Hyperparameter Search in Machine Learning

Marc Claesen, Bart De Moor

STADIUS Center for Dynamical Systems, Signal Processing and Data Analytics
KU Leuven, Department of Electrical Engineering (ESAT) – iMinds, Department of Medical IT
Kasteelpark Arenberg 10, box 2446, 3001 Leuven, Belgium
marc.claesen@esat.kuleuven.be, bart.demoor@esat.kuleuven.be

Abstract

We describe the hyperparameter search problem in the field of machine learning and discuss its main challenges from an optimization perspective. Machine learning methods attempt to build models that capture some element of interest based on given data. Most common learning algorithms feature a set of hyperparameters that must be determined before training commences. The choice of hyperparameters can significantly affect the resulting model's performance, but determining good values can be complex; hence a disciplined, theoretically sound search strategy is essential.

1 Introduction

Machine learning research focuses on the development of methods that are capable of capturing some element of interest from a given data set. Such elements include but are not limited to coherent structures within data (clustering) or the ability to predict certain target values based on given characteristics, which may be discrete (classification) or continuous (regression).

A large variety of learning methods exist, ranging from biologically inspired neural networks [7] over kernel methods [29] to ensemble models [9, 11]. A common trait in these methods is that they are parameterized by a set of hyperparameters λ, which must be set appropriately by the user to maximize the usefulness of the learning approach. Hyperparameters are used to configure various aspects of the learning algorithm and can have wildly varying effects on the resulting model and its performance.

Hyperparameter search is commonly performed manually, via rules-of-thumb [19, 20] or by testing sets of hyperparameters on a predefined grid [28]. These approaches leave much to be desired in terms of reproducibility and are impractical when the number of hyperparameters is large [10]. Due to these flaws, the idea of automating hyperparameter search is receiving increasing amounts of attention in machine learning, for instance via benchmarking suites [15] and various initiatives.[1] Automated approaches have already been shown to outperform manual search by experts on several problems [2, 5].

We briefly introduce some key challenges inherent to hyperparameter search in Section 2. The combination of all these hurdles make hyperparameter search a formidable optimization task. In Section 3 we give a succinct overview of the current state-of-the-art in terms of algorithms and available software.

1.1 Example: controlling model complexity

A key balancing act in machine learning is choosing an appropriate level of model complexity: if the model is too complex, it will fit the data used to construct the model very well but generalize poorly to unseen data (overfitting); if the complexity is too low the model won't capture all the information in the data (underfitting). This is often referred to as the bias-variance trade-off [12, 17], since a complex model exhibits large variance while an overly simple one is strongly biased. Most general-purpose methods feature hyperparameters to control this trade-off; for instance via regularization as in support vector machines and regularization networks [16, 18].

1.2 Formalizing hyperparameter search

The goal of many machine learning tasks can be summarized as training a model $\mathcal{M}$ which minimizes some predefined loss function $\mathcal{L}(\mathbf{X}^{(te)};\ \mathcal{M})$ on given test data $\mathbf{X}^{(te)}$. Common loss functions include

[1] Such as `http://www.automl.org/` and `https://www.codalab.org/competitions/2321`.

mean squared error and error rate. The model $\mathcal{M}$ is constructed by a learning algorithm $\mathcal{A}$ using a training set $\mathbf{X}^{(tr)}$; typically involving solving some (convex) optimization problem. The learning algorithm $\mathcal{A}$ may itself be parameterized by a set of hyperparameters λ, e.g. $\mathcal{M} = \mathcal{A}(\mathbf{X}^{(tr)};\ \lambda)$. An example model $\mathcal{M}$ is a support vector machine classifier with Gaussian kernel [29], for which the training problem $\mathcal{A}$ is parameterized by the regularization constant C and kernel bandwidth σ, i.e. $\lambda = [C, \sigma]$.

The goal of hyperparameter search is to find a set of hyperparameters $\lambda^\star$ that yield an optimal model $\mathcal{M}^\star$ which minimizes $\mathcal{L}(\mathbf{X}^{(te)};\ \mathcal{M})$. This can be formalized as follows [10]:

$$\lambda^\star = \arg\min_{\lambda} \mathcal{L}\big(\mathbf{X}^{(te)};\ \mathcal{A}(\mathbf{X}^{(tr)};\ \lambda)\big) = \arg\min_{\lambda} \mathcal{F}(\lambda;\ \mathcal{A},\ \mathbf{X}^{(tr)}, \mathbf{X}^{(te)},\ \mathcal{L}). \tag{1}$$

The objective function $\mathcal{F}$ takes a tuple of hyperparameters λ and returns the associated loss. The data sets $\mathbf{X}^{(tr)}$ and $\mathbf{X}^{(te)}$ are given and the learning algorithm $\mathcal{A}$ and loss function $\mathcal{L}$ are chosen . Depending on the learning task, $\mathbf{X}^{(tr)}$ and $\mathbf{X}^{(te)}$ may be labeled and/or equal to each other. In supervised learning, a data set is often split into $\mathbf{X}^{(tr)}$ and $\mathbf{X}^{(te)}$ using hold-out or cross-validation methods [14, 24].

2 Challenges in hyperparameter search

The characteristics of the search problem depend on the learning algorithm $\mathcal{A}$, the chosen loss function $\mathcal{L}$ and the data set $\mathbf{X}^{(tr)}$, $\mathbf{X}^{(te)}$, as shown in Equation (1). Hyperparameter search is typically approached as a non-differentiable, single-objective optimization problem over a mixed-type, constrained domain. In this section we will discuss the origins and consequences of challenges in hyperparameter search.

2.1 Costly objective function evaluations

Each objective function evaluation requires evaluating the performance of a model trained with hyperparameters λ. Depending on the available computational resources, the nature of the learning algortihm $\mathcal{A}$ and size of the problem ($\mathbf{X}^{(tr)}$, $\mathbf{X}^{(te)}$) each evaluation may take considerable time. Training times in the order of minutes are considered fast, since days and even weeks are not unheard of [13, 25, 31]. Evaluation time is exacerbated when procedures that train multiple models are employed; for instance to reliably estimate generalization performance [14, 24]. This leads to an increasing need for efficient methods to optimize hyperparameters that require a minimal amount of objective function evaluations.

Additionally, the time required to train and test models can be contingent upon the choice of hyperparameters. Some hyperparameters have an obvious influence on train and/or test time, e.g. the architecture of neural networks [7] and size of ensembles [9, 11]. The influence of hyperparameters can also be subtle, for instance regularization and kernel complexity can significantly affect training time for support vector machines [8].

2.2 Randomness

The objective function often exhibits a stochastic component, which can be induced by various components of the machine learning pipeline, for example due to inherent randomness of the learning algorithm (initialization of a neural network, resampling in ensemble approaches, ...) or due to finite sample effects in estimating generalization performance. This stochasticity can sometimes be addressed via machine learning techniques; but unfortunately such solutions typically dramatically increase the time required per objective function evaluation, limiting their usefulness in some settings.

This inherent stochasticity directly implies that the empirical best hyperparameter tuple, obtained after a given set of evaluations, is not necessarily the true optimum of interest $\lambda^\star$. Fortunately, many search methods are designed to probe many tuples close to the empirical best. If the search region surrounding the empirical optimum is densely sampled, we can determine whether the empirical best was an outlier or not in a post-processing phase, for instance by assuming Lipschitz continuity or smoothness.

2.3 Complex search spaces

The number of hyperparameters is usually small (≤ 5), but it can range up to hundreds for complex learning algorithms [4] or when preprocessing steps are also subjected to optimization [22]. It has been demonstrated empirically that in many cases only a handful of hyperparameters significantly impact performance, though identifying the relevant ones in advance is difficult [2].

Hyperparameters are usually of continuous or integer type, leading to mixed-type optimization problems. Continuous hyperparameters are commonly related to regularization. Common integer hyperparameters are related to network architecture for neural networks [7], size of ensembles [9, 11] or the parameterization of kernels in kernel methods [29].

Some tasks feature highly complex search spaces, in which the very existence of certain hyperparameters are conditional upon the value of others [3, 5, 22]. A simple example is optimizing the architecture of neural networks [7], where the number of hidden layers is one hyperparameter and the size of each layer induces a set of additional hyperparameters, conditional upon the number of layers.

3 Current approaches

A wide variety of optimization methods have been used for hyperparameter search, including particle swarm optimization [26, 27], genetic algorithms [32], coupled simulated annealing [33] and racing algorithms [6]. Surprisingly, randomly sampling the search space was only established recently as a baseline for comparison of optimization methods [2]. Bayesian and related sequential model based optimization techniques using variants of the expected improvement criterion [23] are receiving a lot of attention currently [1, 5, 15, 21, 30], owing to their efficiency in terms of objective function evaluations.

Software packages are being released which implement various dedicated optimization methods for hyperparameter search. Such packages are usually intended to be used in synergy with machine learning libraries that provide learning algorithms [28]. Most of these packages focus on Bayesian methods [3, 22, 30], though metaheuristic optimization approaches are also offered [10]. The increased development of such packages testifies towards the growing interest in automated hyperparameter search.

4 Conclusion

A fully automated, self-configuring learning strategy can be considered the holy grail of machine learning. Though the current state-of-the-art still has a long way to go before this goal can be reached, it is evident that hyperparameter search is a crucial element in its pursuit. Automated hyperparameter search is a hot topic within the machine learning community which we believe can benefit greatly from the techniques and lessons learnt in metaheuristic optimization.

Acknowledgments

This research was funded by the Flemisch Government via the following funding channels: FWO: projects: G.0871.12N (Neural circuits); IWT: TBM-Logic Insulin(100793), TBM Rectal Cancer(100783), TBM IETA(130256), PhD grants (111065); Industrial Research fund (IOF): IOF Fellowship 13-0260; iMinds Medical Information Technologies SBO 2015, ICON projects (MSIpad, MyHealthData); VLK Stichting E. van der Schueren: rectal cancer and the Belgian Federal Government via FOD: Cancer Plan 2012-2015 KPC-29-023 (prostate) and COST: Action: BM1104: Mass Spectrometry Imaging.

References

[1] François Bachoc. Cross validation and maximum likelihood estimations of hyper-parameters of gaussian processes with model misspecification. *Computational Statistics & Data Analysis*, 66:55–69, 2013.

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

[3] James Bergstra, Dan Yamins, and David D Cox. Hyperopt: A Python library for optimizing the hyperparameters of machine learning algorithms. In *Proceedings of the 12th Python in Science Conference*, pages 13–20. SciPy, 2013.

[4] 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, 2013.

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

[6] Mauro Birattari, Zhi Yuan, Prasanna Balaprakash, and Thomas Stützle. F-race and iterated f-race: An overview. In *Experimental methods for the analysis of optimization algorithms*, pages 311–336. Springer, 2010.

[7] Christopher M Bishop et al. Neural networks for pattern recognition. 1995.

[8] Léon Bottou and Chih-Jen Lin. Support vector machine solvers. *Large scale kernel machines*, pages 301–320, 2007.

[9] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[10] Marc Claesen, Jaak Simm, Dusan Popovic, Yves Moreau, and Bart De Moor. Easy hyperparameter search using Optunity. *CoRR*, abs/1412.1114, 2014. `http://www.optunity.net`.

[11] Marc Claesen, Frank De Smet, Johan A.K. Suykens, and Bart De Moor. EnsembleSVM: A library for ensemble learning using support vector machines. *Journal of Machine Learning Research*, 15:141–145, 2014.

[12] Felipe Cucker and Steve Smale. Best choices for regularization parameters in learning theory: on the bias-variance problem. *Foundations of Computational Mathematics*, 2(4):413–428, 2002.

[13] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, pages 1223–1231, 2012.

[14] Bradley Efron and Gail Gong. A leisurely look at the bootstrap, the jackknife, and cross-validation. *The American Statistician*, 37(1):36–48, 1983.

[15] 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.

[16] Theodoros Evgeniou, Massimiliano Pontil, and Tomaso Poggio. Regularization networks and support vector machines. *Advances in Computational Mathematics*, 13(1):1–50, 2000.

[17] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992.

[18] Trevor Hastie, Saharon Rosset, Robert Tibshirani, and Ji Zhu. The entire regularization path for the support vector machine. In *Journal of Machine Learning Research*, pages 1391–1415, 2004.

[19] Geoffrey E Hinton. A practical guide to training restricted boltzmann machines. In *Neural Networks: Tricks of the Trade*, pages 599–619. Springer, 2012.

[20] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification, 2003.

[21] 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, 2011.

[22] Frank Hutter, Holger H Hoos, Kevin Leyton-Brown, and Thomas Stützle. ParamILS: an automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36(1):267–306, 2009.

[23] Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

[24] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 1137–1145, 1995.

[25] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[26] Shih-Wei Lin, Kuo-Ching Ying, Shih-Chieh Chen, and Zne-Jung Lee. Particle swarm optimization for parameter determination and feature selection of support vector machines. *Expert systems with applications*, 35(4):1817–1824, 2008.

[27] Michael Meissner, Michael Schmuker, and Gisbert Schneider. Optimized particle swarm optimization (OPSO) and its application to artificial neural network training. *BMC bioinformatics*, 7(1):125, 2006.

[28] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[29] Bernhard Schölkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[30] 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.

[31] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

[32] Jinn-Tsong Tsai, Jyh-Horng Chou, and Tung-Kuan Liu. Tuning the structure and parameters of a neural network by using hybrid taguchi-genetic algorithm. *Neural Networks, IEEE Transactions on*, 17(1):69–80, 2006.

[33] Samuel Xavier-de Souza, Johan AK Suykens, Joos Vandewalle, and Désiré Bollé. Coupled simulated annealing. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 40(2):320–335, 2010.