Stanford MS&E 215 / CS 264: Guarantees for automated algorithm configuration

Ellen Vitercik*
vitercik@stanford.edu

May 27, 2025

Today, we will discuss *automated algorithm configuration*, a broadly applicable way of using ML to optimize the parameters of any parameterized algorithm, such as an integer programming (IP) solver. Integer programming solvers like CPLEX and Gurobi each come with over one hundred tunable parameters. In automated algorithm configuration, our goal is to use a data-driven approach to find optimized, application-specific parameter settings. This lecture is based on a paper by Gupta and Roughgarden [2], which was later covered in a book chapter by Balcan [1].

1 Notation and problem definition

Our goal is to optimally configure an algorithm with parameters $\boldsymbol{\theta}$. We use the notation Π to denote the set of all possible inputs to this algorithm. For example, if the algorithm is an integer programming solver, then $\pi \in \Pi$ would be an integer program. To model the application domain, we assume there is an application-specific distribution $\mathcal{D}$ over Π. For example, $\mathcal{D}$ might be a distribution over the particular routing IPs that a Bay Area shipping company has to solve on a day-to-day basis. Alternatively, if Π is a known benchmark set, then $\mathcal{D}$ might be the uniform distribution over this set.

We use the notation $r_{\boldsymbol{\theta}}(\pi) \in \mathbb{R}$ to denote the "performance" of the algorithm parameterized by $\boldsymbol{\theta}$ on the input π, measured, for example, by its runtime or the quality of the solution returned by the algorithm (we will see an example of this in Section 2).

Our goal is to find a parameter setting $\boldsymbol{\theta}$ with good expected performance $\mathbb{E}_{\pi\sim\mathcal{D}}[r_{\boldsymbol{\theta}}(\pi)]$. For example, if $\mathcal{D}$ represents the distribution of routing IPs that a shipping company has to solve on a day-to-day basis, then $\mathbb{E}_{\pi\sim\mathcal{D}}[r_{\boldsymbol{\theta}}(\pi)]$ represents the expected performance (e.g., runtime) of our solver on IPs that we'll encounter in the future. Since we do not know the distribution $\mathcal{D}$, we will sample a "training set" $S = \{\pi_1, \ldots, \pi_N\} \sim \mathcal{D}^N$ and find a parameter setting $\boldsymbol{\theta}$ with good empirical performance

$$\frac{1}{N}\sum_{i=1}^{N} r_{\boldsymbol{\theta}}(\pi_i). \tag{1}$$

This lecture will address two questions:

*These notes are course material and have not undergone formal peer review. Please feel free to send me any typos or comments.

Question 1. *Is it possible to find a parameter setting $\hat{\boldsymbol{\theta}}$ that provably optimizes Equation* (1)*?*

Question 2. *How many sample instances do we need to avoid overfitting? In other words, how many samples are sufficient to ensure that if we find a parameter setting $\hat{\boldsymbol{\theta}}$ with good empirical performance, then its expected future performance $\mathbb{E}_{\pi\sim\mathcal{D}}[r_{\hat{\boldsymbol{\theta}}}(\pi)]$ will also be good?*

2 Maximum weight independent set

For concreteness, this lecture is centered on a parameterized algorithm for the maximum weight independent set (MWIS) problem. (In the homework, you will apply the same ideas to another computational problem. Moreover, Balcan [1] provides a more general treatment in her book chapter.) The input is a graph $G = (V, E)$ with $n = |V|$ vertex weights $w_1, \ldots, w_n \geq 0$. For simplicity, we assume these weights are normalized so that $\sum w_i \leq 1$. The set $S \subseteq V$ is an *independent set* if no two vertices in S are connected by an edge. The goal is to find an independent set S with maximum weight

$$\sum_{i\in S} w_i.$$

For each vertex $i \in V$, let $N(i) = \{j : (i, j) \in E\}$ be the neighborhood of i. A classic greedy heuristic for MWIS is to greedily add vertices in decreasing order of

$$\frac{w_i}{1 + |N(i)|}$$

while maintaining independence. Gupta and Roughgarden [2] propose a parameterized variant of this heuristic, which is defined by a parameter $\theta \geq 0$. It greedily adds vertices in decreasing order of

$$\frac{w_i}{(1 + |N(i)|)^\theta}$$

while maintaining independence.

In this context, we will measure algorithmic performance based on the weight of the set that the algorithm returns. In particular, if S is the set returned by the algorithm parameterized by θ on input G, then

$$r_\theta(G) = \sum_{i\in S} w_i.$$

Instantiating the notation from Section 1, $\mathcal{D}$ will be a distribution over graphs. Given a "training set" of graphs $S = \{G_1, \ldots, G_N\} \sim \mathcal{D}^N$, our algorithm configuration goal is to find a parameter setting that maximizes

$$\frac{1}{N}\sum_{i=1}^{N} r_\theta(G_i). \tag{2}$$

To this end, we will begin with a useful lemma.

Lemma 2.1 (Gupta and Roughgarden [2])**.** *Let G be a graph with vertex weights $w_1, \ldots, w_n$. There are $\binom{n}{2}$ thresholds in $[0, \infty)$ such that if θ and θ' lie between the same consecutive thresholds, then $r_\theta(G) = r_{\theta'}(G)$.*

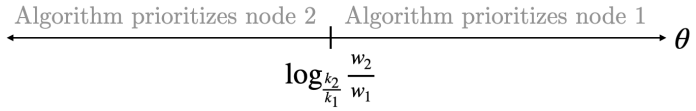


Figure 1: For θ smaller than the threshold, the greedy algorithm parameterized by θ would add node 1 to the independent set before node 2. Otherwise, it would add node 2 before node 1.

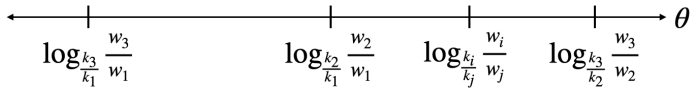


Figure 2: In any interval between consecutive thresholds, the algorithm returns the exact same independent set.

Proof. To simplify notation, we let $k_i = |N(i)| + 1$. The greedy algorithm parameterized by θ would add node 1 to the independent set before node 2 if

$$\frac{w_1}{k_1^\theta} \geq \frac{w_2}{k_2^\theta} \qquad \Leftrightarrow \qquad \theta \geq \log_{\frac{k_2}{k_1}} \frac{w_2}{w_1},$$

as illustrated in Figure 1. Applying the same logic for all pairs of nodes $i, j \in [n]$, we obtain a total of $\binom{n}{2}$ thresholds, as illustrated in Figure 2. If θ and θ' lie between the same consecutive thresholds, the algorithm parameterized by θ adds the same set of vertices to the independent set as the algorithm parameterized by θ'. Consequently, $r_\theta(G) = r_{\theta'}(G)$, as illustrated in Figure 3. □

Across all graphs $G_1, \ldots, G_N$ in the training set, we have a total of $N\binom{n}{2}$ thresholds. These thresholds partition the positive reals into $M = N\binom{n}{2} + 1$ intervals $I_1, I_2, \ldots, I_M$ such that if $\theta, \theta' \in I_j$, then for all graphs $i \in [N]$, $r_\theta(G_i) = r_{\theta'}(G_i)$. In other words,

$$\frac{1}{N} \sum_{i=1}^{N} r_\theta(G_i) = \frac{1}{N} \sum_{i=1}^{N} r_{\theta'}(G_i).$$

This observation suggests the following natural algorithm for finding a parameter setting $\hat{\theta}$ that maximizes Equation (2):

1. Compute the $N\binom{n}{2}$ thresholds corresponding to all graphs $G_1, \ldots, G_N$ in the training set.
2. Calculate Equation (2) using an arbitrary θ between subsequent thresholds.
3. Return the parameter setting $\hat{\theta}$ that maximizes Equation (2).

We've now answered Question 1 for this problem.

3 Statistical guarantees

We now move on to Question 2: how many sample instances do we need to avoid overfitting? In other words, how many samples are sufficient to ensure that for the parameter setting $\hat{\theta}$ returned by the algorithm above, $\mathbb{E}_{\pi \sim \mathcal{D}}\left[r_{\hat{\boldsymbol{\theta}}}(G)\right]$ is also large?

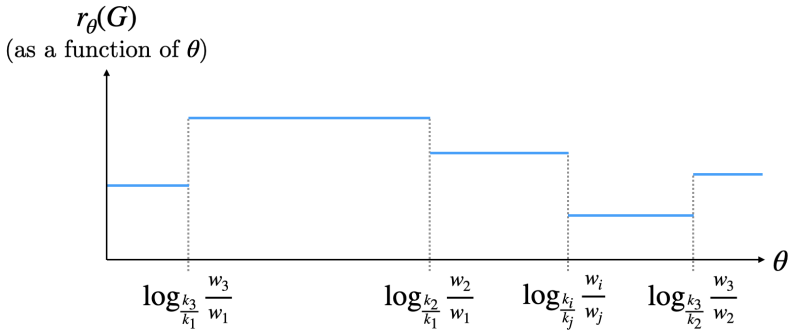
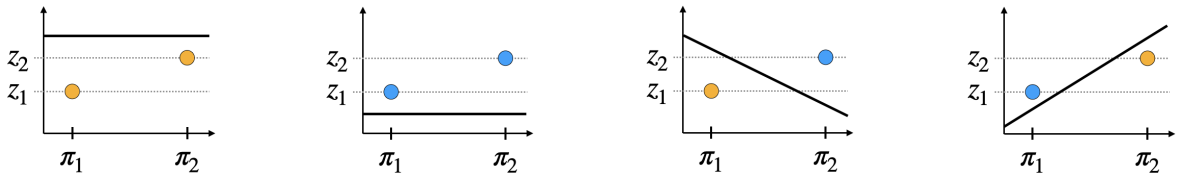


Figure 3: Illustration of Lemma 2.1.



(a) Above/above (b) Below/below (c) Above/below (d) Below/above

Figure 4: Pseudo-dimension illustration. We pick two values $\pi_1, \pi_2 \in [0, 1]$ and two targets $z_1, z_2 \in \mathbb{R}$ such that we can "achieve all above/below patterns" using affine functions.

3.1 Pseudo-dimension

We will use *pseudo-dimension* to answer Question 2. Pseudo-dimension is a complexity measure that applies to any class $\mathcal{R}$ of real-valued functions that map a set Π to some interval $[-H, H]$. We first state pseudo-dimension informally in words. It's ok if this informal description doesn't make complete sense at first; it's best illustrated with a picture. Informally, the pseudo-dimension of $\mathcal{R}$, denoted $\text{Pdim}(\mathcal{R})$, is the size of the largest set $\{\pi_1, \ldots, \pi_N\} \subseteq \Pi$ such that for some set of *targets* $z_1, z_2, \ldots, z_N \in \mathbb{R}$, we can "achieve all above/below patterns" using functions in $\mathcal{R}$. You are probably wondering: what are these targets, and what does it mean to "achieve all above/below patterns"? Let's illustrate with an example.

Let $\Pi = [0, 1]$ and let $\mathcal{R}$ be the set of all affine functions over $\mathbb{R}$. An affine function $r_{a,b}(x)$ is defined by a slope a and offset b, with $r_{a,b}(x) = ax + b$. We will show that $\text{Pdim}(\mathcal{R}) \geq 2$. To do so, we need to identify two values $\pi_1, \pi_2 \in [0, 1]$, together with two targets $z_1, z_2 \in \mathbb{R}$, such that we can "achieve all above/below patterns" using affine functions. In Figure 4, we choose two arbitrary points $\pi_1 < \pi_2$ and two targets $z_1 < z_2$. The affine (constant) function in Figure 4a achieves the "above/above" pattern because when we evaluate the affine function at π_1, it is larger than (or "above") the target z_1, and when we evaluate the function at π_2, it is larger than z_2. Meanwhile, the affine function in Figure 4b achieves the "below/below" pattern because when we evaluate the affine function at π_1, it is smaller than (or "below") the target z_1, and when we evaluate the function at π_2, it is smaller than z_2. Similarly, the affine function in Figure 4c achieves the "above/below" pattern and the affine function in Figure 4d achieves the "above/below" pattern.

To move toward the formal definition of pseudo-dimension, note that if r_1 is the function

in Figure 4a, then

$$\begin{pmatrix} \mathbf{1}_{\{r_1(\pi_1) \le z_1\}} \\ \mathbf{1}_{\{r_1(\pi_2) \le z_2\}} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Similarly, if r_2, r_3, and r_4 are the functions in Figures 4b-4d, then

$$\begin{pmatrix} \mathbf{1}_{\{r_2(\pi_1) \le z_1\}} \\ \mathbf{1}_{\{r_2(\pi_2) \le z_2\}} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \qquad \begin{pmatrix} \mathbf{1}_{\{r_3(\pi_1) \le z_1\}} \\ \mathbf{1}_{\{r_3(\pi_2) \le z_2\}} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \text{and} \quad \begin{pmatrix} \mathbf{1}_{\{r_4(\pi_1) \le z_1\}} \\ \mathbf{1}_{\{r_4(\pi_2) \le z_2\}} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

With this intuition, we now formally define pseudo-dimension.

Definition 3.1. The pseudo-dimension of $\mathcal{R}$ is the size of the largest set $\{\pi_1, \dots, \pi_N\} \subseteq \Pi$ such that for some set of targets $z_1, z_2, \dots, z_N \in \mathbb{R}$

$$\left| \left\{ \begin{pmatrix} \mathbf{1}_{\{r(\pi_1) \le z_1\}} \\ \vdots \\ \mathbf{1}_{\{r(\pi_N) \le z_N\}} \end{pmatrix} : r \in \mathcal{R} \right\} \right| = 2^N.$$

Pseudo-dimension implies an upper bound on the number of samples we need to avoid overfitting:

Theorem 3.2 (Pollard [3])**.** *For $\epsilon, \delta \in (0,1)$, let*

$$N = O\left(\frac{\mathrm{Pdim}(\mathcal{R})}{\epsilon^2} \log \frac{1}{\delta} \right).$$

With probability at least $1 - \delta$ over $\pi_1, \dots, \pi_N \sim \mathcal{D}$, for all $r \in \mathcal{R}$,

$$\left| \frac{1}{N} \sum_{i=1}^{N} r(\pi_i) - \mathop{\mathbb{E}}_{\pi \sim \mathcal{D}} [r(\pi)] \right| \le \epsilon H.$$

3.2 Back to maximum weight independent set

Let's return back to MWIS. In Section 2, we determined that for N graphs $G_1, \dots, G_N$, there are $M = N\binom{n}{2} + 1$ intervals $I_1, I_2, \dots, I_M$ such that if $\theta, \theta' \in I_j$, then for all graphs $i \in [N]$, $r_\theta(G_i) = r_{\theta'}(G_i)$. Said another way,

$$\begin{pmatrix} \mathbf{1}_{\{r_\theta(G_1) \le z_1\}} \\ \vdots \\ \mathbf{1}_{\{r_\theta(G_N) \le z_N\}} \end{pmatrix} = \begin{pmatrix} \mathbf{1}_{\{r_{\theta'}(G_1) \le z_1\}} \\ \vdots \\ \mathbf{1}_{\{r_{\theta'}(G_N) \le z_N\}} \end{pmatrix}.$$

Therefore,

$$\left| \left\{ \begin{pmatrix} \mathbf{1}_{\{r_\theta(G_1) \le z_1\}} \\ \vdots \\ \mathbf{1}_{\{r_\theta(G_N) \le z_N\}} \end{pmatrix} : \theta \ge 0 \right\} \right| \le N\binom{n}{2} + 1. \tag{3}$$

This fact allows us to bound the pseudo-dimension of the set $\mathcal{R} = \{r_\theta : \theta > 0\}$.

Theorem 3.3 (Gupta and Roughgarden [2])**.** *Let $\mathcal{R} = \{r_\theta : \theta > 0\}$. Then Pdim$(\mathcal{R}) = O(\log n)$.*

Proof. Let $N = \text{Pdim}(\mathcal{R})$. By definition, there exist graphs $G_1, \ldots, G_N$ and targets $z_1, \ldots, z_N$ such that

$$2^N = \left| \left\{ \begin{pmatrix} \mathbf{1}_{\{r_\theta(G_1) \le z_1\}} \\ \vdots \\ \mathbf{1}_{\{r_\theta(G_N) \le z_N\}} \end{pmatrix} : \theta > 0 \right\} \right| \le N \binom{n}{2} + 1,$$

where the inequality follows from Equation (3). By the following lemma, this means that $\text{Pdim}(\mathcal{R}) = N = O(\log n)$.

Lemma 3.4 (Lemma A.2 by Shalev-Shwartz and Ben-David [4])**.** *Let $a \ge 1$ and $b > 0$. If $x < a \log x + b$, then $x < 4a \log(2a) + 2b$.*

□

Finally, if we combine this pseudo-dimension bound with Theorem 3.2, we obtain the following guarantee: for $\epsilon, \delta \in (0, 1)$, let

$$N = O\left(\frac{\log n}{\epsilon^2} \log \frac{1}{\delta} \right).$$

Then with probability at least $1 - \delta$ over $G_1, \ldots, G_N \sim \mathcal{D}$, for all $\theta > 0$,

$$\left| \frac{1}{N} \sum_{i=1}^{N} r_\theta(G_i) - \mathop{\mathbb{E}}_{G \sim \mathcal{D}} [r_\theta(G)] \right| \le \epsilon.$$

We have now answered Question 2! Given $N = O\left(\frac{\log n}{\epsilon^2} \log \frac{1}{\delta}\right)$ sample graphs, we know that since $\hat{\theta}$ (the parameter setting we obtained in Section 2) leads to independent sets with large weight on average

$$\frac{1}{N} \sum_{i=1}^{N} r_{\hat{\theta}}(G_i),$$

it will also lead to independent sets with large weight in expectation $\mathbb{E}_{G \sim \mathcal{D}} [r_{\hat{\theta}}(G)]$. In fact, as extra credit in the homework, you will show that $\hat{\theta}$ is nearly optimal in expectation:

$$\mathop{\mathbb{E}}_{G \sim \mathcal{D}} [r_{\hat{\theta}}(G)] \ge \max_{\theta \ge 0} \mathop{\mathbb{E}}_{G \sim \mathcal{D}} [r_\theta(G)] - 2\epsilon.$$

References

[1] Maria-Florina Balcan. Data-driven algorithm design. In Tim Roughgarden, editor, *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, 2020.

[2] Rishi Gupta and Tim Roughgarden. A PAC approach to application-specific algorithm selection. *SIAM Journal on Computing*, 46(3):992–1017, 2017.

[3] David Pollard. *Convergence of Stochastic Processes*. Springer, 1984.

[4] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.