Speeding Up Classification Algorithms in Big Data Environments: A review

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.

The process of finding an optimal 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 $\mathcal{D}_{valid}$.
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 or approximate a hypothesis $\hat{h}_\lambda := \arg\min_{h_{\lambda,w}} e(\mathcal{D}_{train}|h_{\lambda,w})$, 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 usually 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

The runtime of hyperparameter optimization depends upon multiple independent variables:

1. **Number of iterations T:** During optimization multiple hyperparameter configurations $\lambda_1, \dots, \lambda_T$ will be evaluated against $\mathcal{D}_{valid}$. T is usually fixed when using a grid search or a random search. After evaluating T configurations, the best one is chosen. This method assumes that $f(\lambda_i)$ is independent of $f(\lambda_j)$ for all pairs $\lambda_i \neq \lambda_j$. We will see that this strong assumption of independence is not necessarily true which in turn allows estimating $\mathbb{E}[f]$ to guide the search.
2. **Training dataset size S:**

2.1 Bayesian Optimization

2.2 Learning Curve Extrapolation

3 Parameter optimization

3.1 Bag of Little Bootstraps

3.2 Sample Size Selection

3.3 Subsampling for Logistic Regression

3.4 Local Learning

4 Conclusion

TODO