Online function minimization with convex random relu expansions

Bliek, Laurens; Verhaegen, Michel; Wahls, Sander

DOI
10.1109/MLSP.2017.8168109

Publication date
2017

Document Version
Final published version

Published in
Proceedings 2017 27th IEEE International Workshop on Machine Learning for Signal Processing (MLSP 2017)

ONLINE FUNCTION MINIMIZATION WITH CONVEX RANDOM RELU EXPANSIONS

Laurens Bliek, Michel Verhaegen, and Sander Wahls

Delft Center for Systems and Control, Delft University of Technology,
Mekelweg 2, 2628 CD, Delft, Netherlands

ABSTRACT

We propose CDONE, a convex version of the DONE algorithm. DONE is a derivative-free online optimization algorithm that uses surrogate modeling with noisy measurements to find a minimum of objective functions that are expensive to evaluate. Inspired by their success in deep learning, CDONE makes use of rectified linear units, together with a nonnegativity constraint to enforce convexity of the surrogate model. This leads to a sparse and cheap to evaluate surrogate model of the unknown optimization objective that is still accurate and that can be minimized with convex optimization algorithms. The CDONE algorithm is demonstrated on a toy example and on the problem of hyper-parameter optimization for a deep learning example on handwritten digit classification.

Index Terms— Deep learning, surrogate modeling, Bayesian optimization, derivative-free optimization

1. INTRODUCTION

Many practical optimization problems do not satisfy the assumptions that are present in traditional continuous optimization algorithms. Examples of these assumptions are that the derivative of the function to be optimized is known, or that there is at least a mathematical expression for the function, or that the function can be evaluated quickly and accurately. But the outcome of a simulation or algorithm, for example, can depend on many parameters and can suffer from noise. In many cases it is undesirable to try a new set of parameters and check for improvement multiple times, which is what happens in grid search or random search techniques.

Several paths have been taken in alleviating this problem. Most derivative-free optimization algorithms [1] are able to operate without the assumptions mentioned above. The ones that seem most fit to deal with noisy and expensive measurements are in the class of Bayesian optimization algorithms [2, 3, 4]. These algorithms use the available data and a prior to fit a probabilistic surrogate model and then use this model to decide where the next measurement should be taken. Hyper-parameter optimization is just one of the many examples where Bayesian optimization algorithms have shown their potential.

Another algorithm that is based on surrogate models is the DONE algorithm [5]. The surrogate model used in this algorithm is a random feature expansion (RFE) [6], which is updated every time a new measurement comes in. At each iteration of the algorithm a measurement of the objective is taken, then the surrogate model is updated, and then a new measurement location is proposed based on the minimum of the surrogate model. Using RFEs as a surrogate model makes it possible to get a fixed computational complexity per iteration by using recursive least squares updates. In comparison, Bayesian optimization algorithms become slower over time. The DONE algorithm was shown to outperform a popular Bayesian optimization algorithm on several tasks, such as the tuning of an optical beam-forming network [5].

This paper proposes an adaptation of the DONE algorithm called CDONE that has several advantages:

- There are less hyper-parameters to tune.
- The surrogate model is convex.
- The surrogate model is evaluated faster.
- The surrogate model is inherently sparse.

The DONE algorithm already had few hyper-parameters to tune, but having even less is a big advantage. The convexity allows convex optimization algorithms to be used to find the global minimum of the model, as opposed to finding a local minimum. The last two advantages make it possible to find the global minimum efficiently.

The convex model used in this paper is a combination of RFEs and rectified linear units (ReLUs) [7], which will be explained in the next section. Section 3 describes the CDONE algorithm. A comparison with the DONE algorithm is given in Section 4. The results of both algorithms on an artificial example and on a hyper-parameter tuning problem for deep learning are described in Section 5. Finally, Section 6 contains conclusions and recommendations for future work.

This research was supported in part by the European Research Council Advanced Grant Agreement under Grant 339681 and in part by the Dutch Technology Foundation STW under Project 13336.

2. RANDOM RELU EXPANSIONS

RFEs [6] have gained popularity recently due to their ability to approximate kernels with low dimensionality. They are defined as a weighted sum of basis functions with random parameters, and can be trained with conventional regularized linear least squares techniques. Since the number of basis functions and the values of the random parameters stay fixed, these models are particularly fit for problems with many data samples. Theoretical approximation guarantees for RFEs are available for the L_2 norm [8] and for the L_∞ norm [6]. Practically relevant results for the L_2 norm with models trained with regularized least squares are also available [5]. In short, any continuous function on a compact domain can be approximated arbitrarily well if the number of basis functions are large enough. The approximation error scales with the inverse square root of the number of basis functions. In practice, however, the approximation accuracy is sensitive to hyper-parameters such as the probability distribution of the random parameters. Recommendations for these hyper-parameters in the case of random cosine features are given in [5].

At the same time, ReLUs [7] have become a popular activation function in deep neural networks because of the inherent sparsity and the ability to circumvent the vanishing gradient problem. Even shallow ReLU networks can act as universal approximators [7]. In this work we use random features based on ReLUs, to make use of the advantages of both principles.

Define the ReLU $\phi : \mathbb{R} \to \mathbb{R}$ as

$$\phi(z) = \begin{cases} z, \ z > 0, \\ 0, \ z \leq 0. \end{cases} \tag{1}$$

Then, a Random ReLU expansion (RRE) is a model of the form

$$\mathrm{RRE}(\mathbf{x}) = c_D - c_{D-1} + \sum_{k=1}^{D-2} c_k \phi(\mathbf{w}_k^T \mathbf{x} + b_k), \tag{2}$$

with $\mathbf{w}_k \in \mathbb{R}^d$ and $b_k \in \mathbb{R}$ being realizations of i.i.d. random variables from continuous probability distributions. We assume $c_k \geq 0 \ \forall k$, so the first two terms are required for a bias that also allows the model to approximate negative values. With this assumption, the model is a convex function of $\mathbf{x}$.

In the area of neural networks, the parameters $c_k, \mathbf{w}_k$ and b_k are trained with stochastic gradient descent or similar algorithms. In the RRE model, however, $\mathbf{w}_k$ and b_k are chosen randomly, and finding the optimal parameters c_k is a convex optimization problem.

3. THE CDONE ALGORITHM

We present an adaptation to the DONE algorithm [5], with ReLU basis functions and a convexity constraint $c_k \geq 0 \ \forall k$, to find the minimum of an unknown function f using noisy measurements y_i. To initialize the CDONE algorithm, an initial guess $\mathbf{x}_1$ is needed, together with its corresponding measurement y_1. The random parameters $\mathbf{w}_k$ and b_k are drawn independently from their probability distributions and remain fixed for the whole duration of the algorithm. In this paper we have used the uniform distribution on $[-1, 1]$ for both $\mathbf{w}_k$ and b_k. The algorithm then repeats the following three steps:

3.1. Fitting the surrogate model

To fit the RRE to the data $(\mathbf{x}_i, y_i)$ for iterations $i = 1, \ldots, n$, while imposing a convexity constraint, the following regularized nonnegative linear least squares problem needs to be solved:

$$\min_{\mathbf{c}} \sum_{i=1}^{n} (y_i - \mathrm{RRE}(\mathbf{x}_i; \mathbf{c}))^2 + \lambda ||\mathbf{c}||_2^2, \tag{3}$$

$$\mathbf{s.t.} \ c_k \geq 0, \ k = 1, \ldots, D. \tag{4}$$

Here, λ is a regularization parameter, which can be chosen quite small in practice (e.g. $\lambda = 10^{-8}$) because the convexity constraint already helps in preventing overfitting. Being less sensitive to this parameter is a big advantage over the DONE algorithm. The above optimization problem is a nonnegative least squares problem. This problem is convex and can be solved with, for example, an active set method [9].

3.2. Finding the minimum of the surrogate model

After fitting the RRE model with optimal coefficients $\mathbf{c}^*$, we find the minimum of this model:

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \in X} \mathrm{RRE}(\mathbf{x}; \mathbf{c}^*). \tag{5}$$

Here, X is a convex compact set, e.g. $X = [-1, 1]^d$. In the original DONE algorithm, only a local minimum of the surrogate model is found. The initial guess provided to the solver is the current measurement $\mathbf{x}_i$, plus a small perturbation to aid in exploration. However, because the RRE in the CDONE algorithm is convex, we can find the global minimum in this case with a convex optimization algorithm. There is also no need to add an extra exploration step by perturbing the initial guess.

The original DONE algorithm uses second-order optimization methods like the L-BFGS method [9]. Because of the structure of the RRE however, we propose a steepest descent method with a backtracking line search and gradient projection [9] for the CDONE algorithm. It can be seen that the RRE is a piecewise linear function, so first-order approximations are exact in a certain set, and the line search should help in taking the largest possible step within this set.

3.3. Choose a new measurement point

The found global optimum of the RRE is used to determine a new measurement point. Although the RRE model is convex, the original objective f might be non-convex, so a small random perturbation ξ is added to $\mathbf{x}^*$ for exploration purposes. Only a local approximation of f around its (local) minimum is needed, and the RRE becomes more accurate around this point as new measurements are added. A new measurement is taken at $\mathbf{x}_{i+1} = \mathbf{x}^* + \xi$ (after projecting onto X), which leads to a new value y_{i+1}, and the algorithm repeats at step 1.

4. COMPARISON WITH THE DONE ALGORITHM

Several factors influence the computational costs of the CDONE algorithm, compared to the DONE algorithm. First of all, the basis functions used in the DONE algorithm are cosines, whereas the CDONE algorithm uses ReLU basis functions. Although this has no influence on the order of complexity, some computation time can be saved by implementing the ReLU basis function with a simple IF-statement as in (1). This is faster than calculating a cosine. Another time saver is the sparsity of the RRE model, which occurs in two ways: sparsity of the basis functions, and sparsity of the nonnegative weights c_k. The first case occurs if $\mathbf{w}_k^T\mathbf{x} + b_k \leq 0$, in which case a scalar multiplication and addition do not have to be computed. This inherent sparsity is one of the reasons ReLUs are used in deep learning. The second case occurs if $c_k = 0$ after fitting the model. In this case, $\phi(\mathbf{w}_k^T\mathbf{x} + b_k)$ does not have to be computed, saving a vector-vector multiplication of the same size as the input $\mathbf{x}$, a vector addition, and an IF-statement. The next section illustrates how often this happens in practice.

The convexity of the model used in the CDONE algorithm allows convex optimization algorithms to find the global minimum of the model. In the DONE algorithm, finding the global minimum is an intractable problem due to the non-convexity of the model. Furthermore, unlike the DONE algorithm, the CDONE algorithm does not include an exploration step by perturbing the initial guess of this convex optimization algorithm. For each iteration, this saves on computation time equal to the time required to draw a random vector of the same dimension as the input $\mathbf{x}$.

The only part of the CDONE algorithm that could increase its computation time when compared to the DONE algorithm, is the fitting of the surrogate model. Step 1 of the CDONE algorithm, fitting the surrogate model, is the most computationally expensive step of the algorithm. In the original DONE algorithm, a recursive least squares update was used to reduce the computation time of this step. In the CDONE algorithm, this should be changed to a recursive nonnegative least squares update. Several algorithms exist for this purpose [10, 11, 12, 13], all with varying numerical stability, accuracy, and computational complexity. The approach in [13], based on time-, order-, and active-set-recursion, seems the most fit for this problem. With this implementation, the nonnegative least squares problem (3)-(4) can be solved recursively in $O(D^2)$, just like in the DONE algorithm, provided that the active set recursion can be carried out in $O(1)$ steps. In this paper, we did not use a recursive algorithm to solve (3)-(4), but applied the active set method directly for ease of use. We do plan to investigate a recursive implementation in the future.

In Section 5 we note that the active constraints between two subsequent iterations of the CDONE algorithm differ only by 2 on average in a simple test case. This implies that the average order of complexity of the fitting step of the CDONE algorithm could indeed be reduced to $O(D^2)$ in practice.

5. NUMERICAL EXAMPLES

In this section we test the CDONE algorithm on two numerical examples: finding the minimum of a convex function perturbed by noise, and finding the optimal hyper-parameters of a deep learning classification problem.

5.1. Minimizing a noisy convex function

As a test case, consider the function

$$f(\mathbf{x}) = \sqrt{\mathbf{x}^T\mathbf{x}} - 5, \tag{6}$$

with $\mathbf{x} \in \mathbb{R}^2$. We have access to this function via noisy measurements $y(\mathbf{x}) = f(\mathbf{x}) + 0.01\eta$, where η has a standard normal distribution.

The minimum of f is found with four variations of the DONE algorithm: the standard DONE algorithm, the DONE algorithm with ReLU basis functions instead of cosines (DONE RELU), the CDONE algorithm as presented in this paper, and the CDONE algorithm with exponential linear units [14] (ELUs) as basis functions (CDONE ELU). The comparison with the smoother ELUs is made to determine the effect of the smoothness of the basis functions. All algorithms used $D = 500$ basis functions and $N = 500$ measurements, with a regularization parameter of $\lambda = 10^{-2}$ for DONE and DONE RELU, and $\lambda = 10^{-8}$ for CDONE and CDONE ELU. The convexity constraints of the latter two algorithms reduce the risk of overfitting to noise, so they need less regularization as a consequence. The variance of the exploration parameter was set to 10^{-4} for all algorithms. The DONE and DONE RELU algorithms used the standard normal distribution for their respective $\mathbf{w}_k$ parameters, which is the default approach that works well in practice [5]. The experiment was repeated 100 times starting from random initial guesses in $[-1, 1]^2$.

Table 1 shows the distance of the found minimum $\mathbf{x}^*$ to the true minimum, with the mean and standard deviation from 100 runs, as well as the average computation time in seconds. Please note that the computation time can be improved for all

	DONE	DONE RELU	CDONE	CDONE ELU
Mean	0.0132	0.0150	0.0155	0.0244
Std	0.0074	0.0073	0.0091	0.0148
Time	55.735	64.266	27.991	34.427

Table 1: Final distance to the true minimum of the convex function, averaged over 100 runs, and average computation time in seconds.

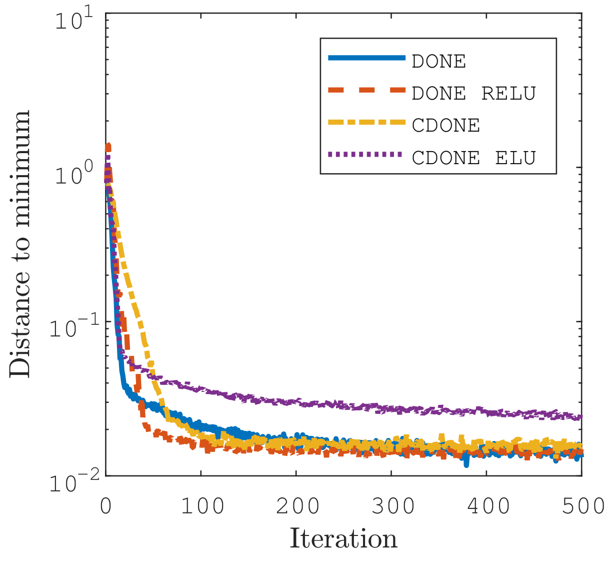


Fig. 1: Distance to the true minimum of the convex function, averaged over 100 runs.

four algorithms, as we used (adaptations of) a slower version of the DONE algorithm available online [15]. Furthermore, the CDONE and CDONE ELU implementations do not yet use a recursive algorithm for fitting the surrogate model, as mentioned in Section 4. Figure 1 shows how the average distance progresses over time. It can be seen that the CDONE algorithm achieves a similar accuracy as the other variations, with CDONE ELU performing slightly worse. However, a larger difference between the variations can be seen in Figure 2. This figure shows the mean number of nonzero coefficients c_k, $k = 1, \ldots, D$. The CDONE algorithm uses only about 16 out of all 500 available basis functions. Furthermore, the set of basis functions that are used remains fairly constant as can be seen in Figure 3. On average, the difference between the active set of coefficients $\{k : c_k = 0\}$ for a particular iteration and the next is less than 1, although this difference can go up to around 15 in one of the earlier iterations. We conclude that the CDONE algorithm has a high accuracy compared to the number of used basis functions, and that there is potential for efficient implementations of the minimization step of this algorithm by exploiting the sparsity.

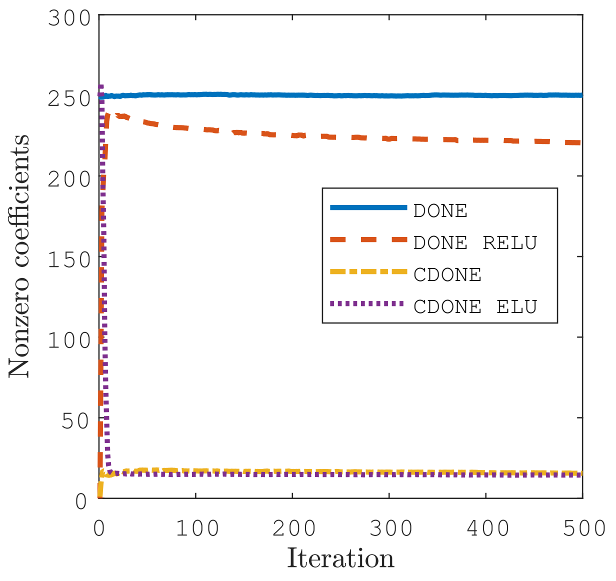


Fig. 2: Average number of nonzero coefficients per iteration.

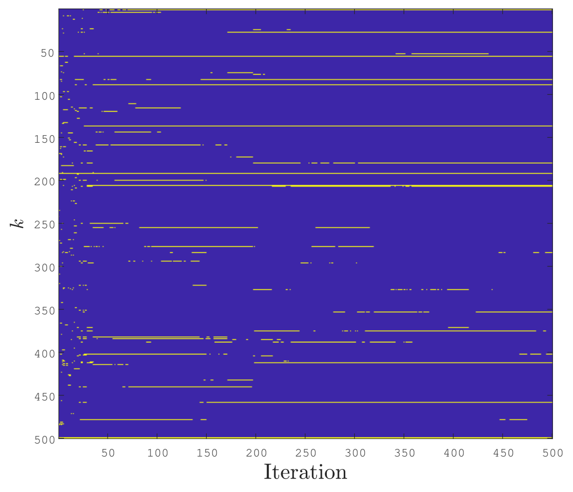
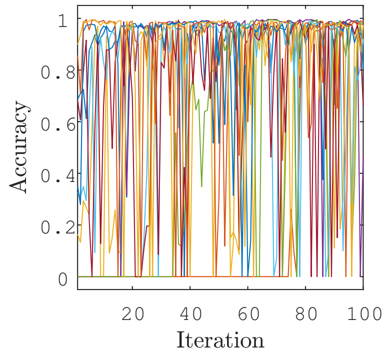


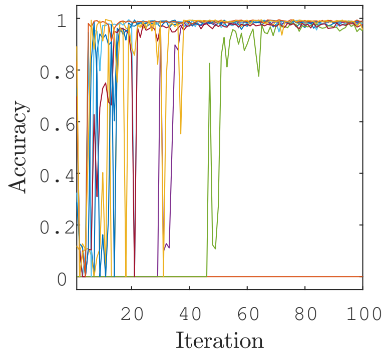
Fig. 3: Nonzero coefficients pattern for one of the runs of CDONE. Yellow indicates $c_k > 0$ for a particular basis function k at that iteration, while blue indicates $c_k = 0$.

5.2. Hyper-parameter optimization for deep learning

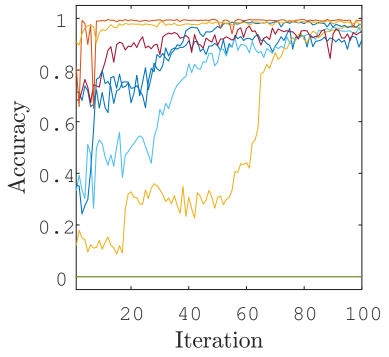
In our second experiment we consider the problem of hyper-parameter optimization. The task we consider is a handwritten digits recognition example [16], where a deep neural network is trained to classify handwritten digits. We modify this example so that the following eight hyper-parameters are considered unknown, even though values are given in the example: height, width and stride of the filter of the convolutional layer, height, width and stride for the max pooling layer, the



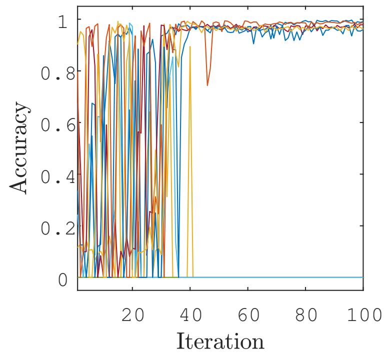
(a) DONE



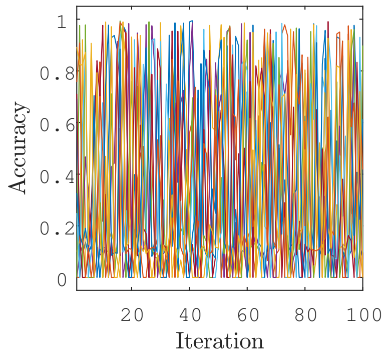
(b) DONE_RELU



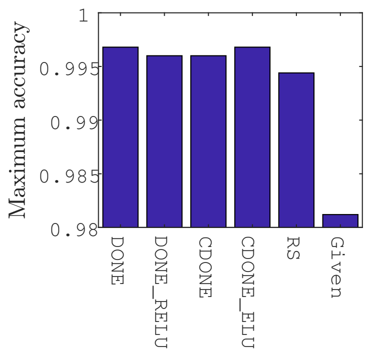
(c) CDONE



(d) CDONE_ELU



(e) Random Search



(f) Best results

Fig. 4: Accuracy of the current iteration of various algorithms, on the test set for the deep learning example. Ten individual runs are shown. The last plot shows the accuracy of the best iteration of all the runs, where the last column shows the accuracy given in the original example.

maximal number of epochs, and the initial learning rate. The example ends by showing the accuracy on the given test set after training.

The function we wish to minimize takes an 8-dimensional input and converts it from the set $[-1, 1]^8$ to realistic values for the hyper-parameters. Hyper-parameters that should have integer values are rounded. Then we run the example with these values for the hyper-parameters, and we take -1 times the accuracy on the test data as the output to be minimized.

Figure 4 shows the accuracy for the same four algorithm variations as in the previous example, for 10 runs, starting from 10 different initial guesses. The initial guesses were shared by the different algorithms. Note that some initial guesses were so bad that the accuracy was precisely 0, and the algorithms had trouble getting out of this part of the hyper-parameter space. The last plot shows the best result found by each algorithm, as well as the result provided in the original example [16]. All algorithms gave better results than the results given in the example. We used the same settings as in the previous example, but the number of basis functions and measurements were changed to $D = 800$ and $N = 100$, respectively. We also make a comparison with a random search over the hyper-parameter space. The random search can provide good hyper-parameter settings in just a few iterations, and so does the DONE algorithm. However, an advantage of the CDONE algorithm is that it stays near the currently best found solution and keeps improving. This allows the user to perform the original task while the hyper-parameters are being optimized. This is important in online applications, such as aberration correction for fluorescence microscopy [17], where the quality of the solution should not deteriorate during the optimization procedure. The most stable behavior, with a clear convergence plot, is found in the CDONE algorithm, although this algorithm had trouble with the worst initial guesses.

The number of nonzero coefficients in CDONE fluctuated between 20 and 60 in this example. We again conclude that the performance of the CDONE algorithm is very high compared to the number of basis functions that are actually used. We also conclude that the CDONE algorithm can be used for non-convex optimization problems despite the convexity constraint, and that the convexity constraint gives rise to stable behavior of the algorithm.

6. CONCLUSION

The DONE algorithm, a derivative-free optimization algorithm for finding the minimum of an objective using noisy measurements, has been adapted by introducing rectified linear units and a nonnegativity constraint. The constraint makes sure that the surrogate model of the objective is convex, allowing its global minimum to be found with convex optimization algorithms. The adapted CDONE algorithm has less hyper-parameters to tune, since the convexity helps in reducing the need for regularization and exploration of the surrogate model. Having less hyper-parameters to tune is crucial for certain tasks, especially when the algorithm is used for finding the optimal hyper-parameters of another algorithm or simulation. Furthermore, the surrogate model benefits from sparsity and can be evaluated efficiently.

The CDONE algorithm has been tested on an artificial example and on the problem of hyper-parameter optimization for a deep neural network classifier for handwritten digits. Using a lower effective number of basis functions because of the sparsity, the CDONE algorithm still exhibited high final accuracy and provided stable behavior, at the cost of slower convergence. In the future we will further exploit this sparsity in efficient implementations of all steps of the algorithm.

7. REFERENCES

[1] Andrew R Conn, Katya Scheinberg, and Luis N Vicente, *Introduction to derivative-free optimization*, SIAM, 2009.

[2] Donald R Jones, Matthias Schonlau, and William J Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.

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

[4] Jonas Mockus, *Bayesian approach to global optimization: theory and applications*, vol. 37, Springer Science & Business Media, 2012.

[5] Laurens Bliek, Hans R. G. W. Verstraete, Michel Verhaegen, and S. Wahls, "Online Optimization with Costly and Noisy Measurements using Random Fourier Expansions," *IEEE Trans. Neural Netw. Learn. Syst., to be published*.

[6] Ali Rahimi and Benjamin Recht, "Uniform approximation of functions with random bases," in *Communication, Control, and Computing, 2008 46th Annual Allerton Conference on*. IEEE, 2008, pp. 555–561.

[7] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio, "Maxout networks," *arXiv preprint arXiv:1302.4389*, 2013.

[8] Ali Rahimi and Benjamin Recht, "Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning," in *Advances in neural information processing systems*, 2009, pp. 1313–1320.

[9] Stephen Wright and Jorge Nocedal, "Numerical optimization," *Springer Science*, vol. 35, pp. 67–68, 1999.

[10] Yunmin Zhu, X Rong Li, et al., "Recursive least squares with linear constraints," *Communications in Information & Systems*, vol. 7, no. 3, pp. 287–312, 2007.

[11] Jie Chen, Cédric Richard, José Carlos M Bermudez, and Paul Honeine, "Nonnegative least-mean-square algorithm," *IEEE Transactions on Signal Processing*, vol. 59, no. 11, pp. 5225–5235, 2011.

[12] Vítor H Nascimento and Yuriy V Zakharov, "Rls adaptive filter with inequality constraints," *IEEE Signal Processing Letters*, vol. 23, no. 5, pp. 752–756, 2016.

[13] Konrad Engel and Sebastian Engel, "Recursive least squares with linear inequality constraints," *Optimization and Engineering*, vol. 16, no. 1, pp. 1–26, 2015.

[14] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[15] MATLAB Central File Exchange, "DONE algorithm," www.mathworks.com/matlabcentral/fileexchange/61288-done-algorithm, retrieved June 6, 2017.

[16] The MathWorks Inc., "Create simple deep learning network for classification," www.mathworks.com/help/nnet/examples/create-simple-deep-learning-network-for-classification.html, retrieved June 6, 2017.

[17] Paolo Pozzi, Dean Wilding, Oleg Soloviev, Hans Verstraete, Laurens Bliek, Gleb Vdovin, and Michel Verhaegen, "High speed wavefront sensorless aberration correction in digital micromirror based confocal microscopy," *Optics Express*, vol. 25, no. 2, pp. 949–959, 2017.