Model Complexity for Supervised Learning: Why Simple Models Almost Always Work Best, And Why It Matters for Applied Research*

Marco Morucci

Department of Political Science, Michigan State University,
moruccim@msu.edu

Arthur Spirling

Department of Politics, Princeton University
arthur.spirling@princeton.edu

Abstract

Inspired by other fields, political scientists have embraced the use of supervised learning for prediction, inference, measurement and description. In doing so, they typically use flexible models of considerable complexity that have proved successful in non-social science settings. Yet there appear to be profound limits to the payoff of such approaches, at least relative to the alternative of using very simple (generalized linear) models for such tasks. We explain why this is, how to identify the problems for which this will be true, and what to do about it. We show that the *intrinsic dimension* of political science data is low, and this means returns to complexity are muted or non-existant. We provide a theory of "data curation" to explain this state of affairs. Our approach allows us to diagnose when simple models are optimal, and to provide advice for practitioners seeking to use machine learning.

*First version: April 1, 2024. This version: September 27, 2024. Thanks to Xilin Yang for excellent research assistance. We received very helpful comments from Dave Armstrong, Mike Burnham, Carlos Cinelli, Nicholas Cole, Ray Duch and Seo Eun Yang on an earlier draft. We are grateful for feedback from audiences at APSA, LSE, MapleMeth, MPSA, Polmeth, University of Oxford.

1 Introduction

Machine learning is not new to political science, but recent times have seen a surge in interest (see, e.g., Arnold et al., 2023; Grimmer et al., 2022, for recent overviews). Focussing specifically on *supervised* learning—that is, situations where one has (pre) labeled data and the task is to predict well—we see researchers applying modern techniques to a broad range of problems and data types, including classification (Torres and Cantú, 2022), matching processes (Acharya et al., 2022) and polarization (Goet, 2019). The promise of these approaches is beguiling. At best, the claim is that with relatively little thought about the ideal form of the model for the outcome, we can nonetheless provide accurate predictions for very complex relationships. This logic has been extended to studying vital human-centered systems like wars (Colaresi and Mahmood, 2017), justice decisions (Ben-Michael et al., 2021) and family welfare policy (Salganik et al., 2020).

Yet the uptake of such approaches has been slower than might be expected given this excitement. This may be because the primary focus of much political science work is (causal) inference, rather than prediction *per se*, though prediction is a pre-requisite for many causal inference problems (see e.g., Cranmer and Desmarais, 2017; Chernozhukov et al., 2018). Or it may be that scholars would use more complex techniques if they were familiar with them and saw their power—an idea that motivates treatments by e.g. Montgomery and Olivella (2018). An alternative story is that these models simply do not work as well as hoped: that is, there appear to be few performance benefits over basic baseline models (e.g. De Marchi et al., 2004; Kapoor and Narayanan, 2023). Combined with concerns about interpretation and worries about the ethics of "black box" approaches for the real world (e.g. Rudin, 2019), we contend that the use of new flexible machine learning algorithms is disincentivized for the median political scientist. This is a concern because despite having important questions related to prediction and inference, political science has been "left behind" as machine learning and deep learning revolutionizes related fields (e.g. LeCun et al., 2015). This paper explores this impasse, explains why we are here, and what we should do about it.

We will argue that the primary problem with modern machine learning methods is that they are a poor match for most of our datasets in political science. While the 'true' data generating process for political outcomes is likely very complicated, the datasets that arrive to us are not. Specifically, the tabular data we typically use simply does not exhibit sufficient subtleties or the non-linearities that complex models are designed to exploit. Our first contribution is an empirical one. We show that much of our end-use data in the

discipline has low *intrinsic dimension*. This means that a small number of variables in a simple model do approximately as well at predicting the relevant target as a large number of variables in a complex model. We will introduce some new methods from other disciplines (e.g. Li et al., 2018) to estimate how "reducible" a given data set is, and provide some innovation on those methods for political science purposes. Essentially, this involves fitting many models to different random subsets of the data, and increasing the number of parameters (the complexity) in each case. For each increase in the complexity, we report a plausible upper bound on the performance of the implied model involving that number of parameters. The key diagnostic then is where in complexity terms—i.e. what smaller number of parameters—model performance is (almost) as good as it can get relative to fitting a very large number of parameters. For a set of well-known data sets from Comparative Politics, International Relations, American politics and sociology we will show that this number is relatively low. Put otherwise: complex models will typically offer *some* improvement over very simple ones, but that gain is often negligible—and can be zero or even negative. Crucially, we contrast our discipline's data to that found in areas like computer vision where unstructured "raw" input such as image pixels are more common. Those datasets *do* show heavy returns to complexity. The difference can be an order of magnitude or more in terms of the number of weights needed for a statistical model to fit the datasets (political science v computer science) equally well.

Our second contribution is a formal argument for why this is. This requires some theory on *data curation*—that is, how researchers put data sets together in the real world in order to model (very complicated) phenomena. Our contention is that scholars think in fundamentally linear ways when pulling in variables that they hypothesize are related to outcomes. This makes sense: political science is a theory-heavy discipline, and gathering data is subject to a cognitive and financial budget constraint. Moreover, work correctly advising against models with large and uninterpretable sets of coefficients on "control" variables for (causal) inference reasons (e.g. Achen, 2005) is misinterpreted in the prediction context. The consequence is that it is almost impossible to improve over simple models—like linear regression and logit—that capture the fundamental nature of the relationship between X and Y. The exception to this is when data is 'raw' and uncurated for regression-style problems—text, audio and visual data has this form. We provide a typology of machine learning data types to explain this difference. In particular, we argue that most machine learning problems can be broken up along two dimensions—their true complexity, and the amount of noise in the data. The issue in political science is that the data sets we use contain a great deal of noise, but the true complexity of the data is low through curation. Thus complex models have nothing to do, and little to learn.

To reiterate: this does not mean that political processes are inherently simple or linear; our point is that the data as it arrives to us for modeling purposes has those properties.

Our third contribution is to propose a way forwards given the above. Specifically, we offer high-level advice as to where the payoff from complex modeling is most likely to exist and be large. This affects both how one should model extant data, but also how it should be gathered.

In the next section, we document the fact that complex models rarely outperform simple ones in social science. In Section 3 we introduce methods for estimating intrinsic dimension, and show how to adjust these ideas for political science problems. In Section 4 we apply these routines to various political science and sociology problems such as predicting war and the *Fragile Families* challenge. We demonstrate that simple models rule the day. In Section 5 we propose a new theory of *data curation* that might drive this general pattern. We then provide some advice to practitioners in the field. In Section 7 we conclude.

2 Complex Models Fail to Outperform Simple Ones in Social Science

To fix ideas, we will use the term "simple" to refer to generalized linear model (GLM) approaches for prediction problems. We mean specifically the logistic regression (typically fit by maximum likelihood) for a binary outcome, and the classical linear regression model (fit via ordinary least squares) for a continuous one.[1] We will use the term "complex" to refer to more flexible, non-GLM approaches such as random forests, neural networks, support vector machines and so on. We acknowledge this labeling is somewhat arbitrary, insofar as e.g. a decision tree may be very parsimonious in practice; but the "simple" vs "complex" distinction nonetheless captures how practitioners view and discuss these techniques and how they teach them. Note that we are referring to the simplicity (or not) of *models* here, not the data *itself*. For that purpose "complexity" is even less well-defined, though we shall impose some structure below.

In a helpful review article, Montgomery and Olivella (2018) clarify the circumstances in which analysts should reach for more complex machine learning tools—random forests in their case—as opposed to simpler approaches. They argue that "traditional statistical" (GLM) techniques "are often insufficiently flexible to capture nuances in the data—such as complex nonlinear functional forms and deep interactions—when no clear a priori expectations exist." They note that modern supervised learning approaches avoid these

[1] In principle, we could also include other GLMs like the probit or the Poisson regression, but we do not see them used much for prediction in political science. In addition, we would include common variants on the logit, such the multinomial or ordinal logistic regression.

drawbacks and are thus a useful alternative for high-dimensional data. We do not dispute these points or similar ones made by many other authors. But we have two observations. First, to the extent supervised learning models have become more widely used, the improvements over standard tools have been mixed.[2] This is most obvious for direct prediction problems where the task is to accurately forecast a target variable because doing so has intrinsic value.

Second, and a point that Montgomery and Olivella (2018) themselves make with respect to tree-based models, these techniques have not proved particularly popular in political science—especially relative to their use in computer science and statistics. Indeed, in a survey of articles in three top political science journals (2016–2021), Arnold et al. (2023) find just 65 papers that used "machine learning", many of which are, in fact, discussing unsupervised techniques.[3]

One explanation for low uptake of machine learning in our discipline is that those methods are primarily for *prediction*, whereas political science research is more concerned with (causal) *parameter estimation*. We disagree. First, because these things are not alternatives, nor in competition. For example, methods that convert predictions to estimates with good statistical guarantees have existed for some time (see,e.g., Ratkovic, 2023). These approaches perform better than standard parametric models in many settings (Fuhr et al., 2024) and we would thus expect to see them used more often for this purpose. Second, of course as a narrow empirical matter, there are many published works that do focus entirely on prediction (for example the papers that we replicate in Section 4). So prediction is not unimportant or ignored. Finally, we contend that the reason why prediction as a task itself has not been embraced more by the field is precisely *because* no model, simple or complex, seem to perform adequately well in predicting many social science outcomes. We give two examples of this phenomenon now.

2.1 *Fragile Families* and Predicting Conflict

The first is the *Fragile Families Challenge*. This is a collaborative effort that assessing whether life outcomes for an individual can be predicted from data collected during the individual's childhood (Salganik et al., 2020). Between March and August 2017 160 researchers submitted predictions for six childhood development outcomes made by fitting models to the same training data, using a wide array of statistical and machine learning methods—both simple and complex models. The predictions were then scored on the

[2] This is not unique to social science; see Nusinovici et al. (2020) for an example in medicine, Vallejos and McKinnon (2013) for geology, Shwartz-Ziv and Armon (2022) more generally.

[3] And surveying the AJPS for this paper, we found just 16 cases of applied supervised learning for the period 2016–2023.

same separate test dataset. Results from the evaluation showed that, first, all models give very similar predictions, meaning that the predictions output by a complex algorithm on the dataset are not much different from those output by simple logistic regression (Salganik et al., 2020). Second, all these predictions are bad insofar as the highest out of sample R^2 achieved across all six outcomes was around 0.23, while the lowest around 0.03. (Salganik et al., 2020, Fig. 3).

The second is the prediction of instances of conflict both within and across countries. This is obviously vital for many policy decisions, and has been of interest applied political scientists for many years (see, e.g., Fearon and Laitin, 2003). Researchers have turned to machine learning methods of varying degrees of complexity for this task: from logistic regression (Ward et al., 2010), to Random Forests (Colaresi and Mahmood, 2017; Hill and Jones, 2014; Hegre et al., 2019), deep Neural Networks (Beck et al., 2000; Brandt et al., 2022), and models specific for network-connected data (Cranmer and Desmarais, 2017; Minhas et al., 2016; Dorff et al., 2020). While improvements have been made, competitions have found that predictions made by a simple benchmark model often lead to equal/better error than many more complicated models (Vesco et al., 2022). And to the extent that those more complex models have historically outperformed simpler ones there is evidence that the gain is small (De Marchi et al., 2004), if it exists at all (Kapoor and Narayanan, 2023).

One response might be that such "narrow" prediction problems are not what machine learning is most useful for in social science—instead, such techniques are for intermediate purposes like generating propensity scores or for measurement (Knox et al., 2022; Montgomery and Olivella, 2018). But we take this as a frank admission that machine learning—with complex models—has not revolutionized political science practice in a way consistent with other disciplines. And it still seems plausible that part of this is a lack of improvement over simpler models. But why is this? That is, why do simple models continue to perform so well *relative* to more complex alternatives, albeit sometimes underwhelming in an absolute sense?

2.2 Why don't complex models do better?

One set of arguments, advanced by Rudin (2019) and others, begins with the observation that many problems in real data are such that very different models perform approximately similarly (see also Hand, 2006). A reason for this is that the data is finite, and thus there are some parts that are not perfectly modeled, i.e. there is uncertainty. And if this "Rashomon set" of similarly performing models is generally large, then it likely contains simple models along with the more complicated options—though they may be difficult to locate

(Semenova et al., 2022). Note that in the Rudin (2019) case, simple models are axiomatically preferred because they are more *interpretable*, i.e. one can understand why the model makes the predictions it does.

A related empirical finding is that on tabular data specifically, models that are non-GLM but still relatively simple, like trees, tend to outperform very complex ones (like neural nets). The reasons for this are not clear, but one issue is that very high dimensional functions are too smooth and struggle in the presence of uninformative variables—that is, they get waylaid by features that do not help in predicting the target of interest (Grinsztajn et al., 2022). More generally, when observation numbers are small, nonlinear models need not improve over linear ones simply because there is insufficient data to estimate higher order terms (e.g. Strang et al., 2018). If these are characteristics shared by social science data, then it may explain why it has proved hard to do better than simple approaches.

Finally, there is evidence that there is more structure in the world than is initially appreciated: Goldblum et al. (2023), for instance, note that many very large, nominally unstructured, machine learning datasets can be heavily compressed. This implies hidden organization, and is compatible with earlier observations that the same quite basic models perform well across a very large range of applications (Fernández-Delgado et al., 2014). More directly: simple models do well even when we do not expect them to because the world has more order than it initially appears.

To summarize: we know that for many applied problems in (social) science, simple models do surprisingly well, sometimes better, than more complex ones. If one combines this observation with a preference for interpretable approaches, then it is hard to motivate anything outside the linear model family—at least as a first cut on a problem. On the other hand, we also see situations such as in the *Fragile Families Challenge*, where simple models do best but their overall performance is not particularly impressive. But we do not know exactly *why* this is—that is, what features of the data generating process in social science lead to this situation. Consequently we don't know how to generally diagnose when a more complex model will be useful—either in theory or in practice. This is our task in the next sections.

3 Political Science Data has Low Intrinsic Dimension

Our central claim is that almost all political science tabular data has low *intrinsic dimension*. We will be more formal about this momentarily. For now, the intuition is that the act of carefully compiling small numbers of covariates for observations leads to a situation where there is little structure left to learn. That is,

our data tends to be already so coarsened and structured that a small model, combining just a few variables in a basically linear way, does as good a job as any more complex technique. To be clear, the claim is not that low intrinsic dimension is by definition identical to outcomes being linearly additive functions of inputs; rather it is that these properties are commensurate and highly associated with each other. And again, it does not mean that the "true" complexity of political phenomena is inherently low; rather, we make it look low by the way we put datasets together.

3.1 Focus Datasets

We will show that this is true for a collection of datasets that represent all areas of the discipline and beyond, and that have all been analyzed with (typically complex) prediction techniques at least once in print. These data are:

1. ***Fragile Families*** from Salganik et al. (2020), as described above. There are three (binary) targets to predict: (household) Eviction, (primary caregiver) job training, (primary caregiver) layoff. We denote this data "FF" (eviction, job training, layoff) in what follows.

2. **Civil War**: the original data is from Muchlinski et al. (2016), though our source was Colaresi and Mahmood (2017). The problem is to "forecast the binary observation of the presence or absence of a civil war onset". We denote this data "Colaresi" in what follows.

3. **International Conflict**: the original data is from Beck et al. (2000). The problem is to predict a binary variable coded as 1 if a state is "engaged in an international conflict, 0 if it is at peace." We denote this data "Beck et al" in what follows.

4. **Political Instability**: the original data is from Goldstone et al. (2010). The problem is to forecast political instability, which can assume many forms including "the onsets of both violent civil wars and nonviolent democratic reversals". We denote this data "Goldstone et al" in what follows.

5. **American Politics**: the original data is from Blackwell (2013), though our source was Montgomery and Olivella (2018). The problem is to predict "whether a Democratic candidate has "gone negative" in a given week" of a campaign, for both incumbents and non-incumbents in US elections.

6. **Image Data**: the data is CIFAR-10 Krizhevsky et al. (2009) and the task is to correctly classify pictures into one of 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck).

The use of standard computer science datasets such as CIFAR-10 may be surprising, not least because classifying images is not a standard operation in political science (though see Torres and Cantú, 2022). But we include these data precisely for calibration reasons: we know that complex models such as convolutional neural nets have generated huge improvements for such problems. Given this, we suspect such data differs in important (i.e. intrinsic dimension) ways from the standard tabular arrangement in political science and thus provides a useful comparison benchmark.

3.2 Intuition: High Dimensional Problems May Only Require Getting a Few Things Right

We will use versions of the methods given in Li et al. (2018) to compute the intrinsic dimension of various problems in social science. To introduce those ideas we begin with an intuitive example. Suppose the task is to estimate the ideal points of legislators in a parliament. There might be 600 members and thus we seek to $\hat{\theta}_1, \hat{\theta}_2 \dots \hat{\theta}_{600}$ where each θ is a position on some left-right (single) dimension or membership of a binary-valued ideological grouping. The parameter vector to be estimated, $\theta^{(D)}$, has dimension $D = 600$—there is one slot available in it for each MP. Using only a roll call matrix of votes and no other information about party or bills, we will go from a initial parameter vector θ_0^D to a final parameter vector θ_*^D. We will do this via some model estimation routine, for instance a Bayesian item-response approach or clustering routine operating on the data matrix.

For the purposes of exposition, suppose that voting behavior in the legislature is highly cohesive and disciplined, perhaps due to ideology or some government-vs-opposition dynamic. So even though there are many MPs, there are just two evenly balanced voting blocs: 300 MPs who vote together one way and 300 who vote the opposite way on every single division (subject to perhaps very minor deviations to avoid identification problems when it comes to estimation). Consequently, while we have a parameter vector of length 600, each θ_i only takes one of only (approximately) two values.

The researcher realizes that if they can correctly identify the leaders of the two groups, and use them as end points of the IRT continuum or as centroids for the clusters, then the problem will be "solved" insofar no model will fit the data better than that one. But, of course, every MP within each group votes very similarly, so finding the two opposing *deputy* leaders will work just as well in terms of overall fit as finding the leaders. As will starting with the two opposing chief whips. In fact, so long as the researcher finds two members *not in the same group* to place as end points, they can do no better—from the perspective of fitting the data into the two (correct) clusters or along one latent dimension. But it will not work—the fit will be

non-optimal—if they take two members from the same group as end points of their dimension or centroids of their clusters.

In the language of Li et al. (2018), this problem is "highly redundant." Once the researcher has identified two members from opposite groups, there are many similar solutions which will result in a model that fits identically well. Indeed, the researcher can choose any one of the other 299 members from the first group, and any one of 299 members from the second (opposing) group and they will be no worse off. The original dimension of the problem was $D = 600$; but so long the researcher correctly identifies 2 slots in the parameter vector to build their model from, they can be no better off. In the original notation, this is

$$D = d_{\text{int}} + s$$

where the *intrinsic dimension* d_{int} is 2 and s is the dimensionality of the solution set (i.e. the number of other ways we could have set the problem up such that the performance is the same). Here that is 299+299=598. And thus we have $600 = 2 + 598$ as required.

3.3 Intrinsic Dimension in theory: optimization problem

The idea inherent in intrinsic dimension is much more general than the specific example of the ideal points. We are more formal about why in Supporting Information A. But we summarize the theoretical logic informally by supposing that the goal is to predict some outcome Y based on $\mathbf{X}$ covariates. As always, this prediction will be best when the loss—the difference between the true value of Y and what we predict it to be—is minimized. The problem is that we don't know what function (what model), f^*, will do this in practice. The assumption we and many others make is that this 'best model' can be found in the class of functions produced by a *deep neural network* (DNN), an extremely flexible technique that can approximate almost any surface of interest.

We will define the intrinsic dimension of a given data set as being the smallest number of weights that a DNN must have in order to perfectly represent the best function for the problem of predicting Y from X. This formulation makes the optimization problem theoretical tractable as a search over DNNs with different numbers of weights. Of course, minimizing the loss by optimizing weights is not unique to DNNs. So it is worth contrasting the idea here to the usual practice in political science. Suppose we had a simple three-variable logistic regression to predict some outcome. Traditional optimization would means finding

the 'best' values for the constant, $\hat{\beta}_1, \hat{\beta}_2$ and $\hat{\beta}_3$. For a DNN, it would mean finding the 'best' values for (i.e. ways for) combining and recombining linear and non-linear functions of the variables to predict the outcomes. Generally, such neural nets will require that we optimize many more weights than for the logistic regression. But the payoff is a more flexible and better fitting model. For a DNN, a model with D weights has all smaller models contained with in it, so as long as pick a large enough space we should find the optimal function for predicting Y within it.

The fact that a network uses a large number of D weights (say, 1000) and also fits well is *prima facie* evidence that the problem is complex. But per Li et al. (2018) and others (e.g. Goldblum et al., 2023) this need not be true: it could be that one could estimate a *much smaller* number of parameters and come remarkably close to the performance of a very big model. That is, we could imagine that there are many redundant weights, and that the intrinsic dimension is low. To see how this could be the case, consider again the example of a 3-variable logistic regression but assume that the true best predictor—i.e. the actual data generating process —only uses two of the variables, i.e., $f^*(\mathbf{x}) = \sigma(\alpha^* + \beta_1^* x_1 + \beta_2^* x_2)$, where σ is the sigmoid function. Here there are 3 learnable weights, so the intrinsic dimension of the problem is $D = 3$. Since we do not know this, we instead fit a logistic regression with all three variables plus the constant, resulting in a model with $D = 4$. The promise of the DNN is that if we try to minimize the loss we will encounter at least one solution in which $\alpha = \alpha^*, \beta_1 = \beta_1^*, \beta_2 = \beta_2^*$, and $\beta_3 = 0$. In this sense the coefficient β_3 is "redundant", and we can conclude that the intrinsic dimension of our problem is actually $D = 3$.[4]

In the case of the simple three β logistic regression in the previous section, we could proceed by trying one variable in the prediction, then two, then three, then the permutations thereof. So a total of 8 combinations (including one that has only an intercept). But in the case of a $D = 1000$ network this is clearly more challenging: first, even for the most naive case, there are many more thousands of permutations. But more importantly, the nature of these networks is that one cannot generally try a single combination at a time—later layers need estimates from earlier layers, which in turn need them from even earlier layers and so on. We give technical details in Supporting Information B, but suffice it to say it is generally not possible to optimize the relevant function directly or indirectly (i.e. just increasing the dimensional of the problem each time until the result is "good enough"). The next subsection describes how one *can* proceed.

[4]Note that, even if we fit a model with more weights than the true best predictor, the optimization routine will always result in at least one solution with all the redundant parameters set to 0. This means that a model class with D larger than the problem's intrinsic dimension will always contain the true best predictor.

3.4 Intrinsic Dimension in practice: Random Subspace Training

The Li et al. (2018) solution to this problem that we use is *random subspace training*. This differs to the conventional way to solve neural nets, which is to optimize the full D-dimensional set of weights at every training step. Instead, the intuition is that one takes a step for a random and much smaller subset of the full length-D parameter vector $\theta^{(D)}$. In the case of our parliamentary voting problem, rather than trying to optimize our $\theta^{(600)}$ ideal points at every step of estimation, we might instead try to randomly pick 2, update those and see how well we now fit the data with that model.

As a practical matter, training a random subspace starts with the following identity:

$$\theta^{(D)} = \theta_0^{(D)} + P\theta^{(d)}. \tag{1}$$

Here the 'full' parameter vector $\theta^{(D)}$ is set equal to the sum of two terms. The first is $\theta_0^{(D)}$, the vector of random initial values for the weights: this will not be trained, and is fixed for the duration of the estimation. The second is a matrix P multiplied by a much smaller parameter vector $\theta^{(d)}$. The P matrix is also random, and will not be trained. In contrast, $\theta^{(d)}$ will be updated, though it starts out as all zeros and thus $\theta^{(D)}$ begins as equal to $\theta_0^{(D)}$. Optimization proceeds by taking steps in d space and thus updating $\theta^{(d)}$. Because P and $\theta_0^{(D)}$ are not trained, each step will also update θ^D but from a smaller space. In this sense, P is a *projection matrix* that maps a larger vector of length D into a smaller one of length d.

3.5 Demonstration: Toy Example

To see how the basic idea of the projection might work, we consider an (admittedly contrived) example of a $D = 2$ dimensional problem that we can reduce to a $\theta^{(d)}$ in only one dimension. The practical situation might be a two weight neural network that we are trying to reduce to one weight. We will suppose that those weights would be estimated to be

$$\theta^{(D)} = \begin{bmatrix} 0.5 \\ -1.0 \end{bmatrix}$$

Using Equation (1) we need to randomly generate a $D \times d$ (2×1) matrix P. Suppose this is (randomly) drawn to be

$$P = \begin{bmatrix} -0.4 \\ 2.4 \end{bmatrix}$$

We also need a randomly drawn $D \times d$ matrix for $\theta_0^{(D)}$ which turns out to be

$$\theta_0^{(D)} = \begin{bmatrix} 0.3 \\ 0.2 \end{bmatrix}$$

Now, simply put: can we find a (one dimensional) value of $\theta^{(d)}$ which will allow Equation (1) to be true for the initialized matrices here? The answer is obviously yes and assuming we optimize properly, $\widehat{\theta^{(d)}} = -\frac{1}{2}$ in this case. That is

$$\begin{bmatrix} 0.5 \\ -1.0 \end{bmatrix} = \begin{bmatrix} 0.3 \\ 0.2 \end{bmatrix} + \begin{bmatrix} -0.4 \\ 2.4 \end{bmatrix} \cdot -\frac{1}{2}.$$

Of course, different initializations of P and $\theta_0^{(D)}$ would lead to different values of $\hat{\theta^{(d)}}$. But we would still obtain a lower-dimensional representation of the weights we learned with a more complex model.

3.6 Properties: theoretical and experimental

The method is straightforward and, as we have seen, the logic can be shown with toy examples. We will give more technical details about implementation below. But what guarantees do we have that it finds the correct (smallest) dimensionality of the problem at hand? There are two results presented in Li et al. (2018) and we paraphrase them here. First, if the d chosen is too small (relative to the true d_{int}) then it will generally be the case that no solutions will be found. This is good news insofar we will not end up thinking the problem is more redundant than it really is. On the other hand, when the d being examined is large enough, we should generally be able to find the correct value of d_{int}. This is also good news, though this is not guaranteed for all cases. While we would like firmer theoretical properties, our experiments—reported in Supporting Information C—suggest reasons for optimism. In particular, sweeping through various values of d using the methods here recovers "correct" results for social science data.[5]

3.7 Why Intrinsic Dimension?

Having explained how intrinsic dimension methods work, readers may nonetheless wonder why we don't use some other, simpler, technique. We fully recognize that there there are many possible measures of data

[5] A note here is that these intrinsic dimension techniques are a type of regularization *per se* so one doesn't need to or want to also regularize explicitly—Supporting Information C.2 has more information.

complexity (see e.g. Lorena et al., 2019). But our approach (courtesy of Li et al., 2018) has some distinct advantages. First, it doesn't measure complexity of the *data* (unlike, e.g. principal components); it measures complexity of *models* that fit the data well. This is closer to our estimand, if we want to understand why complex models (seemingly) under-perform. Second, DNNs—that underpin the approach—specifically have some nice features. For one, they can approximate a vast array of real data generating processes (linear and non-linear, simple and complex). And it is straightforward to compare across DNNs fit to the same problem just by counting their weights—one needs no further calibration (*cf* e.g. random forests where are there are several parameter choices to be summarized).

4 Results: The Intrinsic Dimension of Political Science Data is Low

We now show that, for a sample of benchmark datasets in political science, the intrinsic dimension of the model that best fits them is generally low, and there are rarely gains in performance from fitting more complex models to these datasets.

As explained in Section 3, our methodology is to get progressively closer to the true intrinsic dimension of a given prediction problem by starting with a very complex model class, which will likely contain the population model for the data, and then progressively shrink it via random subspace training until we start seeing substantial increases in test-set error. Specifically, we start with a "full size" model, and then move through eight smaller ones for dimensions $d = 50, 100, 250, 500, 1000, 2000, 3000, 5000$. As we do so, we record several performance statistics (Accuracy, Precision, Recall, AUC) on the test set for each intrinsic dimension. Supporting Information D gives more technical details on our exact procedure.

4.1 Focus Data Sets

Our results for the eight focus data sets we discussed above are displayed in Figure 1. Each column is a data set, and each row is a performance metric. The top row is accuracy, then precision, recall and area under the curve. These are defined in the usual way and in general, higher is better. That is, higher accuracy implies a model performing better, higher precision is better than lower precision and so on. The x-axis for every cell is in terms of (log scale) intrinsic dimension. As we move from left to right on that x-axis, we are increasing the "size" of the solution in the intuitive sense that we are estimating more and more parameters. More precisely, in keeping with our presentation in Section 3, as we move along the x-axis we are allowing

for a larger and larger parameter vector to approximate the 'true' number of parameters (*a priori* unknown to us). In each cell, the horizontal black solid line records the "90%" solution—i.e. when we obtain 90% of the best possible accuracy/precision/recall and AUC achieved under any number of parameters for this problem.

The first column displays the results for CIFAR-10, the computer science image data. As expected given the known success of techniques like convolutional neural nets, there are clear returns to complex models. As we move along the x-axis, we see that all performance metrics clearly increase—albeit non-monotonically—as we allow our model to become more complex.

But this pattern is not nearly so stark for the Montgomery, Goldstone, Beck et al and Colaresi data. We do see some improvements as intrinsic dimension increases, but the lines are basically flat after an (log) intrinsic dimension number of (log) 6 or so. That is, the returns to more complex models are muted at best, and non-existent at worst. To compare more precisely: these data sets allow optimal model performance once they reach around $\exp(6) \approx 400$ or $\exp(8) \approx 3000$ weights; CIFAR-10 needs at least one, perhaps two or three, orders of magnitude more than this—around $\exp(10) \approx 20000$ to $\exp(12) \approx 1000000$ weights. For the political science data, one could make the case that bigger models are better, but the data is essentially as well-described as is possible with smaller models. Bigger models are just not needed—a point similar to the one made some 20 years ago by De Marchi et al. (2004) regarding Beck et al. (2000), in particular.

The final three columns refer to the *Fragile Families* data and here the lesson is even starker. Specifically, the returns to complexity—at least for precision and AUC—are *at best* flat. And for accuracy and recall, there is some evidence that more complex models deliver *worse* results than simple ones. This is, of course, in keeping with Salganik et al. (2020)'s original report. But it is nonetheless remarkable that after some small increases in the complexity of the machine learning functions, performance so quickly degrades. That is, more complex models do not merely fail to help much—they actively hurt.

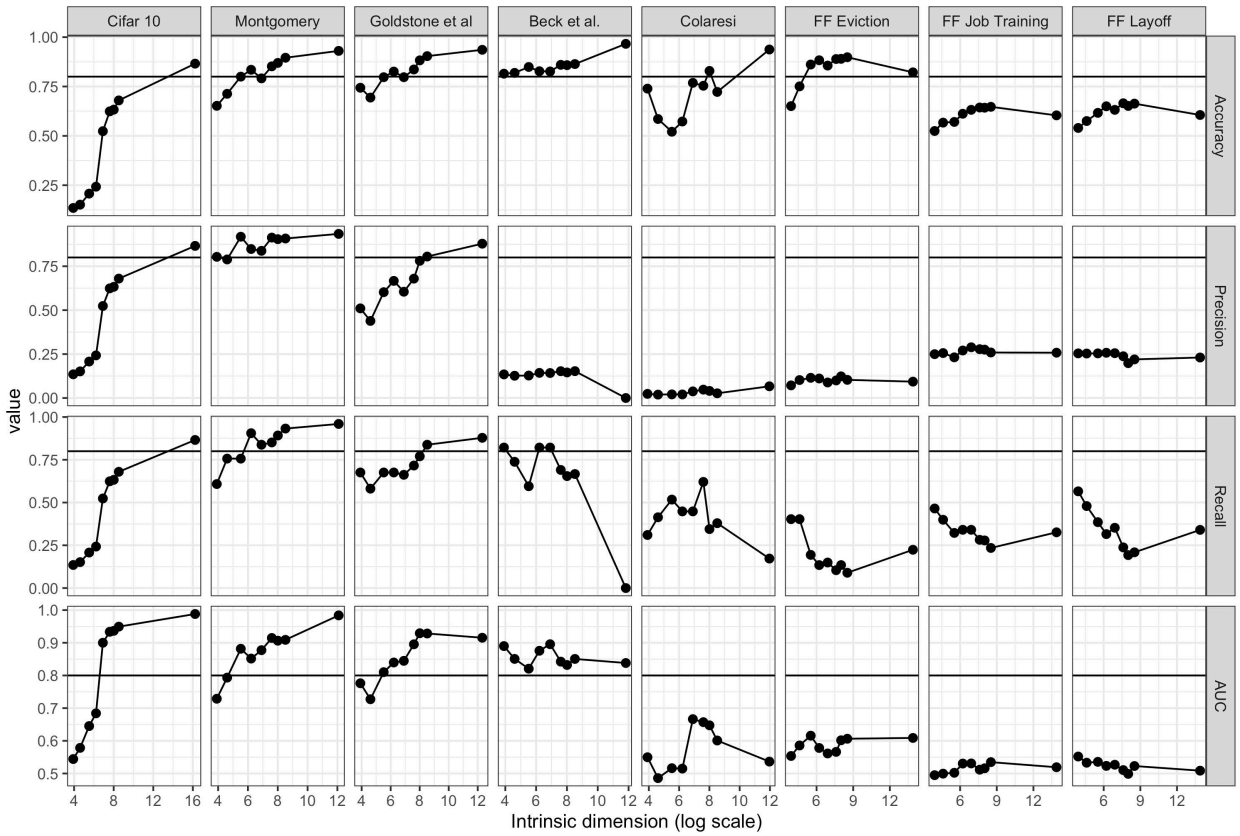


Figure 1: Political Science data sets have low intrinsic dimension, and the returns to complex models are minimal.

5 Why is Intrinsic Dimension So Low? The Curation Problem

The repeated finding above is two-fold: first, it is hard to improve over simple models. While gains do occur with (much) more complex functional forms, they are muted and often negligible. Second, for at least some social science data, no model appears to do especially well in terms of predicting outcomes we care about. Why do these patterns occur? What is it about social science, as opposed to other fields, that generates data where machine learning fails to make large improvements over simple baselines?

We think that at least part of the problem is the way political scientists gather their data sets. In particular, our argument is that the data sets are highly "curated." By this we mean that the data has been gathered, coarsened and structured in such way that it is almost impossible to improve over very simple models in predicting an outcome of interest. This happens because researchers tend to gather and process their data with a theoretical model of the world already in mind: a formal model, a story, a theory they wish to test; and

these models are, at least compared to modern ML models, simple in nature. To restate a point we have made above, the claim is not that elections or wars or protests are themselves simple systems and trivial to predict: it is that the datasets we gather to model them are generally devoid of highly complex structure that machine learning might exploit. Here, we suspect that a contributing factor in political science specifically is work encouraging researchers to keep statistical *models* simple, use limited covariates, and to avoid "garbage can regressions" for interpretation reasons (Achen, 2002, 2005). This is good advice in the correct context, but is misapplied; scholars erroneously infer that simple *datasets* are inherently good—including for prediction.

To fix ideas, consider supervised learning problems where there are a pair of random variables $\mathbf{X}, Y$, such that $\mathbf{X} \in \mathbb{R}^{\mathbf{P}}$ and $Y \in \mathbb{R}$. We assume the researcher would like to predict Y the target (say, civil war) using the variables or features in $\mathbf{X}$. The target and the features have a joint distribution in the real world; i.e. some values of $\mathbf{X}$ are more likely to appear alongside certain values of Y than others. Although there is one 'true' joint distribution, the researcher does not have access to it. So they must gather a sample of observations, and then model the relationship between the target and the feature values they record.

Importantly, analysts do not know *a priori* the correct set of the p variables—and variable values—to gather. It is not hard to see why this might lead to collecting an incomplete version of $\mathbf{X}$ in practice. Perhaps most obviously, it could be that it does not occur to the researcher—in the sense of theory—that a given variable should be recorded. But it could be also that some variables are more costly than others to obtain for every observation, and researchers have a budget constraint.

5.1 A Toy Example of the Curation Problem

It is not difficult to show how this logic might cause problems for machine learning models. As a toy example, suppose that Y is continuous and the true data generating process is

$$Y = 1 + 0.2x_1 + \frac{1}{1 + \exp(-6(\frac{x_1 \times x_2}{\sqrt{2}}) - \frac{1}{3})} + \epsilon$$

where ϵ is a noise term. Clearly, this is a complicated and non-linear DGP, involving two variables x_1 and x_2. But suppose that the researcher only has access—due to theory or some other constraint—to x_1. They note that the correlation of x_1 and Y is relatively large and negative (around -0.5), and their plot of the variables is as in Figure 2.

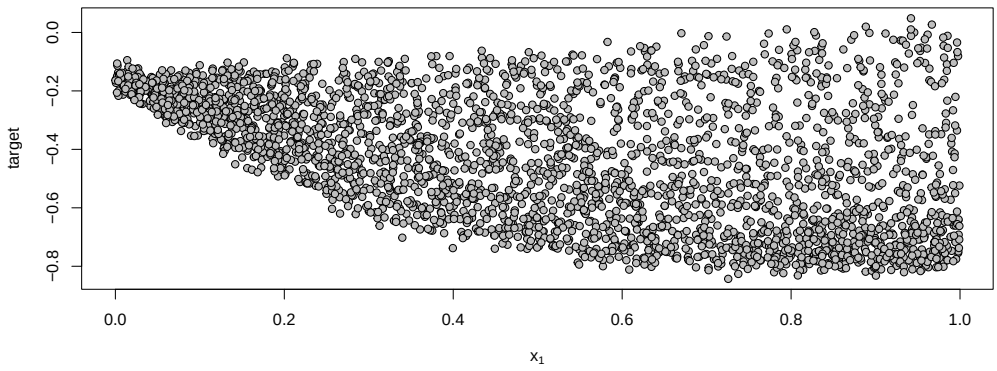


Figure 2: Plotting the target Y against x_1 implies a linear relationship is not unreasonable.

Inspection of Figure 2 suggests that, although the relationship between Y and x_1 is not quite linear, a linear regression may not be unreasonable given the circumstances. So the analyst runs this regression, and it explains around (R^2) 25 percent of the variance in the target. Meanwhile, the coefficient on x_1 is statistically significant ($p < 0.01$). As a final check, the analyst adds a squared term x_1^2 to the regression right-hand side. This provides a very small improvement in fit ($R^2 =\sim 0.27$). They interpret their regression and this completes the analysis: there appears to be little return to a more complex (higher polynomial) model.

The "problem" of course is that there exists another variable x_2 that the researcher does not have access to. Had they done so, they might have plotted the variables as in Figure 3.

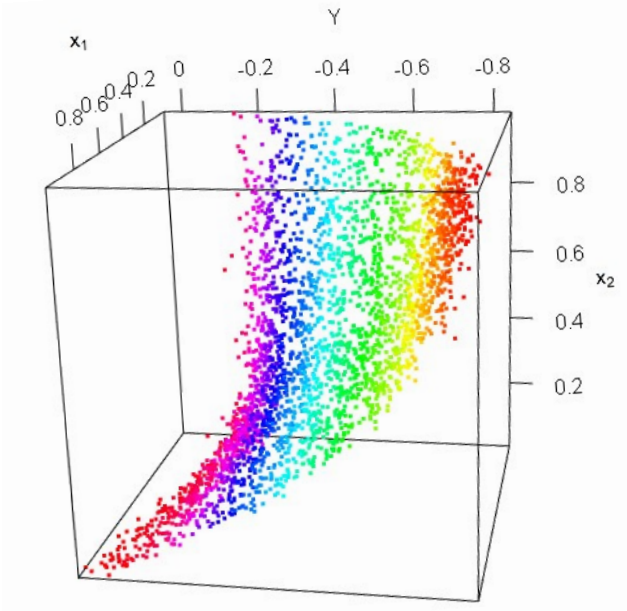


Figure 3: Plotting the target Y against x_1 and x_2 implies a non-linear relationship.

From here, some more complex machine learning model might be the natural next step. As it is, with access to just one curated x_1 variable, the payoffs from subtle functional forms are minimal.

5.2 Data Curation in General

To explore the logic of data curation more generally, suppose that instead of two particular variables, x_1 and x_2, we have two *sets* of variables: $\mathbf{X}$ and $\mathbf{Z}$. In Supporting Information E we set up this case in some technical detail and explore its properties. Here we will give the intuition of the results.

As usual, we want to model Y and we have in mind an additive set up. That is, $Y = f(\mathbf{X}) + g(\mathbf{Z}) + \epsilon$ where ϵ is a statistical error term. Here, f and g are unknown functions of the independent variables: they could be linear or non-linear, additive or multiplicative, and so on. The error term will have the "usual" properties, meaning it is uncorrelated with $\mathbf{X}$ (or $\mathbf{Z}$); it is also uncorrelated with $f(\mathbf{X})$ conditioned on $\mathbf{X}$, meaning that the error term is generally not larger or smaller depending on the value of the function of $\mathbf{X}$ at that particular value of $\mathbf{X}$. We will assume that the same is true of $g(\mathbf{Z})$ too.

Consider a researcher who does not know the true data generating process involves both $\mathbf{X}$ and $\mathbf{Z}$. This may be due to some limits from existing theory, empirics or the literature. Instead, the researcher hypothesizes that Y depends *only* on $\mathbf{X}$ (and not $\mathbf{X}$ and $\mathbf{Z}$). That is, they posit $Y = f(\mathbf{X}) + \nu$, where again, ν is a statistical error term with the usual properties. Luckily, in this first ideal case, existing theory is particularly good. Consequently, the researcher "knows" the true form of f (for instance, a linear regression equation with the variables interacted in a particular way). But, to reiterate, they do not know that in reality their posited statistical error $\nu = g(\mathbf{Z}) + \epsilon$ is the sum of genuine sampling error, and the omitted variables in $\mathbf{Z}$.

Therefore, the researcher proceeds to collect a dataset that does not include any information on $\mathbf{Z}$. The researcher now wants to estimate a model for the relationship between Y and $\mathbf{X}$. We will start with a "best case" scenario in which we assume that the value of $g(\mathbf{Z})$ is uncorrelated with the value of $\mathbf{X}$. This might arise in a causal style setup where $\mathbf{Z}$ is not a confounder for the relationship between $\mathbf{X}$ and Y. As we said, in this particular situation, the researcher happens to know f—the function that correctly captures the relationship between $\mathbf{X}$ and Y—exactly. That model will have some performance—say, mean square error—$\mathrm{MSE}(f)$. Importantly, though more complicated functions of $\mathbf{X}$ exist, say $h(\mathbf{X})$, the $\mathrm{MSE}(f) < \mathrm{MSE}(h)$. That is, in this particular situation, the researcher will see no returns to more elaborate modeling of $\mathbf{X}$ than the one they used. In Supporting Information E we give formal evidence of this claim, but the

intuition is straightforward. Put crudely, it is that the 'best-ness' of f for modeling Y as a function of $\mathbf{X}$ does not vary as $\mathbf{Z}$ varies in this case; so given you do not have access to $\mathbf{Z}$ anyway, you can do no better than use f. Still, the consequence is that data curation (not knowing $\mathbf{Z}$) has lead to zero returns to complex modeling.

But perhaps this is too contrived a scenario. We know, in practice, that we do see (very) small improvements over simple models all the time—enough to keep researchers interested in the prospect of machine learning. This small improvement could happen in many ways, but two routes are especially informative to explore.

First, suppose that we allow $h(\mathbf{X})$ to be highly correlated with $g(\mathbf{Z})$ for at least some values of $\mathbf{X}$. For example, suppose the 'true' $g(\mathbf{Z})$ is $\sin(z)$ where $z = x_1^2$ is a variable in $\mathbf{X}$. We choose some complex high-dimensional function as $h(\mathbf{X})$ and this comes close to approximating the real $g(\mathbf{Z})$. In other words, the model we end up fitting to the data learns something about $g(\mathbf{Z})$ using only $Y - f(\mathbf{X})$. As we demonstrate in Supporting Information E.1, it is possible to then show that the performance of this model is better than f alone. Still, we would not expect the improvement of h over f to be very large, because we are *de facto* trying to learn "everything" from $\mathbf{X}$ alone while trying not to overfit.

A second case of interest is where, unlike our ideal type above, $\mathbf{Z}$ is now a confounder for the (causal) relationship between $\mathbf{X}$ and Y. The very nature of confounding is that we can predict $\mathbf{X}$ from $\mathbf{Z}$, and vice versa. But if this is true, there exists some function of $\mathbf{X}$ which will approximate $g(\mathbf{Z})$ and if we find it, we will do a better job of modeling Y than we would otherwise. We show how that could work in Supporting Information E.2. However, as in the first case above, there is a bias-variance tradeoff. Thus it is unclear how large—if it exists at all—we expect the gains of complexity to be in such a scenario.

The point of this exercise is that returns to complexity can appear to exist in curated, tabular data. But they will typically be small, and somewhat illusionary insofar as the "power" of machine learning in these cases trades on fundamental problems of data omission, including confounding.

5.3 A Typology of Data Sets

We believe that in other fields, where complex models have made great improvements over simpler approaches, the cognitive and/or financial limits we noted above do that bind in the same way. For example, consider a problem common in Computer Science: determining whether a given image is of a cat or a dog (see, e.g., Shorten and Khoshgoftaar, 2019). In principle, one could have "theory" about what variables

ought to predict the label, and gather those from the image data (the pixels). But in practice, it simply using the pixels themselves to predict the label—perhaps via a complex functional form—gets one much of the way there. This is simply not true in political science; or at least for types of prediction problems we discussed above.

This leads us to a typology of data problems, which we present in Table 1. Inevitably, this is a simplification of reality but we believe it is a helpful one. There are two axes on which the typology turns. First, whether the underlying data generating process is truly complex or not. Second, whether the data available for modeling the relationship is noisy or not. Starting at the top left, and moving clockwise we have:

1. "Fundamental Relations": low complexity, low noise. Examples of such functions are found in engineering or physics, where the relationship between X and Y is simple and noise either minimal or well-understood. As an example, consider the relationship between altitude and the boiling point of water. This is approximately linear and can be well modeled with a simple regression. We do not think such relationships are as common in political science, although some "law like" behaviors in elections (between votes and seats, for example) might appear so. In this quadrant, the return to complex models—over and above simple ones—is minimal. And simple models already do well in terms of overall fit.

2. "Learnable Problems": high complexity, low noise. Examples of such functions are found in image recognition—such as dog v cat detection—where complex models outperform simple ones, and the overall fit of such complex models is generally excellent. We do not think these cases are common in political science. We suspect this precisely because we do not see huge gains in performance from (say) neural nets or random forests over and simple logit or linear models.

3. "Unlearnable Problems": high complexity, high noise. We think much human interaction data has this form. The functions that determine various human outcomes like happiness, or lifetime income or the success of business ideas are complicated and involve very large numbers of variables. They are also noisy insofar as we measure both the target and the covariates with error, and there are various complicated selection mechanisms that give raise to the data we observe. In this quadrant, complex models may well do better than simple ones, but no model predicts well. That is, there is a *relative* positive payoff to complex modeling, but not an *absolute* one.

4. "Curated Data": low complexity, high noise. Our prevailing argument is that most political science data—especially the tabular variety—exists in this quadrant. It may start life in the "Unlearnable" quadrant, but migrates left as analysts coarsen, recode and structure the data set. It is high noise because fundamentally the processes to be modelled—like international conflict—are noisy. But there is no return to complex modeling, because the data has been denuded of the variables and the various interactive relationships that would otherwise have allowed this. A feature of this quadrant is that there is little return to complex models, and in many cases the overall performance over the learners is poor.

		Complexity of True DGP	
		Low	High
Noise	Low	**Fundamental Relations** - simple models perform well - muted return to complexity - e.g. predicting boiling point from altitude	**Learnable Problems** - simple models do poorly - complex models do well - e.g. image recognition
	High	**Curated Data** - tabular data in political science - all models bad, no returns to complexity - e.g. predicting war, unemployment, negative ads	**Unlearnable Problems** - complex, unstructured problems all models bad, mild returns to complexity -e.g. predicting happiness, relationships

Table 1: Data Curation Typology

6 Advice to Practitioners

What practical steps should researchers take given our findings and theory above? As our audience here, we have in mind a political scientist who has a (yet un-modeled) data set available to them for some prediction or inference purpose. We have four broad suggestions:

1. **Know your problem**. By this we mean understand how the data set was put together and thus where it might be located in the typology we introduced above. This may be an introspective matter if the researcher themselves gathered it. Part of this will be its basic characteristics such as how large and (un)structured the data is, along with the number of variables and how they were gathered or processed before being tabulated for use. Per our results above this should give an immediate sense of how large the returns to a complex (as opposed to a simple) modeling strategy are likely to be. Concretely: if the highly structured, highly curated, and contains a relatively small number of variables, chances are returns will be minimal. Comparing your data to our focus datasets—just in terms of its approximate structure—should some immediate guidance on possibilities.

2. **Start simple**. Our consistent result above is that, for good reason, it is very hard to outperform simple models like linear regressions and logits. So, one should start with such simple models, unless one has very good reason to do otherwise. At the very least this will establish a good baseline for performance. This is *a fortiori* true if you value ease of interpretation (in the sense of De Marchi et al., 2004; Rudin, 2019).

3. **Do Diagnostics**. In the limit, one can take our software and run the intrinsic dimension detection algorithms on your data. We found that this consistently correctly characterizes the returns to complexity for the sorts of problems political scientists are likely to face. If the resulting plots look like the ones for the unstructured image data, then you can expect good returns to complexity. If not, you cannot.

4. **Get More and Better Data**. If the idea is to predict better (i.e. have a lower loss), then your political science data set may not be fit for purpose. Put differently: you cannot model yourself out of bad data. So augment, clean, de-noise and expand your data with unstructured variables and try again—starting with something simple first.

7 Discussion: Just Do OLS?

Supervised learning, specifically as used for predicting outcomes in social systems, is in the spotlight. Interest from scientists and policy-makers is intense. Perhaps inevitably, early optimism about its promise is giving way to more nuanced, pessimistic evaluations. In particular, as we argued above, scholars have found that it is surprisingly hard to outperform simple models—such as linear regression or logit—for a wide diversity of problems. We think recognition of these "limits" is important. This is because "de-hyping" complex approaches avoids mistaken findings and mis-investments of resources. But it is also important because simple tools have value *per se*. We embrace both the desire to use supervised learning in political science, and the desire to be clear-eyed about what it can actually deliver in practice. Our efforts here, at a high level, are designed to encourage thoughtful use of machine learning. And we encourage that use in light of our findings—and those by others—that complex models are not very helpful for most political science problems.

Why is this? First, a broad answer: we showed that the data we for modeling does not contain the highly complex structure that machine learning is so good at exploiting. Specifically, this is true of our *tabular* data, where observations are the rows, and the columns are carefully collected, edited, cleaned covariates. Where the data is not tabular, for example, when using image information to predict outcomes, our answer differs somewhat. There, complex models can do well—or at least well enough to justify their increased overhead and interpretation costs. We showed this using some new tools from computer science that we adapted to political science and that allow us to obtain the intrinsic dimension of a problem. We will make those tools more generally available so others can use them as a potential diagnostic for their own data—and thus for their own modeling strategies.

Noting that most tabular data does not seem to lend itself to complex models might save users some work, but we also tried to explain the deeper reasons as to why this occurs. We argued, and therein presented a formal account, that those curating data tend to do so in a fundamentally simple and "linear" way. This is not a surprise, because that is what political science theory encourages us to do; given basic cognitive constraints, we gather variables in a form that we anticipate is important and interpretable. And that form is straightforward. Of course, there are other features that make this issue worse in political science—we often simply lack examples of phenomena like "World Wars" (very small n) or the outcome in question is not directly observed but is noisily measured (like 'democracy"). But we believe curation is a central issue.

What follows from this? The headline is that most data in political science is not amendable to complex modeling but that news is "good" or "bad" depending on your perspective. It is good insofar as, for many prediction problems, one is unlikely to need more than the linear model ("just do OLS") or a straightforward GLM. It seems plausible that this logic extends also to cases where highly curated politics data is being used to produce *causal* estimates via machine learning (see e.g. Mullainathan and Spiess, 2017, for discussion). But this "unreasonable effectiveness of Linear Regression" (Facure, 2023, 95) may be bad however if such models do not predict well in absolute terms. That is, there clearly exist situations (e.g. Salganik et al., 2020) where all predictions are low quality, and the fact that a model is simple is cold comfort given our actual aims. In those cases, better and more (unstructured) data is the answer. But it is hard to know exactly how to gather it, and how to avoid curation more generally. We leave these things for future work.

References

Avidit Acharya, Kirk Bansak, and Jens Hainmueller. Combining outcome-based and preference-based matching: A constrained priority mechanism. Political Analysis, 30(1):89–112, 2022.

Christopher H Achen. Toward a new political methodology: Microfoundations and art. Annual review of political science, 5(1):423–450, 2002.

Christopher H Achen. Let's put garbage-can regressions and garbage-can probits where they belong. Conflict Management and Peace Science, 22(4):327–339, 2005.

Christian Arnold, Luka Biedebach, Andreas Küpfer, and Marcel Neunhoeffer. The role of hyperparameters in machine learning models and how to tune them. Political Science Research and Methods, 2023.

Andrew R Barron. Universal approximation bounds for superpositions of a sigmoidal function. IEEE Transactions on Information theory, 39(3):930–945, 1993.

Nathaniel Beck, Gary King, and Langche Zeng. Improving quantitative studies of international conflict: A conjecture. American Political science review, 94(1):21–35, 2000.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. Proceedings of the National Academy of Sciences, 116(32): 15849–15854, 2019.

Eli Ben-Michael, D James Greiner, Kosuke Imai, and Zhichao Jiang. Safe policy learning through extrapolation: Application to pre-trial risk assessment. arXiv preprint arXiv:2109.11679, 2021.

Matthew Blackwell. A framework for dynamic causal inference in political science. American Journal of Political Science, 57(2):504–520, 2013.

Patrick T Brandt, Vito D'Orazio, Latifur Khan, Yi-Fan Li, Javier Osorio, and Marcus Sianan. Conflict forecasting with event data and spatio-temporal graph convolutional networks. International Interactions, 48(4):800–822, 2022.

Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. Double/debiased machine learning for treatment and structural parameters. The Econometrics Journal, 21(1):C1–C68, 01 2018.

Michael Colaresi and Zuhaib Mahmood. Do the robot: Lessons from machine learning to improve conflict forecasting. Journal of Peace Research, 54(2):193–214, 2017.

Skyler J Cranmer and Bruce A Desmarais. What can we learn from predictive modeling? Political Analysis, 25(2):145–166, 2017.

Alicia Curth, Alan Jeffares, and Mihaela van der Schaar. A u-turn on double descent: Rethinking parameter counting in statistical learning. In Thirty-seventh Conference on Neural Information Processing Systems, 2023. URL https://openreview.net/forum?id=O0Lz8XZT2b.

Scott De Marchi, Christopher Gelpi, and Jeffrey D Grynaviski. Untangling neural nets. American Political Science Review, 98(2):371–378, 2004.

Cassy Dorff, Max Gallop, and Shahryar Minhas. Networks of violence: Predicting conflict in nigeria. The Journal of Politics, 82(2):476–493, 2020.

Matheus Facure. Causal Inference in Python. " O'Reilly Media, Inc.", 2023.

James D Fearon and David D Laitin. Ethnicity, insurgency, and civil war. American political science review, 97(1):75–90, 2003.

Manuel Fernández-Delgado, Eva Cernadas, Senén Barro, and Dinani Amorim. Do we need hundreds of classifiers to solve real world classification problems? The journal of machine learning research, 15(1): 3133–3181, 2014.

Jonathan Fuhr, Philipp Berens, and Dominik Papies. Estimating causal effects with double machine learning–a method evaluation. arXiv preprint arXiv:2403.14385, 2024.

Niels D Goet. Measuring polarization with text analysis: Evidence from the uk house of commons, 1811–2015. Political Analysis, 27(4):518–539, 2019.

Micah Goldblum, Marc Finzi, Keefer Rowan, and Andrew Gordon Wilson. The no free lunch theorem, kolmogorov complexity, and the role of inductive biases in machine learning. arXiv preprint arXiv:2304.05366, 2023.

Jack A Goldstone, Robert H Bates, David L Epstein, Ted Robert Gurr, Michael B Lustik, Monty G Marshall, Jay Ulfelder, and Mark Woodward. A global model for forecasting political instability. American journal of political science, 54(1):190–208, 2010.

Justin Grimmer, Margaret E Roberts, and Brandon M Stewart. Text as data: A new framework for machine learning and the social sciences. Princeton University Press, 2022.

Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? Advances in Neural Information Processing Systems, 35:507–520, 2022.

David J. Hand. Classifier Technology and the Illusion of Progress. Statistical Science, 21(1):1 – 14, 2006.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

Håvard Hegre, Marie Allansson, Matthias Basedau, Michael Colaresi, Mihai Croicu, Hanne Fjelde, Frederick Hoyles, Lisa Hultman, Stina Högbladh, Remco Jansen, et al. Views: A political violence early-warning system. Journal of peace research, 56(2):155–174, 2019.

Daniel W Hill and Zachary M Jones. An empirical evaluation of explanations for state repression. American Political Science Review, 108(3):661–687, 2014.

Sayash Kapoor and Arvind Narayanan. Leakage and the reproducibility crisis in machine-learning-based science. Patterns, August 2023. ISSN 2666-3899. doi: 10.1016/j.patter.2023.100804. URL https://www.cell.com/patterns/abstract/S2666-3899(23)00159-9. Publisher: Elsevier.

Dean Knox, Christopher Lucas, and Wendy K Tam Cho. Testing causal theories with learned proxies. Annual Review of Political Science, 25:419–441, 2022.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. nature, 521(7553):436–444, 2015.

Chunyuan Li, Heerad Farkhoor, Rosanne Liu, and Jason Yosinski. Measuring the intrinsic dimension of objective landscapes. In International Conference on Learning Representations, 2018.

Ana C Lorena, Luís PF Garcia, Jens Lehmann, Marcilio CP Souto, and Tin Kam Ho. How complex is your classification problem? a survey on measuring classification complexity. ACM Computing Surveys (CSUR), 52(5):1–34, 2019.

Sanae Lotfi, Marc Finzi, Sanyam Kapoor, Andres Potapczynski, Micah Goldblum, and Andrew G Wilson. Pac-bayes compression bounds so tight that they can explain generalization. Advances in Neural Information Processing Systems, 35:31459–31473, 2022.

Shahryar Minhas, Peter D Hoff, and Michael D Ward. A new approach to analyzing coevolving longitudinal networks in international relations. Journal of Peace Research, 53(3):491–505, 2016.

Jacob M Montgomery and Santiago Olivella. Tree-based models for political science data. American Journal of Political Science, 62(3):729–744, 2018.

David Muchlinski, David Siroky, Jingrui He, and Matthew Kocher. Comparing random forest with logistic regression for predicting class-imbalanced civil war onset data. Political Analysis, 24(1):87–103, 2016.

Sendhil Mullainathan and Jann Spiess. Machine learning: an applied econometric approach. Journal of Economic Perspectives, 31(2):87–106, 2017.

Simon Nusinovici, Yih Chung Tham, Marco Yu Chak Yan, Daniel Shu Wei Ting, Jialiang Li, Charumathi Sabanayagam, Tien Yin Wong, and Ching-Yu Cheng. Logistic regression was as good as machine learning for predicting major chronic diseases. Journal of clinical epidemiology, 122:56–69, 2020.

Philipp Petersen and Felix Voigtlaender. Optimal approximation of piecewise smooth functions using deep relu neural networks. Neural Networks, 108:296–330, 2018.

Marc Ratkovic. Relaxing assumptions, improving inference: integrating machine learning and the linear regression. American Political Science Review, 117(3):1053–1069, 2023.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature machine intelligence, 1(5):206–215, 2019.

Matthew J Salganik, Ian Lundberg, Alexander T Kindel, Caitlin E Ahearn, Khaled Al-Ghoneim, Abdullah Almaatouq, Drew M Altschul, Jennie E Brand, Nicole Bohme Carnegie, Ryan James Compton, et al.

Measuring the predictability of life outcomes with a scientific mass collaboration. Proceedings of the National Academy of Sciences, 117(15):8398–8403, 2020.

Lesia Semenova, Cynthia Rudin, and Ronald Parr. On the existence of simpler machine learning models. In Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency, pages 1827–1858, 2022.

Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. Journal of big data, 6(1):1–48, 2019.

Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. Information Fusion, 81:84–90, 2022.

Benjamin Strang, Peter van der Putten, Jan N van Rijn, and Frank Hutter. Don't rule out simple models prematurely: A large scale benchmark comparing linear and non-linear classifiers in openml. In Advances in Intelligent Data Analysis XVII: 17th International Symposium, IDA 2018,'s-Hertogenbosch, The Netherlands, October 24–26, 2018, Proceedings 17, pages 303–315. Springer, 2018.

Michelle Torres and Francisco Cantú. Learning to see: Convolutional neural networks for the analysis of social science data. Political Analysis, 30(1):113–131, 2022.

JA Vallejos and SD McKinnon. Logistic regression and neural network classification of seismic records. International Journal of Rock Mechanics and Mining Sciences, 62:86–95, 2013.

Paola Vesco, Håvard Hegre, Michael Colaresi, Remco Bastiaan Jansen, Adeline Lo, Gregor Reisch, and Nils B Weidmann. United they stand: Findings from an escalation prediction competition. International Interactions, 48(4):860–896, 2022.

Michael D Ward, Brian D Greenhill, and Kristin M Bakke. The perils of policy by p-value: Predicting civil conflicts. Journal of peace research, 47(4):363–375, 2010.

Dmitry Yarotsky. Error bounds for approximations with deep relu networks. Neural Networks, 94:103–114, 2017.

Online Supporting Information for *Model Complexity for Supervised Learning*

Contents (Appendix)

A Intrinsic Dimension theory: technical details **2**

B Intrinsic Dimension estimation: technical details **4**

C Properties of Random Subspace Training approach **5**

C.1 Experiments 5

C.2 Regularization 7

D Modeling Details **8**

E Data Curation: Formal Details **9**

E.1 Case I: approximating the true function $g(\mathbf{Z})$ 10

E.2 Case II: confounding 11

A Intrinsic Dimension theory: technical details

The idea inherent in intrinsic dimension is much more general than the specific example of the ideal points we gave above. To see why, suppose the goal was to predict some outcome $Y \in \mathbb{R}$ based on P-many covariates $\mathbf{X} \in [-1, 1]^P$, both of which are random variables with joint distribution $P_{(\mathbf{X},Y)}$. Here the problem is to find the optimal combinations of $\mathbf{X}$s, i.e. some function $f(\mathbf{X})$, such that our predictions, $\hat{Y} = f(\mathbf{X})$ are as close to the truth (the actual Y) as possible. That is, we want to minimize the *population loss*, $\mathcal{L}(f(\mathbf{X}), Y)$, which we might measure via the mean square error (for a continuous outcome) or hinge loss (for a categorical ones). In notation, we wish to solve:

$$f^* \in \arg\min_f \mathbb{E}_{P_{(\mathbf{X},Y)}}[\mathcal{L}(f(\mathbf{X}), Y)].$$

Obviously, we do not know anything about the precise nature of f^*; therefore we *must* make some assumptions in order to allow for prediction. The assumption we (and virtually all others wishing to solve a prediction problem) make is that $f^* \in \mathcal{F}_{\text{DNN}}(D)$: that is, that the population best predictor for our data lives within a *class* of functions, $\mathcal{F}_{\text{DNN}}(D)$, which we can characterize with a *deep neural network*

(DNN).[6]

In our case, we will assume that $\mathcal{F}_{\text{DNN}}(D)$ is the set of all Deep Neural Networks with *at most* D many learnable weights and a fixed, but unspecified, architecture. This assumption and its equivalents are at the core of nonparametric statistical learning: an analyst wishing to predict some outcome must choose a model based on whether they think the unknown population best predictor will be contained in the class of the chosen model. For the case of DNNs, we know that the model class contains a vast array of potential best predictor functions (see, e.g., Barron, 1993; Yarotsky, 2017; Petersen and Voigtlaender, 2018), and that so long as D, the number of parameters of the DNN, is chosen to be large enough, one can be quite confident that the population best predictor, f^*, will be contained in $\mathcal{F}_{\text{DNN}}(D)$.[7]

This assumption is useful to us in two ways. First it allows us to precisely establish a definition for the *intrinsic dimension* of the prediction problem under study:

Definition 1. *(Intrinsic Dimension) The Intrinsic Dimension of a prediction problem with predictors X, outcome Y having joint distribution $P_{(X,Y)}$, and loss function $\mathcal{L}$ is the smallest integer D such that $f^* \in \mathcal{F}_{DNN}(D)$, where $f^* \in \arg\min_f \mathbb{E}_{P_{(\mathbf{X},Y)}}[\mathcal{L}(f(\mathbf{X}), Y)]$.*

This definition states that the intrinsic dimension of any prediction problem is the smallest number of weights that a DNN should have in order to perfectly represent the best predictor for the problem, f^*. If the best predictor is not very complex (e.g., a linear regression with P weights), then D will be small, but if the best predictor *is* a complex function, then D must be larger in order for a DNN with D many weights to be able to fully capture f^*.

Second, the assumption allows us to formulate the optimization problem of finding the best predictor as an optimization problem over the D-many learnable weights of our DNN: each model f in $\mathcal{F}_{\text{DNN}}(D)$ is uniquely defined[8] by its weights: $\mathbf{W}_f = [w_{f1}, \ldots, w_{fD}] \in \mathbb{R}^D$, therefore, if $f^* \in \mathcal{F}_{\text{DNN}}(D)$, then so is f^*, i.e., there exists a vector of weights $\mathbf{W}^*$ that uniquely defines $f^* = f_{\mathbf{W}^*}$, and the problem of finding the

[6] Technically speaking, the standard assumption we make is that f^* can be arbitrarily closely approximated by a DNN with at most D weights, that is: for every $\epsilon > 0$ there exists a D_ϵ and a function $h \in \mathcal{F}_{\text{DNN}}(D_\epsilon)$ such that, for every $\mathbf{x}$ we have $\|f^*(\mathbf{x}) - h(\mathbf{x})\| \leq \epsilon$.

[7] This is also largely independent of the network architecture (Yarotsky, 2017; Petersen and Voigtlaender, 2018): as long as there are enough learnable weights in the network it largely does not matter how the layers are connected for optimal approximation of any suitable f^*

[8] In the sense that there exists a function h, which can be thought of as the architecture of the DNN, such that for all $f \in \mathcal{F}_{\text{DNN}}(D)$ and all $\mathbf{x} \in [-1, 1]^P$ we have $f(\mathbf{x}) = h(\mathbf{x}, \mathbf{W}_f)$.

best predictor becomes:

$$f^* \in \underset{\mathbf{f}\in\mathcal{F}_{\text{DNN}}(\mathbf{D})}{\arg\min}\ \mathbb{E}_{P_{(X,Y)}}[\mathcal{L}(f(X),Y)] = \mathbf{W}^* \in \underset{\mathbf{W}\in\mathbb{R}^D}{\arg\min}\,\mathbb{E}_{P_{(X,Y)}}[\mathcal{L}(f_{\mathbf{W}}(X),Y)],$$

and this is much easier to solve in practice than searching for an unknown function.

A point we need to stress about our definition of intrinsic dimension is that D as learned by a DNN may not be the same as the smallest number of parameters of the best predictor of y among *any* class of functions. For example, if $f^*(x) = \sin(\beta x)$ the only parameter here is β so the dimensionality is 1; however, a DNN will learn the $\sin$ function as the composition of many weights, implying that D will be greater than 1. This is not an issue for the purposes of how we use DNNs to quantify complexity: so long as the same large DNN is fit to all the datasets the dimension of which we wish to compare, the comparison will still be meaningful. This is because the number of weights the same DNN needs to represent a function will not vary across datasets.

B Intrinsic Dimension estimation: technical details

One of the goals of this paper is to evaluate the intrinsic dimension of several problems in the social sciences. Given the theory introduced above, and a data set, how is this done? Formally, the problem we wish to solve, for a given $P_{(\mathbf{X},Y)}$ and loss function $\mathcal{L}$ is:

$$D^* \in \underset{D\in\mathbb{Z}}{\arg\min}\left[\min_{\mathbf{W}\in\mathbb{R}^D}\mathbb{E}_{P_{(\mathbf{X},Y)}}[\mathcal{L}(f_{\mathbf{W}}(X),Y)]\right].$$

From the running logistic regression example, it would seem that there is a clear and simple path to finding the intrinsic dimension of any given problem: simply choose D large enough so that $\mathcal{F}_{\text{DNN}}(D)$ contains f^*, solve $\mathbf{W}^* = \arg\min_{\mathbf{W}\in\mathbb{R}^D}\mathbb{E}_{P_{(\mathbf{X},Y)}}[\mathcal{L}(f_{\mathbf{W}}(\mathbf{X}),Y)]$, and then count how many elements of $\mathbf{W}^*$ are nonzero to find the intrinsic dimension.

Unfortunately, this does not work in practice for several reasons: first, obviously we can never solve this population minimization problem directly, as we can never observe the value of the loss over the entire population. We instead have a sample; that is, a dataset of iid replicates of $\mathbf{X}, Y$, defined as $\mathcal{D} = \{(\mathbf{X}_i, Y_i)\}_{i=1}^N$.

So we settle for finding the minimizer of the analogue of the loss in this sample:

$$\widehat{\mathbf{W}} \in \underset{\mathbf{W} \in \mathbb{R}^D}{\arg\min} \sum_{i=1}^{N} \mathcal{L}(f_{\mathbf{W}}(\mathbf{X}_i), Y_i).$$

This procedure adds substantial *statistical noise* to the optimal DNN weights we will estimate. This implies that the $\widehat{\mathbf{W}}$ that we find empirically may be arbitrarily different from the population-optimal $\mathbf{W}^*$. In turn, this means that the number of nonzero elements of the former may be nowhere close to that of the latter. In addition, the inherent noisiness of the finite-sample problem will make it highly unlikely that $\widehat{\mathbf{W}}$ will have many nonzero elements, if any at all: by definition $\widehat{\mathbf{W}}$ tries to minimize the loss of each individual observation in the sample, meaning that some elements of $\mathbf{W}$ may be used to simply reduce error for an individual observation and no other. This problem is commonly known as *overfitting*: the model will tune itself to fit the training sample as close as possible during training, thus failing to predict Y well for any new data point not in the training set. Finally, both in the sample and in the population, there may be many equivalent values $\widehat{\mathbf{W}}$ that achieve minimal loss and there is no way to guarantee that solving the loss minimization problem once will result in the optimal $\widehat{\mathbf{W}}$ with lowest intrinsic dimensionality, and solving the problem multiple times to obtain different solution is likely to be infeasible either computationally or theoretically.

C Properties of Random Subspace Training approach

C.1 Experiments

Li et al. (2018) provide evidence that random subspace training of DNNs does a good job of approximating the intrinsic dimension for many different prediction problems. But it could conceivably be the case that it does not work well for common scenarios in political science. So we perform additional simulation experiments to verify here.

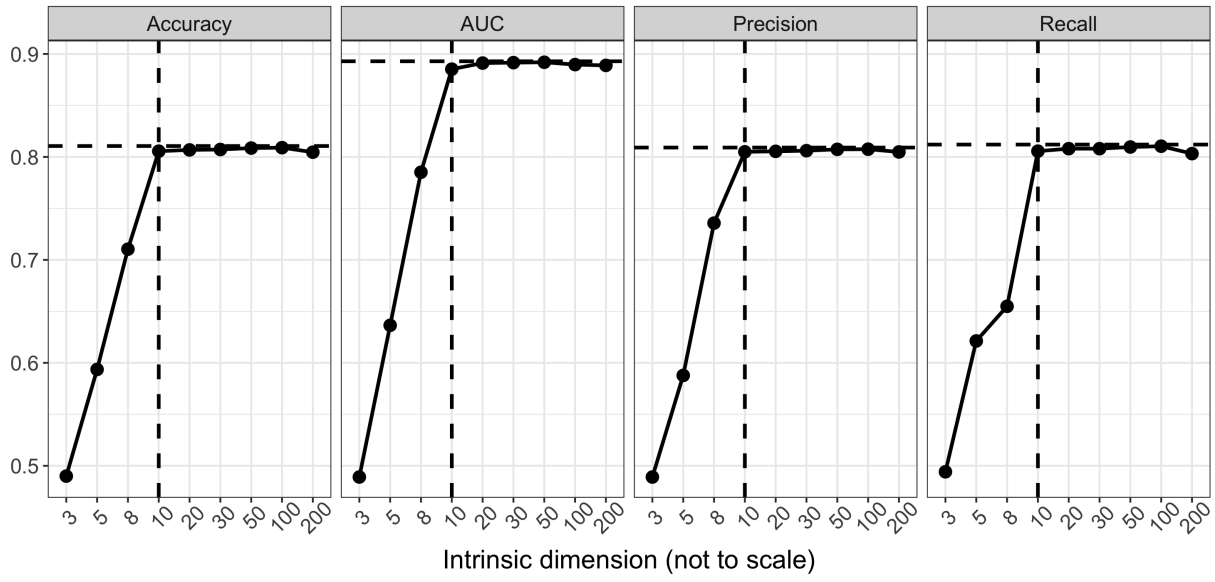
For these experiments we generate data from a simple logistic regression model with 10 covariates, i.e.,

we draw, for $i = 1, \ldots, N$ and $j = 1, \ldots, 10$:

$$\beta_j \sim \text{Uniform}(-1, 1)$$
$$\mathbf{X}_i \sim \text{Uniform}(-1, 1)$$
$$p_i \sim \sigma(\mathbf{X}_i \beta)$$
$$Y_i \sim \text{Bernoulli}(p_i).$$

We generate data for $N = 10000$ training observations and $N = 5000$ test observations. We then fit a logistic regression (the true parametric model) to the training data and measure performance on the test data. In this case the true intrinsic dimensionality is $D = 10$ as there are 10 free parameters in the true best predictor. Finally, we apply our search procedure as described in the rest of the paper to the data generated in this way. We evaluate model performance for $d = 3, 5, 8, 10, 20, 30, 50, 100, 200$. If our approach is successful, then the model we train via random subspaces will underperform for $d < 10$, and reach optimal performance at $d = 10$, which is the same as the true dimension of the problem. Furthermore, bigger models will offer no additional gains. This is exactly what we see: the results are presented in Figure 4. Our models

Figure 4: Random subspace training simulation results



Note: Dashed vertical line is the true ID of the problem and dashed horizontal lines are the performance metric levels achieved by the true parametric model. The solid line is the ID/performance of our estimated models.

achieve performance similar to the true model on all metrics when $d = 10$, there is no performance gain

on any metric from fitting more complex models to the data, while models with $d < 10$ fail to achieve the performance level of the true model.

Overall these experiments presents compelling evidence that our search procedure does recover the true intrinsic dimension of a given problem, and can, therefore offer substantial insight on the true complexity of the datasets we analyze.

C.2 Regularization

For completeness we clarify the relationship between regularization as commonly intended in machine learning and intrinsic dimension. As Li et al. (2018, supplement) note, choosing an intrinsic dimension smaller than D is, *inherently* a form of regularization. By "regularization" we mean any methodology that restricts the possible values that may be assigned to the learnable parameters of a given model. This implies that choosing a (relatively) small d and then training a model using a random projection will result in a lower likelihood of overfitting the training data, and consequently of better generalization performance.

The advantage of our use of intrinsic dimension over other forms of regularization is that it allows us to characterize the theoretical complexity of a given prediction problem, while simultaneously avoiding overfitting. To reiterate, intrinsic dimension *is* regularization. Because this is true, successfully computing the intrinsic dimension for a given problem—by explicitly searching over possible ds—requires that the base model whose weights are to be projected on a smaller space should not employ any *other* form of (explicit or implicit) regularization. That is, one must not also employ techniques like dropout or L_2 regularization. This is because such approaches (further) lower the intrinsic dimension of the model that the random subspace technique is being applied to. This could lead one to erroneously conclude that the intrinsic dimension of a given problem is smaller or larger than it actually is.

This is also why it is not necessary for intrinsic dimension estimation to train (potentially) over-parametrized models until they reach a double descent point (Belkin et al., 2019) and naturally self-regularize to the point where they do not overfit anymore (Curth et al., 2023). Our grid search procedure will naturally find an intrinsic dimension that is near-optimal by just lowering the intrinsic dimension until the model is properly regularized. If a lower-dimensional model does better than a higher-dimension model one can assume that the higher intrinsic dimension model could potentially do as well as the optimal lower intrinsic dimension model (but no better) if trained until optimal double descent was reached. We note that, for general datasets, it is very hard in a practical sense to reliably induce double descent in overparametrized models during

training. Therefore it is desirable that a methodology for intrinsic dimension estimation does not require this.

D Modeling Details

To perform the subspace optimization, we adopt the same methodology as Li et al. (2018), combined with some of the improvements given in Lotfi et al. (2022). For each dataset, we identify the set of outcome variables, Y, and predictor variables, $\mathbf{X}$, used by the original author of the data. We start by fitting a very large Resnet (He et al., 2016) model to the data and progressively decrease the intrinsic dimension of the model, recording several performance statistics (Accuracy, Precision, Recall, AUC) on a separate test set for each intrinsic dimension, which are reported in Figure 1.

Table 2: Architecture of the base model used in the empirical analysis.

Layer	Dimension	Connection	Activation	BatchNorm
Input	N predictors	–	No	No
Hidden	N predictors x 256	Direct	ReLU	Yes
Residual 1	256 x 256	Residual	ReLU	Yes
Residual 2	256 x 256	Residual	ReLU	Yes
Output	256 x N outputs	Direct	No/Sigmoid/Softmax	No
Min. number of learnable parameters: 131584				

Our "full-size" model is a feedforward Resnet with the architecture given in Table 2, where we apply a different activation function to the model output depending on the data type of Y: we use the sigmoid function for binary outcomes, the softmax function for multi-class outcomes, and we use no activation for real-valued outcomes. We also employ different loss functions depending on Y: we use cross-entropy for binary or multi-class outcomes, and mean squared error for real-valued outcomes. We choose a 4-layer neural network with residual connections in the middle two hidden layers as this model is still somewhat simple, but its size (given by the number of learnable parameters) is already extremely large compared to most models found in political science. We will not explain in too much depth what each of the modeling elements in Table 2 is, but see e.g., Torres and Cantú (2022) for a user-friendly explanation of the DNN architecture elements used here. We choose batch-normalized residual layers as these have been shown to introduce stability during training without explicitly regularizing model parameters (He et al., 2016), a fact which we have empirically verified during the training of our models.

After training and testing our full-size model on each of our focus datasets, we train and test eight progressively smaller models: we train models for $d = 50, 100, 250, 500, 1000, 2000, 3000, 5000$. To compute a random projection from the D weights of our full model to the d intrinsic weights we actually train, we use the FiLM + Kronecker sum projector introduced in Lotfi et al. (2022), which is an advanced sampling scheme to generate random projection matrices that facilitate model convergence during training.

E Data Curation: Formal Details

In order to describe a slightly more general notion of data curation, we will be looking at the sample case in which we have two variables of interest, $\mathbf{X}$, and $\mathbf{Z}$ as before. We consider an additive model for our data in which $Y = f(\mathbf{X}) + g(\mathbf{Z}) + \epsilon$, where ϵ is a statistical error term, satisfying $\mathbb{E}[\epsilon|\mathbf{X}] = 0, \mathbb{E}[\epsilon f(\mathbf{X})|\mathbf{X}] = 0, \mathbb{E}[\epsilon g(\mathbf{Z})|\mathbf{X}] = 0$, and f and g are unknown functions of the independent variables of interest. Consider a researcher who is interested in modeling Y: because of existing theory, empirics, literature, etc., the researcher hypothesizes that Y depends *only* on the variables in $\mathbf{X}$, and not on those in $\mathbf{Z}$, that is, they posit $Y = f(\mathbf{X}) + \nu$, where again, ν is a statistical error term. In this ideal case, existing theory is particularly good and the researcher "knows" the true form of f (for example, a linear regression equation), however the do not know that, in reality, their posited statistical error $\nu = g(\mathbf{Z}) + \epsilon$ is the sum of genuine sampling error, and the omitted variables in $\mathbf{Z}$.

Therefore, they proceed to collect a dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n$ of iid draws from the distribution of $(\mathbf{X}, Y)$ that does not include any information on $\mathbf{Z}$. The researcher now wants to estimate a model for the relationship between Y and $\mathbf{X}$.

Let us consider first the benign case in which $\mathbb{E}[g(\mathbf{Z})|\mathbf{X}] = 0$. This is a very common case, for example, when we are interested in estimating the causal relationship between $\mathbf{X}$ and Y and $Y \perp\!\!\!\perp \mathbf{Z}|\mathbf{X}$, that is $\mathbf{Z}$ is not a confounder of the relationship of interest. As an intermediate step in estimating a causal quantity of interest, the researcher will have to model Y as a function of $\mathbf{X}$, and therefore wants to fit the model that best captures this relationship. Suppose that the researcher fits exactly the function f to the data. Note first that, by assumption $\mathbb{E}[Y|\mathbf{X}] = f(\mathbf{X})$, and therefore the resulting model will have MSE as follows:

$$\text{MSE}(f) = \mathbb{E}[(f(\mathbf{X}) - Y)^2] = \mathbb{E}_{\mathbf{X}}\mathbb{E}[(\mathbb{E}[Y|\mathbf{X}] - Y)^2|\mathbf{X}] = \mathbb{E}_{\mathbf{X}}[\mathbb{V}[Y|\mathbf{X}]].$$

Now suppose instead that the researcher fits a more complicated model to the data, h, whose only constraint is that it satisfies $\mathbb{E}[h(\mathbf{X})g(\mathbf{Z})|\mathbf{X}] = 0$, and $\mathbb{E}[h(\mathbf{X})\epsilon|\mathbf{X}] = 0$. This model will have MSE equal to:

$$\begin{aligned}
\text{MSE}(h) &= \mathbb{E}[(h(\mathbf{X} - Y)^2)] = \mathbb{E}[(h(\mathbf{X}) - f(\mathbf{X}) - g(\mathbf{Z}) - \epsilon)^2] \\
&= \mathbb{E}[(h(\mathbf{X}) - f(\mathbf{X}))^2] + \mathbb{E}[(g(\mathbf{Z}) + \epsilon)^2] - 2\mathbb{E}[(g(\mathbf{Z}) + \epsilon)(h(\mathbf{X}) - f(\mathbf{X}))] \\
&= \mathbb{E}[(h(\mathbf{X}) - f(\mathbf{X}))^2] + \mathbb{E}[(g(\mathbf{Z}) + \epsilon)^2] \\
&- 2\mathbb{E}_{\mathbf{X}}\mathbb{E}[g(\mathbf{Z})h(\mathbf{X})|\mathbf{X}] + 2\mathbb{E}_{\mathbf{X}}\mathbb{E}[g(\mathbf{Z})f(\mathbf{X})|\mathbf{X}] + 2\mathbb{E}_{\mathbf{X}}\mathbb{E}[\epsilon h(\mathbf{X})|\mathbf{X}] - 2\mathbb{E}_{\mathbf{X}}\mathbb{E}[\epsilon f(\mathbf{X})|\mathbf{X}] \\
&= \mathbb{E}[(h(\mathbf{X}) - f(\mathbf{X}))^2] + \mathbb{E}[(g(\mathbf{Z}) + \epsilon)^2] \\
&= \mathbb{E}[(h(\mathbf{X}) - f(\mathbf{X}))^2] + \mathbb{E}_{\mathbf{X}}\mathbb{V}[Y|\mathbf{X}],
\end{aligned}$$

and since $\mathbb{E}[(h(\mathbf{X}) - f(\mathbf{X}))^2]$ is strictly nonnegative, we have that $\text{MSE}(h) \geq \text{MSE}(f)$ for any h that satisfies the constraints. This shows that no function h can achieve a population MSE lower than the function f with respect to which the data was curated in the first place.

This case may be too extreme, as in practice improvements over simple models are observable even in curated datasets. These improvements can be explained in either of two ways.

E.1 Case I: approximating the true function $g(\mathbf{Z})$

First, it is very reasonable to remove the constraint $\mathbb{E}[h(\mathbf{X})g(\mathbf{Z})|\mathbf{X}] = 0$ from the family of functions that h is chosen from. In practice, this means that the model we fit to the data will be able to glean some of $g(\mathbf{Z})$ from the error term $Y - f(\mathbf{X})$, and, therefore, potentially achieve lower MSE than f. Specifically, in this case we have:

$$\text{MSE}(h) = \mathbb{E}[(h(\mathbf{X}) - f(\mathbf{X}))^2] + \mathbb{E}_{\mathbf{X}}\mathbb{V}[Y|\mathbf{X}] - 2\text{Cov}(h(\mathbf{X}), g(\mathbf{Z})),$$

and therefore h will be an improvement over f as long as $2\mathbb{E}[h(\mathbf{X})g(\mathbf{Z})] > \mathbb{E}[(h(\mathbf{X}) - f(\mathbf{X}))^2]$: the covariance between $g(\mathbf{Z})$ and $h(\mathbf{X})$ must be strong enough to offset the bias $\mathbb{E}[(h(\mathbf{X}) - f(\mathbf{X}))^2]$. Even if this were the case, it is unlikely that the improvement of h over f will be too substantial: this is due to the bias-variance tradeoff inherent in all models, as trying to learn $g(\mathbf{Z})$ from the residual $Y - f(\mathbf{X})$, while learning $f(\mathbf{X})$ at the same time is essentially a noise-minimization problem, which will lead to overfitting in any finite sample.

E.2 Case II: confounding

The second case in which more complex models may give an improvement over simple ones in curated data is one in which $\mathbb{E}[g(\mathbf{Z})|\mathbf{X}] \neq 0$, that is $\mathbf{Z}$ is an omitted confounder in the relationship between $\mathbf{X}$ and Y and cannot be ignored safely. Fitting f to the data will result in an MSE equal to:

$$\text{MSE}(f) = \mathbb{E}[(g(\mathbf{Z}) + \epsilon)^2],$$

and fitting a, potentially complex, function h to the same data will result in a population MSE of:

$$\text{MSE}(h) = \mathbb{E}[(g(\mathbf{Z}) + \epsilon)^2] + \mathbb{E}[(h(\mathbf{X}) - f(\mathbf{Z}))^2] + 2\mathbb{E}[g(\mathbf{Z})f(\mathbf{X})] - 2\mathbb{E}[h(\mathbf{X})g(\mathbf{Z})].$$

Now there is an additional term, $2\mathbb{E}[g(\mathbf{Z})f(\mathbf{X})]$, that can result in an offset to the bias and overall MSE lower than $\text{MSE}(f)$. Again, it is hard to define specific cases in which these latter two terms will be enough to offset the bias in the first two, especially in view of the fact that, once again, h will be subject to the bias-variance tradeoff.