

No Free Lunch Theorem

Aayush Bajaj

July 10, 2026

Contents

1	motivation	1
2	the supervised-learning theorem	2
2.1	setting	2
2.2	statement	2
2.3	proof sketch: the averaging argument	2
3	the optimisation theorem	3
4	the matching-matrix picture	4
5	an exhaustive check	4
6	what it does and does not imply	5
6.1	it does imply	5
6.2	it does not imply	5
7	see also	6

averaged over *all possible problems*, every learning algorithm is exactly as good as random guessing — and every optimiser is exactly as good as blind enumeration. this sounds like nihilism but is actually the sharpest possible argument *for* inductive bias: an algorithm can only beat chance on some problems by losing to chance on others, so the whole game of machine learning is choosing whose lunch to eat.

the name comes from the american saloon practice of ‘free’ lunches that you paid for in beer

1 motivation

- every practical learner extrapolates: it predicts labels at points it has never seen. the training data alone say *nothing* about those points — any completion of the target function is logically consistent with what was observed.
- so where does generalisation come from? not from the data. from **assumptions** — smoothness, sparsity, low complexity, translational invariance — baked into the model class and the algorithm. the no-free-lunch (nfl) theorems make this precise: strip away all assumptions (formally: put a uniform prior over targets) and all algorithms collapse to the same expected performance.
- there are two classic formalisations: wolpert’s 1996 result for supervised learning (*the lack of a priori distinctions between learning algorithms*, neural computation 8(7)) and wolpert & macready’s 1997 result for search/optimisation (*no free lunch theorems for optimization*, ieee trans. evolutionary computation 1(1)).

2 the supervised-learning theorem

2.1 setting

keep everything finite so the counting is honest:

- input space $\mathcal{X}$ with $|\mathcal{X}| = N$, label space $\mathcal{Y} = \{0, 1\}$.
- a **target** is any function $f : \mathcal{X} \rightarrow \mathcal{Y}$; there are 2^N of them.
- a training set $\mathcal{D} = \{(x_i, f(x_i))\}_{i=1}^m$ on m distinct inputs $X_{\mathcal{D}} \subset \mathcal{X}$.
- a **learning algorithm** A is any map (deterministic or randomised) from training sets to hypotheses $h = A(\mathcal{D}) : \mathcal{X} \rightarrow \mathcal{Y}$.
- score generalisation by **off-training-set** (ots) error

$$\text{err}_{\text{ots}}(h, f) = \frac{1}{N - m} \sum_{x \notin X_{\mathcal{D}}} \mathbb{1}[h(x) \neq f(x)]. \quad (1)$$

wolpert insists on ots error: on the training inputs themselves a lookup table is unbeatable, so counting them only rewards memorisation

2.2 statement

theorem (wolpert, 1996 — uniform-average form). fix the training inputs $X_{\mathcal{D}}$ and average over all 2^N targets with equal weight. then for *any* two learning algorithms A and B ,

$$\frac{1}{2^N} \sum_f \mathbb{E}[\text{err}_{\text{ots}}(A(\mathcal{D}_f), f)] = \frac{1}{2^N} \sum_f \mathbb{E}[\text{err}_{\text{ots}}(B(\mathcal{D}_f), f)] = \frac{1}{2}, \quad (2)$$

where $\mathcal{D}_f$ is the training set labelled by f and the inner expectation is over any randomness in the algorithm. uniform-averaged ots error is *algorithm-independent*, and equal to coin-flipping.

2.3 proof sketch: the averaging argument

the entire theorem is one observation about conditioning:

- **condition on the data.** fix the observed labels on $X_{\mathcal{D}}$. under the uniform prior, the 2^{N-m} targets consistent with those observations are equally likely — the prior factorises across inputs, so seeing f on $X_{\mathcal{D}}$ tells you *nothing* about f elsewhere. each unseen value $f(x)$, $x \notin X_{\mathcal{D}}$, remains an independent fair coin.
- **any guess is a coin call.** whatever $h(x)$ the algorithm outputs at an unseen x — however it was computed, however clever — it disagrees with the still-uniform $f(x)$ with probability exactly $\frac{1}{2}$.
- **sum out.** linearity of expectation over the $N - m$ unseen points gives conditional expected ots error $\frac{1}{2}$ for every algorithm and every realisation of the data; averaging over data sets and targets preserves it. ■

two upgrades come for free:

- the argument never used what A is, so it holds for humans, deep nets, and the algorithm “predict the opposite of what gradient boosting predicts”. averaged uniformly, cross-validating between two learners is no better than anti-cross-validating (picking the one with the *worse* validation score) — see [cross validation](#) for why it still works in practice.

- uniformity is sufficient but not necessary: the average is algorithm-independent for any prior on targets under which unseen labels stay conditionally unbiased given the sample.¹

3 the optimisation theorem

wolpert & macready (1997) proved the search-space twin. setting:

- finite search space $\mathcal{X}$, finite value space $\mathcal{Y}$, objective $f : \mathcal{X} \rightarrow \mathcal{Y}$ (all $|\mathcal{Y}|^{|\mathcal{X}|}$ of them possible).
- a (black-box) **search algorithm** a picks its next point as any function of the points visited so far and their observed values, never revisiting a point.
- after m steps the algorithm owns a trace $d_m = \{(x_1, y_1), \dots, (x_m, y_m)\}$; performance is any functional $\Phi(d_m^y)$ of the observed *values* $d_m^y = (y_1, \dots, y_m)$ — best-so-far, time-to-optimum, whatever.

theorem (wolpert & macready, 1997). for any two algorithms a and b , any m , and any value sequence d_m^y ,

$$\sum_f P(d_m^y \mid f, m, a) = \sum_f P(d_m^y \mid f, m, b). \quad (3)$$

summed over all objectives, every algorithm generates every possible sequence of observed values equally often — hence any performance measure Φ has identical all-functions average for all algorithms. simulated annealing, evolutionary strategies, and “enumerate $\mathcal{X}$ in alphabetical order” tie.

proof sketch (induction on m):

- **base.** the first query x_1 is fixed by the (empty) history. as f ranges uniformly over $\mathcal{Y}^{\mathcal{X}}$, the value $f(x_1)$ is uniform on $\mathcal{Y}$ — regardless of *which* x_1 the algorithm chose. so the distribution of y_1 is algorithm-free.
- **step.** condition on the trace d_{m-1} . the algorithm’s next query x_m is a fresh, unvisited point, and under the uniform prior $f(x_m)$ is still uniform on $\mathcal{Y}$ conditioned on everything seen: distinct inputs carry independent values. so $P(y_m \mid d_{m-1})$ is uniform for every algorithm, and the product telescopes into an algorithm-independent $P(d_m^y)$. ■

same engine as before: **the uniform prior makes unvisited coordinates independent of visited ones**, so no strategy for choosing where to look next can matter. a later sharpening (schumacher, vose & whitley, 2001) shows nfl holds for a set of functions exactly when that set is closed under permutations of $\mathcal{X}$ — a vanishing sliver of the priors you might actually hold.

¹wolpert’s full framework (the “extended bayesian framework”) states the results for arbitrary priors $P(f)$ and characterises when algorithm-independence holds; the uniform case is the memorable corollary. see the 1996 paper linked above.

references

Goodfellow, Ian (2016). *Deep Learning*, MIT Press. Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome (2009). *The Elements of Statistical Learning*, Springer. Wasserman, Larry (2010). *All of Statistics: A Concise Course in Statistical Inference*, Springer.

4 the matching-matrix picture

think of a performance table: rows are algorithms, columns are problems, cell brightness is performance. nfl says every row is a *permutation of the same multiset* — you can rearrange where your wins land, never how many you get. an algorithm is a bet about which columns you will face.

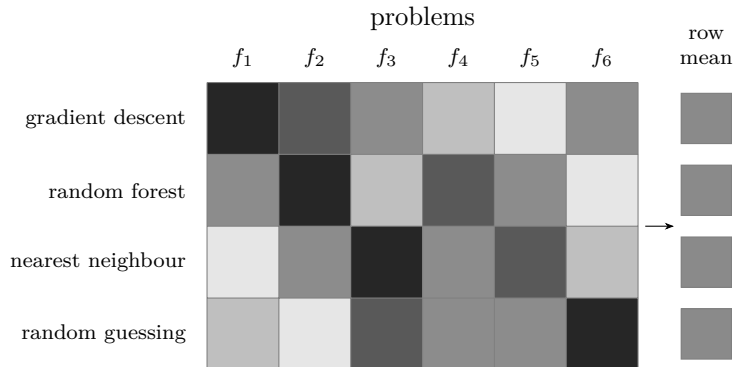


Figure 1: the nfl matching matrix: darker = better performance. every row (algorithm) contains the same multiset of values, only permuted; the row means (right) are identical. choosing an algorithm is choosing where the dark cells sit.

5 an exhaustive check

the theorem is finite, so verify it by brute force: $|\mathcal{X}| = 5$ inputs, all $2^5 = 32$ binary targets, train on the first $m = 3$ inputs, and pit four very different “algorithms” against each other on the two unseen points.

```

1  import itertools
2  import numpy as np
3
4  N, m = 5, 3                                # |X| and training-set size
5  train_idx = [0, 1, 2]
6  test_idx = [3, 4]
7
8  def learn_majority(labels):                  # predict the training majority everywhere
9      vote = int(sum(labels) * 2 > len(labels))
10     return lambda x: vote
11
12  def learn_minority(labels):                  # deliberately perverse: predict the
13     ↪ minority
14     vote = int(sum(labels) * 2 > len(labels))
15     return lambda x: 1 - vote
16
17  def learn_zero(labels):                      # ignore the data entirely
18     return lambda x: 0
19
20  def learn_parity(labels):                    # something "clever": parity of seen labels
21     p = sum(labels) % 2
22     return lambda x: (x + p) % 2
23
24  learners = {"majority": learn_majority, "minority": learn_minority,
25             "always-0": learn_zero, "parity": learn_parity}
26
27  for name, learner in learners.items():
28      total = 0.0
29      for f in itertools.product([0, 1], repeat=N): # all 32 targets
30          h = learner([f[i] for i in train_idx])
31          total += np.mean([h(x) != f[x] for x in test_idx])

```

```
31 print(f"{name:>9}: mean ots error over all targets = {total / 2**N:.4f}")
```

```
majority: mean ots error over all targets = 0.5000
minority: mean ots error over all targets = 0.5000
always-0: mean ots error over all targets = 0.5000
parity: mean ots error over all targets = 0.5000
```

sensible, perverse, lazy, and cute all land on exactly $\frac{1}{2}$. the uniform average launders every strategy into a coin.

6 what it does and does not imply

6.1 it does imply

- **inductive bias is not optional.** a learner with no preference among hypotheses consistent with the data cannot generalise at all. every working algorithm smuggles in assumptions — knn assumes local constancy, linear models assume additivity, convnets assume translation-equivariant local structure (Goodfellow, Ian, 2016). the assumptions *are* the algorithm.
- **no universally superior learner.** claims of the form “method x beats method y” are only meaningful relative to a distribution over problems. this is why honest empirical work fixes a problem family first; the point recurs throughout the model-assessment chapters of (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009).
- **specialisation has a price sheet.** gains on the problems you care about are funded, exactly, by losses on problems you (hopefully) never meet. tuning a learner to a domain sharpens the matching-matrix row toward your columns.
- **the meta-level is not exempt.** model selection, hyperparameter search, automl — these are also algorithms over the same problem space, so nfl applies to them too; there is no assumption-free way to *choose* an algorithm either.

6.2 it does not imply

- **“all algorithms are equal in practice.”** the uniform prior is a wildly unphysical assumption: almost all of the 2^N targets are incompressible noise, look-up tables with no structure whatsoever. the problems reality serves up — vision, language, physical dynamics — occupy a vanishingly small, heavily structured corner of problem space. on that corner, algorithms whose biases match the structure genuinely dominate, which is the empirical content of deep learning’s success (Goodfellow, Ian, 2016).
- **“theory is useless, just try everything.”** the opposite: since performance is purchased with assumptions, the productive question is *which assumptions hold in my domain* — and that is a modelling question, answerable with domain knowledge and diagnostics, not with more benchmark sweeps.
- **“nfl forbids general-purpose learners.”** it forbids *universal* ones. a learner biased toward low-complexity, compositional, smooth targets is still hopeless on white noise, but that bias is broad enough to cover most problems humans pose — general-purpose in the only sense that matters.
- **free lunches reappear the moment the prior is non-uniform.** real prob-

a uniform-random labelling of the 2^{16} four-by-four binary images needs 65536 bits to describe; nothing in nature writes those

lem distributions are not permutation-closed, so the 2001 sharpening says nfl simply does not bind on them. the theorem describes the desert, not the buffet you actually eat at.

the bayesian reading is clean: nfl says that with a flat prior the posterior over unseen labels equals the prior — inference cannot begin until you put mass somewhere (Wasserman, Larry, 2010). the bias-variance ledger of [the bias-variance decomposition](#) is the accounting system for what that mass costs you.

7 see also

- [the bias-variance decomposition](#) — the price of the inductive bias nfl forces you to buy
- [cross validation](#) — model selection under nfl: works because real problems are not uniform
- [curse of dimensionality](#) — the other impossibility result every learner must dodge
- [loss functions](#) — the performance measures Φ that nfl averages over
- [kernel methods](#) — inductive bias made explicit as a choice of kernel