L20 - The No Free Lunch Theorem

Nature Inspired Search and Optimisation

Per Kristian Lehre

March 15, 2018

Outline

Black Box Optimisation[1]

Function class F

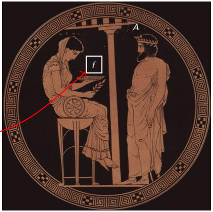


Photo: E. Gerhard (1846).

[Droste et al., 2006]

[1]We assume maximisation of functions.

Black Box Optimisation[1]

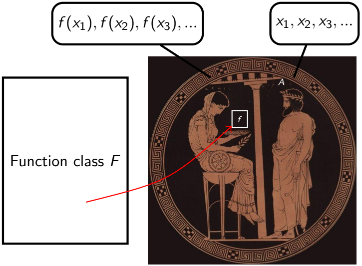


Photo: E. Gerhard (1846).

[Droste et al., 2006]

[1]We assume maximisation of functions.

Black Box Optimisation[1]

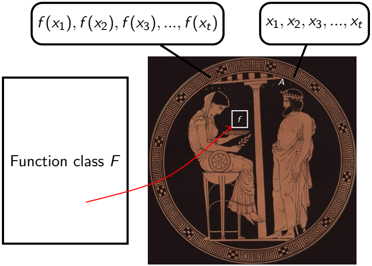


Photo: E. Gerhard (1846).

[Droste et al., 2006]

- **Runtime** of algorithm A on f

$$T_{A,f} := \min_{t \in \mathbb{N}} \{t \mid \forall y\ f(x_t) \geq f(y)\}$$

- **Average case** expected runtime

$$\mathbf{E}\left[T_{A,F}\right] := \sum_{f \in F} \mathbf{Pr}\left[f\right] \mathbf{E}\left[T_{A,f}\right]$$

[1]We assume maximisation of functions.

Black Box Optimisation Algorithms

Black Box Optimisation Algorithm

Choose some probability distribution p_0 on X.
Sample a search point $x_0 \in X$ according to p_0.
$R_0 := \{x_0\}$
Evaluate the fitness $f(x_0)$
for $t = 1$ **to** $|X|$ **do**
    $H_t := (x_0, f(x_0)), \ldots, (x_{t-1}, f(x_{t-1}))$
    Given H_t, choose a prob. distr. p_t on $X \setminus R_{t-1}$.
    Sample a search point $x_t \in X \setminus R_{t-1}$ according to p_t
    $R_t := \{x_0, \ldots, x_t\}$
    Evaluate the fitness $f(x_t)$

The No Free Lunch Theorem

Theorem ([Wolpert and Macready, 1997])

Let X and $Y \subset \mathbb{R}$ be any finite sets, and let F be the set of all functions $f : X \to Y$.

Then for any black box optimisation algorithms A and B,

$$\mathbf{E}\left[T_{A,F}\right] = \mathbf{E}\left[T_{B,F}\right].$$

The average case runtime over F is the same for all black box optimisation algorithms. (Multiple evaluations of same search point counted once.)

The Generalized No Free Lunch Theorem

Theorem ([Wolpert and Macready, 1997])

Let X and $Y \subset \mathbb{R}$ be any finite set, and let F be any set of functions $f : X \to Y$ which is closed under permutation.

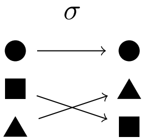
Then for any black box optimisation algorithms A and B

$$\mathbf{E}\left[T_{A,F}\right] = \mathbf{E}\left[T_{B,F}\right].$$

Class of functions *closed under permutations* (c.u.p.)

Definition (F c.u.p.)

If function f is in class F,
then for any permutation σ,
function $f \circ \sigma$ is also in F.

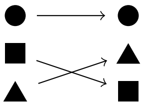


Class of functions *closed under permutations* (c.u.p.)

Definition (F c.u.p.)

If function f is in class F,
then for any permutation σ,
function $f \circ \sigma$ is also in F.

σ



Example

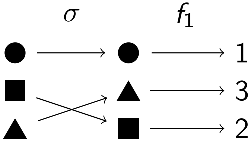
x	$f_1(x)$
●	1
■	2
▲	3

The class $F = \{f_1\}$ is **not** closed under permutation.

Class of functions *closed under permutations* (c.u.p.)

Definition (F c.u.p.)

If function f is in class F,
then for any permutation σ,
function $f \circ \sigma$ is also in F.



Example

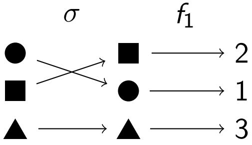
x	$f_1(x)$	$f_2(x)$
●	1	1
■	2	3
▲	3	2

The class $F = \{f_1, f_2\}$ is **not** closed under permutation.

Class of functions *closed under permutations* (c.u.p.)

Definition (F c.u.p.)

If function f is in class F,
then for any permutation σ,
function $f \circ \sigma$ is also in F.



Example

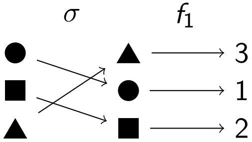
x	$f_1(x)$	$f_2(x)$	$f_3(x)$
●	1	1	2
■	2	3	1
▲	3	2	3

The class $F = \{f_1, f_2, f_3\}$ is **not** closed under permutation.

Class of functions *closed under permutations* (c.u.p.)

Definition (F c.u.p.)

If function f is in class F,
then for any permutation σ,
function $f \circ \sigma$ is also in F.



Example

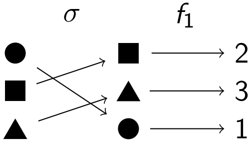
x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$
●	1	1	2	3
■	2	3	1	1
▲	3	2	3	2

The class $F = \{f_1, f_2, f_3, f_4\}$ is **not** closed under permutation.

Class of functions *closed under permutations* (c.u.p.)

Definition (F c.u.p.)

If function f is in class F,
then for any permutation σ,
function $f \circ \sigma$ is also in F.



Example

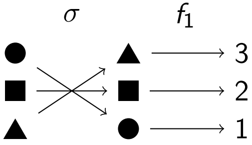
x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$
●	1	1	2	3	2
■	2	3	1	1	3
▲	3	2	3	2	1

The class $F = \{f_1, f_2, f_3, f_4, f_5\}$ is **not** closed under permutation.

Class of functions *closed under permutations* (c.u.p.)

Definition (F c.u.p.)

If function f is in class F,
then for any permutation σ,
function $f \circ \sigma$ is also in F.



Example

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
●	1	1	2	3	2	3
■	2	3	1	1	3	2
▲	3	2	3	2	1	1

The class $F = \{f_1, f_2, f_3, f_4, f_5, f_6\}$ is closed under permutation.

The Generalized No Free Lunch Theorem

Theorem ([Wolpert and Macready, 1997])

Let X and $Y \subset \mathbb{R}$ be any finite set, and let F be any set of functions $f : X \rightarrow Y$ which is closed under permutation.

Then for any black box optimisation algorithms A and B

$$\mathbf{E}\left[T_{A,F}\right] = \mathbf{E}\left[T_{B,F}\right].$$

NFL Proof Idea - 1

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
●	1	1	2	3	2	3
■	2	3	1	1	3	2
▲	3	2	3	2	1	1

NFL Proof Idea - 1

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
●	1	1	2	3	2	3
■	2	3	1	1	3	2
▲	3	2	3	2	1	1

$\boxed{f_1}$

- The adversary selects a function from the class, e.g., f_1.

NFL Proof Idea - 1

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
●	1	1	2	3	2	3
■	2	3	1	1	3	2
▲	3	2	3	2	1	1

$\boxed{f_1}$

- The adversary selects a function from the class, e.g., f_1.
- Alg. knows fitness function is either f_1, or f_2, or ... or f_6.

NFL Proof Idea - 1

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
●	1	1	2	3	2	3
■	2	3	1	1	3	2
▲	3	2	3	2	1	1

$\boxed{f_1}$

- The adversary selects a function from the class, e.g., f_1.
- Alg. knows fitness function is either f_1, or f_2, or ... or f_6.
- Alg. asks for the function value of ● and gets $f(\bullet) = 1$.
  - Only functions f_1 and f_2 consistent with $f(\bullet) = 1$.

NFL Proof Idea - 1

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
●	1	1	2	3	2	3
■	2	3	1	1	3	2
▲	3	2	3	2	1	1

$\boxed{f_1}$

- The adversary selects a function from the class, e.g., f_1.
- Alg. knows fitness function is either f_1, or f_2, or ... or f_6.
- Alg. asks for the function value of ● and gets $f(●) = 1$.
  - Only functions f_1 and f_2 consistent with $f(●) = 1$.
- Problem reduced to function class $F(●, 1)$.

NFL Proof Idea - 2

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
●	1	1	2	3	2	3
■	2	3	1	1	3	2
▲	3	2	3	2	1	1

After revealing that $f(x) = b$, the problem is reduced to

$$F(x, b) := \{f \in F \mid f(x) = b\}, \tag{1}$$

i.e., the subset of functions f consistent with $f(x) = b$.

The following two claims can be proved (which we do not do here)

- The class $F(x, b)$ is closed under permutations.

NFL Proof Idea - 2

x	$f_1(x)$	$f_2(x)$	$f_3(x)$	$f_4(x)$	$f_5(x)$	$f_6(x)$
●	1	1	2	3	2	3
■	2	3	1	1	3	2
▲	3	2	3	2	1	1

After revealing that $f(x) = b$, the problem is reduced to

$$F(x, b) := \{f \in F \mid f(x) = b\}, \tag{1}$$

i.e., the subset of functions f consistent with $f(x) = b$.

The following two claims can be proved (which we do not do here)

- The class $F(x, b)$ is closed under permutations.
- The classes $F(x, b)$ and $F(y, b)$ are isomorphic.

NFL Proof 1/2 - Deterministic Black Box Algorithms

- Proof by induction over the size of the search space X.
  - Step 1: Show that NFL holds when $|X| = 1$.
  - Step 2: Assume that NFL holds when $|X| = N$.
    Show that it also holds when $|X| = N + 1$.
- Consider any two black box algorithms A and B
  - Assume A chooses x as first search point.
  - Assume B chooses y as first search point.

NFL Proof 1/2 - Deterministic Black Box Algorithms

- Proof by induction over the size of the search space X.
  - Step 1: Show that NFL holds when $|X| = 1$.
  - Step 2: Assume that NFL holds when $|X| = N$.
    Show that it also holds when $|X| = N + 1$.
- Consider any two black box algorithms A and B
  - Assume A chooses x as first search point.
  - Assume B chooses y as first search point.

Step 1: Search space X contains only one search point.

- A and B find the optimum in first iteration.
- So NFL trivially holds when $|X| = 1$.

NFL Proof 1/2 - Deterministic Black Box Algorithms

Step 2: Assume NFL holds when $|X| = N$,
and that $\max_x f(x) = b^*$.

$$\mathbf{E}\left[T_{A,F}\right] = \mathbf{Pr}\left[f(x) = b^*\right] \cdot 1 + \sum_{b \neq b^*} \mathbf{Pr}\left[f(x) = b\right] \cdot \left(1 + \mathbf{E}\left[T_{A,F(x,b)}\right]\right).$$

$$\mathbf{E}\left[T_{B,F}\right] = \mathbf{Pr}\left[f(y) = b^*\right] \cdot 1 + \sum_{b \neq b^*} \mathbf{Pr}\left[f(y) = b\right] \cdot \left(1 + \mathbf{E}\left[T_{B,F(y,b)}\right]\right).$$

NFL Proof 1/2 - Deterministic Black Box Algorithms

Step 2: Assume NFL holds when $|X| = N$,
and that $\max_x f(x) = b^*$.

$$\mathbf{E}\left[T_{A,F}\right] = \mathbf{Pr}\left[f(x) = b^*\right] \cdot 1 + \sum_{b \neq b^*} \mathbf{Pr}\left[f(x) = b\right] \cdot (1 + \mathbf{E}\left[T_{A,F(x,b)}\right]).$$

$$\mathbf{E}\left[T_{B,F}\right] = \mathbf{Pr}\left[f(y) = b^*\right] \cdot 1 + \sum_{b \neq b^*} \mathbf{Pr}\left[f(y) = b\right] \cdot (1 + \mathbf{E}\left[T_{B,F(y,b)}\right]).$$

- $F(x, b)$ and $F(y, b)$ closed under permutations (by Claim 1.)

NFL Proof 1/2 - Deterministic Black Box Algorithms

Step 2: Assume NFL holds when $|X| = N$,
and that $\max_x f(x) = b^*$.

$$\mathbf{E}\left[T_{A,F}\right] = \mathbf{Pr}\left[f(x) = b^*\right] \cdot 1 + \sum_{b \neq b^*} \mathbf{Pr}\left[f(x) = b\right] \cdot \left(1 + \mathbf{E}\left[T_{A,F(x,b)}\right]\right).$$

$$\mathbf{E}\left[T_{B,F}\right] = \mathbf{Pr}\left[f(y) = b^*\right] \cdot 1 + \sum_{b \neq b^*} \mathbf{Pr}\left[f(y) = b\right] \cdot \left(1 + \mathbf{E}\left[T_{B,F(y,b)}\right]\right).$$

- $F(x, b)$ and $F(y, b)$ closed under permutations (by Claim 1.)
- $F(x, b)$ and $F(y, b)$ are the same problem. (Claim 2.)

NFL Proof 1/2 - Deterministic Black Box Algorithms

Step 2: Assume NFL holds when $|X| = N$,
and that $\max_x f(x) = b^*$.

$$\mathbf{E}\left[T_{A,F}\right] = \mathbf{Pr}\left[f(x) = b^*\right] \cdot 1 + \sum_{b \neq b^*} \mathbf{Pr}\left[f(x) = b\right] \cdot (1 + \mathbf{E}\left[T_{A,F(x,b)}\right]).$$

$$\mathbf{E}\left[T_{B,F}\right] = \mathbf{Pr}\left[f(y) = b^*\right] \cdot 1 + \sum_{b \neq b^*} \mathbf{Pr}\left[f(y) = b\right] \cdot (1 + \mathbf{E}\left[T_{B,F(y,b)}\right]).$$

- $F(x, b)$ and $F(y, b)$ closed under permutations (by Claim 1.)
- $F(x, b)$ and $F(y, b)$ are the same problem. (Claim 2.)
- $\mathbf{E}\left[T_{A,F(x,b)}\right] = \mathbf{E}\left[T_{B,F(y,b)}\right]$. (Induction hypothesis.)

NFL Proof 1/2 - Deterministic Black Box Algorithms

Step 2: Assume NFL holds when $|X| = N$, and that $\max_x f(x) = b^*$.

$$\mathbf{E}[T_{A,F}] = \mathbf{Pr}[f(x) = b^*] \cdot 1 + \sum_{b \neq b^*} \mathbf{Pr}[f(x) = b] \cdot (1 + \mathbf{E}\left[T_{A,F(x,b)}\right]).$$

$$\mathbf{E}[T_{B,F}] = \mathbf{Pr}[f(y) = b^*] \cdot 1 + \sum_{b \neq b^*} \mathbf{Pr}[f(y) = b] \cdot (1 + \mathbf{E}\left[T_{B,F(y,b)}\right]).$$

- $F(x, b)$ and $F(y, b)$ closed under permutations (by Claim 1.)
- $F(x, b)$ and $F(y, b)$ are the same problem. (Claim 2.)
- $\mathbf{E}\left[T_{A,F(x,b)}\right] = \mathbf{E}\left[T_{B,F(y,b)}\right]$. (Induction hypothesis.)

By the induction principle, we can conclude that

$$\Longrightarrow \mathbf{E}[T_{A,F}] = \mathbf{E}[T_{B,F}] = \mathbf{E}[T_F],$$

i.e., the average runtime is independent of the algorithm.

NFL Proof 2/2 - Randomised Black Box Algorithms

There is a finite number of deterministic algorithms $A_1, A_2, ..., A_m$, because we assumed a deterministic search space X.

Consider any randomised algorithm A

- A makes some decisions by tossing a coin
- A can make all coin tosses first, then proceed "determinstically" according to the outcomes of coin tosses
- i.e. A chooses probabilistically a deterministic algorithm A_i

NFL Proof 2/2 - Randomised Black Box Algorithms

There is a finite number of deterministic algorithms $A_1, A_2, ..., A_m$, because we assumed a deterministic search space X.

Consider any randomised algorithm A

- A makes some decisions by tossing a coin
- A can make all coin tosses first, then proceed "determinstically" according to the outcomes of coin tosses
- i.e. A chooses probabilistically a deterministic algorithm A_i

$$
\begin{aligned}
\mathbf{E}\left[T_{A,F}\right] &= \sum_{i,j} \mathbf{Pr}\left[A \text{ uses } A_i \cap \text{adversary chooses } f_j\right] \cdot \mathbf{E}\left[T_{A_i,f_j}\right] \\
&= \sum_{i} \mathbf{Pr}\left[A \text{ uses } A_i\right] \cdot \sum_{j} \mathbf{Pr}\left[\text{adversary chooses } f_j\right] \cdot \mathbf{E}\left[T_{A_i,f_j}\right] \\
&= \sum_{i} \mathbf{Pr}\left[A \text{ uses } A_i\right] \cdot \mathbf{E}\left[T_{A_i,F}\right] \\
&= \mathbf{E}\left[T_F\right]
\end{aligned}
$$

How realistic is the NFL scenario?

Assume the class F of fitness functions $f : A \to B$, where

- A is the set of bitstrings of length 100.
- B is the set of integers represented by 32 bits.

How many different fitness functions are there in this class?

$$|F| = |B|^{|A|} = \left(2^{32}\right)^{2^{100}}.$$

How large are the programs that implement these functions?

How realistic is the NFL scenario?

Assume the class F of fitness functions $f : A \to B$, where

- A is the set of bitstrings of length 100.
- B is the set of integers represented by 32 bits.

How many different fitness functions are there in this class?

$$|F| = |B|^{|A|} = \left(2^{32}\right)^{2^{100}}.$$

How large are the programs that implement these functions?

By a simple counting argument, the length of at least half of the fitness functions must have shortest programs of length at least

$$\log |F| - 1 \text{ bits} \geq 32 \cdot 2^{100} - 1 \text{ bits} \geq 10^{20} \text{ terabytes.}$$

$\Longrightarrow$ Conditions in NFL theorem rarely hold in practice.

Almost No Free Lunch

Theorem ([Droste et al., 2002])

Given any black-box optimisation algorithm A and function

$$f : \{0,1\}^n \rightarrow \{0,1,...,N-1\}.$$

There exist at least $N^{2^{n/3}-1}$ functions

$$f^* : \{0,1\}^n \rightarrow \{0,1,...,N\}$$

which agree with f on all but at most $2^{n/3}$ inputs such that

- *A does find the optimum of f^* within $2^{n/3}$ steps with a probability bounded above by $2^{-n/3}$.*
- *Exponentially many of these functions have the additional property that their evaluation time, circuit size representation, and Kolmogorov complexity is only by an additive term of $O(n)$ larger than the corresponding complexity of f.*

NFL - Conclusion

- No single best search heuristic on **all** problems.
- In *design* and *analysis* of search heuristics, it is necessary to consider function classes that are not *c.u.p.*
  - assume a certain type of "fitness landscape"
  - assume a certain type of "structural" property.
- Runtime differences still possible on subclasses.
  - F *c.u.p.*, $F_1 \cup F_2 = F$ and $F_1 \cap F_2 = \emptyset$.
  - A outperforms B on $F_1 \implies B$ outperforms A on F_2.
- NFL conditions will not occur in practice.
- Almost NFL in restricted scenario
  - Small modifications to an easy function can make it hard.

References I

Droste, S., Jansen, T., and Wegener, I. (2002).
Optimization with randomized search heuristics–the (a)nfl theorem, realistic scenarios, and difficult functions.
Theoretical Computer Science, 287(1):131–144.

Droste, S., Jansen, T., and Wegener, I. (2006).
Upper and lower bounds for randomized search heuristics in black-box optimization.
Theory of Computing Systems, 39(4):525–544.

Wolpert, D. H. and Macready, W. G. (1997).
No free lunch theorems for optimization.
IEEE Transactions on Evolutionary Computation, 1(1):67–82.