How Bad Is Worst Case Data If You Know Where It Comes From: Review & Summary

Vafa Behnam Roudsari

April 2020

1 Introduction

Paper: How bad is worst-case data if you know where it comes from? (Chen, J.; Valiant, G.; Valiant, P) [1]

A lot of standard statistical inference work assumes that our data is i.i.d and that any arbitrary partition of data can be exchangeable with each other. This paper deals with the consequences of relaxing this assumption, i.e. under what conditions can we learn statistics about one dataset from another dataset (called train and test in the paper)? It is pretty apparent that if there is no connection between the train and test set, that inference from the train set to the test set is impossible. But what of all the scenarios in between these two extreme cases?

More formally, suppose we are given a set of training data, and are asked to estimate the mean of another set of data, called the test data. How well can we approximate the test set, if we have knowledge (or sample access) of the partition distributions governing how the train and test datasets are created? The answer is that it depends on how the two datasets are partitioned, and this dependence in the paper is captured by the performance of the "worst-case" data, i.e. if on the most adversarial set of data learning is futile, then we cannot expect to learn, as in the fully disjoint case above.

As we will soon see, the data is not the stochastic quantity in this context, but rather sets of partitions are, which is an unfamiliar scenario for statisticians. The justification here is, fixed on a sample of partitions, to tether our analysis to the worst-possible case. The assumption here is that the worst-case data will reveal structure about how difficult the learning problem is. Surprisingly, this worst-case analysis has a connection to the P vs. NP issue, and might suggest why worst-case analysis hasn't generally caught on in statistics.

The most restrictive assumption of this paper is knowledge or sample access to a partition distribution P, which splits some large set into test and train. It is critical therefore to investigate the nature of P, and to see what techniques exist to approximate or model it. For example, what is the connection of P to the propensity score in statistics? Can we interpret train/test split, standard

machine learning nomenclature, the same way we could treatment/control? If so, what are the connections of this paper to the causal inference literature?

2 Evaluating Performance

Optimization implicitly includes two components, evaluating the performance of a proposed set of inputs, and then selection of the best inputs. In most of the scenarios typical to statistics or machine learning, the former is trivial and immediate[1]: since we assume the data is fixed (after a sampling process)[2], we simply plug in these values to the distance function. As we'll soon see, introducing "worst-case data" to statistical analysis raises fundamental difficulties in this vein.

Hence, let us present what we mean by performance. Adopting the same notation as the paper, let A_i, B_i be the training and test set respectively, x the (worst case) data. We define $a_i \in \mathbb{R}^N$ as the weights of our linear function, and $b_i \in \mathbb{R}^N$ where $(b_i)_j = \frac{1}{|B_i|}$ for j in the test set and 0 otherwise. Hence, $b_i^T x$ is equal to the mean of the vector x, and is what we are trying to predict. Therefore, the worst-case mean squared error between our prediction, whose weights are a_i (as we restricted our predictors to the set of linear predictors) and which can only take in training data identified from A_i, and the mean of the vector x can be written in the form:

$$\max_{x_1,\ldots,x_n \in [-1,1]} \frac{1}{m} \sum_{i=1}^{m} ((a_i - b_i)^T x)^2$$

$$= \max_{x_1,\ldots,x_n \in \{-1,1\}} x^T \left(\frac{1}{m} \sum_{i=1}^{m} (a_i - b_i)^T (a_i - b_i) \right) x \tag{1}$$

where in (1) we simply expanded the squared term, and converted the problem from the interval to a tuple, as the maximum of the semidefinite form as a function of x must occur on the endpoints.

The conversion of the optimization problem from an interval to a tuple is very subtle but it reveals the difficulty of the problem at hand. Optimizing over integers is NP-hard whereas, under certain conditions, optimizing over intervals is solvable in a polynomial function of the number of variables. The intuition is that intervals, as convex sets, add structure that can be exploited by optimization algorithms.

This problem, in the second formulation, is now the Max-Cut and the more general Grothendieck problem, and we can now leverage the vast literature that have concerned themselves with these problems. The immediate upshot of this literature is that these problems are NP-hard.

This is a pretty depressing result, since it is very difficult to even verify the performance of any predictor function, much less find a good one. This

[1] A notable exception is policy evaluation in reinforcement learning.

[2] More technically what is called the "realization" of a random variable.

result seems to have far ranging consequences for statistics, as doing a worst case analysis of the mean squared error, ubiquitous in statistics, is an NP-hard problem. This might very well be the reason why statistics generally does not do such "worst-case data" analysis.

We can therefore only hope to compute approximations of the performance (which as we'll see, is within a $\pi/2$ factor of the optimal), and the bounds of this paper are near as tight as they can be (assuming no innovations regarding P vs. NP).

3 Optimization Formulation of Max-Cut and Grothendieck Problem

Here we emphasize the mathematical programming formulations of the Max-Cut problem, as opposed to its geometric or graph formulation, for which there are many resources available [2, 3].

In the language of mathematical optimization, given a graph consisting of vertices V and edges E the Max-Cut problem is, (thanks to Rajat Mittal's lecture notes [2]),

$$\begin{aligned} \underset{y_1,\ldots,y_n}{\text{maximize}} \quad & \sum_{(i,j)\in E} \frac{1 - y_i^T y_j}{2} \\ \text{subject to} \quad & y_i, y_j \in \{-1, 1\}\ i = 1, ..., n \end{aligned}$$

This is an integer programming problem, which is NP-hard. A strategy here is to convert our problem into one that is easier to solve, then to round that solution back into the original domain of the problem. As a result, we can try to relax the integer constraint to an interval constraint.

$$\begin{aligned} \underset{y_1,\ldots,y_n}{\text{maximize}} \quad & \sum_{(i,j)\in E} \frac{1 - y_i^T y_j}{2} \\ \text{subject to} \quad & y_i \in [-1, 1] \qquad i = 1, ..., n \end{aligned}$$

While this is a linear program that can be solved in polynomial runtime, this unfortunately does not round too well back into the integer domain. Taking this principle further, we now instead consider vectors rather than real numbers on the interval.

$$\begin{aligned} \underset{y_1,\ldots,y_n}{\text{maximize}} \quad & \sum_{(i,j)\in E} \frac{1 - y_i^T y_j}{2} \\ \text{subject to} \quad & ||y_i|| \leq 1 \qquad i = 1, ..., n \end{aligned}$$

This is a semidefinite programming problem, which is a generalization of linear programming, and can be solved in polynomial time. However, unlike the linear programming problem above, rounding back into the integer domain is much more accurate here. We do not go into detail into how Goemans and

Williamson round back into the integer domain, but their routine ends up in achieving a performance within $\frac{\pi}{2}$ of the optimal integer programming max-cut formulation. It also turns out this technique also achieves the same guarantees for the more general Grothendieck problem, as proven by the authors in Proposition 1.

Converting our original problem to the semidefinite relaxation as stated in (1) (noting that we are now optimizing over vectors v, not scalars x) we yield,

$$\begin{array}{ll} \underset{v_1,\ldots,v_N}{\text{maximize}} & \sum_{j,k=1}^{N} {v_j}^T \left(\frac{1}{m} \sum_{i=1}^{m} (a_i - b_i)^T (a_i - b_i) \right) v_k \\ \text{subject to} & ||v_j|| \leq 1 \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad j = 1, ..., n \end{array}$$

This is equivalent to the following matrix formulation, where $V_{j,k} = v_j^T v_k$ and $h : \mathbb{R}^N \times \mathbb{M}_{n,n}(\mathbb{R}) \to \mathbb{R}$ by $h(a_{s_i}, V) = \sum_{i=1}^{m} \sum_{j,k=1}^{N} (a_{s_i} - b_{s_i})_j (a_{s_i} - b_{s_i})_k V_{(j,k)}$.

$$\begin{array}{ll} \underset{V}{\text{maximize}} & h(a_{si}, V) \\ \text{subject to} & V \succeq 0 \\ & V(j,j) \leq 1 \; j = 1, ..., n \end{array}$$

where $V \succeq 0$ denotes V has eigenvalues greater than 0 (so the set of positive semidefinite matrices). Let $Z = \{V \in \mathbb{M}_{n,n}(\mathbb{R}) | V \succeq 0, V(j,j) \leq 1\}$ to avoid writing the constraints every time. $\mathbb{M}_{n,n}(\mathbb{R})$ denotes the set of all $n \times n$ matrices with real valued entries.

Since the objective is convex, and our constraints are over convex sets, this problem is able to be solved in polynomial time.

4 Selecting the Best Linear Algorithm

Now that we can compute our performance to a multiplicative $\pi/2$ factor in polynomial time, we turn to finding an algorithm that can actually perform well. Surprisingly, the difficulty of the problem is contained within evaluation of performance, as opposed to selecting the best performing inputs, which, as we'll soon see, can be computed with standard programming techniques after some clever massaging and matrix algebra. The goal is now,

$$\min_{a_i} \max_{V \succeq 0, V_{(j,j)} \leq 1} h(a_{s_i}, V)$$

In other words, we seek a set of weights for the training data, a_i (or equivalently a linear function) such that it is closest to estimating the mean of the test set, on worst possible data. The authors propose the following algorithm to find the weights a_i.

Algorithm 2 Sampling algorithm to approximate the best linear estimator

Input: Accuracy parameter $\epsilon > 0$; t random samples from the joint distribution of training and test sets, $(A_{s_1}, b_{s_1}), \ldots, (A_{s_t}, b_{s_t})$, where each $A_i \subset \{1, \ldots, n\}$ is the set of training set indices in the i^{th} case and each b_i is a vector with uniform values over the test set in the i^{th} case as in Definition 4; and the actual instance to predict, specified by (A, b) and the data x_A.

For an $n \times n$ matrix V and a set $A_i \in \{1, \ldots, n\}$, let V_{A_i} denote V restricted to the rows in A_i, and let V_{A_i,A_i} denote V restricted to *both* rows and columns in A_i.

1. Compute the concave maximization

$$\widetilde{V} = \underset{V \succeq \epsilon, V_{j,j} \leq 1}{\arg\max} \sum_{i=1}^{t} b_{s_i}^T (V - V_{(A_{s_i})}^T V_{(A_{s_i},A_{s_i})}^{-1} V_{(A_{s_i})}) b_{s_i} \tag{10}$$

2. Output the estimate $x_A \widetilde{V}_{(A,A)}^{-1} \widetilde{V}_{(A)} b$.

An immediate question is why are the eigenvalues greater than ϵ as opposed to 0? As we'll see in Lemma 3, the reciprocal of the eigenvalues will bound the derivative of the quadratic form inside the sum, which is critical in our approximations, since $\frac{1}{\epsilon}$ for $\epsilon \to 0$ is unbounded. Thankfully, by Lemma 2, this approximation only makes us an additive constant away from the optimal value.

The proof of polynomial algorithmic runtime, and of convergence to the true solution are coupled, since the analytic structure of the problem will allow us to formulate the problem in a way that can be solved by standard mathematical programming techniques in polynomial time.

We begin with Sion's generalization of Von Neumann's minimax theorem [4]. Here we are allowed to interchange the min and max operations,

$$\min_{a_i} \max_{V \succeq 0, V_{(j,j)} \leq 1} h(a_{s_i}, V) = \max_{V \succeq 0, V_{(j,j)} \leq 1} \min_{a_i} h(a_{s_i}, V)$$

The conditions of the minimax theorem are satisfied, as the domains of both optimizations are convex, convex in $h(\cdot, V)$ for fixed $V \in Z$, and concave in $h(a_{si}, \cdot)$ for fixed a_{si}. For proofs of these claims please refer to the paper.

In a sense, satisfying the conditions of the minimax theorem is more important than the actual application of the theorem, since it is now revealed that we have a min-max optimization which has a lot of structure and can be done in polynomial time. However, we proceed with the switch, and minimize the inside because this will help with the analysis down the line (and in the case where we have samples as opposed to a full description.

Rewriting in vector notation, we have, for fixed s_i, $h(a_{s_i}, V) = \nabla_{a_{si}} (a_{si} - b_{si})^T V (a_{si} - b_{si}) = 2V(a_{si} - b_{si})$. Setting this to 0, a solution, that also satisfies the constraint that $(a_{si})_j = 0$ for $j \notin A_{si}$, involves using row restrictions (or submatrices). The intuition here is that we set the rows that do not correspond to training data, A_{si}, to a row of 0s, and for this end we introduce $V_{A_{s_i}}$ to mean V row-reduced to only the indices in A_{si}, while $V_{(A_{s_i}),(A_{s_i})}$ to mean V both row and column reduced to the indices in A_{si}.

Explicitly, the solution $a^*_{s_i} = V^{-1}_{(A_{s_i}),(A_{s_i})} V_{(A_{s_i})} b_{s_i}$. Note the function h seen as a function of a_{si} is convex. Hence, a^*_{si} is a global optimum for the constrained problem if it satisfies the two relevant KKT conditions: 1) feasibility $(a^*_i)_j = 0$ for $j \notin A_i$ and 2) stationarity $\nabla_{a_i} h(a^*_i) = 0$.

To see that $(V^{-1}_{(A_{s_i}),(A_{s_i})} V_{(A_{s_i})} b_{s_i})_j = 0$ for $j \notin A_{si}$. Note that the j-th coordinate of $(V_(A_{si}))_j$ is equal to 0, by definition of the row restriction to A_{si}. Second, to see $2V(a^*_{s_i} - b_{si}) = 0$, rearrange to yield $VV^{-1}_{(A_{s_i}),(A_{s_i})} V_{(A_{s_i})} b_{si} = Vb_{si}$. The analysis can be simplified if we analyze each coordinate j separately as opposed to the vector as a whole. We can split this into two cases. Case 1: $j \in A_{si}$. Immediate that $(b_{si})_j = 0$, so both sides are equal. Case 2: $j \notin A_{si}$. I think it's because $(VV^{-1}_{(A_{s_i}),(A_{s_i})} V_{(A_{s_i})})_j = V_j$, yielding the equality on both sides.

If our sample indexed by s_i consists of all possible partitions of P, we have actually just proven our algorithm finds a linear predictor within $\frac{\pi}{2}$ of optimal, which is formalized in the following corollary.

Corollary. *Let $a^*_{s_i} = V^{-1}_{(A_{s_i}),(A_{s_i})} V_{(A_{s_i})} b_{s_i}$ (as above), where we have access to all partitions of P, and let $a^{**}_{s_i}$ be the optimal set of weights. Then*

$$\mathop{\mathbb{E}}_{(A,B)\sim P} \Big(\sum_{i=1}^{|A|} a^*_{s_i} x_{A_i} - \frac{1}{|B|} \sum_{i=1}^{|B|} x_{B_i}\Big)^2$$

$$\leq \frac{\pi}{2} \Big[\sup_{(x_1,\ldots,x_n):||x_i|\leq 1} \Big(\mathop{\mathbb{E}}_{(A,B)\sim P} \Big[\sum_{i=1}^{|A|} a^{**}_{s_i} x_{A_i} - \frac{1}{|B|} \sum_{i=1}^{|B|} x_{B_i}\Big]^2\Big)\Big]$$

Note we implicitly used the assumption that we have knowledge of the full partition distribution P (or equivalently, all the possible train-test splits (A_{si}, B_i)), through how we defined $a^*_{s_i}$. However, this might be unrealistic, as in certain scenarios we don't have full access to P, or P's support is too large. At this point, we proceed with using concentration inequalities to bound the probability of sample means diverging from the true expectation.

4.1 Introducing Stochasticity (Or: Epsilon Nets to the Rescue!)

If s_i is not all the possible partitions of P, then we can try to probablistically bound how much the samples deviates from their mean by usage of concentration inequalities. To avoid writing the quadratic product every time, let us define a function $q : \mathbb{M}_{n,n}(\mathbb{R}) \times \mathbb{N} \to \mathbb{R}$ where $q(V, A_{s_i}) = b^T_{s_i}(V - V^T_{A_{si}} V^{-1}_{(A_{si},A_{si})} V_{A_{si}}) b_{s_i}$.

If our function is bounded between $[0, 1]$, we can apply the Chernoff-Hoeffding bound. By Lemma 1 of the paper, it is indeed.

Lemma. *(Lemma 1 of the paper)*
For fixed $V \in Z$ and A_{si}, $0 \leq q(V, A_{si}) \leq 1$

Proof in paper.

Hence, by Chernoff-Hoeffding, for a single fixed $V \in Z$,

$$P(\frac{1}{n}\sum_{i=1}^{t} q(V, A_{s_i}) - \mathop{\mathbb{E}}_{(A,B)\sim P} q(V, A_{s_i}) \geq \epsilon) \leq 2e^{-2t\epsilon^2}$$

This is a first step. However, we want to bound this event for all possible matrices $V \in Z$ simultaneously. However, how do we guarantee this for not only one matrix, but all matrices in Z? A basic approach is to simply use the union bound, i.e. $\cup_{i=1}^{N} P(A_i) \leq \sum_{i=1}^{N} P(A_i)$. Unfortunately, this set of such matrices is an infinite set, so $N = \infty$, and the RHS side may diverge, leading to a useless bound.

One strategy is to use the ϵ'-net. An ϵ'-net for Z is a set of matrices with the property that for any arbitrary matrix $V \in Z$, there is at least one element within the net that is at most ϵ' away from V. Luckily, in this case the ϵ'-net of Z is finite (it has cardinality $e^{poly(n)/\epsilon'}$), and it allows us to apply the basic union bound now, with the guarantee the RHS sum does not diverge. Letting $\epsilon' = \epsilon^3/poly(n)$,

$$\bigcup_{V \in \epsilon'\text{-}net(Z)} P\Big(\frac{1}{n}\sum_{i=1}^{t} q(V, A_{s_i}) - \mathbb{E}q(V, A_{s_i}) \geq \epsilon\Big) \leq$$

$$\sum_{V \in \epsilon'\text{-}net(Z)} P\Big(\frac{1}{n}\sum_{i=1}^{t} q(V, A_{s_i}) - \mathbb{E}q(V, A_{s_i}) \geq \epsilon\Big) \leq e^{poly(n)/\epsilon'} e^{-2t\epsilon^2}$$

If we let the number of samples $t = \frac{poly(n)}{\epsilon'\epsilon^2}$, we yield,

$$\leq e^{poly(n)/\epsilon'} e^{-2(poly(n)\epsilon^2/\epsilon'\epsilon^2)} = e^{poly(n)/\epsilon'(1-2)} = e^{-1/\epsilon^3} \leq \epsilon$$

Now all of the matrices in the ϵ'-net have probabilistic guarantees. How do we go from guarantees for ϵ'-net(Z) to Z itself? As we'll see in more detail in Lemma 3, we use a combination of the boundedness of the derivative of the quadratic form, and the ϵ' closeness of the matrix and its ϵ'-net approximation.

(At this point you may have notice we have been using "ϵ'-closeness" of two matrices very loosely. We need to define a metric on matrices to make rigorous what "closeness" actually means. But ultimately, the particular choice of a metric does not matter, as in any finite dimensional vector space, all norms are equivalent up to a multiplicative constant, i.e. for any two norms $||\cdot||$ and $||\cdot||_*$ $\exists C, D \in \mathbb{R}$ such that $C||V - V'|| \leq ||V - V'||_* \leq D||V - V'||$)

The last thing we need is to bound the derivative of q, when seen as a function of V. This will come in handy because we can then argue, for a matrix V' that

is ϵ'-near to V, if the derivative of q does not move too much, then $q(V', A_{si})$ and $q(V, A_{si})$ are close to one another.

Lemma. *(Lemma 3 in paper)*
Fix $V \in Z$. $\frac{\partial}{\partial V_{j,k}} q(V, A_{si}) \leq \frac{poly(n)}{\epsilon^2}$

Proof. It is useful to reiterate why Lemma 3 is able to bound the derivative of the quadratic form. $V \succeq \epsilon \implies V_{(A_{si}),(A_{si})} \succeq \epsilon \implies V^{-1}_{(A_{si}),(A_{si})} \preceq \frac{1}{\epsilon}$. By matrix calculus, $\frac{\partial}{\partial V_{j,k}} V^{-1}_{(A_{si}),(A_{si})} = (v_j^{-1})^T (v_k^{-1})$ where v_j^{-1}, v_k^{-1} are columns of $V^{-1}_{(A_{si}),(A_{si})}$. Note that each of these columns are individually bounded by $\frac{1}{\epsilon}$. This then implies $||\frac{\partial}{\partial V_{j,k}} V^{-1}_{(A_{si}),(A_{si})}|| = ||(v_j^{-1})^T (v_k^{-1})|| \leq \frac{1}{\epsilon}\frac{1}{\epsilon}$. Then, we have $\frac{\partial}{\partial V_{j,k}} q(V, A_{si}) \leq \frac{poly(n)}{\epsilon^2}$, which is the claim of the lemma (how do I get to this last step from the second last step?).

□

So how can we put the boundedness of the derivative to good use? A function with bounded derivative is Lipschitz continuous, which means the distance of the function evaluated at two inputs is a multiplicative constant of the distance of its inputs (which is ϵ). To see this explicitly, note that ϵ'-net(Z) is an open and convex set (a neighborhood of a convex set is still convex). With complete abuse of notation, by generalization of the mean value theorem to multiple valued inputs, for fixed A_{si}, there exists some V^* in ϵ'-net(Z), such that,

$$|q(V, A_{s_i}) - q(V', A_{s_i})| = \frac{\partial}{\partial V_{j,k}} q(V^*, A_{si}) ||V - V'||$$

By Lemma 3, we know the derivative is bounded. Also if V' is the ϵ'-net approximation of V, then the norm $||V - V'||$ is also bounded by ϵ'.

$$\leq \frac{poly(n)}{\epsilon^2} \frac{\epsilon^3}{poly(n)} = \epsilon$$

We have all the ingredients necessary to prove our bounds. For a fixed training sample A_{s_i},

$$\text{Let } \tilde{V} = \underset{V \succeq \epsilon, V_{j,j} \leq 1}{\text{argmax}} \frac{1}{N} \sum_{s_i} q(V, A_{s_i})$$

$$\text{and } \hat{V} = \underset{V \succeq 0, V_{j,j} \leq 1}{\text{argmax}} \underset{(A,B) \sim P}{\mathbb{E}} q(V, A)$$

.

Note the two differences between the two, the former computes the argmax over a sample of the partitions whereas the latter computes it over the whole support of P. Furthermore, the latter requires matrices with non-negative eigenvalues, whereas the former is a subset of these matrices with all eigenvalues at least ϵ.

$$\underset{(A,B)\sim P}{\mathbb{E}} q(\tilde{V}, A_{s_i}) \underset{\text{Lemma 3}}{\geq} \underset{(A,B)\sim P}{\mathbb{E}} q(\tilde{V}', A) - \epsilon \underset{\text{Chernoff-Hoeffding}}{\geq} \frac{1}{N}\sum_{s_i} q(\tilde{V}', A_{s_i}) - 2\epsilon$$

$$\underset{\text{Lemma 3}}{\geq} \frac{1}{N}\sum_{s_i} q(\tilde{V}, A_{s_i}) - 3\epsilon \underset{\tilde{V} \text{ is max}}{\geq} \frac{1}{N}\sum_{s_i} q(\hat{V}', A_{s_i}) - 3\epsilon$$

$$\underset{\text{Chernoff-Hoeffding}}{\geq} \underset{(A,B)\sim P}{\mathbb{E}} q(\hat{V}', A_{s_i}) - 4\epsilon \underset{\text{Lemma 3}}{\geq} \underset{(A,B)\sim P}{\mathbb{E}} q(\hat{V}, A_{s_i}) - 5\epsilon$$

Finally, we return to the perennial eigenvalue question. By restricting our eigenvalues to ϵ as opposed to 0, how far off are we from our solution? Thankfully, this lemma says "not much".

Lemma. *(Lemma 2 in paper)*

$V^* = \arg\max_{V \succeq \epsilon} q(V, A)$ *is at most* ϵm *away from* $V^{**} = \arg\max_{V \succeq 0} q(V, A)$

Proof in paper.

Hence, $q(\hat{V}, A_{s_i})$ is at most ϵ away from the optimal matrix when the eigenvalue constraint is relaxed from ϵ to 0. Finally, this is itself within $\frac{\pi}{2}$ of the optimal linear estimator, concluding our proof.

5 Discussions and Future Directions

One direction is to directly incorporate how divergence between the test and train distributions affects the learning problem. This connection is only made in terms of the worst-case data, but it is not immediate what the worst-case data implies how different the two distributions are. We already know the two extreme cases, that fully divergent distributions cannot learn; and that two fully overlapping distributions are only subject to standard iid restrictions. Can we more generally explicitly model a tradeoff between how different the two distributions, and data required to achieve a given level of approximation (For example, very different train and test distributions would require more data to achieve a given level of approximation). Maybe taking this perspective of the problem liberates us from Max-Cut, and we can therefore have more practically useful bounds.

Of course, another way that might obviate the P vs. NP issue is to use an average case analysis as opposed to a worst-case analysis. While this would likely introduce distributional assumptions on our data, unlike the current setting, it might be worthwhile to get practically better bounds and as a way to have tighter bounds that are not dependent on potentially pathological worst-case data.

Another direction centers on the interpretation of the partition function P. What are the connections between P and the propensity score in statistics? Can the propensity score operate as an approximation of P? Is conditional independence on covariates as restrictive as full knowledge of the partition distribution P? Is there a connection to the causal inference literature in statistics? Is it as simple as relabeling test/train – standard nomenclature in machine learning – to treatment/control?

References

[1] J. Y. Chen, G. Valiant, and P. Valiant, "How bad is worst-case data if you know where it comes from?" *CoRR*, vol. abs/1911.03605, 2019. [Online]. Available: http://arxiv.org/abs/1911.03605

[2] R. Mittal, "Lecture 10: Approximation algorithm for max cut." [Online]. Available: https://www.cse.iitk.ac.in/users/rmittal/prev_course/s18/reports/10gw.pdf

[3] P. Harsha, "Lec. 3: Maxcut and introduction to inapproximability." [Online]. Available: http://www.tcs.tifr.res.in/~prahladh/teaching/2009-10/limits/lectures/lec03.pdf

[4] M. Sion, "On general minimax theorems." *Pacific J. Math.*, vol. 8, no. 1, pp. 171–176, 1958. [Online]. Available: https://projecteuclid.org:443/euclid.pjm/1103040253