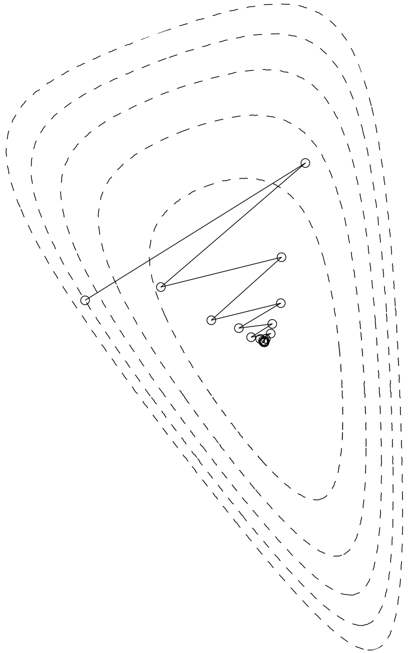


Optimization Algorithms

No Free Lunch

Marc Toussaint
Technical University of Berlin
Winter 2024/25

References

- Toussaint: *The Bayesian Search Game*. In Theory and Principled Methods for Designing Metaheuristics, Springer, 2012.
- Igel & Toussaint: *On Classes of Functions for which No Free Lunch Results Hold*. Information Processing Letters, 86, p. 317-321, 2003.
- Wolpert & Macready. *No free lunch theorems for optimization*. IEEE Transactions on Evolutionary Computation, 1(1):67–82, 1997.

No Free Lunch Theorem – Problem Setting

[Following *The Bayesian Search Game* (2012)]

- Finite(!) space X
- Distribution $P(f)$ over functions $f:\ X \to Y$
- A **non-revisiting** algorithm $\mathcal{A}$ generates queries x_t and observations $y_t = f(x_t)$. Formally, a probabilistic algorithm is defined by

$$P(x_t \,|\, x_{1:t\text{-}1}, y_{1:t\text{-}1}; \mathcal{A})$$

  and $P(x_1; \mathcal{A})$.

No Free Lunch Theorem – Problem Setting

[Following *The Bayesian Search Game* (2012)]

- Finite(!) space X
- Distribution $P(f)$ over functions $f:\ X \to Y$
- A **non-revisiting** algorithm $\mathcal{A}$ generates queries x_t and observations $y_t = f(x_t)$.
  Formally, a probabilistic algorithm is defined by
  $$P(x_t \,|\, x_{1:t\text{-}1}, y_{1:t\text{-}1}; \mathcal{A})$$
  and $P(x_1; \mathcal{A})$.
- Therefore, $\mathcal{A}$ interacting with random function f generates the joint process:

$$P(f, x_{1:T}, y_{1:T}; \mathcal{A}) = P(f)\ P(y_1 \,|\, x_1, f)\ P(x_1; \mathcal{A})\ \prod_{t=2}^{T} P(y_t \,|\, x_t, f)\ P(x_t \,|\, x_{1:t\text{-}1}, y_{1:t\text{-}1}; \mathcal{A})$$

No Free Lunch Theorem

- **Theorem:**

$$\exists h : Y \to \mathbb{R} \text{ s.t. } \forall K \in \mathcal{N}^+, \{x_1, .., x_K\} \subset X :\ P(f_{x_1}, .., f_{x_K}) = \prod_{k=1}^{K} h(f_{x_k}) \qquad (1)$$

$$\iff \qquad \forall_{\mathcal{A}}, \forall_T :\ P(y_{1:T}; \mathcal{A}) = \prod_{i=1}^{T} h(y_i) \quad \text{(independent of } \mathcal{A}\text{)} \qquad (2)$$

- In words:
  $P(f)$ factorizes $\quad \Leftrightarrow \quad$ all $\mathcal{A}$ generate the same random observations

[Proof later]

No Free Lunch Theorem – Comments

- Interpreting the LHS:
  - $P(f_{x_1},..,f_{x_K}) = \prod_{k=1}^K h(f_{x_k})$ factorizes i.i.d.
  - **There is no mutual information between any** $f(x_1), f(x_2), x_1 \neq x_2$, $I(f(x_1), f(x_2)) = 0$
  - Observing $f(x_1)$ reveals no information whatsoever on what $f(x_2)$ might be
  - Any (non-repeating!) algorithm is equally blind and uninformed about what future observations might be, not matter how it collected past information $(x_{1:t\text{-}1}, y_{1:t\text{-}1})$

No Free Lunch Theorem – Comments

- Interpreting the LHS:
  - $P(f_{x_1}, .., f_{x_K}) = \prod_{k=1}^K h(f_{x_k})$ factorizes i.i.d.
  - **There is no mutual information between any** $f(x_1), f(x_2), x_1 \neq x_2$, $I(f(x_1), f(x_2)) = 0$
  - Observing $f(x_1)$ reveals no information whatsoever on what $f(x_2)$ might be
  - Any (non-repeating!) algorithm is equally blind and uninformed about what future observations might be, not matter how it collected past information $(x_{1:t\text{-}1}, y_{1:t\text{-}1})$

- Often we have a performance metric (see later); but "all observations $P(y_t \mid ...; \mathcal{A})$ are indep. of $\mathcal{A}$" is stronger and implies equal expected performance with whatever metric
- Traditional statement: "Averaged over *all* problem instances, any algorithm performs equally. (E.g. equal to random.)"
- "there is no one algorithm that works best for every problem"

No Free Lunch Theorem – Comments

- The classical citation is Wolpert & Macready (1997), but is less general than the above and proof overly complicated and less clear in my view.
  - "Averaging over all problems" $\rightarrow$ expectation w.r.t. $P(f)$
  - "set of functions closed under permutation" $\rightarrow$ $P(f)$ factorizes
  - Our Theorem is strong $\Leftrightarrow$, not just $\Rightarrow$ (Igel & Toussaint, 2004)

NFL Proof

- We defined the process $P(f, x_{1:T}, y_{1:T}; \mathcal{A})$ previously
- Basic definitions of probabilities to prove $\Rightarrow$:

$$\begin{aligned}
P(y_t \,|\, x_{1:t\text{-}1}, y_{1:t\text{-}1}; \mathcal{A}) &= \sum_{x_t \in X} \Big[\sum_f P(y_t \,|\, x_t, f)\ P(f \,|\, x_{1:t\text{-}1}, y_{1:t\text{-}1}) \Big]\ P(x_t \,|\, x_{1:t\text{-}1}, y_{1:t\text{-}1}; \mathcal{A}) \\
&= \sum_{x_t \in X} P(f_{x_t} \!=\! y_t \,|\, x_{1:t\text{-}1}, y_{1:t\text{-}1})\ P(x_t \,|\, x_{1:t\text{-}1}, y_{1:t\text{-}1}; \mathcal{A}) \\
&= \sum_{x_t \in X} h(y_t)\ P(x_t \,|\, x_{1:t\text{-}1}, y_{1:t\text{-}1}; \mathcal{A}) = h(y_t)\ .
\end{aligned}$$

Last line: $\mathcal{A}$ is non-revisiting, and $P(f_{x_t} \!=\! y_t \,|\, x_{1:t\text{-}1}, y_{1:t\text{-}1}) = P(f_{x_t} \!=\! y_t) = h(y_t)$.

NFL Proof

- We defined the process $P(f, x_{1:T}, y_{1:T}; \mathcal{A})$ previously
- Basic definitions of probabilities to prove $\Rightarrow$:

$$
\begin{aligned}
P(y_t \,|\, x_{1:t\text{-}1}, y_{1:t\text{-}1}; \mathcal{A}) &= \sum_{x_t \in X} \Big[\sum_f P(y_t \,|\, x_t, f)\; P(f \,|\, x_{1:t\text{-}1}, y_{1:t\text{-}1}) \Big]\; P(x_t \,|\, x_{1:t\text{-}1}, y_{1:t\text{-}1}; \mathcal{A}) \\
&= \sum_{x_t \in X} P(f_{x_t} \!=\! y_t \,|\, x_{1:t\text{-}1}, y_{1:t\text{-}1})\; P(x_t \,|\, x_{1:t\text{-}1}, y_{1:t\text{-}1}; \mathcal{A}) \\
&= \sum_{x_t \in X} h(y_t)\; P(x_t \,|\, x_{1:t\text{-}1}, y_{1:t\text{-}1}; \mathcal{A}) = h(y_t)\ .
\end{aligned}
$$

Last line: $\mathcal{A}$ is non-revisiting, and $P(f_{x_t} \!=\! y_t \,|\, x_{1:t\text{-}1}, y_{1:t\text{-}1}) = P(f_{x_t} \!=\! y_t) = h(y_t)$.

- Prove $\Leftarrow$ by explicitly constructing algorithms that generate different outputs when $P(f)$ is non-factored. [Details in *The Bayesian Search Game*, 2012]

No Free Lunch for Optimization

- Consider the problem $\min_{x \in X} f(x)$ for finite X
- Also here, an algorithm $\mathcal{A}$ is defined by $P(x_k \mid x_{1:t\text{-}1}, y_{1:t\text{-}1}; \mathcal{A})$
- A typical performance metric could be **regret**

$$R(T) = \sum_{t=1}^{T} y_t - y^*$$

No Free Lunch for Optimization

- Consider the problem $\min_{x\in X} f(x)$ for finite X
- Also here, an algorithm $\mathcal{A}$ is defined by $P(x_k \,|\, x_{1:t\text{-}1}, y_{1:t\text{-}1}; \mathcal{A})$
- A typical performance metric could be **regret**

$$R(T) = \sum_{t=1}^{T} y_t - y^*$$

- But if for a non-repeating(!) $\mathcal{A}$, $P(y_t)$ is indep. of $\mathcal{A}$, so is the expected regret

No Free Lunch for Machine Learning

- Given data $D = \{(x_i, y_i)\}_{i=1}^n$, find a predictor $\hat f : \ X \to y$ that minimizes expected loss $\mathbb{E}\Big\{\ell(\hat f(x^*), f(x^*))\Big\}$ for a future query x^*, where $f(x^*)$ is the ground truth
- A learning algorithm $\mathcal{A}$ is a predictive distribution $P(y \,|\, x^*, D; \mathcal{A})$ (i.e., a mapping from D to a prediction $P(y \,|\, x^*)$ for a new query x^*)
- Assume X is finite and $x^* \notin D$ (non-repeating!)

- But if $P(f)$ factorizes so that $P(f(x^*){=}y) = h(y)$ is fully independent from D (zero mutual information), then no algorithm can learn anything or predict better than the prior.

Bayes' Theorem

$$P(X|D) = \frac{P(D|X)}{P(D)}\, P(X)$$

$$\text{posterior} = \frac{\text{likelihood} \cdot \text{prior}}{\text{normalization}}$$

- But if X is indep. from D, then there is nothing to learn or predict better than the prior $P(X)$

Conclusions from NFL?

- NFL is an almost trivial theorem, what is non-trivial is what to make of it

- NFL is an almost trivial theorem, what is non-trivial is what to make of it

- First pressing question:
  - Does NFL also hold for continuous X? What would it mean that $P(f)$ is factorized, or $I(f(x_1), f(x_2)) = 0$, for any $x_1 \neq x_2$ in continous X?

- NFL is an almost trivial theorem, what is non-trivial is what to make of it

- First pressing question:
  - Does NFL also hold for continuous X? What would it mean that $P(f)$ is factorized, or $I(f(x_1), f(x_2)) = 0$, for any $x_1 \neq x_2$ in continous X?
- Thoughts on conclusions from NFL:
  - Become aware, in your methods, what actually you are assuming - you must assume something
  - Fight back if anybody ever states "we don't (want to) make assumptions" (e.g. in a talk on RL that claims it can solve any problem without assumptions)
  - There is no Artificial General Intelligence if general would mean "making NO assumptions". So, the AGI community (say, Marcus Hutter) must make some assumptions – what are they *exactly*?
  - What are assumptions we would "generally" accept to make in our physical universe? (In case we care about AI specifically in our physical universe.)
  - What are algorithms that literally start by making assumptions about $P(f)$ and then derive an optimal algorithm for that $P(f)$? (see Bayesian Search Game...)

NFL in continuous domains

- The LHS describes $P(f)$ with $I(f(x_1), f(x_2)) = 0$ for any $x_1 \neq x_2$
  - How can we define probability distributions over functions (over continuous X) in the first place?

NFL in continuous domains

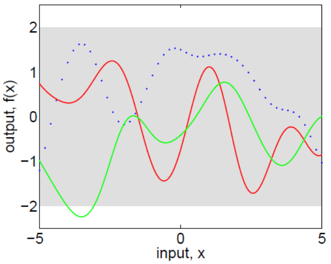
- The LHS describes $P(f)$ with $I(f(x_1), f(x_2)) = 0$ for any $x_1 \neq x_2$
  - How can we define probability distributions over functions (over continuous X) in the first place?

- A typical way to define distributions over $f : \mathbb{R}^n \to \mathbb{R}$ is as a **Gaussian Process**:
  - For every finite set $\{x_1, .., x_M\}$, the function values $f(x_1), .., f(x_M)$ are Gaussian distributed with mean and covariance

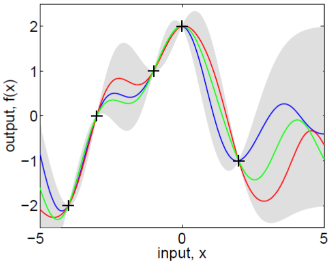
$$\mathbb{E}\{f(x_i)\} = \mu(x_i) \qquad \text{(often zero)}$$
$$\mathbb{E}\{[f(x_i) - \mu(x_i)][f(x_j) - \mu(x_j)]\} = k(x_i, x_j)$$

    where, $\mu(x)$ is called **mean function**, and $k(x, x')$ is called **covariance function**
  - μ and k generalize the notion of *mean vector* μ_x and *covariance matrix* $\Sigma_{xx'}$ from finite $x \in \{1, .., n\}$ to continuous $x \in \mathbb{R}^n$

GP examples



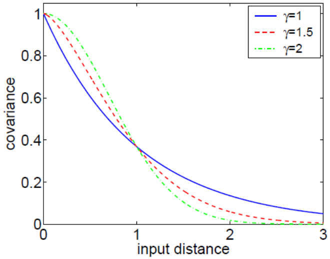
(a), prior

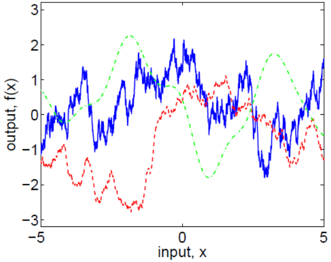


(b), posterior

(from Rasmussen & Williams)

GP examples: different covariance functions





(from Rasmussen & Williams)

- These are examples from the γ-exponential covariance function

$$k(x, x') = \exp\{-|(x - x')/l|^\gamma\}$$

NFL in continuous domains

- Back to NFL: the LHS requires $I(f(x_1), f(x_2)) = 0$, which would mean, for GPs, zero covariance function $k(x, x') = 0$ for any $x \neq x'$

NFL in continuous domains

- Back to NFL: the LHS requires $I(f(x_1), f(x_2)) = 0$, which would mean, for GPs, zero covariance function $k(x, x') = 0$ for any $x \neq x'$

- At first sight this might seem ok, but
  - Auger & Teytaud clarify that "zero-covariance GP" is not a proper Lebesgue measure over function
  - Conversely, they state that for any Lebesgue meassure the LHS does not hold (and claim that Lebesgue meassures are the only sensible kind of $P(f)$)

A. Auger and O. Teytaud: *Continuous lunches are free plus the design of optimal optimization algorithms.* Algorithmica, 2008

NFL in continuous domains

- Back to NFL: the LHS requires $I(f(x_1), f(x_2)) = 0$, which would mean, for GPs, zero covariance function $k(x, x') = 0$ for any $x \neq x'$

- At first sight this might seem ok, but
  - Auger & Teytaud clarify that "zero-covariance GP" is not a proper Lebesgue measure over function
  - Conversely, they state that for any Lebesgue meassure the LHS does not hold (and claim that Lebesgue meassures are the only sensible kind of $P(f)$)

A. Auger and O. Teytaud: *Continuous lunches are free plus the design of optimal optimization algorithms.* Algorithmica, 2008

- Beyond my expertise as non-mathematician
- But the point of NFL remains the same: one would only have to replace "non-revisiting" by "non-near-revisiting" or so.

NFL in continuous domains – conclusions

- Whether NFL holds in continuous domains depends on what $P(f)$ you consider mathematically sound
- The core point remains that if $I(f(x_1), f(x_2)) = 0$ (for non-close x_1, x_2), no non-(near)-revisiting algorithm can be smart

- Gaussian Processes are the simplest instance for assuming non-zero $I(f(x_1), f(x_2)) \neq 0$, by assuming Gaussian dependencies between $x \neq x'$
  $\Rightarrow$ GPs became a standard assumption to explicitly design algorithms exploiting that assumption and evading NFL

Become aware, in your methods, what actually you are assuming - you must assume something

- What did our optimization algorithms assume so far?

Assumptions in continuous optimization

- f is continously differentiable $f \in C^1$!
  - The limits exist! Clearly there are "correlations" when approaching infinitesimally!
  - Sure we can predict to (infinitesimally close) points: The gradient gives an accurate 1st order Taylor prediction (in the vicinity)
  - We can predict to go downhill following the gradient.
  - All this would not be possible with NFL assumptions.

Assumptions in continuous optimization

- f is continously differentiable $f \in C^1$!
  - The limits exist! Clearly there are “correlations” when approaching infinitesimally!
  - Sure we can predict to (infinitesimally close) points: The gradient gives an accurate 1st order Taylor prediction (in the vicinity)
  - We can predict to go downhill following the gradient.
  - All this would not be possible with NFL assumptions.
- Lipschitz continuity of $\nabla f(x)$ (assumption of SGD convergence)

Assumptions in continuous optimization

- f is continously differentiable $f \in C^1$!
  - The limits exist! Clearly there are "correlations" when approaching infinitesimally!
  - Sure we can predict to (infinitesimally close) points: The gradient gives an accurate 1st order Taylor prediction (in the vicinity)
  - We can predict to go downhill following the gradient.
  - All this would not be possible with NFL assumptions.
- Lipschitz continuity of $\nabla f(x)$ (assumption of SGD convergence)
- Strong convexity assumption (eigenvalues λ of the Hessian $\nabla^2 f(x)$ bounded by $m < \lambda < M$) (exponential convergence of line search)

Assumptions in continuous optimization

- f is continously differentiable $f \in C^1$!
  - The limits exist! Clearly there are “correlations” when approaching infinitesimally!
  - Sure we can predict to (infinitesimally close) points: The gradient gives an accurate 1st order Taylor prediction (in the vicinity)
  - We can predict to go downhill following the gradient.
  - All this would not be possible with NFL assumptions.
- Lipschitz continuity of $\nabla f(x)$ (assumption of SGD convergence)
- Strong convexity assumption (eigenvalues λ of the Hessian $\nabla^2 f(x)$ bounded by $m < \lambda < M$) (exponential convergence of line search)

- All assumptions are *local*, and were used to characterize local convergence behavior

Assumptions made in AGI

- Kolmogorov & Solomonoff complexity
  (also not my expertise...)

  Lattimore & Hutter: *No free lunch versus Occam's razor in supervised learning*. In Algorithmic Probability and Friends. Bayesian Prediction and Artificial Intelligence, 2013

  Baum, Hutter, & Kitzelmann: *Artificial general intelligence*. In Proceedings of the Third Conference on Artificial General Intelligence, 2010

- Occam's rasor: $P(f)$ is higher for "simpler" functions f. Assuming all (relevant) f are computable, simpler = of lower Kolmogorov/Solomonoff complexity.
- Obvious algorithm to exploit this universal prior: Sort all f by complexity, test each in order – will be better than random.
- Can also define optimal algorithms (optimal AGI) under this universal complexity prior

What are assumptions we would “generally” accept to make in our physical universe? (In case we care about AI specifically in our physical universe.)

What are assumptions we would “generally” accept to make in our physical universe? (In case we care about AI specifically in our physical universe.)

- Beyond full discussion here. Some thoughts:
  - physics $\leftrightarrow$ space$\times$time; things (fields/objects); local(!) interactions between things; invariances(!)

What are assumptions we would "generally" accept to make in our physical universe? (In case we care about AI specifically in our physical universe.)

- Beyond full discussion here. Some thoughts:
  - physics $\leftrightarrow$ space$\times$time; things (fields/objects); local(!) interactions between things; invariances(!)
  - images $\leftrightarrow$ invariances; neighboring pixels correlated $\leftrightarrow$ convolutional features, hierarchies, CNN
  - time series $\leftrightarrow$ Markovian, maybe smooth $\leftrightarrow$ HMMs, MDPs, control, etc, etc
  - Robotics, Language, Text, humans, animals, etc etc

What are algorithms that literally start by making assumptions about $P(f)$ and then derive an optimal algorithm for that $P(f)$?

Optimal Optimization

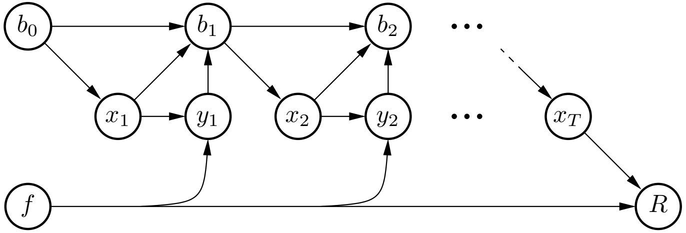
- Optimization can be formalized as a sequential decision problem (MDP):
  - Start with a prior $b_0 = P(f)$
  - Choose a query x_t based on b_t (policy, acquisition function)
  - Query x_t, observe y_t, update data D, update belief $b_t \leftarrow P(f \mid D)$, iterate

$$\begin{tikzcd}
b_0 \arrow[rr] \arrow[dr] & & b_1 \arrow[rr] \arrow[dr] & & b_2 & \cdots \arrow[dr, dashed] & & \\
& x_1 \arrow[ur] \arrow[r] & y_1 \arrow[u] & x_2 \arrow[ur] \arrow[r] & y_2 \arrow[u] & \cdots & x_T \arrow[dr] & \\
f \arrow[rrrrrrr] \arrow[uurr, bend right] \arrow[uurrrr, bend right] & & & & & & & R
\end{tikzcd}$$

[Bayesian Search Game]

Optimal Optimization

- Optimization can be formalized as a sequential decision problem (MDP):
  - Start with a prior $b_0 = P(f)$
  - Choose a query x_t based on b_t (policy, acquisition function)
  - Query x_t, observe y_t, update data D, update belief $b_t \leftarrow P(f \mid D)$, iterate



$$\begin{tikzcd}
b_0 \arrow[rr] \arrow[dr] & & b_1 \arrow[rr] \arrow[dr] & & b_2 & \cdots \arrow[dr, dashed] & \\
& x_1 \arrow[r] \arrow[ur] & y_1 \arrow[u] & x_2 \arrow[r] \arrow[ur] & y_2 \arrow[u] & \cdots & x_T \arrow[d] \\
f \arrow[urr] \arrow[urrrr] \arrow[rrrrrr] & & & & & & R
\end{tikzcd}$$

[Bayesian Search Game]

- This defines a *known* decision process, for which we can define an optimal policy
  - Can in principle be computed using Dynamic Programming – but intractable

Bayesian Optimization in a nutshell

- We maintain a particular belief $b_t = P(f \mid D)$, namely a *Gaussian Process*

Bayesian Optimization in a nutshell

- We maintain a particular belief $b_t = P(f \mid D)$, namely a *Gaussian Process*

- Don't plan an optimal query policy, but use a 1-step heuristic:
- An **acquisition function** $\alpha(x, b_t)$ characterizes how "interesting" it is to query x next, and defines the policy

$$x_t = \operatorname*{argmax}_x \alpha(x, b_t)$$

- Analogies:
  - $\alpha(x, b_t)$ is a descriminative function for the next decision
  - $\alpha(x, b_t)$ is like a Q-function $Q(b_t, x)$ for the next decision (but not learned)

to be continued with Bayesian Optimization...