowledge of $\mathbf{x}_{\text{opt}}$ also permits use of the gap metric, $(f(\mathbf{x}_1)-f_{\text{best}}[T])/(f(\mathbf{x}_1)-f(\mathbf{x}_{\text{opt}}))$ (Huang et al., 2006; Brochu et al., 2010) which is cleanly scaled between 0 and 1. Cumulative regret, $\sum_{i=1}^{T}(f(\mathbf{x}_i) - f(\mathbf{x}_{\text{opt}}))$, penalizes suggestions which do not improve f_{best} (Srinivas et al., 2010; Bull, 2011) and thus may be valuable for an online automated machine learning setting.

The f_{LB} term in the **Area Under Curve** metric serves only a cosmetic purpose – its omission would produce an equivalent *averaged* f_{best} metric. An averaging of the gap metric could be similarly considered. Many AutoML problems have variable computational cost for different $\mathbf{x}_{\text{opt}}$ values (e.g., SVM training time as related to the box constraint), thus the total resources (e.g., CPU time) expended on function evaluations can also be an important metric. The time required for the entire SMBO may also be relevant when comparing software implementations (Martinez-Cantin, 2014). Furthermore, there could exist different judgments for each module of the Bayesian optimization scheme, which parallels the discussion in Hoffman and Shahriari (2014).

3. Ranking Demonstration

We conducted numerical experiments on functions from McCourt (2016), a test suite derived from Gavana (2013); information is provided there explaining exactly which functions were used. The use of such artificial test functions is well established within the Bayesian optimization community and plays an important role in current research (Snoek et al., 2015; Hernández-Lobato et al., 2015; González et al., 2015). The metrics described in Section 2 were recorded over 30 optimizations per algorithm per function and aggregated as described in Section 2.1. All algorithms were terminated at 80 function evaluations, unless the function was in dimension $d < 4$ when optimization was terminated after $20d$ evaluations. This decision was made for simplicity of comparison and, in a real AutoML setting, each method may have a preferred stopping criteria which should be observed.

Four Bayesian optimization methods are studied in our evaluation. SPEARMINT (Snoek et al., 2012, 2014b) (the HIPS implementation in Snoek et al. (2014a)) and SIGOPT, which is derived from MOE (Clark et al., 2014), use Gaussian processes to model f. SMAC (Hutter et al., 2011a,b) uses random forests to model f; we utilized the python wrapper pySMAC (Falkner, 2014). HYPEROPT (Bergstra et al., 2013b) uses tree-structured Parzen estimators to facilitate the optimization, and we used the standard python library implementation (Bergstra et al., 2013a) in our experiments.

In addition to these Bayesian methods, we also consider grid search (GRID), random search (Bergstra and Bengio, 2012) (RANDOM), and particle swarm optimization (PSO) (Kennedy and Eberhart, 1995). For grid search, we randomly sample without replacement from a grid designed

Table 2: Comparison between U / t tests

ALGORITHM	BORDA	FIRSTS	TOP THREE
HYPEROPT	162 / 153	12 / 15	48 / 38
SIGOPT	297 / 289	39 / 49	59 / 58
SMAC	47 / 41	3 / 5	14 / 10
SPEARMINT	253 / 222	24 / 19	47 / 46
GRID	50 / 34	5 / 6	14 / 12
PSO	162 / 129	9 / 18	51 / 52
RANDOM	40 / 36	3 / 5	9 / 8

Table 3: Comparison with/without AUC results

ALGORITHM	BORDA	FIRSTS	TOP THREE
HYPEROPT	138 / 120	8 / 10	31 / 31
SIGOPT	236 / 211	35 / 42	43 / 43
SMAC	42 / 34	1 / 2	6 / 6
SPEARMINT	183 / 169	9 / 26	32 / 40
GRID	34 / 28	2 / 3	5 / 6
PSO	129 / 119	4 / 8	33 / 37
RANDOM	36 / 29	2 / 3	5 / 5

with roughly 1 million grid points, a necessary adaptation in higher dimensional settings. For PSO, we used the pyswarm (Lee, 2014) implementation with $2d$ particles.

In Section 2.1.2, we explained our preference for the Mann-Whitney U test over traditional t tests for comparing optimization performance. Table 2 displays a stark shift in FIRSTS results with relative consistency in TOP THREE, suggesting that the parametric tests are making inappropriate assumptions regarding the best performers. Note that both the t tests and U tests were conducted with the $\alpha = .0005$ significance explained in Section 2.1.

As discussed in Section 2.1, the ranking scheme we propose involves a hierarchy of metrics: first the **Best Found** and second the **Area Under Curve**, with more possible if desired. Table 3 gives an example of the contrast between ranking with and without AUC. The decrease in FIRSTS without a similar decrease in TOP THREE again suggests that the AUC test is helping better identify the top performers for each function. It is also of note that the better resolution (reduction in ties) also increases the Borda scores overall.

4. Conclusions

The necessity for black-box optimization for, among other things, hyperparameter optimization has yielded the development of a variety of optimization methods, which has created a need to compare them. We have proposed a strategy for which a set of problems of interest can be provided and a set of optimization methods can be ranked based on their performance on those problems. This ranking strategy utilizes nonparametric statistical analysis to avoid potential problems associated with non-normality. It also allows for a hierarchy of metrics by which optimization methods can be judged, providing more ability to refine the rankings. Future work will include additional statistical analysis on the skewness of optimization results from methods other than random search and experimentation with this ranking strategy to translate performance of optimizers on constructed functions to specific AutoML problems.

References

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554, 2011.

James Bergstra, Daniel Yamins, and David Cox. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. https://github.com/hyperopt/hyperopt, 2013a.

James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of The 30th International Conference on Machine Learning*, pages 115–123, 2013b.

James Bergstra, Brent Komer, Chris Eliasmith, and David Warde-Farley. Preliminary evaluation of hyperopt algorithms on HPOLib. In *ICML workshop on AutoML*, 2014.

Eric Brochu, Tyson Brochu, and Nando de Freitas. A bayesian interactive optimization approach to procedural animation design. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 103–112. Eurographics Association, 2010.

Adam D Bull. Convergence rates of efficient global optimization algorithms. *The Journal of Machine Learning Research*, 12:2879–2904, 2011.

Scott Clark, Eric Liu, Peter Frazier, JiaLei Wang, Deniz Oktay, and Norases Vesdapunt. MOE: A global, black box optimization engine for real world metric optimization. https://github.com/Yelp/MOE, 2014.

Adam Coates, Honglak Lee, and Andrew Y Ng. An analysis of single-layer networks in unsupervised feature learning. *JMLR workshop and conference proceedings*, 15:215–223, 2011.

Wade D Cook, Boaz Golany, Michal Penn, and Tal Raviv. Creating a consensus ranking of proposals from reviewers partial ordinal rankings. *Computers & Operations Research*, 34(4):954–965, 2007.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1–30, 2006.

Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91(2):201–213, 2002.

Cynthia Dwork, Ravi Kumar, Moni Naor, and Dandapani Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th international conference on World Wide Web*, pages 613–622. ACM, 2001.

Katharina Eggensperger, Matthias Feurer, Frank Hutter, James Bergstra, Jasper Snoek, Holger Hoos, and Kevin Leyton-Brown. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice*, 2013.

Katharina Eggensperger, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Efficient benchmarking of hyperparameter optimizers via surrogates. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Matthias Ehrgott. *Multicriteria optimization*. Springer Science & Business Media, 2006.

Stefan Falkner. pysmac : simple python interface to SMAC. https://github.com/automl/pysmac, 2014.

Michael P. Fay and Michael A. Proschan. On assumptions for hypothesis tests and multiple interpretations of decision rules. *Statistics Surveys*, 4:1–39, 2010.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, pages 2944–2952, 2015.

Peter C Fishburn. Axioms for lexicographic preferences. *The Review of Economic Studies*, 42(3): 415–419, 1975.

Andrea Gavana. AMPGO global optimization benchmark functions. https://github.com/andyfaff/ampgo, 2013.

Javier González, Michael Osborne, and Neil D Lawrence. GLASSES: Relieving the myopia of bayesian optimisation. In *NIPS Workshop on Bayesian Optimization*, 2015.

Egbert Harzheim. *Ordered Sets*, volume 7 of *Advances in Mathematics*. Springer, 2005.

José Miguel Hernández-Lobato, Michael A Gelbart, Matthew W Hoffman, Ryan P Adams, and Zoubin Ghahramani. Predictive entropy search for bayesian optimization with unknown constraints. *Proceedings of The 32nd International Conference on Machine Learning*, 2015.

Matthew W Hoffman and Bobak Shahriari. Modular mechanisms for bayesian optimization. In *NIPS Workshop on Bayesian Optimization*, 2014.

Deng Huang, Theodore T Allen, William I Notz, and Ning Zeng. Global optimization of stochastic black-box systems via sequential kriging meta-models. *Journal of global optimization*, 34(3): 441–466, 2006.

Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, pages 507–523. Springer, 2011a.

Frank Hutter, Holger H Hoos, Kevin Leyton-Brown, Kevin Murphy, and Steve Ramage. SMAC: Sequential model-based algorithm configuration. http://www.cs.ubc.ca/labs/beta/Projects/SMAC/, 2011b.

Johannes Jahn. *Vector optimization*. Springer, 2009.

James Kennedy and Russell C. Eberhart. Particle swarm optimization. In *Proceedings of the 1995 IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, Perth, Australia, IEEE Service Center, Piscataway, NJ, 1995.

Viktor Y Korolev and Irina G Shevtsova. On the upper bound for the absolute constant in the Berry-Esseen inequality. *Theory of Probability & Its Applications*, 54(4):638–658, 2010.

Abraham Lee. pyswarm : Particle swarm optimization (PSO) with constraint support. https://github.com/tisimst/pyswarm, 2014.

Ruben Martinez-Cantin. Bayesopt: A bayesian optimization library for nonlinear optimization, experimental design and bandits. *The Journal of Machine Learning Research*, 15(1):3735–3739, 2014.

Ruben Martinez-Cantin, Nando de Freitas, Arnaud Doucet, and José A Castellanos. Active policy learning for robot planning and exploration under uncertainty. In *Robotics: Science and Systems*, pages 321–328, 2007.

Michael McCourt. Optimization Test Functions. https://github.com/sigopt/evalset, 2016.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

Jasper Snoek, Kevin Swersky, Hugo Larochelle, Michael Gelbart, Ryan P Adams, and Richard S Zemel. Spearmint : Bayesian optimization software. https://github.com/HIPS/Spearmint, 2014a.

Jasper Snoek, Kevin Swersky, Richard S Zemel, and Ryan Prescott Adams. Input warping for bayesian optimization of non-stationary functions. In *International Conference on Machine Learning*, 2014b.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Md. Mostofa Ali Patwary, Prabhat, and Ryan P. Adams. Scalable bayesian optimization using deep neural networks. In *International Conference on Machine Learning*, 2015.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design. In *Proceedings of the 27th International Conference on Machine Learning*, 2010.

Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855. ACM, 2013.

D. Wackerly, W. Mendenhall, and R.L. Scheaffer. *Mathematical Statistics with Applications*. Cengage Learning, 2014.