New Generation Computing manuscript No.
(will be inserted by the editor)

Comprehensive analysis of gradient-based hyperparameter optimization algorithms

O. Y. Bakhteev · V. V. Strijov

Received: date / Accepted: date

Abstract The paper investigates hyperparameter optimization problem. Hyperparameters are the parameters of model parameter distribution. The adequate choice of hyperparameter values prevents model overfit and allows it to obtain higher predictive performance. Neural network models with large amount of hyperparameters are analyzed. The hyperparameter optimization for models is computationally expensive. The paper proposes modifications of various gradient-based methods to simultaneously optimize many hyperparameters. The paper compares the experiment results with the random search. The main impact of the paper is hyperparameter optimization algorithms analysis for the models with high amount of parameters. To select precise and stable models the authors suggest to use two model selection criteria: cross-validation and evidence lower bound. The experiments show that the models optimized using the evidence lower bound give higher error rate than the models obtained using cross-validation. These models also show greater stability when data is noisy. The evidence lower bound usage is preferable when the model tends to overfit or when the cross-validation is computationally expensive. The algorithms are evaluated on regression and classification datasets.

Keywords gradient descent · hyperparameter optimization · model selection · neural networks · classification · regression

The research was made possible by Government of the Russian Federation (Agreement 05.Y09.21.0018)

O. Y. Bakhteev
Moscow Institute of Physics and Technology
Tel.: +74991354163
E-mail: bakhteev@phystech.edu

V. V. Strijov
Moscow Institute of Physics and Technology

FRCCSC of the Russian Academy of Sciences

1 Introduction

The paper analyzes hyperparameter optimization problem. *Model* is a function superposition, which solves a classification or regression problem. *Model hyperparameters* are the parameters of model parameter distribution.

The paper focuses on the neural network models. The parameter optimization problem is computationally expensive because of training loss non-convexity. The amount of model parameters can reach millions [25] and such models optimization requires multiple days[29]. The hyperparameter values can influence significantly the quality of models [33,30]. These two facts make the hyperparameter optimization problem very important. The adequate choice of hyperparameter values also helps to prune redundant parameters to make the model more compact and stable [10,15].

In this paper gradient-based algorithms are analyzed. They optimize a large amount of hyperparameters within a short time in contrast to gradient-free algorithms [22]. The complexity of hyperparameter optimization is comparable to the model parameter optimization complexity, therefore, the parameters and hyperparameters are optimized in a single procedure. As a baseline, a random search algorithm is used. The pros and cons of all the algorithms are illustrated in Table 1. Opposing to papers [9,24,20] this research does not focus on the hold-out cross-validation and analyzes algorithms in general. The analyzed algorithms are implemented as a toolbox [1]. The authors propose a number of analyzed algorithms modifications: we extend all the algorithms to work with validation loss that can contain hyperparameters, for DrMad algorithm [9] we add an additional parameter that regularizes a length of analyzed parameter update trajectory, for HOAG algorithm [24] we use stochastic gradient descent for large linear system solving because of high dimension of the linear system. The main theoretical impact of the paper is the analysis of algorithm behavior with various validation loss functions and a quality and stability analysis of the resulting models. The experiments were conducted using cross-validation and evidence lower bound as a model selection criterion. Opposing to the paper [24], where the optimization algorithms comparison can also be found, this paper presents the results on the large datasets, such as WISDM [18] and MNIST [19], where the number of hyperparameters is significant. The experimental results show that the gradient-based algorithms are significantly more effective when the number of hyperparameters is large.

Related works. The papers [4,5] propose hyperparameters optimization strategies based on the random search. The current gold-standard methods [27,14] propose hyperparameter optimizations based on the probabilistic models construction for the optimal hyperparameter value prediction. These methods are ineffective whenever the number of hyperparameters is large [22].

In [22,7,9,24,20] the gradient-based hyperparameter optimization methods are suggested. The papers [9,22] propose the gradient-based methods that use reverse differentiation method. This method is similar to backpropagation. The main idea of this approach is to restore the full history of parameter

Algorithm	**Type**	**Pros**	**Cons**
Random search	Stochastic	Easy to implement	Ineffective in high dimensions (curse of dimension)
Greedy [20]	Gradient-based	Can be used in inner model parameter optimization	Greedy and non-optimal: the algorithm is correct whenever the loss function Hessian is identical.
HOAG [24]	Gradient-based	Fast convergence	The algorithm requires additional parameter configuration: it is required to solve the linear system, which depends on the Hessian of a loss function. The error in linear system solution influences the final model quality.
DrMAD [9]	Gradient-based	Considers the trajectory of parameter updates and the optimization algorithm.	Can be instable because of gradient explosion or vanishing. The algorithm is correct when the trajectory of parameter update is linear, which is mostly not true for the complex non-convex models.

Table 1: The analyzed algorithms properties.

updates in order to optimize hyperparameters using this history. In general this procedure is computationally expensive. In [20] the authors propose to use only the last update of parameters at each optimization iteration. This method can be considered as a greedy hyperparameter optimization algorithm. In [22] the model parameters are optimized using stochastic gradient descent with momentum, which allows to use less memory for storing the parameter update history. In [9] propose to linearize the history trajectory to effectively restore the update history. In [24] use an approximation of the exact hyperparameter gradient, which can be calculated when the problem satisfies some regular conditions.

For the hyperparameter optimization various model selection criteria can be used [21,6]. In [21,6,17,28] marginal likelihood or *evidence* is used. In [17, 28] the authors consider the problem of model selection and hyperparameter optimization for the linear regression problem. One of the marginal likelihood approximation methods is an *evidence lower bound*, which can be obtained using variational inference [6]. In [12] a stochastic version of the variational inference is proposed. In [26] the authors analyze the relation between the gradient-based evidence lower bound estimations and MCMC methods. An alternative model selection criterion is a minimum description length [11] that characterizes statistical complexity of the model. In [10] an evidence lower

bound estimation method for the neural networks is proposed. The authors analyze the relation between the evidence lower bound and minimum description length.

The other model selection criterion is the cross-validation [2,17]. One of this criterion problem is its high computational cost [32,16]. In [13,3] the authors analyze the variance of cross-validation model performance estimation. In [17] the authors compare hyperparameter values obtained during hyperparameter optimization using various model selection criteria.

2 Problem statement

There is given a dataset

$$\mathfrak{D} = \{(\mathbf{x}_i, y_i)\}, i = 1, \dots, m. \tag{1}$$

It contains the object matrix $\mathbf{X}$ and the label vector $\mathbf{y}$. The label y_i is in finite set, $y_i \in \mathbb{Y} = \{1, \dots, Z\}$, in case of Z-class classification problem. For the regression problem the label y_i is in the real-value subset $y_i \in \mathbb{Y} \subset \mathbb{R}$.

A model f predicts the variable y_i:

$$f : \mathbb{R}^n \to \mathbb{Y}, \quad \mathbf{w} \in \mathbb{R}^u.$$

It is differentiable with respect to parameters.

Introduce the probabilistic interpretation of the model f for the classification and regression problems.

Regression problem. Suppose the dependent variable y is normally distributed:

$$\mathbf{y} = \mathcal{N}\big(\mathbf{f}, \mathbf{I}\big), \quad \text{where } \mathbf{f} = \mathbf{f}(\mathbf{w}, \mathbf{X}). \tag{2}$$

Define the likelihood function $p(\mathbf{y}|\mathbf{X}, \mathbf{w})$:

$$\log p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = -\frac{m}{2}\log(2\pi) - \frac{1}{2}\big(\mathbf{y} - \mathbf{f}(\mathbf{w}, \mathbf{X})\big)^{\mathsf{T}}\mathbf{I}\big(\mathbf{y} - \mathbf{f}(\mathbf{w}, \mathbf{X})\big).$$

Classification problem. For the 2-class classification problem suppose the dependent variable y is distributed binomially:

$$\mathbf{y} = \mathcal{B}\big(1 - \mathbf{f}, \mathbf{f}\big), \tag{3}$$

where $\mathbf{f}$ is the probability that objects from the matrix $\mathbf{X}$ belong to the first class. For the multiclass classification problem the dependent variable y is distributed multinomially and r-th component f_r is the probability that objects from the matrix $\mathbf{X}$ belong to the class r. Define the likelihood function

$$\log p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \sum_{\mathbf{x}, y \in \mathbf{X}, \mathbf{y}} \sum_{r=1}^{Z} [y = r] \log f_r(\mathbf{w}, \mathbf{x}).$$

For both classification (3) and regression (2) problems define the prior distribution $p(\mathbf{w}|\mathbf{A})$:

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{A}^{-1}), \tag{4}$$

where $\mathbf{A}^{-1} = \text{diag}[\alpha_1, \ldots, \alpha_u]^{-1}$ is the covariance diagonal matrix. The hypotheses (2), (3) and (4) do not contradict each other since the normal distribution is unbounded [8].

Since the high dimensionality of parameter space and loss functions non-convexity some model selection methods use parameters of multiple models instances. The authors denote a vector of model parameters by $\mathbf{w}$ and by $\boldsymbol{\theta} \in \mathbb{R}^s$ a vector of parameters for multiple instances or model parameters distribution. The details of $\boldsymbol{\theta}$ construction are given in the model selection methods description.

Introduce an example of hyperparameter optimization using the coherent Bayesian inference. Optimize model parameters $\boldsymbol{\theta} = \mathbf{w}$ according to the model parameter distribution $p(\mathbf{w}|\mathbf{X}, \mathbf{y}, \mathbf{A})$:

$$\hat{\boldsymbol{\theta}} = \arg\max\big(-L(\boldsymbol{\theta}, \mathbf{A})\big) = p(\mathbf{w}|\mathbf{X}, \mathbf{y}, \mathbf{A}) = \frac{p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w}|\mathbf{A})}{p(\mathbf{y}|\mathbf{X}, \mathbf{A})}. \tag{5}$$

Calculate the hyperparameter $\mathbf{A}$ posterior distribution:

$$p(\mathbf{A}|\mathbf{X}, \mathbf{y}) \propto p(\mathbf{y}|\mathbf{X}, \mathbf{A})p(\mathbf{A}),$$

where $\propto$ means proportionality.

Suppose $p(\mathbf{A})$ is uniform on a large interval. The hyperparameter optimization problem is:

$$Q(\boldsymbol{\theta}, \mathbf{A}) = p(\mathbf{y}|\mathbf{X}, \mathbf{A}) = \int_{\mathbf{w} \in \mathbb{R}^u} p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w}|\mathbf{A}) \to \max_{\text{diag}(\mathbf{A}) = [\alpha_1, \ldots, \alpha_u] \in \mathbb{R}^n}. \tag{6}$$

Formulate the hyperparameter optimization problem. Given functions L and Q that characterize the model selection method. The loss function L optimizes parameters $\boldsymbol{\theta}$. The validation function Q evaluates the model predictive performance. The problem is to optimize the parameters $\boldsymbol{\theta}$ and the hyperparameters $\mathbf{A}$:

$$\hat{\mathbf{A}} = \arg\max_{[\alpha_1, \ldots, \alpha_u] \in \mathbb{R}^n} Q(\hat{\boldsymbol{\theta}}(\mathbf{A}), \mathbf{A}), \tag{7}$$

$$\hat{\boldsymbol{\theta}}(\mathbf{A}) = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^s} L(\boldsymbol{\theta}, \mathbf{A}). \tag{8}$$

Introduce the functions L, Q and $\boldsymbol{\theta}$ for various model selection methods.

Basic method. Optimize the model parameters $\boldsymbol{\theta}$ and hyperparameters $\mathbf{A}$ using the whole dataset $\mathfrak{D}$ and the same function for both optimization and model evaluation with one model instance, $L = -Q$:

$$Q(\boldsymbol{\theta}, \mathbf{A}) = -L(\boldsymbol{\theta}, \mathbf{A}) = \log p(\mathbf{y}, \mathbf{w}|\mathbf{X}, \mathbf{A}) = \log p(\mathbf{y}|\mathbf{X}, \mathbf{w}) + \log p(\mathbf{w}|\mathbf{A}). \tag{9}$$

Describe two model selection methods that prevent model overfitting.

Cross-validation. Split the dataset $\mathfrak{D}$ into k equal parts: $\mathfrak{D} = \mathfrak{D}_1 \sqcup \dots \sqcup \mathfrak{D}_k$.

Run k model optimizations for each subset. The vector $\boldsymbol{\theta}$ is a concatenation of model parameters $\mathbf{w}_k$ for each optimization instance:

$$\boldsymbol{\theta} = [\mathbf{w}_1, \dots, \mathbf{w}_k].$$

Construct the loss function L as an average negative log posterior probability over the remaining part of each split $\mathfrak{D}_1, \dots, \mathfrak{D}_k$:

$$L(\boldsymbol{\theta}, \mathbf{A}) = -\frac{1}{k}\sum_{c=1}^{k}\Big(\frac{k}{k-1}\log p(\mathbf{y} \setminus \mathbf{y}_c | \mathbf{X} \setminus \mathbf{X}_c, \mathbf{w}_c) + \log p(\mathbf{w}_c | \mathbf{A})\Big). \tag{10}$$

Construct the validation function Q as an average log likelihood over k splits:

$$Q(\boldsymbol{\theta}, \mathbf{A}) = \frac{1}{k}\sum_{c=1}^{k} k \log p(\mathbf{y}_c | \mathbf{X}_c, \mathbf{w}_c).$$

Evidence Lower Bound. As in the basic method description let $L = -Q$. It is an evidence lower bound:

$$\log p(\mathbf{y}|\mathbf{X}, \mathbf{A}) \geq -\mathrm{D}_{\mathrm{KL}}\big(q(\mathbf{w})||p(\mathbf{w}|\mathbf{A})\big) + \int_{\mathbf{w}} q(\mathbf{w}) \log p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \mathbf{A}) d\mathbf{w} \approx \tag{11}$$

$$\approx \sum_{i=1}^{m} \log p(y_i|\mathbf{x}_i, \mathbf{w}_i) - D_{\mathrm{KL}}\big(q(\mathbf{w})||p(\mathbf{w}|\mathbf{A})\big) = -L(\boldsymbol{\theta}, \mathbf{A}) = Q(\boldsymbol{\theta}, \mathbf{A}),$$

where q is an auxiliary distribution called variational distribution. Let q be a normal distribution:

$$q \sim \mathcal{N}(\boldsymbol{\mu}_q, \mathbf{A}_q^{-1}), \tag{12}$$

where $\mathbf{A}_q = \mathrm{diag}[\alpha_1^q, \dots, \alpha_u^q]^{-1}$ is a diagonal covariance matrix and $\boldsymbol{\mu}_q$ is a mean vector. The divergence D_{KL} between 2 Gaussian variables is

$$D_{\mathrm{KL}}\big(q(\mathbf{w})||p(\mathbf{w}|\mathbf{f})\big) = \frac{1}{2}\big(\mathrm{Tr}[\mathbf{A}\mathbf{A}_q^{-1}] + (\boldsymbol{\mu} - \boldsymbol{\mu}_q)^{\mathsf{T}}\mathbf{A}(\boldsymbol{\mu} - \boldsymbol{\mu}_q) - u + \ln|\mathbf{A}^{-1}| - \ln|\mathbf{A}_q^{-1}|\big).$$

The vector of optimized parameters $\boldsymbol{\theta}$ is a vector of variational distribution q parameters:

$$\boldsymbol{\theta} = [\mathrm{diag}(\mathbf{A}_q), \mu_1, \dots, \mu_u].$$

3 Gradient-based hyperparameter optimization methods

In this section we analyze hyperparameter optimization methods based on gradient descent. Suppose the parameters $\boldsymbol{\theta}$ are also optimized using gradient-based methods.

Definition 1 A stochastic gradient descent operator T estimates the parameters $\boldsymbol{\theta}'$ using their previous values $\boldsymbol{\theta}$ with random subset of dataset:

$$\boldsymbol{\theta}' = T(\boldsymbol{\theta}, \mathbf{A}, \mathfrak{D}) = \boldsymbol{\theta} - \gamma \nabla L(\boldsymbol{\theta}, \mathbf{A})_{\mathfrak{D}=\hat{\mathfrak{D}}},$$

where $\hat{\mathfrak{D}}$ is a random subset of $\mathfrak{D}$.

Optimize the parameters $\boldsymbol{\theta}$ with η steps of stochastic gradient descent:

$$\hat{\boldsymbol{\theta}} = T \circ T \circ \cdots \circ T(\boldsymbol{\theta}_0, \mathbf{A}) = T^{\eta}(\boldsymbol{\theta}_0, \mathbf{A}), \quad T(\boldsymbol{\theta}, \mathbf{A}) = \boldsymbol{\theta} - \gamma \nabla L(\boldsymbol{\theta}, \mathbf{A}), \quad (13)$$

where γ is the learning rate, $\boldsymbol{\theta}_0$ is the initial values of the vector $\boldsymbol{\theta}$.

Redefine the optimization problem according to the definition of operator T:

$$\hat{\mathbf{A}} = \underset{\operatorname{diag}(\mathbf{A}) \in \mathbb{R}^n}{\arg\max} \, Q\big(T^{\eta}(\boldsymbol{\theta}_0, \mathbf{A})\big). \quad (14)$$

Solve the optimization problem (14) using gradient-based methods. The exact gradient calculation $\nabla_{\mathbf{A}} Q(T^{\eta}(\boldsymbol{\theta}_0, \mathbf{A}))$ is intractable because of the inner optimization operator T. The scheme of the hyperparameter optimization.

1. In range from 1 to l, where l is the number of the hyperparameter optimization iterations:
2. Initialize parameters $\boldsymbol{\theta}$.
3. Solve optimization problem (14) and obtain the new hyperparameter values $\mathbf{A}'$
4. Set $\mathbf{A} = \mathbf{A}'$.

Introduce methods of numerical solution for this optimization problem. Their main properties are listed in Table 2.

Greedy algorithm. Update the hyperparameter $\mathbf{A}$ using gradient descent, which depends only on the last update of parameters $\boldsymbol{\theta}$. Optimize the hyperparameters and parameters in a single optimization procedure. Update the hyperparameter $\mathbf{A}$ at every iteration:

$$\mathbf{A}' = \mathbf{A} + \gamma_{\mathbf{A}} \nabla_{\mathbf{A}} Q\big(T(\boldsymbol{\theta}, \mathbf{A}), \mathbf{A}\big) = \mathbf{A} + \gamma_{\mathbf{A}} \nabla_{\mathbf{A}} Q\big(\boldsymbol{\theta} - \gamma \nabla L(\boldsymbol{\theta}, \mathbf{A}), \mathbf{A}\big)\big),$$

where $\gamma_{\mathbf{A}}$ is the learning rate for the hyperparameter optimization.

Algorithm	**Type**	**Optimization iteration complexity**	**Correctness suppositions**
Random search	stochastic	$O(\eta u\|\hat{\mathfrak{D}}\|)$	-
Greedy [20]	gradient-based	$O(\eta u\|\hat{\mathfrak{D}}\|)$	$\mathbf{H}(\boldsymbol{\theta}) = \mathbf{I}$
HOAG [24]	gradient-based	$O(\eta u\|\hat{\mathfrak{D}}\|+o)$, where o is a complexity of linear equation solution	first derivatives of Q and second derivatives of L are Lipshitz functions; $\det\mathbf{H} \neq \mathbf{0}$;
DrMAD [9]	gradient-based	$O(\eta u\|\hat{\mathfrak{D}}\|)$	Parameter trajectory$\boldsymbol{\theta} = \boldsymbol{\theta}_0, \dots \boldsymbol{\theta}_\eta$ is linear

Table 2: The analyzed algorithms complexity and correctness. The model is a multilayer perceptron optimized using backpropagation with the basic model selection method (9).

HOAG. Obtain approximate hyperparameter gradient values $\nabla_{\mathbf{A}} Q\big(T^\eta(\boldsymbol{\theta}_0, \mathbf{A})\big)$ using the following formula approximation of the of the gradient:

$$\nabla_{\mathbf{A}} Q\big(T^\eta(\boldsymbol{\theta}_0, \mathbf{A})\big) = \nabla_{\mathbf{A}} Q(\boldsymbol{\theta}, \mathbf{A}) - (\nabla_{\mathbf{A}} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}, \mathbf{A}))^{\mathsf{T}} \mathbf{H}(\boldsymbol{\theta})^{-1} \nabla_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \mathbf{A}),$$

where $\mathbf{H}$ is the Hessian of the function L with respect to the parameters $\boldsymbol{\theta}$.

The algorithm of hyperparameter gradient $\nabla_{\mathbf{A}} Q$ approximation:

1. Optimize the parameters $\boldsymbol{\theta} = T^\eta(\boldsymbol{\theta}_0, \mathbf{A})$.
2. Solve linear system for a vector $\boldsymbol{\lambda}$: $\mathbf{H}(\boldsymbol{\theta})\boldsymbol{\lambda} = \nabla_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \mathbf{A})$.
3. Compute the approximate hyperparameter gradient: $\hat{\nabla}_{\mathbf{A}} Q = \nabla_{\mathbf{A}} Q(\boldsymbol{\theta}, \mathbf{A}) - \nabla_{\mathbf{A}} \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}, \mathbf{A})^{\mathsf{T}} \boldsymbol{\lambda}$.

The final update rule is

$$\mathbf{A}' = \mathbf{A} + \gamma_{\mathbf{A}} \hat{\nabla}_{\mathbf{A}} Q. \tag{15}$$

The computational cost of the Hessian $\mathbf{H}(\boldsymbol{\theta})$ evaluation from step 2 is high. In this paper we use stochastic gradient descent for solving this linear system.

DrMad. For the hyperparameter gradient evaluation restore the full history of η parameters updates starting from the initial value $\boldsymbol{\theta}_0$. For the simplification of this procedure suppose that the trajectory of parameters $\boldsymbol{\theta}$ update is linear:

$$\boldsymbol{\theta}^\tau = \boldsymbol{\theta}_0 + \frac{\tau}{\eta} T(\boldsymbol{\theta}). \tag{16}$$

The algorithm of approximate hyperparameter gradient calculation:

1. Optimize parameters $\boldsymbol{\theta} = T^\eta(\boldsymbol{\theta}_0, \mathbf{A})$.
2. Let $\hat{\nabla}\mathbf{A} = \nabla_{\mathbf{A}} Q(\boldsymbol{\theta}, \mathbf{A})$.
3. Let $\hat{\nabla}\boldsymbol{\theta} = \nabla_{\boldsymbol{\theta}} Q(\boldsymbol{\theta}, \mathbf{A})$.
4. In range from $\tau = \eta$ to 1:

5. Compute $\boldsymbol{\theta}^{\tau}$(16).
6. $d\mathbf{v} = \gamma\hat{\nabla}\boldsymbol{\theta}$.
7. $\hat{\nabla}\mathbf{A} = \hat{\nabla}\mathbf{A} + d\mathbf{v}\nabla_{\mathbf{A}}\nabla_{\boldsymbol{\theta}}L$.
8. $\hat{\nabla}\boldsymbol{\theta} = \hat{\nabla}\boldsymbol{\theta} - d\mathbf{v}\nabla_{\boldsymbol{\theta}}\nabla_{\boldsymbol{\theta}}L$.

The final update rule is analogous to (15). The algorithm is instable when the learning rate γ is large [9]. For the stability of the algorithm we discard first 5% of values $\boldsymbol{\theta}$ and calculate only each τ_{step} step of parameter updates history:

$$\boldsymbol{\theta}^{\tau} = \boldsymbol{\theta}_{\tau_0} + \frac{\tau}{\eta}T(\boldsymbol{\theta}), \quad \tau \in \{\tau_0, \tau_o + \tau_{\text{step}}, \ldots, \eta - \tau_{\text{step}}, \eta\}, \tag{17}$$

where $\tau_0 = \lceil 0.05 \cdot \eta \rceil$.

4 Experiments

For the evaluation of the analyzed algorithms we conducted a series of computational experiments. A synthetic dataset, WISDM [18] and MNIST [19] datasets were used. All datasets were splitted into Train and Test subsets. The algorithms were evaluated on the Test subsets. For each dataset and each algorithm we ran 5 optimizations. The results were averaged. We used the following evaluation criteria.

1. Quality: the best value of Q: $\hat{Q} = \max_{j\in\{1,\ldots,l\}} Q^j$.
2. Convergence: the number of iterations to have a validation value greater than 99% of the best value $\hat{Q}$:

$$\arg\min_j : \frac{Q^j - Q^0}{\hat{Q} - Q^0} \geq 0.99,$$

   where Q^0 is the value of Q before the hyperparameter optimization.
3. Error function E:

$$E = \text{RMSE}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) = \left(\frac{1}{|\mathbf{X}_{\text{test}}|}\sum_{\mathbf{x}\in\mathbf{X}_{\text{test}}, y\in\mathbf{y}_{\text{test}}} (f(\mathbf{x}, \mathbf{w}) - y)\right)^{\frac{1}{2}}$$

   for the regression problem and

$$E = \text{Acc}(\mathbf{X}_{\text{test}}, \mathbf{y}_{\text{test}}) = 1 - \frac{1}{|\mathbf{X}_{\text{test}}|}\sum_{\mathbf{x}\in\mathbf{X}_{\text{test}}, y\in\mathbf{y}_{\text{test}}} [f(\mathbf{x}, \mathbf{w}) \neq y]$$

   for the classification problem.
4. The error function E_σ for the model with noisy dataset:

$$\text{RMSE}_\sigma = \text{RMSE}(\mathbf{X}_{\text{test}} + \boldsymbol{\varepsilon}, \mathbf{y}_{\text{test}}),$$

$$\text{Acc}_\sigma = \text{Acc}(\mathbf{X}_{\text{test}} + \boldsymbol{\varepsilon}, \mathbf{y}_{\text{test}}), \quad \boldsymbol{\varepsilon} \sim \mathcal{N}\big(\mathbf{0}, \sigma\mathbf{I}\big).$$

The random search was used as a baseline. The number of the random search iterations equaled to the number l of gradient-based hyperparameter optimization algorithms iterations. It was set to $l = 50$ for the synthetic dataset and WISDM dataset, $l = 25$ for the MNIST dataset. For the loss function L and validation function Q cross-validation (10) with $k = 4$ and evidence lower bound (11) were used.

For the neural network models we set all the hyperparameters equal for each layer when used cross-validation (10). The full diagonal parameterization of the hyperparameters was used for all the linear models and for the neurla network models when we used evidence lower bound (11).

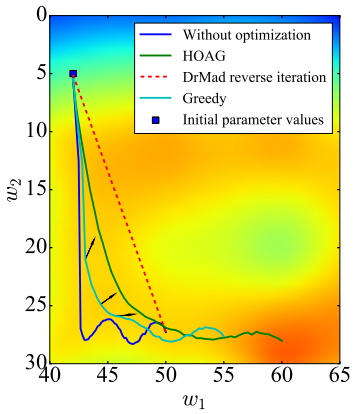


Fig. 1: An example of parameter update trajectories. The color displays the value of the validation function Q. Greedy algorithm optimizes hyperparameter during the parameter optimization, therefore it has the light blue trajectory between the optimized green trajectory of HOAG algorithm and the dark blue trajectory of parameters without hyperparameter optimization. DrMAD uses a linearized dashed parameter trajectory during hyperparameter optimization procedure.

The hyperparameters were initialized with uniform distribution $\mathcal{U}(a, b)$, where $a = -2, b = 10$ for the synthetic dataset and $a = -4, b = 10$ for the WISDM and MNIST datasets.

We calibrated the hyperparameter learning rate $\gamma_{\mathbf{A}}$ for each algoritm using grid search: $\{r \cdot 10^s, s \leq 1, r \in \{1, 25, 50, 75\}\}$. The largest value of $\gamma_{\mathbf{A}}$ was chosen if the final hyperparameter value $\mathbf{A}$ satisfied the condition:

$$a_{\min} \leq \min(\mathbf{A}), \quad \max(\mathbf{A}) \leq b_{\max},$$

where $a_{\min} = -2.5, b_{\max} = 10.5$ for the synthetic dataset and $a_{\min} = -5, b_{\max} = 11$ for the WISDM and MNIST datasets. We calibrated $\gamma_{\mathbf{A}}$ with

a small amount of iterations: $l = 50$ for the syntetic dataset, $l = 10$ for the WISDM and $l = 5$ for the MNIST dataset. Whenever each algorithm showed instability in further experiments (gradient explosion or overflow) the value of $\gamma_{\mathbf{A}}$ was lowered. The parameter τ_{step} set to 1 for syntetic dataset and WISDM and $\tau_{\text{step}} = 10$ for the MNIST dataset.

Synthetic dataset. The synthetic dataset was generated according to the rule:

$$\mathbf{y} = \mathbf{x} + \boldsymbol{\varepsilon}, \quad \mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad \boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),$$

where $m = 40, \quad n = 1$.

The regression model $\mathbf{f}$ uses the following features: $\{\mathbf{x}^0, \ldots, \mathbf{x}^9, \mathbf{sin}(\mathbf{x}), \mathbf{cos}(\mathbf{x})\}$. The unregularized regression model with this such ratio of object number and feature number tends to overfit significantly. The goal of this experiment was to analyze if the hyperparameter optimization could protect such models from overfit.

Fig. 2 plots the resulting polynoms. The evidence lower bound criterion (11) gave non-overfitted polynoms, which parameters are close to the linear models.

WISDM. The WISDM dataset contains a set of records from accelerometer. Each record has three coordinates that correspond to the accelerometer axes. As a set of features we used first 199 records from each 200 sequential records. We used l_2 norm of the 200th records as a label and required to predict.

A neural network with 10 neurons on the hidden layer was used:

$$\mathbf{f} = \mathbf{W}_2 \cdot \mathbf{RELU}(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2,$$

where $\mathbf{W}_1, \mathbf{b}_1$ — hidden layer parameters, $\mathbf{W}_2, \mathbf{b}_2$ — output layer parameters,

$$\mathbf{RELU}(\mathbf{x}) = \max(\mathbf{0}, \mathbf{x}).$$

Fig. 3 shows the plots with RMSE and best validation loss value $\hat{Q}$ for the WISDM dataset. DrMad and HOAG algorithms showed significantly lower results than greedy algorithm. The random search showed good results in case of cross-validation, when the number of hyperparameters is small. When the number of hyperparameters is large the greedy and HOAG algorithm showed good results. HOAG requires more iteration number l than the greedy algorithm for the convergence.

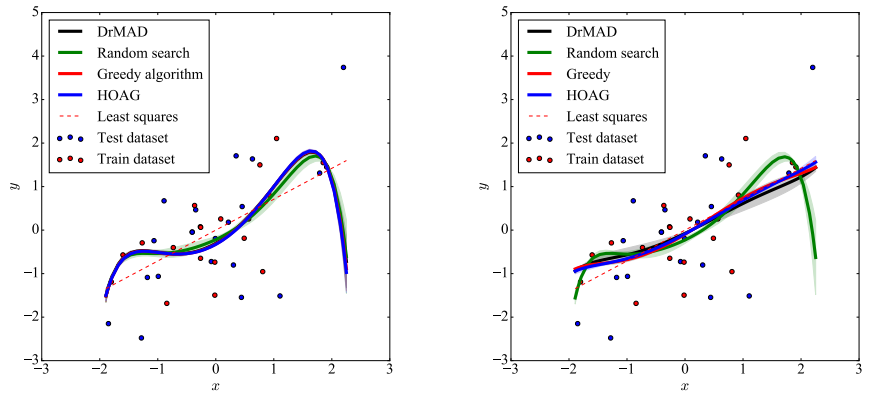
MNIST. The MNIST dataset contains a set of images of handwritten digits. We used a neural network with 300 neurons on the hidden layer and softmax output.

Fig. 4 plots the error E and quality $\hat{Q}$. For the evidence lower bound criterion, the models with highest $\hat{Q}$ value have also highest error E. Nevertheless, these models showed the best results for the stability, when the noise in dataset is high. Fig. 5 plots the error E_σ. Models with highest evidence lower bound have the lowest error when the Test dataset is noisy. We interpret this as an ability to select the model with highest generalization ability.

Algorithm	L, Q	$Q(\boldsymbol{\theta}, \mathbf{A})$	Convergence	E	$E_{0.25}$	$E_{0.5}$
			Synthetic			
Random search	(10)	**-171.6**	**26.2 ± 20.0**	**1.367**	**1.410**	**1.555**
Greedy	(10)	-172.5	30.0 ± 24.5	1.421	1.439	1.536
DrMAD	(10)	-174.1	40.2 ± 16.1	1.403	1.424	**1.512**
HOAG	(10)	-174.7	29.4 ± 24.0	**1.432**	1.463	1.553
Random Search	(11)	-63.5	32.4 ± 18.7	1.368	1.426	1.546
Greedy	(11)	-25.5	**1.2 ± 0.4**	1.161	1.174	1.193
DrMAD	(11)	**-25.1**	10.6 ± 0.8	1.157	1.163	1.184
HOAG	(11)	-25.8	10.8 ± 1.5	**1.141**	**1.149**	**1.177**
			WISDM			
Random search	(10)	**-1086661.1**	22.0 ± 19.3	**0.660**	**0.670**	**0.690**
Greedy	(10)	-1086707.1	**15.4 ± 17.2**	0.707	0.723	0.769
DrMAD	(10)	-1086708.2	29.2 ± 8.0	0.694	0.708	0.742
HOAG	(10)	-1086733.5	28.2 ± 7.13	0.701	0.724	0.753
Random search	(11)	-35420.4	14.4 ± 7.8	0.732	0.755	0.785
Greedy	(11)	**-3552.9**	**1.0 ± 0.0**	**0.702**	**0.730**	**0.767**
DrMAD	(11)	-26091.4	50.0 ± 0.0	0.729	0.753	0.816
HOAG	(11)	-16566.6	49.0 ± 0.0	0.733	0.755	0.801
			MNIST			
Random search	(10)	-3236.4	7.8 ± 1.9	0.981	**0.966**	**0.866**
Greedy	(10)	**-3416.7**	10.8 ± 10.4	0.979	0.962	0.860
DrMAD	(10)	-3469.0	17.0 ± 5.6	**0.982**	0.962	0.831
HOAG	(10)	-3748.6	**8.6 ± 7.3**	0.980	0.961	0.853
Random search	(11)	-1304556.4	14.2 ± 5.7	**0.982**	0.943	0.814
Greedy	(11)	**-11136.2**	**1.0 ± 0.0**	0.977	**0.952**	**0.884**
DrMAD	(11)	-1305432.9	24.6 ± 0.5	**0.982**	0.941	0.813
HOAG	(11)	-280061.6	24.0 ± 0.0	0.981	0.943	0.819

Table 3: Experiment results

As we can see from the experiments, the gradient-based methods give better results than the random search when the number of hyperparameters is large. The best results for all the experiments were obtained by greedy algorithm. DrMad algorithm showed weaker results than greedy and HOAG algorithms because of its instability: the algorithm often had gradient explosions, therefore the learning rate $\gamma_{\mathbf{A}}$ for it was set to low values during calibration.



(a) (b)

Fig. 2: Resulting models for the synthetic dataset: a — cross-validation, b — evidence lower bound

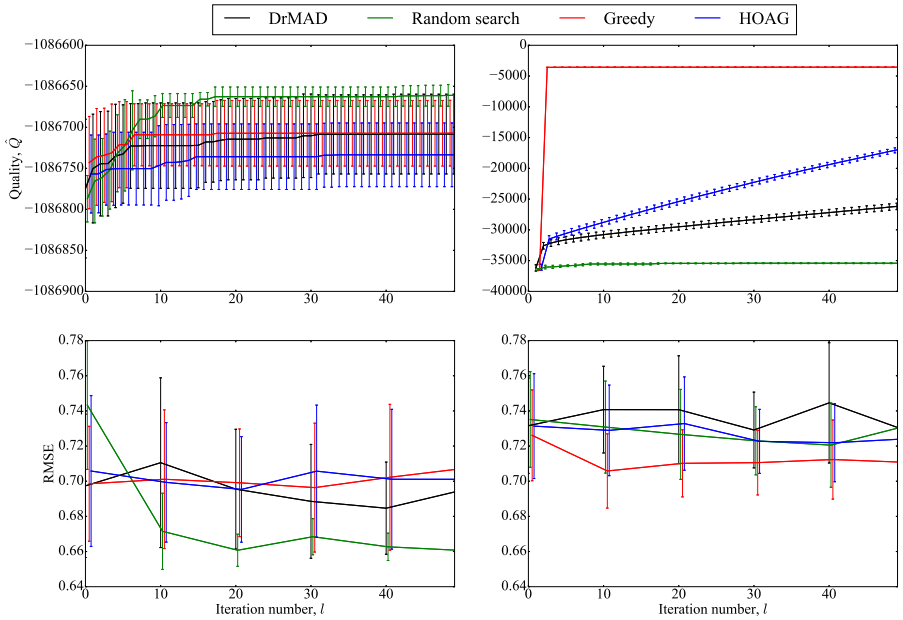


Fig. 3: WISDM, best validation value $\hat{Q}$ and RMSE for cross-validation (left) and evidence lower bound (right)

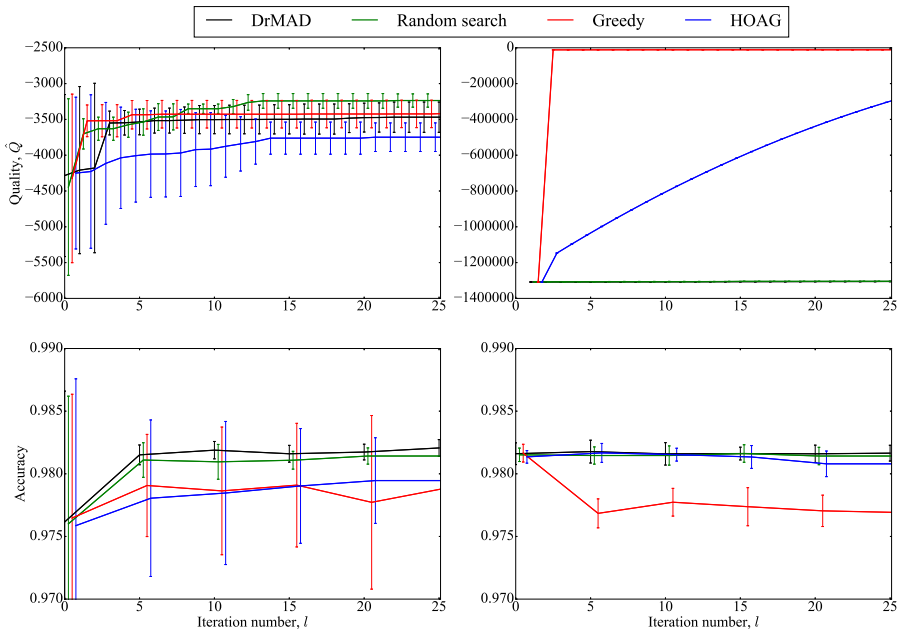


Fig. 4: MNIST, best validation value $\hat{Q}$ and Accuracy for cross-validation (left) and evidence lower bound (right)

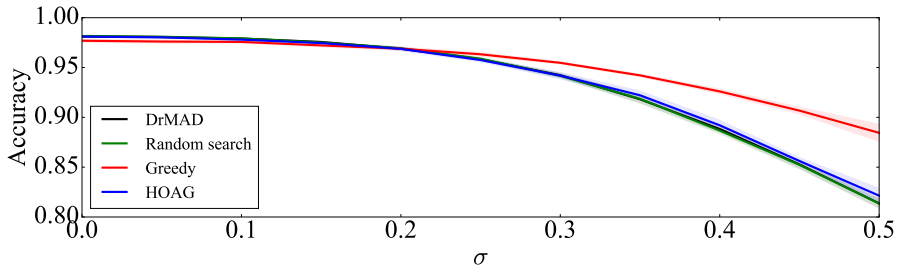


Fig. 5: MNIST, Model accuracy with noise in the Test dataset. The hyperparameters were optimized with evidence lower bound criterion.

5 Discussion

The experiments showed that each algorithm perform effectively and therefore the appropriate hyperparameter optimization method should rely on the amount of hyperparameters and the specific of the problem.

When dealing with small amount of hyperparameters random search showed the best results since search procedure can be employed more effectively than gradient-based optimization in low-dimensional hyperparameters space. For the high-dimensional hyperparameter space both HOAG and greedy

algorithms showed good performance. HOAG algorithm would be more preferable if the model optimization problem is expensive. On the other hand we can schedule greedy hyperparameter optimization to make it less expensive as in [20].

DrMad algorithm showed rather poor results on the MNIST and WISDM datasets. Perhaps, it is so because of high learning rate γ used in experiments. The large value of the learning rate can make DrMad algorithm instable. Two improvements can be proposed. We can use more stable optimization like Adam or AdaGrad for both parameter and hyperparameter optimization. The second improvement was proposed in [9]: we can use more complicated parameter trajectory approximation to make it more similar to the original parameter trajectory. Opposing to HOAG and greedy algorithms, DrMad optimization has prerequisites for not only hyperparameters optimization but also the metaparameters, i.e. the optimization procedure parameters. The opportunity of such optimization using reversed differentiation was shown in [9].

The other interesting aspect of our experiments is the relation between the model error (RMSE or Accuracy) and the value of validation loss Q. The models obtained by the evidence lower bound showed higher error rate than the models obtained using cross-validation on the MNIST and WISDM datasets. These models also showed greater stability when noise was added to the Test datasets. The evidence lower bound showed significantly better results on the synthetic dataset, when the amount of objects in the Train dataset is small. Therefore we conclude that the evidence lower bound usage is preferable when the model tends to overfit or when the cross-validation usage is too computationally expensive. In [10] it is noted that the evidence lower bound optimization required more iterations for the convergence. In our experiments we used the same number of iterations both for the cross-validation and evidence lower bound. The more accurate iteration number calibration can improve the final quality of these models.

6 Conclusion and future work

The paper analyzed the gradient-based hyperparameter optimization algorithms. We adapted the analyzed algorithms for general validation functions and evaluated their performance on the MNIST and WISDM datasets. Two model selection criteria were compared: the cross-validation and evidence lower bound.

The experiments showed that the gradient-based algorithms are effective when the number of hyperparameters is large. The results showed that models obtained using evidence lower bound have higher error rater than models obtained using cross-validation, but they are also more stable when the test dataset contains a lot of noise.

The authors implemented these algorithms as a toolbox available at [1]. The toolbox is developed in Python using Theano [31] and Numpy [23] libraries.

In out future work we are planning to develop the analyzed algorithm and to extend gradient-based algorithms to optimize not only hyperparameters, but also the model optimization parameters. The other object of our future research will is the difference between the cross-validation and evidence lower bound and the theoretical aspects of their properties for the models with large amount of parameters.

References

1. URL https://svn.code.sf.net/p/mlalgorithms/code/Group074/Bakhteev2017Hypergrad/code/
2. Arlot, S., Celisse, A.: A survey of cross-validation procedures for model selection. Statist. Surv. **4**, 40–79 (2010). DOI 10.1214/09-SS054. URL http://dx.doi.org/10.1214/09-SS054
3. Bengio, Y., Grandvalet, Y.: No unbiased estimator of the variance of k-fold cross-validation. J. Mach. Learn. Res. **5**, 1089–1105 (2004)
4. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research **13**(Feb), 281–305 (2012)
5. Bergstra, J.S., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Advances in Neural Information Processing Systems, pp. 2546–2554 (2011)
6. Bishop, C.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006)
7. Domke, J.: Generic methods for optimization-based modeling. In: N.D. Lawrence, M.A. Girolami (eds.) AISTATS, *JMLR Proceedings*, vol. 22, pp. 318–326. JMLR.org (2012). URL http://dblp.uni-trier.de/db/journals/jmlr/jmlrp22.html#Domke12
8. Farcomeni, A.: Bayesian constrained variable selection. Statistica Sinica pp. 1043–1062 (2010)
9. Fu, J., Luo, H., Feng, J., Low, K.H., Chua, T.S.: Drmad: Distilling reverse-mode automatic differentiation for optimizing hyperparameters of deep neural networks. arXiv preprint arXiv:1601.00917 (2016)
10. Graves, A.: Practical variational inference for neural networks. In: J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F. Pereira, K.Q. Weinberger (eds.) Advances in Neural Information Processing Systems 24, pp. 2348–2356. Curran Associates, Inc. (2011)
11. Grnwald, P.: A tutorial introduction to the minimum description length principle. In: Advances in Minimum Description Length: Theory and Applications. MIT Press (2005)
12. Hoffman, M.D., Blei, D.M., Wang, C., Paisley, J.: Stochastic variational inference. J. Mach. Learn. Res. **14**(1), 1303–1347 (2013). URL http://dl.acm.org/citation.cfm?id=2502581.2502622
13. Hornung, R., Bernau, C., Truntzer, C., Stadler, T., Boulesteix, A.L.: Full versus incomplete cross-validation: measuring the impact of imperfect separation between training and test sets in prediction error estimation (2014)
14. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: International Conference on Learning and Intelligent Optimization, pp. 507–523. Springer (2011)
15. Karaletsos, T., Rätsch, G.: Automatic relevance determination for deep generative models. arXiv preprint arXiv:1505.07765 (2015)
16. Krstajic, D., Buturovic, L.J., Leahy, D.E., Thomas, S.: Cross-validation pitfalls when selecting and assessing regression and classification models. Journal of Cheminformatics **6**(1), 1–15 (2014). DOI 10.1186/1758-2946-6-10. URL http://dx.doi.org/10.1186/1758-2946-6-10
17. Kuznetsov, M.P., Tokmakova, A.A., Strijov, V.V.: Analytic and stochastic methods of structure parameter estimation. Informatica **27(3)**, 607–624 (2016). DOI http://www.mii.lt/informatica/pdf/INFO1109.pdf

18. Kwapisz, J.R., Weiss, G.M., Moore, S.A.: Activity recognition using cell phone accelerometers. ACM SigKDD Explorations Newsletter **12**(2), 74–82 (2011)
19. LeCun, Y.: The mnist database of handwritten digits. http://yann. lecun. com/exdb/mnist/ (1998)
20. Luketina, J., Berglund, M., Greff, K., Raiko, T.: Scalable gradient-based tuning of continuous regularization hyperparameters. arXiv preprint arXiv:1511.06727 (2015)
21. MacKay, D.J.C.: Information Theory, Inference & Learning Algorithms. Cambridge University Press, New York, NY, USA (2002)
22. Maclaurin, D., Duvenaud, D., Adams, R.: Gradient-based hyperparameter optimization through reversible learning. In: D. Blei, F. Bach (eds.) Proceedings of the 32nd International Conference on Machine Learning (ICML-15), pp. 2113–2122. JMLR Workshop and Conference Proceedings (2015)
23. Oliphant, T.E.: A guide to NumPy, vol. 1. Trelgol Publishing USA (2006)
24. Pedregosa, F.: Hyperparameter optimization with approximate gradient. In: Proceedings of the 33rd International Conference on Machine Learning (2016)
25. Salakhutdinov, R., Hinton, G.: Learning a nonlinear embedding by preserving class neighbourhood structure. In: M. Meila, X. Shen (eds.) Journal of Machine Learning Research - Proceedings Track, vol. 2, pp. 412–419 (2007)
26. Salimans, T., Kingma, D.P., Welling, M.: Markov chain monte carlo and variational inference: Bridging the gap. In: F.R. Bach, D.M. Blei (eds.) ICML, *JMLR Proceedings*, vol. 37, pp. 1218–1226. JMLR.org (2015)
27. Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M.M.A., Prabhat, M., Adams, R.P.: Scalable bayesian optimization using deep neural networks. In: ICML, pp. 2171–2180 (2015)
28. Strijov, V.V.: Model genetation and selection for regression and classification problems (dsc thesis). Ph.D. thesis, Russian Academy of Sciences, Computing Center (2014)
29. Sutskever, I., Vinyals, O., Le, Q.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada, pp. 3104–3112 (2014)
30. Suzuki, J.: An efficient bayesian network structure learning strategy. New Generation Computing **35**(1), 105–124 (2017). DOI 10.1007/s00354-016-0007-6. URL https://doi.org/10.1007/s00354-016-0007-6
31. Theano Development Team: Theano: A Python framework for fast computation of mathematical expressions. arXiv e-prints **abs/1605.02688** (2016). URL http://arxiv.org/abs/1605.02688
32. Vishnu, A., Narasimhan, J., Holder, L., Kerbyson, D.J., Hoisie, A.: Fast and accurate support vector machines on large scale systems. In: 2015 IEEE International Conference on Cluster Computing, CLUSTER 2015, Chicago, IL, USA, September 8-11, 2015, pp. 110–119 (2015). DOI 10.1109/CLUSTER.2015.26. URL http://dx.doi.org/10.1109/CLUSTER.2015.26
33. Zou, Y., Roos, T.: On model selection, bayesian networks, and the fisher information integral. New Generation Computing **35**(1), 5–27 (2017)