Bias-Variance Tradeoff and the No-Free-Lunch Theorem

Uri Shaham

May 31, 2026

1 The No-Free-Lunch Theorem

We have seen that any hypothesis class $\mathcal{H}$ with finite VC dimension is PAC learnable. But how about richer classes? Why shouldn't we simply choose the richest of all - the set of all functions from domaim $\mathcal{X}$ to $\{0,1\}$. Is this class learnable? If so, there exists a learning algorithm that can universally crack any learning problem, i.e., can achieve good generalization error on any distribution $\mathcal{D}$ (did anyone mention deep learning?) In this section, we will see that this is, unfortunately, impossible. Specifically, we will see that for every learning algorithm A there exists a distribution $\mathcal{D}$ on which it miserably fails.

Theorem 1.1 (No-Free-Lunch)**.** *Let $\mathcal{A}$ be a learning algorithm for binary classification over a domain $\mathcal{X}$. Let $n \leq \frac{|\mathcal{X}|}{2}$ be the training set size. Then there exists a distribution $\mathcal{D}$ over $\mathcal{X} \times \{0,1\}$ such that:*

1. *There exists a function $f : \mathcal{X} \to \{0,1\}$ with $L_{\mathcal{D}}(f) = 0$*
2. *$\mathbb{E}_{S_n \sim \mathcal{D}^n} L_{\mathcal{D}}(\mathcal{A}(S_n)) \geq 1/4$.*

Remark 1.2. *There is nothing special about the number $\frac{1}{4}$ in this theorem; it is just used for convenience. Any number smaller than 1 would work as well.*

Proof. The intuition of the proof is simple: any algorithm trained on $\S_n$ makes predictions on the rest of $\mathcal{X}$. We can always have a distribution in which these predictions are all wrong outside of $\S_n$.

Let $C \subseteq \mathcal{X}$ be of size $2n$. For each possible function $f_i : C \to \{0,1\}$ we create a distribution $\mathcal{D}_i$ on $C \times \{0,1\}$, by placing the probability mass of $\mathcal{D}_i$ uniformly only on instances (x, y) on which $y = f_i(x)$. Thus $L_{\mathcal{D}_i}(f_i) = 0$.

We will now show that for any algorithm A there exists a distribution D_i such that

$$\mathbb{E}_{S_n \sim \mathcal{D}_i^n} L_{\mathcal{D}_i}(\mathcal{A}(S_n)) \geq 1/4.$$

There are $k = (2n)^n$ sequences of length n from C, which we enumerate by $S_1, \ldots S_k$. For sequence $S_j = (x_1, \ldots, x_n)$ we denote by S_j^i the labeled sequence obtained from f_i, i.e., $S_j^i = ((x_1, f_i(x_1)) \ldots, (x_n, f_i(x_n))$. Observe that if the distribution is $\mathcal{D}_i$, the possible training sequences are $S_1^i, \ldots, S_k^i$, and all have the same probability of being sampled. Thus

$$\mathbb{E}_{S_n \sim \mathcal{D}_i^n} L_{\mathcal{D}_i}(\mathcal{A}(S_n)) = \frac{1}{k} \sum_{j=1}^{k} L_{\mathcal{D}_i}(\mathcal{A}(S_j^i)). \tag{1}$$

Recall that we are interested in the worst distribution (in terms of generalization error). Denote the total number of labeling functions on C by $T = 2^{2n}$. We can write

$$\begin{aligned}\max_i \frac{1}{k}\sum_{j=1}^{k} L_{\mathcal{D}_i}(\mathcal{A}(S_j^i)) &\geq \frac{1}{T}\sum_{i=1}^{T}\frac{1}{k}\sum_{j=1}^{k} L_{\mathcal{D}_i}(\mathcal{A}(S_j^i)) \\ &= \frac{1}{k}\sum_{j=1}^{k}\frac{1}{T}\sum_{i=1}^{T} L_{\mathcal{D}_i}(\mathcal{A}(S_j^i)) \\ &= \min_j \frac{1}{T}\sum_{i=1}^{T} L_{\mathcal{D}_i}(\mathcal{A}(S_j^i)). \end{aligned} \tag{2}$$

Now fix some j. Since $|C| = 2n$, and there are n elements in S_j, there are at $p \geq m$ elements in C that do not appear in S_j, which we will denote by $v_1, \dots v_p$. For every h and every i we have

$$\begin{aligned} L_{\mathcal{D}_i}(h) &= \frac{1}{2n}\sum_{x\in C} \mathbb{1}_{h(x)\neq f_i(x)} \\ &\geq \frac{1}{2n}\sum_{r=1}^{p} \mathbb{1}_{h(v_r)\neq f_i(v_r)} \\ &\geq \frac{1}{2p}\sum_{r=1}^{p} \mathbb{1}_{h(v_r)\neq f_i(v_r)}. \end{aligned}$$

Thus,

$$\begin{aligned}\frac{1}{T}\sum_{i=1}^{T} L_{\mathcal{D}_i}(\mathcal{A}(S_j^i)) &\geq \frac{1}{T}\sum_{i=1}^{T}\frac{1}{2p}\sum_{r=1}^{p} \mathbb{1}_{\mathcal{A}(S_j^i)(v_r)\neq f_i(v_r)} \\ &= \frac{1}{2p}\sum_{r=1}^{p}\frac{1}{T}\sum_{i=1}^{T} \mathbb{1}_{\mathcal{A}(S_j^i)(v_r)\neq f_i(v_r)} \\ &\geq \frac{1}{2}\min_r \frac{1}{T}\sum_{i=1}^{T} \mathbb{1}_{\mathcal{A}(S_j^i)(v_r)\neq f_i(v_r)}. \end{aligned} \tag{3}$$

Observe that given r, we can partition the T functions to pairs, where each pair (i, i') differ only in the label of v_r, so that $\mathbb{1}_{\mathcal{A}(S_j^i)(v_r)\neq f_i(v_r)} + \mathbb{1}_{\mathcal{A}(S_j^{i'})(v_r)\neq f_i(v_r)} = 1$. This gives

$$\frac{1}{T}\sum_{i=1}^{T} L_{\mathcal{D}_i}(\mathcal{A}(S_j^i)) \geq \frac{1}{2}. \tag{4}$$

Combining equations (4), (3), (2), (1) completes the proof. □

Corollary 1.3. *Under the conditions of the previous theorem, with probability at least $\frac{1}{7}$ over the choice of $S_n \sim \mathcal{D}^n$, $L_{\mathcal{D}}(\mathcal{A}(S_n)) \geq 1/8$.*

Proof. Define the random variable $Z := 1 - L_{\mathcal{D}}(\mathcal{A}(S_n))$, whose expectation is $\mu := 1 - \mathbb{E}_{S_n \sim \mathcal{D}^n} L_{\mathcal{D}}(\mathcal{A}(S_n))$. By the NFL theorem, $\mu \leq \frac{3}{4}$. By Markov inequality, $\Pr(Z \geq a) \leq \frac{\mu}{a}$. Thus

$$\begin{aligned}
\Pr\left(L_{\mathcal{D}}(\mathcal{A}(S_n)) \geq \frac{1}{8}\right) &= 1 - \Pr\left(L_{\mathcal{D}}(\mathcal{A}(S_n)) \leq \frac{1}{8}\right) \\
&= 1 - \Pr\left(Z \geq \frac{7}{8}\right) \\
&\geq 1 - \frac{8\mu}{7} \\
&\geq 1 - \frac{8 \cdot 3}{7 \cdot 4} \\
&= \frac{1}{7}. \qquad (5)
\end{aligned}$$

□

Corollary 1.4. *The hypothesis class $\mathcal{H}$ of all hypotheses over $\mathcal{X}$ is not PAC-learnable.*

Proof. Assume to the contrary that $\mathcal{H}$ is PAC-learnable. Choose $\epsilon < \frac{1}{8}$ and $\delta < \frac{1}{7}$. Then there must be an algorithm A and an integer $n = n(\epsilon, \delta)$ such that for every distribution $\mathcal{D}$ over $\mathcal{X} \times \{0, 1\}$, if there exists a function f such that $L_{\mathcal{D}}(f) = 0$ then with probability at least $1 - \delta$ over samples $S \sim \mathcal{D}^n$, $L_{\mathcal{D}} A(S) \leq \epsilon$. However, if $|\mathcal{X}| > 2n$, this contradicts the NFL theorem. □

2 Error Decomposition and the Bias-Variance Tradeoff

The corollary tells us that in order to have PAC learnability, we cannot simply choose the class of all hypotheses, and choose some smaller class, applying inductive bias. We typically characterize a hypothesis class $\mathcal{H}$ via the quantity $\epsilon_{\text{app}} := \min_{h \in \mathcal{H}} L_{\mathcal{D}}(h)$, to which we refer as *approximation error*, quantifying the extent of risk we take by choosing the class $\mathcal{H}$, or, put another way, how good is $\mathcal{H}$ in approximating the target labeling. Then, we for each hypothesis $h \in \mathcal{H}$ we specify the extra risk arising from this particular h (compared to $\arg\min_{h \in \mathcal{H}} L_{\mathcal{D}}(\text{h})$, which we denote by $\epsilon_{\text{est}} := Ł_{\mathcal{D}}(h) - \epsilon_{\text{app}}$. The source of the estimation error is in (i) using the training error $Ł_S$ as the objective for learning, and in (ii) finding the minimizer of L_S.

Our goal is naturally to find h which minimizes the overall risk $Ł_D(h)$. The above decomposition introduces the famous *Bias-Variance* tradeoff: By choosing a richer hypothesis class $\mathcal{H}$, we approximate the target labeling better, and decrease the approximation error. However, the estimation error grows as $\mathcal{H}$ becomes richer - we saw this in the case of finite hypothesis case (where ϵ_{est} grows logarithmically with $\mathcal{H}$, and also in the infinite case, having the VC dimension as quantifier for the richness of $\mathcal{H}$.

We frequently call the approximation error *bias* (and use it to describe the goodness of fit of a hypothesis class to the true labeling) and the estimation error *variance*, which quantifying the challenges arising from optimizing over rich classes.

2.1 Bayes Optimal Predictor

Any classification problem has an optimal, unavoidable error, called the *Bayes error*, which is the error of the optimal classifier $h_{Bayes}(x) = \arg\max_y \mathcal{D}(y|x)$ (why is it optimal? hint: let h be some other classifier, find x on which the classifiers do not agree, change the prediction to the Bayes and get a

smaller error). Obviously, unless generating the data ourselves, we never know $\mathcal{D}$, so we cannot simply take the Bayes classifier.

Since the approximation error is at least as large as the Bayes error, it can be decomposed to $\epsilon_{\text{Bayes}} + (\min_{h \in \mathcal{H}} L_{\mathcal{D}}(h) - \epsilon_{\text{Bayes}})$, where $\epsilon_{\text{Bayes}} = L_{\mathcal{D}}(h_{\text{Bayes}})$.

2.2 MSE Decomposition

Assume a regression model, where we specify the data generating distribution $\mathcal{D}$ by $y = h^*(x) + \epsilon$, where h^* is the correct labeling function and ϵ is some zero-mean random variable, independent of x. In the case of mean squared error, we fix an x, and look at expectations with respect to training samples $S \sim D^n$. That is, $h(x)$ is a random variable, as h is a result of training on a particular training sample S.

we can decompose the error to variance and squared bias terms:

$$\begin{aligned}
\mathbb{E}_{S,\epsilon}(h) &= \mathbb{E}_{S,\epsilon}(y - h(x))^2 \\
&= \mathbb{E}_{[S,\epsilon)\sim\mathcal{D}}(y - \mathbb{E}_S[h(x)] + \mathbb{E}_S[h(x)] - h(x)]^2 \\
&= \mathbb{E}_{S,\epsilon}(y - \mathbb{E}_S\, h(x))^2 + \mathbb{E}_S(\mathbb{E}_{x\sim\mathcal{D}}\, h(x) - h(x))^2,
\end{aligned}$$

where the last transition is due to the fact that the cross terms vanish, since $\mathbb{E}_{\mathcal{D}}[h(x) - \mathbb{E}_{\mathcal{D}}[h(x)]] = 0$. Observe that

$$\mathbb{E}_S[\mathbb{E}_S[h(x)] - h(x)]^2 = \text{Var}(h(x)).$$

Furthermore

$$\begin{aligned}
\mathbb{E}_{S,\epsilon}[y - \mathbb{E}_S\, h(x)]^2 &= \mathbb{E}_{S,\epsilon}[h^*(x) + \epsilon - \mathbb{E}_S\, h(x)]^2 \\
&= \text{Var}(\epsilon) + (h^*(x) - \mathbb{E}_S[h(x)])^2, \qquad (6)
\end{aligned}$$

and we write

$$(h^*(x) - \mathbb{E}_S[h(x)])^2 = \text{bias}^2 h(x)$$

to get

$$\mathbb{E}_{S,\epsilon}(h) = \text{Var}(\epsilon) + \text{Var}(h(x)) + \text{bias}^2 h(x)$$

, which is a decomposition of the squared error to variance and squared bias terms, plus an unavoidable error resulting from noisy labeling.

3 Reading

UML ch. 5