HyperNOMAD: Hyperparameter optimization of deep neural networks using mesh adaptive direct search

Sébastien Le Digabel, Dounia Lakhmiri, Christophe Tribes

ICCOPT 2019
2019–08–05
Presentation outline

Blackbox optimization

The MADS algorithm with categorical variables

Hyperparameters Optimization (HPO)

Computational experiments

Discussion
Blackbox optimization

The MADS algorithm with categorical variables

Hyperparameters Optimization (HPO)

Computational experiments

Discussion
Blackbox optimization (BBO) problems

▶ Optimization problem:

\[
\min_{x \in \Omega} f(x)
\]

▶ Evaluations of \( f \) (the objective function) and of the functions defining \( \Omega \) are usually the result of a computer code (a blackbox).

▶ Variables are typically continuous, but in this work, some of them are discrete – integers or categorical variables.
Blackbox optimization

We consider

$$\min_{x \in \Omega} f(x)$$

where the evaluations of $f$ and the functions defining $\Omega$ are the result of a computer simulation (a blackbox).
Blackbox optimization

We consider

$$\min_{x \in \Omega} f(x)$$

where the evaluations of $f$ and the functions defining $\Omega$ are the result of a computer simulation (a blackbox).

- Each call to the simulation may be expensive.
- The simulation can fail.
- Sometimes $f(x) \neq f(x)$.
- Derivatives are not available and cannot be approximated.
Blackbox optimization

The MADS algorithm with categorical variables

Hyperparameters Optimization (HPO)

Computational experiments

Discussion
General framework

```
for ( i = 0; i < nc; ++i )
  if ( i != hat_i ) {
    j = rp.pickup();
    if ( j == hat_i )
      j = rp.pickup();
  }
```

\[ f(x) \]

\[ x \in \Omega ? \]
Mesh Adaptive Direct Search (MADS) in $\mathbb{R}^n$

- [Audet and Dennis, Jr., 2006].

- Iterative algorithm that evaluates the blackbox at some trial points on a spatial discretization called the mesh.

- One iteration = search and poll.

- The search allows trial points generated anywhere on the mesh.

- The poll consists in generating a list of trial points constructed from poll directions. These directions grow dense.

- At the end of the iteration, the mesh size is reduced if no new success point is found.

- Algorithm backed by a convergence analysis.
Initializations \( (x_0, \Delta_0: \text{initial poll size}) \)

Iteration \( k \)

- Let \( \delta^k \leq \Delta^k \) be the mesh size parameter.

Search
- Test a finite number of mesh points.

Poll (if the Search failed)
- Construct set of directions \( D_k \).
- Test poll set \( P_k = \{ x_k + \delta^k d : d \in D_k \} \)
  - with \( \| \delta^k d \| \simeq \Delta_k \)

Updates
- If success
  - \( x_{k+1} \leftarrow \text{success point} \)
  - Increase \( \Delta^k \)
- Else
  - \( x_{k+1} \leftarrow x_k \)
  - Decrease \( \Delta^k \)

\( k \leftarrow k + 1 \), stop if \( \Delta^k \leq \Delta_{\text{min}} \) or go to [1]
Poll illustration (successive fails and mesh shrinks)

\[
\delta^k = 1 \\
\Delta^k = 1
\]

trial points = \{p_1, p_2, p_3\}
Poll illustration (successive fails and mesh shrinks)

\[
\begin{align*}
\delta^k &= 1 \\
\Delta^k &= 1
\end{align*}
\]

\[
\begin{align*}
\delta^{k+1} &= 1/4 \\
\Delta^{k+1} &= 1/2
\end{align*}
\]

trial points = \{p_1, p_2, p_3\} = \{p_4, p_5, p_6\}
Poll illustration (successive fails and mesh shrinks)

\[
\delta^k = 1 \\
\Delta^k = 1
\]

\[
\delta^{k+1} = 1/4 \\
\Delta^{k+1} = 1/2
\]

\[
\delta^{k+2} = 1/16 \\
\Delta^{k+2} = 1/4
\]

trial points = \{p_1, p_2, p_3\} = \{p_4, p_5, p_6\} = \{p_7, p_8, p_9\}
Types of variables in MADS

- MADS has been initially designed for continuous variables.

- Some theory exists for categorical variables [Audet and Dennis, Jr., 2001, Abramson, 2004, Abramson et al., 2009].

- (Other discrete variables now considered in MADS: Integer, binary, granular [Audet et al., 2019]).

- Two kinds of “categorical” variables:
  - Non-orderable and unrelaxable discrete variables.
  - An integer whose value changes the number of variables of the problem.
Example: A thermal insulation system

\begin{align*}
\min_{\Delta x, T, n, M} \quad & \text{power}(\Delta x, T, n, M) \\
\text{s.t.} \quad & \Delta x \geq 0 \quad T_C \leq T \leq T_H \\
& n \in \mathbb{N} \quad M \in \text{Materials}
\end{align*}
MADS with categorical variables

- [Abramson et al., 2009].

- The search is still a finite search on the mesh, free of any rules.

- The poll is the failsafe step that evaluates function values at mesh neighbors for the continuous variables, and in a user-defined set of neighbors $\mathcal{N}(x_k)$.

- This set of neighbors defines a notion of local optimality.
Extended poll

\[ y_k \bullet \ldots \bullet y_k^j \bullet \ldots \bullet z_k \]

\[ x_k \bullet \]

HyperNOMAD: Hyperparameter optimization with MADS
Extended poll
Blackbox optimization

The MADS algorithm with categorical variables

Hyperparameters Optimization (HPO)

Computational experiments

Discussion
HPO with HyperNOMAD

- PhD project of Dounia Lakhmiri.
- We focus on the HPO of deep neural networks.
- Our advantages:
  - Blackbox optimization problem:
    \[
    \text{One blackbox call} = \text{Training} + \text{validation} + \text{test}, \text{for a fixed set of hyperparameters.}
    \]
  - Presence of categorical variables (ex.: number of layers).
  - Existing methods are mostly heuristics
    \[
    (\text{grid search, random search, GAs, etc.})
    \]
- Based on the NOMAD implementation of MADS.
Principle
HyperNOMAD

- HyperNOMAD is the interface between NOMAD and a deep learning platform.
- Based on the PyTorch library.
- Works with preexisting datasets such as MNIST or CIFAR-X, or on a custom data.
- Available at https://github.com/DouniaLakhmiri/HyperNOMAD.

- We consider three types of hyperparameters:
  - Architecture of the neural network.
  - Optimizer.
  - Plus one for the size of mini-batches.

- Number of hyperparameters: \(5n_1 + n_2 + 10\).
Network architecture

A convolutional neural network is a deep neural network consisting of a succession of convolutional layers followed by fully connected layers:

Image from [Deshpande, 2019].
## Hyperparameters for the architecture \((5n_1 + n_2 + 4)\)

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Type</th>
<th>Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of convolutional layers ((n_1))</td>
<td>Categorical</td>
<td>([0;20])</td>
</tr>
<tr>
<td>Number of output channels</td>
<td>Integer</td>
<td>([0;50])</td>
</tr>
<tr>
<td>Kernel size</td>
<td>Integer</td>
<td>([0;10])</td>
</tr>
<tr>
<td>Stride</td>
<td>Integer</td>
<td>([1;3])</td>
</tr>
<tr>
<td>Padding</td>
<td>Integer</td>
<td>([0;2])</td>
</tr>
<tr>
<td>Do a pooling</td>
<td>Boolean</td>
<td>(0) or (1)</td>
</tr>
<tr>
<td>Number of full layers ((n_2))</td>
<td>Categorical</td>
<td>([0;30])</td>
</tr>
<tr>
<td>Size of the full layer</td>
<td>Integer</td>
<td>([0;500])</td>
</tr>
<tr>
<td>Dropout rate</td>
<td>Real</td>
<td>([0;1])</td>
</tr>
<tr>
<td>Activation function</td>
<td>Cat./Int.</td>
<td>ReLU, Sigmoid, Tanh</td>
</tr>
</tbody>
</table>
### Hyperparameters for the optimizer (5)

<table>
<thead>
<tr>
<th>Optimizer</th>
<th>Hyperparameter</th>
<th>Type</th>
<th>Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stochastic Gradient Descent (SGD)</td>
<td>Learning rate</td>
<td>Real</td>
<td>[0;1]</td>
</tr>
<tr>
<td></td>
<td>Momentum</td>
<td>Real</td>
<td>[0;1]</td>
</tr>
<tr>
<td></td>
<td>Dampening</td>
<td>Real</td>
<td>[0;1]</td>
</tr>
<tr>
<td></td>
<td>Weight decay</td>
<td>Real</td>
<td>[0;1]</td>
</tr>
<tr>
<td>Adam</td>
<td>Learning rate</td>
<td>Real</td>
<td>[0;1]</td>
</tr>
<tr>
<td></td>
<td>$\beta_1$</td>
<td>Real</td>
<td>[0;1]</td>
</tr>
<tr>
<td></td>
<td>$\beta_2$</td>
<td>Real</td>
<td>[0;1]</td>
</tr>
<tr>
<td></td>
<td>Weight decay</td>
<td>Real</td>
<td>[0;1]</td>
</tr>
<tr>
<td>Adagrad</td>
<td>Learning rate</td>
<td>Real</td>
<td>[0;1]</td>
</tr>
<tr>
<td></td>
<td>Learning rate decay</td>
<td>Real</td>
<td>[0;1]</td>
</tr>
<tr>
<td></td>
<td>Initial accumulator</td>
<td>Real</td>
<td>[0;1]</td>
</tr>
<tr>
<td></td>
<td>Weight decay</td>
<td>Real</td>
<td>[0;1]</td>
</tr>
<tr>
<td>RMSProp</td>
<td>Learning rate</td>
<td>Real</td>
<td>[0;1]</td>
</tr>
<tr>
<td></td>
<td>Momentum</td>
<td>Real</td>
<td>[0;1]</td>
</tr>
<tr>
<td></td>
<td>$\alpha$</td>
<td>Real</td>
<td>[0;1]</td>
</tr>
<tr>
<td></td>
<td>Weight decay</td>
<td>Real</td>
<td>[0;1]</td>
</tr>
</tbody>
</table>
Blocks of hyperparameters

- **Convolution block**: 2 convolutional layers with
  (number of output channels, kernel size, stride, padding, pooling) = (16, 5, 1, 1, 0) and (7, 3, 1, 1, 1):

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>16</th>
<th>5</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>7</th>
<th>3</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
</table>

- **Fully connected block**: 3 fully connected layers with sizes of output = 1200, 512, 20:

<table>
<thead>
<tr>
<th></th>
<th>3</th>
<th>1200</th>
<th>512</th>
<th>20</th>
</tr>
</thead>
</table>

- **Optimizer block**: SGD with learning rate = 0.1, momentum = 0.9, dampening = $1e^{-4}$, and weight decay = 0:

  |   | 1  | 0.1 | 0.9 | $1e^{-4}$ | 0 |
Blackbox optimization

The MADS algorithm with categorical variables

Hyperparameters Optimization (HPO)

Computational experiments

Discussion
# Average results on MNIST

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Avg accuracy on validation set</th>
<th>Avg accuracy on test set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rand. search [Bergstra and Bengio, 2012]</td>
<td>94.02</td>
<td>89.07</td>
</tr>
<tr>
<td>SMAC [Hutter et al., 2011]</td>
<td>95.48</td>
<td>97.54</td>
</tr>
<tr>
<td>RBFOpt [Diaz et al., 2017]</td>
<td>95.66</td>
<td>97.93</td>
</tr>
<tr>
<td>NOMAD</td>
<td><strong>96.81</strong></td>
<td><strong>97.98</strong></td>
</tr>
</tbody>
</table>

HyperNOMAD: Hyperparameter optimization with MADS
MNIST results with HyperNOMAD

Comparison between HyperNOMAD, TPE and RS when launched from the default starting point of HyperNOMAD, on the MNIST data set. Best solution with HyperNOMAD: 99.61%.
Results on CIFAR-10 (vs Hyperopt)

- Training with 40,000 images, validation/test on 10,000 images.
- One evaluation (training+test) $\simeq 2$ hours (i7-6700@3.4 GHz, GeForce GTX 1070).

![Graphs showing test accuracy vs number of blackbox evaluations for different methods.](image)

(a) Default starting point

(b) From a VGG architecture
<table>
<thead>
<tr>
<th>BBO</th>
<th>MADS and categorical variables</th>
<th>HPO</th>
<th>Computational experiments</th>
<th>Discussion</th>
</tr>
</thead>
</table>

## Blackbox optimization

## The MADS algorithm with categorical variables

## Hyperparameters Optimization (HPO)

## Computational experiments

## Discussion
Discussion

- **HyperNOMAD**: Library for the HPO problem.
- Specialized for convolutional deep neural networks via the PyTorch library.
- Key aspect: Optimize both the architecture and the optimization phase of a deep neural network.
- Based on the blackbox optimization solver NOMAD and its ability to model categorical variables.
- So far: Competitive results with state-of-the-art on the MNIST and CIFAR-10 datasets.
- Future work: Expand the library to other types of problems than classification, provide interfaces to other libraries.
- We thank G. Naniccini for his code and the NVIDIA GPU grant program.
References I


Audet, C. and Dennis, Jr., J. (2001).

Audet, C. and Dennis, Jr., J. (2006).


References II

Deshpande, A. (2019).
A Beginner’s Guide To Understanding Convolutional Neural Networks.

An effective algorithm for hyperparameter optimization of neural networks.

Sequential model-based optimization for general algorithm configuration.

Algorithm 909: NOMAD: Nonlinear Optimization with the MADS algorithm.