HYPERNOMAD: Hyper-parameter optimization of deep neural networks using mesh adaptive direct search

Sébastien Le Digabel, Dounia Lakhmiri, Christophe Tribes

GERAD
GROUP FOR RESEARCH IN
DECISION ANALYSIS

POLYTECHNIQUE
MONTRÉAL
TECHNOLOGICAL
UNIVERSITY

The Optimization Days 2019

2019–05–15

Presentation outline

Blackbox optimization

The MADS algorithm with categorical variables

Hyper-Parameters Optimization (HPO)

Computational experiments

Discussion

Blackbox optimization

The MADS algorithm with categorical variables

Hyper-Parameters Optimization (HPO)

Computational experiments

Discussion

Blackbox optimization (BBO) problems

- Optimization problem:

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

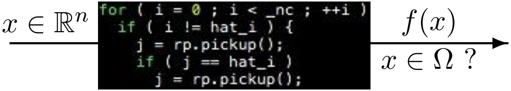
- Evaluations of f (the objective function) and of the functions defining Ω are usually the result of a computer code (a blackbox).
- Variables are typically continuous, but in this work, some of them are discrete – integers or *categorical* variables.

Blackbox optimization

We consider

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

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



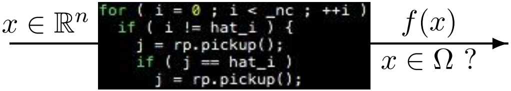
```
for ( i = 0 ; i < _nc ; ++i )
  if ( i != hat_i ) {
    j = rp.pickup();
    if ( j == hat_i )
      j = rp.pickup();
```

Blackbox optimization

We consider

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

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



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

- 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

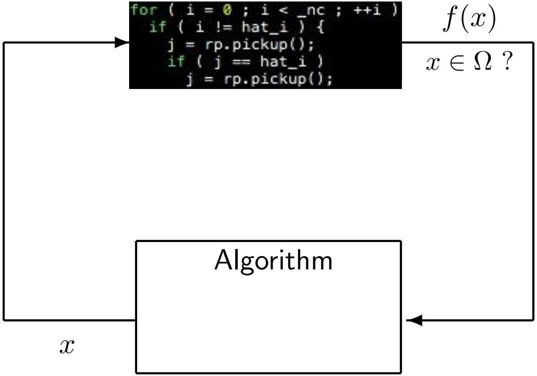
The MADS algorithm with categorical variables

Hyper-Parameters 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();
```

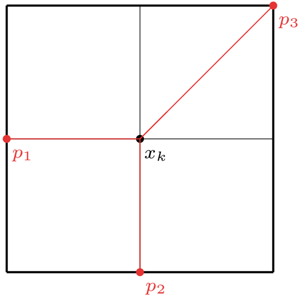
Mesh Adaptive Direct Search (MADS) in $\mathbb{R}^n$

- [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.

```
[0] Initializations ($x_0$, $\Delta_0$: initial poll size )
[1] 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 $\|\delta^k d\| \simeq \Delta_k$
[2] Updates
    if success
        $x_{k+1} \leftarrow$ success point
        increase $\Delta^k$
    else
        $x_{k+1} \leftarrow x_k$
        decrease $\Delta^k$
    $k \leftarrow k+1$, stop if $\Delta^k \leq \Delta_{\min}$ or go to [1]
```

Poll illustration (successive fails and mesh shrinks)

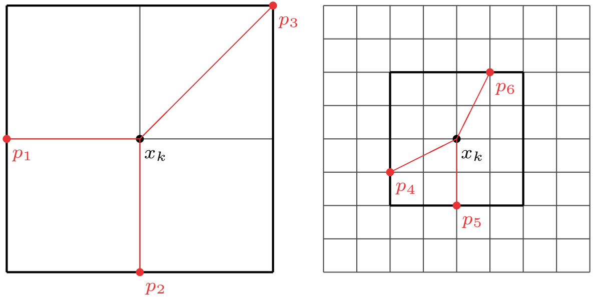
$$\delta^k = 1$$
$$\Delta^k = 1$$



trial points$=\{p_1, p_2, p_3\}$

Poll illustration (successive fails and mesh shrinks)

$$\delta^k = 1 \qquad \delta^{k+1} = 1/4$$
$$\Delta^k = 1 \qquad \Delta^{k+1} = 1/2$$



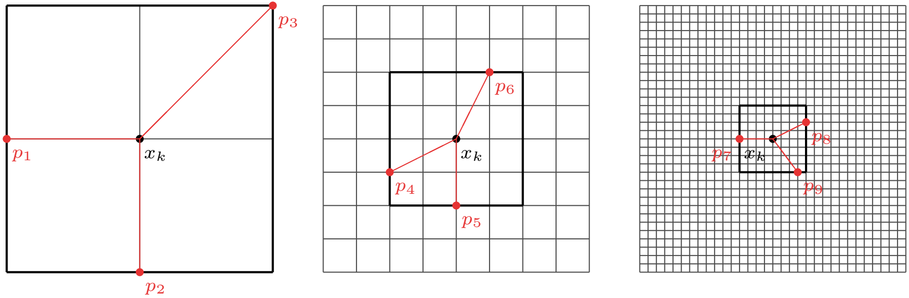
trial points$=\{p_1, p_2, p_3\}$ $= \{p_4, p_5, p_6\}$

Poll illustration (successive fails and mesh shrinks)

$\delta^k = 1$
$\Delta^k = 1$

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

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

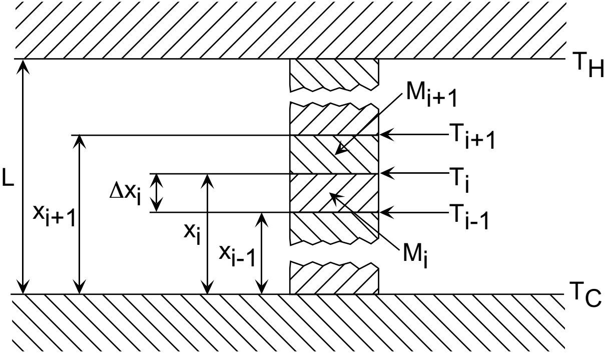


trial points$=\{p_1, p_2, p_3\}$ $= \{p_4, p_5, p_6\}$ $= \{p_7, p_8, p_9\}$

Types of variables in MADS

- MADS has been initially designed for continuous variables.
- Some theory exists for categorical variables [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



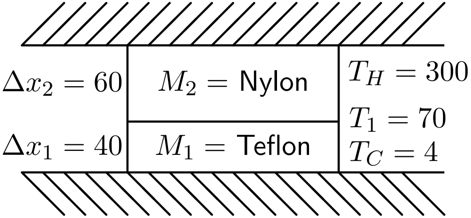
$$\begin{aligned} \min_{\mathbf{\Delta x},\mathbf{T},\mathbf{n},\mathbf{M}} \quad & power(\mathbf{\Delta x}, \mathbf{T}, \mathbf{n}, \mathbf{M}) \\ \text{s.t.} \quad & \mathbf{\Delta x} \geq \mathbf{0} \quad T_C \leq \mathbf{T} \leq T_H \\ & \mathbf{n} \in \mathbb{N} \qquad \mathbf{M} \in Materials \end{aligned}$$

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*.

Illustration of a set of neighbors

Heat shield example:



$\Delta x_2 = 60$

$M_2 =$ Nylon

$\Delta x_1 = 40$

$M_1 =$ Teflon

$T_H = 300$

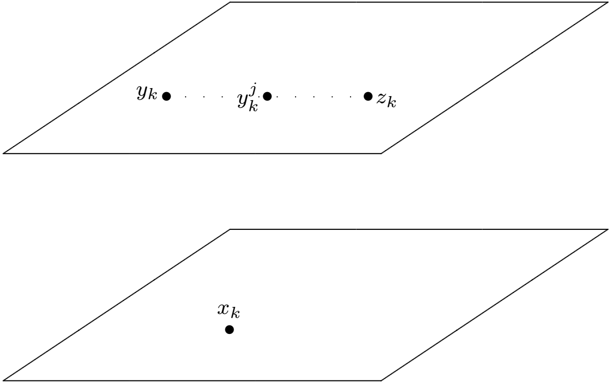
$T_1 = 70$

$T_C = 4$

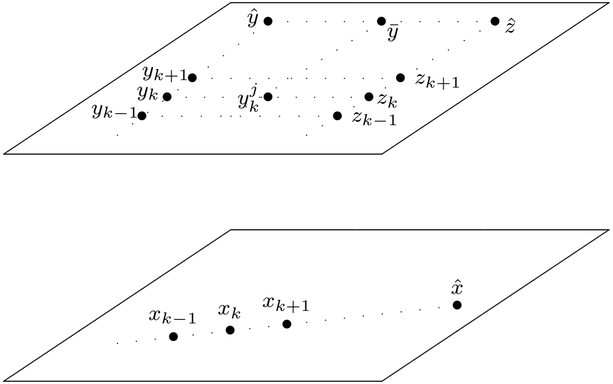
Mesh neighbors: Vary either Δx_1 or Δx_2 or T_1 by $\pm\Delta^k$.

Set of neighbors $\mathcal{N}$:
- Change M_1 to Nylon or to Fiberglass.
- Change M_2 to Teflon or to Fiberglass.
- Add a shield and a material.
- Remove a shield and a material.

Extended poll



Extended poll



Blackbox optimization

The MADS algorithm with categorical variables

Hyper-Parameters Optimization (HPO)

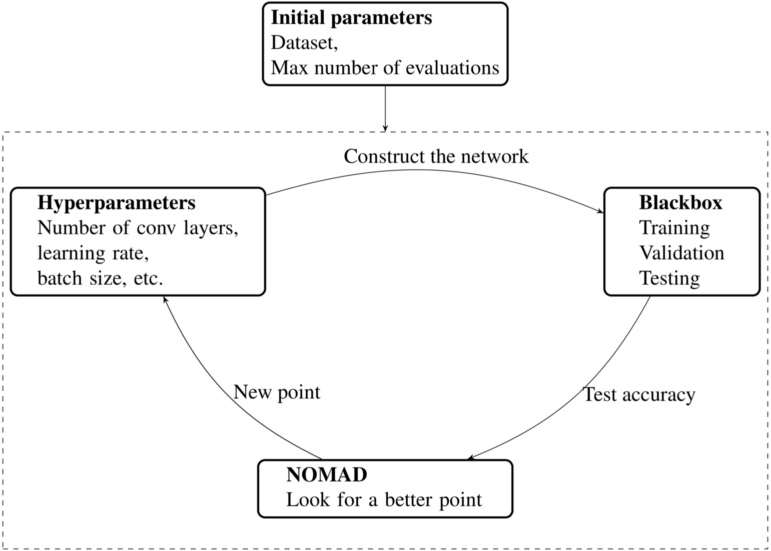
Computational experiments

Discussion

HPO with HYPERNOMAD

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

Principle



Initial parameters
Dataset,
Max number of evaluations

Construct the network

Hyperparameters
Number of conv layers,
learning rate,
batch size, etc.

Blackbox
Training
Validation
Testing

Test accuracy

New point

NOMAD
Look for a better point

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:

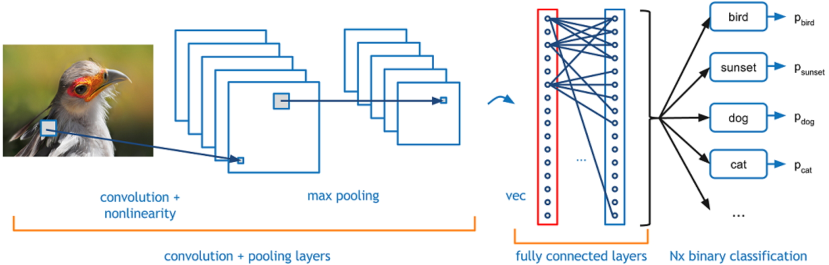


Image from [Deshpande, 2019].

Hyperparameters for the architecture $(5n_1 + n_2 + 4)$

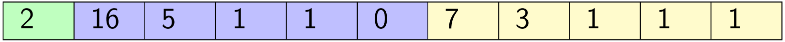
Hyperparameter	**Type**	**Scope**
Number of convolutional layers (n_1)	Categorical	[0, 20]
Number of output channels	Integer	[0, 50]
Kernel size	Integer	[0, 10]
Stride	Integer	[1, 3]
Padding	Integer	[0, 2]
Do a pooling	Boolean	0 or 1
Number of full layers (n_2)	Categorical	[0,30]
Size of the full layer	Integer	[0, 500]
Dropout rate	Real	[0, 1]
Activation function	Categorical	ReLU, Sigmoid, Tanh

Hyperparameters for the optimizer (5)

Optimizer	Hyperparameter	Type	Scope
Stochastic Gradient Descent (SGD)	Learning rate	Real	[0, 1]
	Momentum	Real	[0, 1]
	Dampening	Real	[0, 1]
	Weight decay	Real	[0, 1]
Adam	Learning rate	Real	[0, 1]
	β_1	Real	[0, 1]
	β_2	Real	[0, 1]
	Weight decay	Real	[0, 1]
Adagrad	Learning rate	Real	[0, 1]
	Learning rate decay	Real	[0, 1]
	Initial accumulator	Real	[0, 1]
	Weight decay	Real	[0, 1]
RMSProp	Learning rate	Real	[0, 1]
	Momentum	Real	[0, 1]
	α	Real	[0, 1]
	Weight decay	Real	[0, 1]

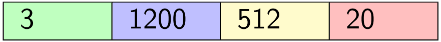
Blocks of hyperparameters

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



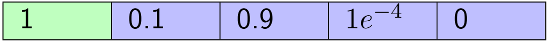
2	16	5	1	1	0	7	3	1	1	1

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



3	1200	512	20

- 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

Datasets currently embedded in HYPERNOMAD

Dataset	Training data	Validation data	Testing data	Number of classes
MNIST	40000	10000	10000	10
Fashion-MNIST	40000	10000	10000	10
EMNIST	40000	10000	10000	10
KMNIST	40000	10000	10000	10
CIFAR-10	40000	10000	10000	10
CIFAR-100	40000	10000	10000	100
STL-10	4000	1000	8000	10

Blackbox optimization

The MADS algorithm with categorical variables

Hyper-Parameters Optimization (HPO)

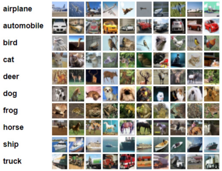
Computational experiments

Discussion

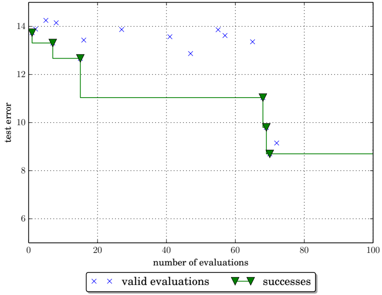
Average results on MNIST

Algorithm	Avg accuracy on validation set	Avg accuracy on test set
Rand. search [Bergstra and Bengio, 2012]	94.02	89.07
SMAC [Hutter et al., 2011]	95.48	97.54
RBFOpt [Diaz et al., 2017]	95.66	97.93
NOMAD w/o cat. var.	95.43	96.51
HYPERNOMAD	**97.54**	**97.95**

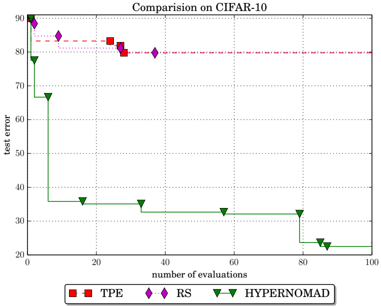
Results on CIFAR-10



- 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).



HYPERNOMAD starting from a VGG architecture



HYPERNOMAD vs Hyperopt (TPE and random search)

Blackbox optimization

The MADS algorithm with categorical variables

Hyper-Parameters 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

Abramson, M. (2004).
Mixed variable optimization of a Load-Bearing thermal insulation system using a filter pattern search algorithm.
Optimization and Engineering, 5(2):157–177.

Abramson, M., Audet, C., Chrissis, J., and Walston, J. (2009).
Mesh Adaptive Direct Search Algorithms for Mixed Variable Optimization.
Optimization Letters, 3(1):35–47.

Audet, C. and Dennis, Jr., J. (2006).
Mesh Adaptive Direct Search Algorithms for Constrained Optimization.
SIAM Journal on Optimization, 17(1):188–217.

Audet, C., Le Digabel, S., and Tribes, C. (2019).
The Mesh Adaptive Direct Search Algorithm for Granular and Discrete Variables.
SIAM Journal on Optimization, 29(2):1164–1189.

Balaprakash, P., Salim, M., Uram, T., Vishwanath, V., and Wild, S. (2018).
DeepHyper: Asynchronous Hyperparameter Search for Deep Neural Networks.
In *2018 IEEE 25th International Conference on High Performance Computing (HiPC)*, pages 42–51.

Bergstra, J. and Bengio, Y. (2012).
Random search for hyper-parameter optimization.
Journal of Machine Learning Research, 13:281–305.

References II

Bouthillier, X. and Tsirigotis, C. (2019).
Oríon: Asynchronous Distributed Hyperparameter Optimization.
`https://github.com/Epistimio/orion`.

Deshpande, A. (2019).
A Beginner's Guide To Understanding Convolutional Neural Networks.
`https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/`.

Diaz, G., Fokoue, A., Nannicini, G., and Samulowitz, H. (2017).
An effective algorithm for hyperparameter optimization of neural networks.
IBM Journal of Research and Development, 61(4):9:1–9:11.

Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011).
Sequential model-based optimization for general algorithm configuration.
In *International Conference on Learning and Intelligent Optimization*, pages 507–523. Springer.

Le Digabel, S. (2011).
Algorithm 909: NOMAD: Nonlinear Optimization with the MADS algorithm.
ACM Transactions on Mathematical Software, 37(4):44:1–44:15.

References III

NS, J. B., Yamins, D., and Cox, D. (2013).
Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures.
In *Proceedings of the 30th International Conference on International Conference on Machine Learning*, volume 28 of *ICML'13*, pages I–115–I–123. JMLR.org.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011).
Scikit-learn: Machine Learning in Python.
Journal of Machine Learning Research, 12:2825–2830.

Snoek, J., Larochelle, H., and Adams, R. P. (2012).
Practical Bayesian optimization of machine learning algorithms.
In *Advances in Neural Information Processing Systems (NIPS) 25*, pages 2960–2968.

Yelp (2014).
Metric Optimization Engine.
`https://github.com/Yelp/MOE`.

Open source libraries for the HPO problem

	Optimization method					Type of variabes		
Name	Grid search	Random search	Bayesian optim.	Model based	Direct search	Real	Int.	Cat.
scikit-learn [Pedregosa et al., 2011]	✓	✓	-	-	-	✓	✓	✓
Hyperopt [NS et al., 2013]	-	✓	✓	-	-	✓	✓	✓
Spearmint [Snoek et al., 2012]	✓	✓	✓	-	-	✓	✓	✓
SMAC [Hutter et al., 2011]	-	-	-	✓	-	✓	✓	✓
MOE [Yelp, 2014]	-	-	✓	-	-	✓	-	-
RBFOpt [Diaz et al., 2017]	-	-	-	✓	-	✓	✓	-
DeepHyper [Balaprakash et al., 2018]	-	✓	-	✓	-	✓	✓	✓
Oríon [Bouthillier and Tsirigotis, 2019]	-	✓	-	-	-	✓	✓	✓
HYPERNOMAD	-	-	-	-	✓	✓	✓	✓