Noisy Black-Box Optimization

Marie-Liesse Cauwet

Université de Liège, Montefiore Institute, SYSTMOD Research Unit

October 28, 2016
About Me ...

- Postdoctoral Fellow at Université de Liège.

- Previously Ph.D. student at INRIA Saclay in France. Thesis: "Uncertainties in Optimization", supervised by Olivier Teytaud.

- Fields of research:
  - Noisy Black-Box Optimization;
  - Markov Decision Process Problems;
  - Decision Theory;
  - Statistics (bootstrap & cross-validation).
Outline

1. Introductory Examples
2. Framework
3. Different Families of Optimization Algorithms
   - Value-based Algorithms
   - Comparison-based Algorithms
   - Experimentations
4. Portfolio of Noisy Optimization Algorithms
5. Conclusion
6. References
Noise?

Forecast error.

Uncertain measurement.

Stochastic Policy.

Stochastic Environment.
Rodolfo Dufo-López, Iván R. Cristóbal-Monreal, and José M. Yusta.

Stochastic-heuristic methodology for the optimization of components and control variables of pv-wind-diesel-battery stand-alone systems. 99:919–935,
Examples in Games

An example of a generated level Dahlskog and Togelius [2014].
1 Introductory Examples

2 Framework

3 Different Families of Optimization Algorithms
   Value-based Algorithms
   Comparison-based Algorithms
   Experimentations

4 Portfolio of Noisy Optimization Algorithms

5 Conclusion

6 References
Optimization Setting

Continuous Black-Box Noisy Optimization Problem

- **Continuous** objective function \( f : \mathcal{D} \subseteq \mathbb{R}^d \rightarrow \mathbb{R} \).
- **Black-box:**
  \[
  x \xrightarrow{\text{Oracle}} f(x).
  \]
- **Noise:** inaccurate output of ‘Oracle’: \( f_{\text{noisy}}(x) \).
- **Additive model of noise:**
  \[
  f_{\text{noisy}}(x) = f(x) + \omega,
  \]
  with \( \mathbb{E}(\omega) = 0 \) and \( \text{Var}(\omega) \) constant > 0.

Goal

Finding the optimum \( x^* \) such that

\[
\forall \ x \in \mathcal{D}, \mathbb{E} \ f_{\text{noisy}}(x) \geq \mathbb{E} \ f_{\text{noisy}}(x^*).
\]
Performance of Optimization Algorithms?

Optimization process

An optimization algorithm:

- **selects** a search point \( x \);
- **evaluates** its corresponding fitness value \( y = f_{noisy}(x) \);
- **recommends** an approximation \( \tilde{x} \) of the optimum \( x^* \).

Computational Cost

The call to the oracle can be:

- fast, whereas internal and communication cost are not negligible;
- not constant;
- **constant and expensive/slow**: indexation in the number of calls the the black-box.
Definition: Simple Regret (SR)

The Simple Regret (SR) is defined by:

\[ SR_n := f(\tilde{x}_n) - f(x^*), \]

with \( \tilde{x}_n \) recommendation after \( n \) function evaluations. 

\( \tilde{x}_n \) random variable \( \implies \) \( SR_n \) random variable.
Definition: Rate of convergence of an algorithm

Let $\mathcal{F}$ be a family of noisy objective functions. If $\exists \alpha > 0$ s.t.

$$\forall f \in \mathcal{F}, \exists C > 0, \exists n_0 \in \mathbb{N}, \text{ s.t. } \forall n \geq n_0, \ ESR_n \leq \frac{C}{n^\alpha}, \quad (\star)$$

then $\alpha_{\text{max}} := \max(\alpha | \alpha \text{ satisfies } \star)$ is the rate of convergence of the algorithm.

Typical of the black-box optimisation setting with additive noise.

Optimal value of $\alpha$?

Upper / lower bound on the rate of convergence?
A Lower Bound on the Rate of Convergence

Lower Bound
For any kind of querying algorithm, in the black-box optimization setting with additive noise,

\[ \alpha \leq 1, \]

see Chen [1988], Polyak and Tsybakov [1990].

Question
Is it possible to reach this optimal rate of convergence, i.e. to find a noisy optimization algorithm with \( \mathbb{E}SR = O(1/n) \)?
1 Introductory Examples

2 Framework

3 Different Families of Optimization Algorithms
   Value-based Algorithms
   Comparison-based Algorithms
   Experimentations

4 Portfolio of Noisy Optimization Algorithms

5 Conclusion

6 References
Value-based vs. Comparison-based Algorithms

Value-based Algorithms: use the **fitness value** itself.

Comparison-based Algorithms: use **only the ranking** of the search points.
1 Introductory Examples

2 Framework

3 Different Families of Optimization Algorithms
   Value-based Algorithms
   Comparison-based Algorithms
   Experimentations

4 Portfolio of Noisy Optimization Algorithms

5 Conclusion

6 References
**Value-based Algorithms**

**Kiefer-Wolfowitz scheme**

\[
\tilde{x}_{n+1} = \tilde{x}_n - a_n \cdot g_n,
\]

\[
g_n := \text{approximate gradient, estimated by finite differences.}
\]

**Kiefer and Wolfowitz [1952]: Symmetric difference quotient**

\[
g_n = \frac{f_{noisy}(\tilde{x}_n + c_n) - f_{noisy}(\tilde{x}_n - c_n)}{c_n}.
\]

**[KW52] \( \alpha = 2/3 \)**

**[Fabian67] \( \alpha \rightarrow 1 \)**

- huge comp. cost (2sd eval. per iter.),
- only even ‘s’.

\[ f(x + c_n) \quad f(x - c_n) \]
Value-based Algorithms

Kiefer-Wolfowitz scheme
\[ \tilde{x}_{n+1} = \tilde{x}_n - a_n \cdot g_n, \]
\[ g_n := \text{approximate gradient, estimated by finite differences.} \]

Fabian [1967]: Averaging differences
\[ g_n^{(i)} = \frac{1}{c_n} \sum_{k=1}^{s/2} v_k \left( f_{\text{noisy}}(\tilde{x}_n + c_n u_k e_i) - f_{\text{noisy}}(\tilde{x}_n - c_n u_k e_i) \right). \]

[KW52] \( \alpha = 2/3 \)

[Fabian67] \( \alpha \to 1 \)
- huge comp. cost (2sd eval. per iter.),
- only even ‘s’.

\[ \tilde{x}_n, \tilde{x}_n + c_n u_1, \tilde{x}_n - c_n u_2, \tilde{x}_n + c_n u_3, \tilde{x}_n - c_n u_4 \]
Value-based Algorithms

**Kiefer-Wolfowitz scheme**

\[ \tilde{x}_{n+1} = \tilde{x}_n - a_n \cdot g_n, \]

\[ g_n := \text{approximate gradient, estimated by finite differences.} \]

**Spall [1987]: Stochastic differences**

\[ g_n^{(i)} = \frac{f_{\text{noisy}}(\tilde{x}_n + c_n \Delta) - f_{\text{noisy}}(\tilde{x}_n - c_n \Delta)}{2c_n \Delta^{(i)}}, \quad \Delta \text{ Bernoulli.} \]

**References**

**[KW52]** \( \alpha = \frac{2}{3} \)

**[Fabian67]** \( \alpha \to 1 \)

- huge comp. cost (2sd eval. per iter.),
- only even ‘s’.

**[Spall87]** \( \alpha = \frac{2}{3} \)

- fast in practise,
- works in noise-free setting.

**[PT90]** \( \alpha \to 1 \)

- fast,
- any ‘s’.
Value-based Algorithms

Kiefer-Wolfowitz scheme

\[ \tilde{x}_{n+1} = \tilde{x}_n - a_n \cdot g_n, \]

\[ g_n := \text{approximate gradient, estimated by finite differences}. \]

Polyak and Tsybakov [1990]: Smoothing stochastic differences

\[ g_n = K(\Delta) \frac{f_{\text{noisy}}(\tilde{x}_n + c_n\Delta) - f_{\text{noisy}}(\tilde{x}_n - c_n\Delta)}{2c_n}, \Delta \text{ uniform}. \]

[KW52] \( \alpha = 2/3 \)

[Fabian67] \( \alpha \rightarrow 1 \)

- huge comp. cost (2sd eval. per iter.),
- only even ‘s’.

[Spall87] \( \alpha = 2/3 \)

- fast in practise,
- works in noise-free setting.

[PT90] \( \alpha \rightarrow 1 \)

- fast,
- any ‘s’.
1 Introductory Examples

2 Framework

3 Different Families of Optimization Algorithms
   Value-based Algorithms
   Comparison-based Algorithms
   Experimentations

4 Portfolio of Noisy Optimization Algorithms

5 Conclusion

6 References
Effect of the noise on Evolution Strategies?

Optimization Algorithm: Evolution Strategy.

**Noise-free** Objective function:

\[ f : \mathbb{R}^2 \rightarrow \mathbb{R} \]
\[ x \mapsto \|x - 0.5\|^2 \]

**Noisy** Objective function:

\[ f : \mathbb{R}^2 \rightarrow \mathbb{R} \]
\[ x \mapsto \|x - 0.5\|^2 + \mathcal{N} \]

Standard Gaussian
Comparison-based algorithms

(1, λ) Self Adaptive Evolution Strategy.

Inputs: Initial values $\tilde{x}_0$, $\sigma_0$, $\tau > 0$, $\lambda \in \mathbb{N}^+$

1: $n \leftarrow 1$ ★ iteration index
2: while not terminate do
3:     $\forall i \in \{1, \ldots, \lambda\}$, $\sigma_{n-1}^i \leftarrow \sigma_{n-1} \exp(\tau G)$ ★ Mutate Step-size
4:     $\forall i \in \{1, \ldots, \lambda\}$, $x_n^i \leftarrow \tilde{x}_{n-1} + \sigma_{n-1}^i G$ ★ Offsprings
5:     Rank $x_n^i, \ldots, x_n^\lambda$ such that

$$f_{noisy}(x_n^{i_1}) \leq f_{noisy}(x_n^{i_2}) \leq \cdots \leq f_{noisy}(x_n^{i_\lambda})$$

6:     $\tilde{x}_n \leftarrow x_n^{i_1}$ ★ Selection
7:     $\sigma_n \leftarrow \sigma_n^{i_1}$ ★ Update Step-size
8:     $n \leftarrow n + 1$
9: end while

return $x_n$
Comparison-based algorithms

\((1, \lambda)\) Self Adaptive Evolution Strategy.

**Inputs:** Initial values \(\tilde{x}_0, \sigma_0, \tau > 0, \lambda \in \mathbb{N}^+\)

1: \(n \leftarrow 1\)  \hspace{1cm} \text{iteration index}

2: \textbf{while} not terminate \textbf{do}

3: \(\forall i \in \{1, \ldots, \lambda\}, \sigma_{n-1}^i \leftarrow \sigma_{n-1}^i \exp(\tau G)\)  \hspace{1cm} \text{Mutate Step-size}

4: \(\forall i \in \{1, \ldots, \lambda\}, x_{n}^i \leftarrow x_{n-1}^i + \sigma_{n-1}^i G\)  \hspace{1cm} \text{Offsprings}

5: Rank \(x_{n}^{i_1}, \ldots, x_{n}^{i_\lambda}\) such that \(f_{\text{noisy}}(x_{n}^{i_1}) \leq f_{\text{noisy}}(x_{n}^{i_2}) \leq \cdots \leq f_{\text{noisy}}(x_{n}^{i_\lambda})\)  \hspace{1cm} \text{Evaluation}

6: \(\tilde{x}_n \leftarrow x_{n}^{i_1}\)  \hspace{1cm} \text{Selection}

7: \(\sigma_n \leftarrow \sigma_{n}^{i_1}\)  \hspace{1cm} \text{Update Step-size}

8: \(n \leftarrow n + 1\)

9: \textbf{end while}

\textbf{return} \(x_n\)
Comparison-based algorithms

\[(1, \lambda) \textbf{ Self Adaptive Evolution Strategy.}\]

\textbf{Inputs:} Initial values \(\tilde{x}_0, \sigma_0, \tau > 0, \lambda \in \mathbb{N}^+\)

1: \(n \leftarrow 1\) \hspace{2cm} \text{\textbf{ iteration index}}

2: \textbf{ while } not terminate \textbf{ do}

3: \(\forall i \in \{1, \ldots, \lambda\}, \sigma_{n-1}^i \leftarrow \sigma_{n-1} \exp(\tau G)\) \hspace{1cm} \text{Mutate Step-size}

4: \(\forall i \in \{1, \ldots, \lambda\}, x_n^i \leftarrow \tilde{x}_{n-1} + \sigma_{n-1}^i G\) \hspace{1cm} \text{Offsprings}

5: Rank \(x_n^{i_1}, \ldots, x_n^{i_{\lambda}}\) such that \hspace{1cm} \text{Evaluation}

\[f_{\text{noisy}}(x_n^{i_1}) \leq f_{\text{noisy}}(x_n^{i_2}) \leq \cdots \leq f_{\text{noisy}}(x_n^{i_{\lambda}})\]

6: \(\tilde{x}_n \leftarrow x_n^{i_1}\) \hspace{1cm} \text{Selection}

7: \(\sigma_n \leftarrow \sigma_n^{i_1}\)

8: \(n \leftarrow n + 1\)

9: \textbf{ end while}

\textbf{return} \(x_n\)

Increase population variance
Comparison-based algorithms

\( (1, \lambda) \) Self Adaptive Evolution Strategy.

**Inputs:** Initial values \( \tilde{x}_0, \sigma_0, \tau > 0 \), resampling tool \( r \), \( \lambda \in \mathbb{N}^+ \)

1: \( n \leftarrow 1 \) \hfill \text{iteration index}
2: \textbf{while} not terminate \textbf{do}
3: \( \forall i \in \{1, \ldots, \lambda\}, \sigma_{n-1}^i \leftarrow \sigma_{n-1} \exp(\tau G) \) \hfill \text{Mutate Step-size}
4: \( \forall i \in \{1, \ldots, \lambda\}, x_n^i \leftarrow \tilde{x}_{n-1} + \sigma_{n-1}^i G \) \hfill \text{Offsprings}
5: \text{Rank } x_n^{i_1}, \ldots, x_n^{i_\lambda} \text{ such that}
6: \( \tilde{x}_n \leftarrow x_n^{i_1} \) \hfill \text{Resampling}
7: \( \sigma_n \leftarrow \sigma_n^{i_1} \) \hfill \text{Selection}
8: \( n \leftarrow n + 1 \) \hfill \text{Update Step-size}
9: \textbf{end while}
10: \textbf{return } x_n
Comparison-based algorithms

### Resampling for ES

[Astete-Morales et al., 2014] on the functions $f : x \mapsto \|x - x^*\|^p$:

- **theoretically**, $r(n) \sim \exp(n)$ or $\frac{1}{\sigma_n} \Rightarrow \mathbb{E}SR_n = O\left(\frac{1}{n^{\alpha}}\right)$, $\alpha = ?$
- **experimentally**, $r(n) \sim \text{poly}(n)$ or $\frac{1}{\sigma_n} \Rightarrow \mathbb{E}SR_n = O\left(\frac{1}{n^{\alpha}}\right)$.

### A lower bound for a large class of CBA

(Astete-Morales, Cauwet, and Teytaud [2015]) on quadratic functions:

**Simple** CBA are **slower** than value-based algorithms $\Rightarrow \alpha \leq 1/2$.

### CBA can be fast!

- [Cauwet and Teytaud, 2016] on quadratic functions $\Rightarrow \alpha = 1$.
- [Hellwig and Beyer, 2016] experimentally, ES $\Rightarrow \alpha = 1$. 
Introduction to Optimization Algorithms

1. Introductory Examples

2. Framework

3. Different Families of Optimization Algorithms
   - Value-based Algorithms
   - Comparison-based Algorithms
   - Experimentations

4. Portfolio of Noisy Optimization Algorithms

5. Conclusion

6. References
Comparison of the rate of convergence of different algorithms.

- •: comparison-based algorithms;
- •: value-based algorithms.
1 Introductory Examples
2 Framework
3 Different Families of Optimization Algorithms
   Value-based Algorithms
   Comparison-based Algorithms
   Experimentations
4 Portfolio of Noisy Optimization Algorithms
5 Conclusion
6 References
Motivations

- Select a Noisy Optimization Algorithm?
- Parameters tuning?
- Counterbalance ‘bad luck’?
**Usually**: Portfolio of Algorithms in **Combinatorial Optimization**.

**New**: Portfolio of Algorithms in **Noisy Optimization**.

**Challenges**

- Stochastic problem;
- Limited budget (the total number of evaluations): how to distribute it?
- Sharing information?
Noisy Optimization Portfolio Algorithm (NOPA)

Input:
- noisy optimization solvers $Solver_1, Solver_2, \ldots, Solver_M$
- function $\text{lag} : \mathbb{N}^* \mapsto \mathbb{N}^*$
- non-decreasing integer sequence $r_1, r_2, \ldots$
- a non-decreasing integer sequence $s_1, s_2, \ldots$

Output:
- approximation $\tilde{x}$ of the optimum $x^*$ of the objective function.

1: $n \leftarrow 1$  
2: $m \leftarrow 1$  
3: $i^* \leftarrow \text{null}$  
4: $\tilde{x} \leftarrow \text{null}$  
5: while not finished do  
6: \hspace{1em} if $m \geq r_n$ then  
7: \hspace{2em} $i^* = \arg\min_{i \in \{1, \ldots, M\}} \hat{E}_{s_n}[f_{\text{noisy}}(\tilde{x}_{i^*, \text{lag}(r_n)})]$  
8: \hspace{2em} $n \leftarrow n + 1$  
9: \hspace{1em} else  
10: \hspace{2em} for $i \in \{1, \ldots, M\}$ do  
11: \hspace{3em} Apply one evaluation for $Solver_i$  
12: \hspace{2em} end for  
13: \hspace{2em} $m \leftarrow m + 1$  
14: \hspace{1em} end if  
15: $\tilde{x} = \tilde{x}_{i^*, m}$  
16: end while  
return $\tilde{x}$

“Orthogonal”  
Periodic comparisons  
Number of resamplings  
Number of selections  
NOPA’s iteration number  
Index of recommended solver  
Recommendation  
Algorithm selection  
Update recommendation
Noisy Optimization Portfolio Algorithm (NOPA)

Input:
- noisy optimization solvers \( \text{Solver}_1, \text{Solver}_2, \ldots, \text{Solver}_M \)
- function \( \text{lag} : \mathbb{N}^* \rightarrow \mathbb{N}^* \)
- non-decreasing integer sequence \( r_1, r_2, \ldots \)
- a non-decreasing integer sequence \( s_1, s_2, \ldots \)

Output:
- approximation \( \tilde{x} \) of the optimum \( x^* \) of the objective function.

1: \( n \leftarrow 1 \)  
2: \( m \leftarrow 1 \)  
3: \( i^* \leftarrow \text{null} \)  
4: \( \tilde{x} \leftarrow \text{null} \)  
5: while not finished do  
6: \hspace{1em} if \( m \geq r_n \) then  
7: \hspace{2em} \( i^* = \arg\min_{i \in \{1, \ldots, M\}} \mathbb{E}_{s_n}[f_{\text{noisy}}(\tilde{x}_{i^*}, \text{lag}(r_n))] \)  
8: \hspace{2em} \( n \leftarrow n + 1 \)  
9: \hspace{1em} else  
10: \hspace{2em} for \( i \in \{1, \ldots, M\} \) do  
11: \hspace{3em} Apply one evaluation for \( \text{Solver}_i \)  
12: \hspace{2em} end for  
13: \hspace{2em} \( m \leftarrow m + 1 \)  
14: \hspace{1em} end if  
15: \hspace{1em} \( \tilde{x} = \tilde{x}_{i^*}, m \)  
16: end while  
return \( \tilde{x} \)
Compare Solver Early: Why this Lag?

- Comparing ‘good’ points: very expensive;

- Algorithms’ ranking is usually stable: no use comparing the very last.
Cauwet, Liu, Rozière, and Teytaud [2016]

- Performs almost as well as the best solver.
- The lag is necessary.
- The comparison budget increases polynomially, slower than the portfolio budget.
Experimentation on Cart-Pole: Liu and Teytaud [2014]

<table>
<thead>
<tr>
<th>Solver</th>
<th>2 neurons, $d = 9$</th>
<th>4 neurons, $d = 17$</th>
<th>8 neurons, $d = 33$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (RSAES)</td>
<td>-0.458033 ± 0.045014</td>
<td>-0.421535 ± 0.045643</td>
<td>-0.351726 ± 0.051705</td>
</tr>
<tr>
<td>2 (Fabian1)</td>
<td>0.002226 ± 5.29923e-05</td>
<td>0.002089 ± 1.57766e-04</td>
<td>0.00221 ± 8.14518e-05</td>
</tr>
<tr>
<td>3 (Fabian2)</td>
<td>0.002318 ± 9.80792e-05</td>
<td>0.002238 ± 1.14289e-04</td>
<td>0.00236 ± 1.51244e-04</td>
</tr>
<tr>
<td>4 (Newton1)</td>
<td>0.002229 ± 6.08973e-05</td>
<td>-0.030731 ± 0.111294</td>
<td>0.002247 ± 1.19829e-04</td>
</tr>
<tr>
<td>5 (Newton2)</td>
<td>0.00227 ± 5.2989e-05</td>
<td>0.002217 ± 7.80888e-05</td>
<td>0.002307 ± 9.96404e-05</td>
</tr>
<tr>
<td>6 (P.12345)</td>
<td>-0.408705 ± 0.068428</td>
<td>-0.3917 ± 0.071791</td>
<td>-0.320399 ± 0.050338</td>
</tr>
<tr>
<td>7 (P.12345 + S.)</td>
<td>-0.42743 ± 0.05709</td>
<td>-0.403707 ± 0.056173</td>
<td>-0.354043 ± 0.069576</td>
</tr>
</tbody>
</table>

Rate of convergence ($\alpha$) for Cart-Pole, a multimodal problem, using Neural Network.

Results

- Fabian’s algorithm and Newton’s algorithm are not able to solve multimodal problem;
- The problem is easily solved by NOPA, because one solver is much better than the others - this makes comparisons easy.
Summary

- Main conclusion:
  - portfolios are classical in combinatorial optimization; (because in C.O. differences between runtimes can be huge);
  - portfolios also make a big difference in noisy optimization; (because in N.O., with lag, comparison cost = small).

(+) A portfolio of solvers = approximately as efficient as the best.

(+) Unfair budget distribution

(−) Sharing not always that good.
Perspectives

- Identifying relevant information for sharing.

- Orthogonality?

- If solver 1 says “I’ll never do better than X” and solver 2 says “I have found at least Y > X” then we can stop 1.
1 Introductory Examples

2 Framework

3 Different Families of Optimization Algorithms
   Value-based Algorithms
   Comparison-based Algorithms
   Experimentations

4 Portfolio of Noisy Optimization Algorithms

5 Conclusion

6 References
Conclusion

- Value-based and Comparison-based Algorithm are competitive;

- Key point: sampling far from the optimum;

- Selection tool.
Take-home message

Noise
Many optimization problem are impacted by noise: in industry, in games, in power systems, ...

If you are lucky ...
Identifying some structure on your optimization problem:
- on the objective function (convexity, ...);
- on the domain (convexity, ...);
- on the stochastic process (independent realization, Markovian process, ...);
- ...
Take-home message

Noise
Many optimization problems are impacted by noise: in industry, in games, ...

Identifying some structure in your optimization problem:
▶ on the objective function (convexity, ...);
▶ on the domain (convexity, ...);
▶ on the stochastic process (independent realization, Markovian process, ...);
▶ ...

Errors
Error in the model vs. error in the optimization process.
Take-home message

Noise
Many optimization problem are impacted by noise: in industry, in games, in power systems, ...

If you are lucky ...
Identifying some structure on your optimization problem:
  ► on the objective function (convexity, ...);
  ► on the domain (convexity, ...);
  ► on the stochastic process (independent realization, Markovian process, ...);
  ► ...

... otherwise
Consider some more generic optimization processes, they are not that bad ...


Jialin Liu and Olivier Teytaud. Meta online learning: experiments on a unit commitment problem. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, Bruges, Belgium, April 2014. URL https://hal.inria.fr/hal-00973397.
