

CS4618 Artificial Intelligence I

Today: Assessment of
Randomised Search Heuristics:
NFL and Beyond

Thomas Jansen

November 2nd

Plans for Today

- 1 Assessing Randomised Search Heuristics
No Free Lunch
- 2 NFL Discussion
Function Classes with NFL
Almost No Free Lunch
- 3 Summary
Summary & Take Home Message

Dealing with the NFL

Remember

NFL Theorem (1996)

For any finite S and R , on average, all black-box algorithms for $\mathcal{F} = R^S$ make an equal number of distinct f evaluations until an arbitrary optimisation goal is reached.

Now

- 1 formalising deterministic black-box algorithms ✓
- 2 formalising optimisation goals ✓
- 3 proof for deterministic search heuristics
- 4 proof for randomised search heuristics
- 5 putting the NFL into perspective (part 1)
- 6 proof of a more general NFL
- 7 putting the NFL into perspective (part 2)

Proving the NFL — Part 1

Remember notation

- deterministic BBA $\hat{=}$ tree
- trace $T_A(f, t) = \langle (s_1, f(s_1)), (s_2, f(s_2)), \dots, (s_t, f(s_t)) \rangle$
- value vectors $V_A(f, t) = (f(s_1), f(s_2), \dots, f(s_t))$

Now proof for deterministic black-box algorithms

Need to prove For all deterministic black-box algorithms A, A' and all optimisation goals $M: \{V_A(f, t) \mid A, f, t\} \rightarrow \mathbb{R}$

$$\frac{\sum_{f \in R^S} M(V_A(f, |S|))}{|R^S|} = \frac{\sum_{f \in R^S} M(V_{A'}(f, |S|))}{|R^S|}$$

Sub-Claim 1 $\forall \text{det. BBA } A, f, g \in R^S:$

$$V_A(f, |S|) = V_A(g, |S|) \Rightarrow f = g \quad \checkmark$$

Another Sub-Claim

Remember

Sub-Claim 1 $\forall \text{det. BBA } A, f, g \in R^S :$
 $V_A(f, |S|) = V_A(g, |S|) \Rightarrow f = g$

Sub-Claim 2 $\forall \text{det. BBA } A: \left| \left\{ V_A(f, |S|) \mid f \in R^S \right\} \right| = \left| R^S \right|$

Proof

Observation $\forall A: \left| \left\{ V_A(f, |S|) \mid f \in R^S \right\} \right| \leq \left| R^S \right|$
 since only f not fixed on left-hand side

Observation $\forall A: \left| \left\{ V_A(f, |S|) \mid f \in R^S \right\} \right| \geq \left| R^S \right|$
 since $f \neq g \Rightarrow V_A(f, |S|) \neq V_A(g, |S|)$ (Sub-Claim 1)

□

Combining What We have

Observation $V_A(f, |S|)$ is vector of function values of length $|S|$

Observation vector of function values of length $|S|$
defines $f: S \rightarrow R$

Thus there are $|R^S|$ different $V_A(f, |S|)$

Therefore \forall det. BBA A, A' :

$$\{V_A(f, |S|) \mid f \in R^S\} = \{V_{A'}(f, |S|) \mid f \in R^S\}$$

Thus
$$\frac{\sum_{f \in R^S} M(V_A(f, |S|))}{|R^S|} = \frac{\sum_{f \in R^S} M(V_{A'}(f, |S|))}{|R^S|}$$

for all M ✓

Observation for deterministic BBA **proof complete** ✓

Randomised Black-Box Algorithms

Observation S, R finite
 \Rightarrow number of different deterministic BBA finite

How can we represent randomised BBA similar to det. BBA?

Idea describe randomised BBA as running in two phases

- ① Randomly decide all random bits.
- ② Compute deterministically
 (looking up the pre-computed random bits).

Observation randomised BBA
 $\hat{=}$ probability distribution over det. BBA

Thus performance of randomised BBA
 $\hat{=}$ weighted sum of performances of det. BBA

Thus performance of all det. BBA equal
 \Rightarrow performance of all randomised BBA equal ✓ □

Discussing the NFL of 1996

NFL Averaged over all functions...

- ... all algorithms perform equally.
- ... no algorithm is better than pure random search.
- hill-descending leads on the top of the highest hill just as quickly as hill-climbing does.

Is this surprising? may be

Should we care? not really

Obvious Nobody optimises all functions.
 Nobody selects objective functions uniformly at random.

Example $R^S = \{f: \{0, 1\}^{40} \rightarrow \{0, 1, \dots, 255\}\}$

consider implementation in Java
 with source code of size $\leq 1\text{GB}$

↪ fraction of implementable functions $< 10^{-2.199.023.256.000\%}$

The NFL of 1996 and Beyond

We know The NFL holding for $\mathcal{F} = R^S$ is **not interesting** since we do not care about all functions.

Does the NFL hold for arbitrary \mathcal{F} ?

Observation **No!**

Consider $\mathcal{F} = \{f\}$.

Clearly algorithms starting with some $s \in S$ with $f(s) = \max\{f(x) \mid x \in S\}$ are **better** than others

Does the NFL hold only for $\mathcal{F} = R^S$?

Unfortunately **No!**

A Sharper NFL

Notation Consider $\sigma \in \text{Perm}(S)$, $f: S \rightarrow R$.
 $\sigma f: S \rightarrow R$ is $\sigma f(s) = f(\sigma^{-1}(s))$ for all $s \in S$

$\mathcal{F} \subseteq R^S$ is called

closed under permutations of the search space (c. u. p.)

$\Leftrightarrow \forall f \in \mathcal{F}, \sigma \in \text{Perm}(S): \sigma f \in \mathcal{F}$

Theorem (NFL, 2001)

For any finite S and R , on average, all black-box algorithms for $\mathcal{F} \subseteq R^S$ make an equal number of distinct f evaluations until an arbitrary optimisation goal is reached if and only if \mathcal{F} is c. u. p.

Proof Strategy

- ① proof for deterministic BBA suffices ✓
- ② \mathcal{F} c. u. p. \Rightarrow NFL for all M
- ③ \mathcal{F} not c. u. p. $\Rightarrow \exists M: \text{no NFL}$

② \mathcal{F} c. u. p. \Rightarrow NFL for all M

Let A, A' det. BBA.

We prove $\{V_A(f, |S|) \mid f \in \mathcal{F}\} = \{V_{A'}(f, |S|) \mid f \in \mathcal{F}\}$

To this end $\forall f \in \mathcal{F}: \exists f' \in \mathcal{F}: V_A(f, |S|) = V_{A'}(f', |S|)$

Consider For given $A, A', f \in \mathcal{F}$

$$T_A(f, |S|) = \langle (s_1, f(s_1)), (s_2, f(s_2)), \dots, (s_{|S|}, f(s_{|S|})) \rangle$$

Let s'_1 label of root in A'

$$\sigma(s_1) := s'_1$$

Observation $\sigma f(s'_1) = f(\sigma^{-1}(s'_1)) = f(s_1)$ ✓

Let s'_{i+1} label of $f(s_i)$ -successor of $\sigma(s_i)$ in A'

$$\sigma(s_{i+1}) := s'_{i+1}$$

Observation A' on σf like A on f ✓

Observation $\sigma f \in \mathcal{F}$ since \mathcal{F} c. u. p. ✓

③ \mathcal{F} not c. u. p. $\Rightarrow \exists M$: no NFL

Consider $f \in \mathcal{F}$, $\sigma \in \text{Perm}(S)$ with $\sigma f \notin \mathcal{F}$

Need to prove \exists optimisation goal M , det. BBA A , A' :
 A and A' different under M

Consider arbitrary det. BBA A for \mathcal{F}

Let $M(V) := \begin{cases} 1 & \text{if } V = V_A(f, |S|) \\ 0 & \text{otherwise} \end{cases}$

Observation $\left(\sum_{g \in \mathcal{F}} M(V_A(g, |S|)) \right) / |\mathcal{F}| = 1 / |\mathcal{F}|$

Define A' by considering $T_A(f, |S|) = \langle (s_1, f(s_1)), \dots, (s_{|S|}, f(s_{|S|})) \rangle$
 Let A' enumerate S in the ordering $\sigma(s_1), \sigma(s_2), \dots, \sigma(s_{|S|})$

Observation $\sigma f(\sigma(s_i)) = f(\sigma^{-1}(\sigma(s_i))) = f(s_i)$

Thus $T_{A'}(\sigma f, |S|) = T_A(f, |S|)$

$\left(\sum_{g \in \mathcal{F}} M(V_{A'}(g, |S|)) \right) / |\mathcal{F}| = 0$, since $\sigma f \notin \mathcal{F}$ \square

Function Classes Without NFL

Remember sharpened **NFL Theorem** (2001)
 For any finite S and R , on average, all black-box algorithms for $\mathcal{F} \subseteq R^S$ make an equal number of distinct f evaluations until an arbitrary optimisation goal is reached if and only if \mathcal{F} is c. u. p.

Are many classes of functions c. u. p.?

Theorem

For any finite S , R , the fraction of non-empty classes of functions $\mathcal{F} \subseteq R^S$ that are c. u. p. equals

$$\left(2^{\binom{|S|+|R|-1}{|S|}} - 1 \right) / \left(2^{|R|^{|S|}} - 1 \right).$$

The Number of c. u. p. Classes

Theorem

For any finite S , R , the fraction of non-empty classes of functions $\mathcal{F} \subseteq R^S$ that are c. u. p. equals

$$\left(2^{\binom{|S|+|R|-1}{|S|}} - 1 \right) / \left(2^{|R|^{|S|}} - 1 \right).$$

Proof

Obvious number of non-empty function classes = $2^{|R|^{|S|}} - 1$ ✓

Open number of non-empty function classes that are c. u. p.


Histograms

Define for $f: S \rightarrow R$
histogram $h_f: R \rightarrow \mathbb{N}_0$ by $h_f(r) = |\{s \in S \mid f(s) = r\}|$

Define $f \sim g$ by $h_f = h_g$

Observation \sim is **equivalence relation**
 reflexive, symmetric, transitive ✓

Define **base class** $B_f = [f]$ for \sim

Observation number of base classes = $\binom{|S|+|R|-1}{|S|}$

 $|R|$ balls, $|S| + 1$ bars, two of these fixed ✓

Observation $\sigma f = g \Rightarrow h_f = h_g \Rightarrow f \sim g$ ✓
 $f \sim g \Rightarrow h_f = h_g \Rightarrow \exists \sigma \in \text{Perm}(S): \sigma f = g$ ✓

Thus $f \sim g \Leftrightarrow \exists \sigma: \sigma f = g$

Base Classes and \mathcal{F} c. u. p.

Let \mathcal{F} c. u. p.

Consider $F_f := \mathcal{F} \cap B_f$

Observations

- $g \notin \mathcal{F} \Rightarrow g \notin F_f$
- $f \in F_f$

Thus $\bigcup_{f \in \mathcal{F}} F_f = \mathcal{F}$

Obvious $F_f \subseteq B_f$

Observation $g \in B_f \Rightarrow \exists \sigma: \sigma g = f \Rightarrow g \in \mathcal{F} \Rightarrow g \in F_f$

Thus $F_f = B_f$ and $\mathcal{F} = \bigcup_{f \in \mathcal{F}} B_f$



Reasoning Against NFL

Consider non-trivial neighbourhood $N: S \times S \rightarrow \{0, 1\}$
($\exists s_1 \neq s_2 \neq s_3 \neq s_4 \in S: N(s_1, s_2) = 1, N(s_3, s_4) = 0$)

Observation neighbourhood **not** preserved
under permutation of search space

Consequences NFL does **not** hold for \mathcal{F} if

- global minima and global maxima are not direct neighbours for all $f \in \mathcal{F}$
- number of local optima is limited for all $f \in \mathcal{F}$
- all $f \in \mathcal{F}$ are smooth in some way defined by means of the neighbourhood
- all $f \in \mathcal{F}$ share some property defined by means of the neighbourhood

Consequence NFL does **not** hold **in practice**

Almost No Free Lunch

ANFL Theorem

Let $S = \{0, 1\}^n$, $R = \{0, 1, \dots, N - 1\}$, $f: S \rightarrow R$, A a randomised BBA.

The number of functions $f': S \rightarrow R \cup \{N\}$ such that A does not find an optimum of f' within $2^{n/3}$ f -evaluations with probability at least $1 - 2^{-n/3}$ is bounded below by $N^{2^{n/3}-1}$.

Of these exponentially many have the additional property that their complexity (measured by evaluation time, circuit size, or Kolmogoroff complexity) is by $O(n)$ larger than that of f .

Consequence If your algorithm is **efficient** on some function f it is necessarily **inefficient** for very many other functions, many of those not too different from f .

Summary & Take Home Message

Things to remember

- NFL: All search heuristics perform equal over a c. u. p. class of functions.
- set of all functions is c. u. p.
- being c. u. p. is rare
- 'natural' function classes not c. u. p.
- ANFL: Still there is no generally good search heuristic.

Take Home Message

- When making too general statements about randomised search heuristics the statements are either trivial or false.
- One needs to consider the class of optimisation problems to achieve good performance.