Speeding Up Classification Algorithms in Big Data Environments

Clemens Damke

Intelligent Systems Group (ISG)
Department of Electrical Engineering, Computer Science and Mathematics
Warburger Straße 100
33098 Paderborn

Abstract. TODO

Keywords: Big Data · Automated Machine Learning

1 Introduction

Over the last few years Big Data processing has become increasingly important in many domains. This increase in the data volume also poses new challenges for machine learning applications. The training time of learners is usually polynomially dependent on the size of the training dataset $\mathcal{D}_{train}$, i. e. $\mathcal{O}(|\mathcal{D}_{train}|^{\alpha}), \alpha \geq 1$. Since training has to be repeated for every iteration of cross-validation and hyperparameter search, always using the entire dataset quickly becomes infeasible. This paper gives an overview of approaches to tackle this problem with a focus on the domain of classification problems.

The process of finding a model can in general be split into two phases:

1. **Hyperparameter search:** Optimizing a vector λ in the hyperparameter space Λ_L of a learner L representing a hypothesis space $\mathcal{H}_\lambda$. A naïve approach to do this is to systematically try configurations using a grid search or a random search over Λ_L. To evaluate the quality of a given λ, L is usually trained on a training dataset $\mathcal{D}_{train}$ using λ. This yields a hypothesis $\hat{h}_\lambda \in \mathcal{H}_\lambda$ that is evaluated using a validation dataset $\mathcal{D}_{valid}$. The goal of hyperparameter optimization is to minimize the empirical error $f(\lambda)$ of $\hat{h}_\lambda$ on $\mathcal{D}_{valid}$, i. e. to find an approximation $\hat{\lambda}$ of $\lambda^* := \arg\min_\lambda f(\lambda)$.
2. **Training or parameter search:** Let w be a vector in the parameter space $W_{\mathcal{H}_\lambda}$, describing a hypothesis $h_{\lambda,w} \in \mathcal{H}_\lambda$ given a hyperparameter configuration λ. The goal of parameter search is to find an approximation $\hat{h}_\lambda$ of the hypothesis $h^*_\lambda := \arg\min_{h_{\lambda,w}} e(\mathcal{D}_{train}|h_{\lambda,w})$, with $e(\mathcal{D}_{train}|h_{\lambda,w})$ being the empirical error of $h_{\lambda,w}$ on a given training dataset $\mathcal{D}_{train}$ according to some error measure. Depending on the learner L, various kinds of optimization methods are used to find this minimum, e. g. Bayesian optimization, quadratic programming or, if $\nabla_w e(\mathcal{D}_{train}|h_{\lambda,w})$ is computable, gradient descent. The quality f of $\hat{h}_\lambda$ is measured by the error on a validation or test dataset, i. e. $f(\lambda) := e(\mathcal{D}_{valid}|\hat{h}_\lambda)$.

This paper is structured according to those two phases. Section 2 describes ways to speed up the hyperparameter search. Section 3 then describes how to improve the

training methods of existing learners. Most of the techniques described in this paper improve upon independent components of the model finding process allowing them to be combined.

2 Hyperparameter optimization

As described in the introduction, the goal of hyperparameter optimization is to find a global minimum of f. Since f is generally unknown, analytical methods or gradient descent cannot usually be applied. The only way the get information about f is to evaluate it, which is costly. There are multiple ways to reduce the total cost of those evaluations:

1. **Number T of evaluations of f:** During optimization multiple hyperparameter configurations $\lambda_1, \dots, \lambda_T$ will be evaluated using f. T is usually fixed when using a grid search or a random search. After evaluating T configurations, the best one is chosen. Those naïve approaches assume that $f(\lambda)$ is independent of $f(\lambda')$ for all pairs $\lambda \neq \lambda'$. We will see that this strong assumption of independence is not necessarily true which in turn allows reducing T.

2. **Training dataset size S:** The performance of a given configuration $f(\lambda)$ is computed by training the learner on $\mathcal{D}_{train}$ which is expensive for big datasets. By training on S instead of $|\mathcal{D}_{train}|$ datapoints the evaluation can be sped up.

3. **Number of parameter search iterations E:** Depending on the learner, training often is an iterative process, e. g. gradient descent. To speed up hyperparameter optimization training could be terminated before convergence.

2.1 FABOLAS

The first approach we will discuss is called FABOLAS (Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets) [Kle+16]. It can be applied to any learner L and is based upon two main ideas:

1. The function f is modeled as a *Gaussian process* (GP) $\hat{f}$ based on the assumption that two configurations λ_1 and λ_2 will perform similar if they are similar according to some kernel $k(\lambda_1, \lambda_2)$. The Gaussian process $\hat{f}$ is used as a surrogate to estimate the expected value and variance of f given λ. Using *Bayesian optimization* f will be probed at promising positions to iteratively improve $\hat{f}$. Hyperparameter configurations that are expected to perform worse than the current optimum will not be probed. This effectively reduces T.

2. The training dataset size S is modeled as an additional hyperparameter of $\hat{f}$. This allows extrapolating the value of f when trained on the complete dataset while only probing smaller subsets which effectively reduces S.

We will now describe how those two ideas can be applied.

Gaussian processes A Gaussian process is a family of random variables $(X_\theta)_{\theta\in\Theta}$, s. t. every finite subset of them follows a multivariate normal distribution. It is described by a prior mean function $\mu_0(\theta) = \mathbb{E}[X_\theta]$ and a positive-definite kernel $k(\theta_1, \theta_2) = Cov(X_{\theta_1}, X_{\theta_2})$. Let $\mathcal{D}_n = \{(\boldsymbol{\theta}_i, \boldsymbol{y}_i)\}_{i=1}^n$ denote a set of observations. Those observations can be used to update the means and variances of the RVs via Bayes rule:

$$\begin{aligned}
\boldsymbol{m} &:= (\mu_0(\boldsymbol{\theta}_1), \ldots, \mu_0(\boldsymbol{\theta}_n))^T \\
\boldsymbol{k}(\theta) &:= (k(\boldsymbol{\theta}_1, \theta), \ldots, k(\boldsymbol{\theta}_n, \theta))^T \\
\boldsymbol{K} &\in \mathbb{R}^{n\times n}, \boldsymbol{K}_{ij} := k(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) \\
\mathbb{E}[X_\theta|\mathcal{D}_n] &:= \mu_n(\theta) = m_0(\theta) + \boldsymbol{k}(\theta)^T \boldsymbol{K}^{-1}(\boldsymbol{y} - \boldsymbol{m}) \qquad (1) \\
Cov(X_\theta, X_{\theta'}|\mathcal{D}_n) &:= k(\theta, \theta') - \boldsymbol{k}(\theta)^T \boldsymbol{K}^{-1} \boldsymbol{k}(\theta') \qquad (2)
\end{aligned}$$

FABOLAS works by modeling the error function f as a Gaussian process $\hat{f} \sim \mathcal{GP}(m, k)$ with parameter set $\Theta := \Lambda \times [0, 1]$ and $\mu_0(\lambda, s) = \mathbb{E}[f(\lambda \,|\, \text{relative training set size } s)]$. To model the covariances between different combinations of hyperparameters and training set sizes, the following product kernel is used:

$$\begin{aligned}
k((\lambda, s), (\lambda', s')) &:= k_{5/2}(\lambda, \lambda') \cdot (\phi^T(s) \cdot \Sigma_\phi \cdot \phi(s')) \qquad (3) \\
k_{5/2}(\lambda, \lambda') &:= \beta \left(1 + \sqrt{5}\, d(\lambda, \lambda') + \frac{5}{3} d^2(\lambda, \lambda')\right) e^{-\sqrt{5}\, d(\lambda, \lambda')} \qquad (4)
\end{aligned}$$

TODO: explain the kernel (Matern kernel, Mahalanobis distance etc.)

Bayesian optimization To find $\arg\min_\lambda f(\lambda)$ the bias and variance of $\hat{f}$ has to be reduced by probing f. The estimated minimum after n probes then is described by $\arg\min_\lambda \mu_n(\lambda, s = 1)$, i. e. the configuration with the smallest predicted error on the full test dataset. To reduce the number of probes required until this minimum converges, an *acquisition function* is used. Its role is to trade-off exploration vs. exploitation of f by describing the expected utility of probing (λ_{n+1}, s_{n+1}) given a set of previous probes $\mathcal{D}_n$. FABOLAS uses an aquisition function that rates configurations by their *information gain* per computation time:

$$\begin{aligned}
a_F(\lambda, s) := \frac{1}{c(\lambda, s)} \mathbb{E}_y \Bigg[& p(y \,|\, \lambda, s, \mathcal{D}_n) \cdot \int p(\hat{\lambda} \in \arg\min_{\lambda'} f(\lambda') \,|\, \mathcal{D}_n \cup \{(\lambda, s, y)\}) \qquad (5) \\
& \cdot \log \frac{p(\hat{\lambda} \in \arg\min_{\lambda'} f(\lambda') \,|\, \mathcal{D}_n \cup \{(\lambda, s, y)\})}{u(\hat{\lambda})} \mathrm{d}\hat{\lambda} \Bigg]
\end{aligned}$$

TODO: explain the aquisition function (Kullback-Leibler divergence, cost estimation model)

2.2 Learning Curve Extrapolation

Domhan et al. [Dom+15]

3 Parameter optimization

3.1 Bag of Little Bootstraps

Kleiner et al. [Kle+11]

3.2 Sample Size Selection

Byrd et al. [Byr+12]

3.3 Subsampling for Logistic Regression

Wang et al. [Wan+17]

3.4 Local Learning

Al-Jarrah et al. [AJ+15]

4 Conclusion

TODO

References

[AJ+15] O. Y. Al-Jarrah, P. D. Yoo, S. Muhaidat, G. K. Karagiannidis, and K. Taha. "Efficient Machine Learning for Big Data: A Review". In: (Mar. 18, 2015). arXiv: 1503.05296v1 [cs.LG] (cit. on p. 4).

[Byr+12] Richard H. Byrd, Gillian M. Chin, Jorge Nocedal, and Yuchen Wu. "Sample size selection in optimization methods for machine learning". In: *Mathematical Programming* 134.1 (2012), pp. 127–155 (cit. on p. 4).

[Dom+15] Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. "Speeding Up Automatic Hyperparameter Optimization of Deep Neural Networks by Extrapolation of Learning Curves". In: *Proceedings of the 24th International Conference on Artificial Intelligence*. IJCAI'15. Buenos Aires, Argentina: AAAI Press, 2015, pp. 3460–3468 (cit. on p. 3).

[Kle+11] Ariel Kleiner, Ameet Talwalkar, Purnamrita Sarkar, and Michael I. Jordan. "A Scalable Bootstrap for Massive Data". In: (Dec. 21, 2011). arXiv: 1112.5016v2 [stat.ME] (cit. on p. 4).

[Kle+16] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. "Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets". In: (May 23, 2016). arXiv: 1605.07079v2 [cs.LG] (cit. on p. 2).

[Wan+17] HaiYing Wang, Rong Zhu, and Ping Ma. "Optimal Subsampling for Large Sample Logistic Regression". In: (Feb. 3, 2017). arXiv: 1702.01166v2 [stat.CO] (cit. on p. 4).