Avoiding Overfitting in Adaptive Data Analysis

Kellie J. MacPhee
CSE 544 – Winter 2018
University of Washington
Final Project Report

Abstract

We investigate the Thresholdout method [2] for allowing (limited) reuse of a dataset (the holdout set) in building machine learning models, while also maintaining the statistical validity of reported results. Over time, such reuse of a holdout set by data analysts naturally occurs, and Thresholdout provides a simple method for integrating, directly into databases, safeguards against the degradation of generalizable results that can occur. We report the results of experiments on both synthetic and real data. Modest gains due to Thresholdout can be seen with both types of data.

1 Introduction

In recent years, a growing body of work has called into question the statistical validity of conclusions reported in research across a wide range of scientific disciplines. This phenomenon has been called the "crisis of reproducibility," in reference to the fact that results in one research paper cannot be reproduced independently by other researchers. Often, these issues stem from a lack of independence between data and analysis. Here we discuss how this problem pertains to machine learning, and investigate a proposed solution arising out of work on differential privacy.

1.1 The problem

The statistical validity of conclusions drawn from a given dataset is dependent on the assumption that the data analyst's hypotheses are fixed ahead of time, so that the structure of any tests run on the dataset has not been informed by the dataset itself. In reality, though, researchers and analysts often interact adaptively with the data: exploring or querying a dataset, forming hypotheses, then querying again to test those hypotheses, and repeating. For example, in building machine learning models, the structure of the model is often informed by the structure of the data. This can appear in terms of data preprocessing (e.g. variable selection, data rescaling, and the use of particular kernels), the choice and complexity of model (e.g. SVM vs. neural network, number of hidden nodes in a neural network), hyperparameter estimation, and other aspects of model selection.

In the ideal setting, an analyst has access to both a training set of data, which can be exploited in any way desired, and a holdout or testing set, which is used *exactly once* to test the validity of the model (which is completely specified before interacting with the holdout set). In reality, though, performance of the model on the holdout set is often used to eliminate possible model structures which do not perform well on the holdout set. This kind of adaptive data analysis can, unfortunately, lead to overfitting of the model to the holdout set, and the formation of invalid conclusions about the accuracy of the model based on its performance on the holdout set.

1.2 Connections to differential privacy

In the past few years, an attempt to address this issue has been made by researchers using ideas from differential privacy. A nice summary of the major results connecting differential privacy and generalizability appears in [2], which is connected to work in [1, 4, 6].

In [2], the authors show that for any differentially private algorithm $\mathcal{M}$ which outputs a function from the data universe $\mathcal{X}$ to $[0, 1]$, with high probability the empirical average of the function that $\mathcal{M}$ produces on a random dataset will be close to the true expectation of that function over the full population. This is analogous to the Chernoff-Hoeffding concentration inequalities, and allows the authors of [2] to conclude the existence of an algorithm that, for any large enough dataset, can with high probability answer a fixed number of adaptively chosen statistical queries up to a given tolerance τ. This algorithm is based on the multiplicative weights update method of [3], but is unfortunately not computationally efficient.

A simpler algorithm, which is computationally efficient and enjoys a similar guarantee, is based on perturbing answers to queries with Laplace noise [2]. This is the method we will consider.

1.3 The Thresholdout method

The Thresholdout method [2] is a specific instance of the computationally efficient method just described, which limits the amount of data that can be obtained from a sequence of (adaptive) queries to the holdout set. Thresholdout answers *counting queries* or *linear queries* asking for the empirical value of a function $\phi : \mathcal{X} \to [0, 1]$ on a reusable holdout set S_h, with either:

1. the answer on the training set S_t (if similar enough to the answer on the holdout set), or
2. the true answer on the holdout set, perturbed by Laplacian noise.

The number of times Thresholdout can provide answers of the second form is limited in advance by a fixed budget B. The analyst can thus potentially ask an exponential number of queries, but can only receive B answers that provide any additional information about the holdout set; the rest simply reiterate what is already known.

The full method is specified in Algorithm 1. Here $\mathcal{E}_S[\phi]$ denotes the empirical value of the function ϕ on the set S.

Algorithm 1: Thresholdout

Input : Training set S_t, holdout set S_h, noise rate σ, budget B, threshold T

Set $\hat{T} \leftarrow T + \gamma$ for $\gamma \sim \text{Lap}(2\sigma)$
for *each* $\phi : \mathcal{X} \to [0, 1]$ **do**
  If $B < 1$, output NULL.
  Else, sample $\xi \sim \text{Lap}(\sigma)$, $\gamma \sim \text{Lap}(2\sigma)$, and $\eta \sim \text{Lap}(4\sigma)$
  if $|\mathcal{E}_{S_h}[\phi] - \mathcal{E}_{S_t}[\phi]| > \hat{T} + \eta$ **then**
    Output $\mathcal{E}_{S_h}[\phi] + \xi$ and set $B \leftarrow B - 1$ and $\hat{T} \leftarrow T + \gamma$
  else
    Output $\mathcal{E}_{S_t}[\phi]$
  end
end

Theoretically, the Thresholdout method as written in Algorithm 1 achieves the following guarantee [2].

Theorem 1.1. *Let S denote a holdout set of size n drawn i.i.d. from a distribution $\mathcal{P}$, and let S_t be an additional dataset over $\mathcal{X}$ (the training set). Consider an algorithm that is given access to S_t and adaptively chooses functions $\phi_1, \ldots, \phi_m$ while interacting with Thresholdout, where $m \geq B > 0$. Let $\beta, \tau > 0$. The inputs to Thresholdout are the datasets S and S_t, and the values B, $\sigma = \tau/(96 \ln(4m/\beta))$, and $T = 3\tau/4$.*

For every $i \in [m]$, we define the counter of overfitting

$$Z_i := |\{j \leq i : |\mathcal{P}[\phi_j] - \mathcal{E}_{S_t}[\phi_j] > \tau/2\}|.$$

That is, Z_i denotes the number of rounds $j \leq i$ at which our empirical estimate of ϕ_j on the training set differs from the true value of ϕ_j (over the distribution $\mathcal{P}$) by at least $\tau/2$.

Additionally, for every $i \in [m]$ let a_i denote the answer of Thresholdout on function $\phi_i : \mathcal{X} \to [0, 1]$. Then

$$\textit{Pr}[\exists i \in [m] \textit{ s.t. } Z_i < B \textit{ and } |a_i - \mathcal{P}[\phi_j]| \geq \tau] \leq \beta,$$

whenever $n \geq n_0$ for

$$n_0 = O\left(\frac{\ln(m/\beta)}{\tau^2} \cdot \min\left\{B, \sqrt{B \ln(\ln(m/\beta)/\tau)}\right\}\right).$$

Essentially, Theorem 1.1 tells us that when the size $|S_h|$ of the holdout set is large enough, with probability $1 - \beta$ the answers outputted by Thresholdout will be within τ of the true answer over the population. (This holds, at least, for rounds at which we have not yet exceeded the budget B for the number of queries on which the training set S_t differs from the population by at least $\tau/2$.)

Note that because the counter Z_i in Theorem 1.1 is not explicitly computable (since we do not know the population averages $\mathcal{P}[\phi_j]$), setting the parameters of Thresholdout using Theorem 1.1 is not entirely straightforward, and must be done heuristically. The aim of this project is to test the effectiveness of the Thresholdout method at preventing overfitting to the holdout set. The parameters we use will be reported in our results.

2 Our tests

We test the performance of Thresholdout in the setting of **variable selection** for building linear models, where a data analyst:

1. Chooses the top k attributes to include in the model, based on the size of their correlations with the labels in the training set S_t. (The analyst only wants to include variables which the labels actually depend upon.)
2. Checks that the selected variables are reasonable using the holdout set S_h, by querying S_h on the same correlations and confirming that the correlations in S_t and S_h have the same sign, and are both above a fixed threshold. Otherwise, drops this variable.

After performing this variable selection, the remaining attributes are used to build a support vector classifier (SVC) based on the training data only. We report the accuracy of this SVC on S_t, S_h, and on "fresh" data unseen by the data analyst during the model building phase.

The problem that we hope Thresholdout will alleviate is not the discrepancy between reported accuracies on S_t and S_h, but rather *the discrepancy between the reported accuracy on S_h and the accuracy on the "fresh" data.* If this discrepancy is large, then the SVC has overfit to the holdout set, and the reported accuracy on the holdout set no longer generalizes to the accuracy on the population.

We will measure the performance of Thresholdout relative to the "standard" holdout reuse, in which the analyst is allowed to ask an unlimited number of queries of the holdout set and obtains exact answers to every query.

All tests have been implemented in Python. After performing the variable selection, the support vector classifiers are generated using the `scikit-learn` software [5].

2.1 Data

We test the effectiveness of the Thresholdout method in two settings. First, we replicate the results of [2] on synthetic data. In particular, we generate data points (rows) with d attributes, where the attributes are independent $N(0, 1)$ random variables. Binary labels are assigned uniformly at random (independent of the attributes) to each data point. *At a population level, there is no connection*

between any attribute and the label. Thus no classifier can have true accuracy above 50% on the population.

Second, we test the effectiveness of the Thresholdout method on real data. We use a dataset of presidential election results by county in 2016, which forms a subset of the "2012 and 2016 Presidential Elections" dataset [7] shared on Kaggle.com. The voting dataset includes $d = 50$ attributes, which are different numeric measures of county demographics (e.g. total population, population density, percentages of various minority groups, percentage of population with various education levels, median household income). The binary labels assigned to each county correspond to whether that county voted majority Democratic or Republican in the election.

3 Results

3.1 Synthetic experiments

Following the experiments in [2], we first test the Thresholdout algorithm on synthetic data, which was generated as described in the introduction. This setting is in a sense the most extreme case: since the attributes are independent of the labels, any degree of fitting to the holdout set is an overfitting, and we expect any adaptive reuse of the holdout set to lead to overestimates of the model's validity.

We note that these experiments are identical to those in [2], except that they used Gaussian rather than Laplacian noise. (This was an empirical decision, not yet supported by the analogous theory.) The results obtained with both types of noise are nearly identical.

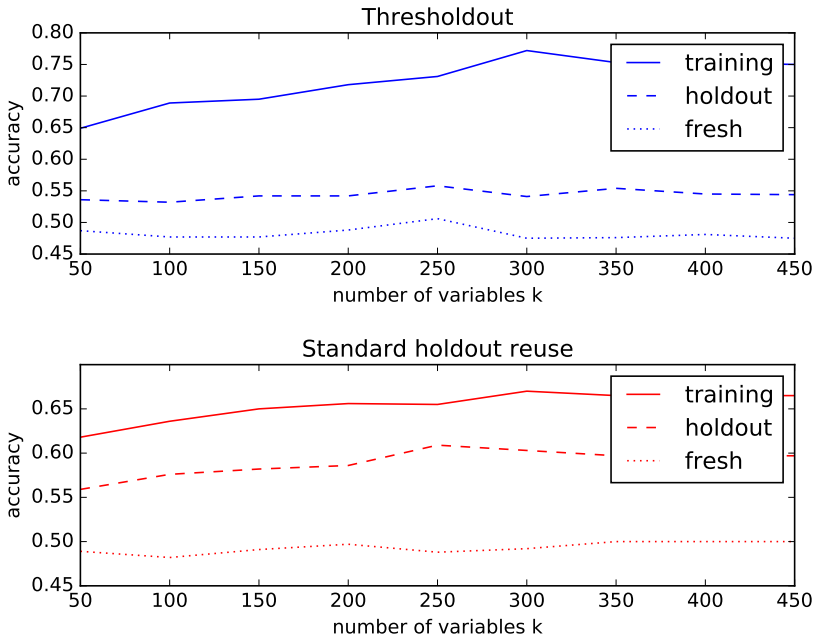

Figure 1: Replication of synthetic experiments in [2]. Here $d = 1000$ and $|S_t| = |S_h| = 1000$. Attributes are independent $N(0, 1)$ random variables and labels are uniform over $\{0, 1\}$. We set $B = 450$, $\sigma = 0.01$, and $T = 0.04$ in the Thresholdout algorithm.

The results, reported in Figure 1, show that the discrepancy between accuracies reported on the holdout set and on fresh data are nontrivial in both the standard holdout reuse and with Thresholdout. However, the discrepancy is significantly smaller when Thresholdout is used, consistently over a varying (maximum) number k of selected variables.

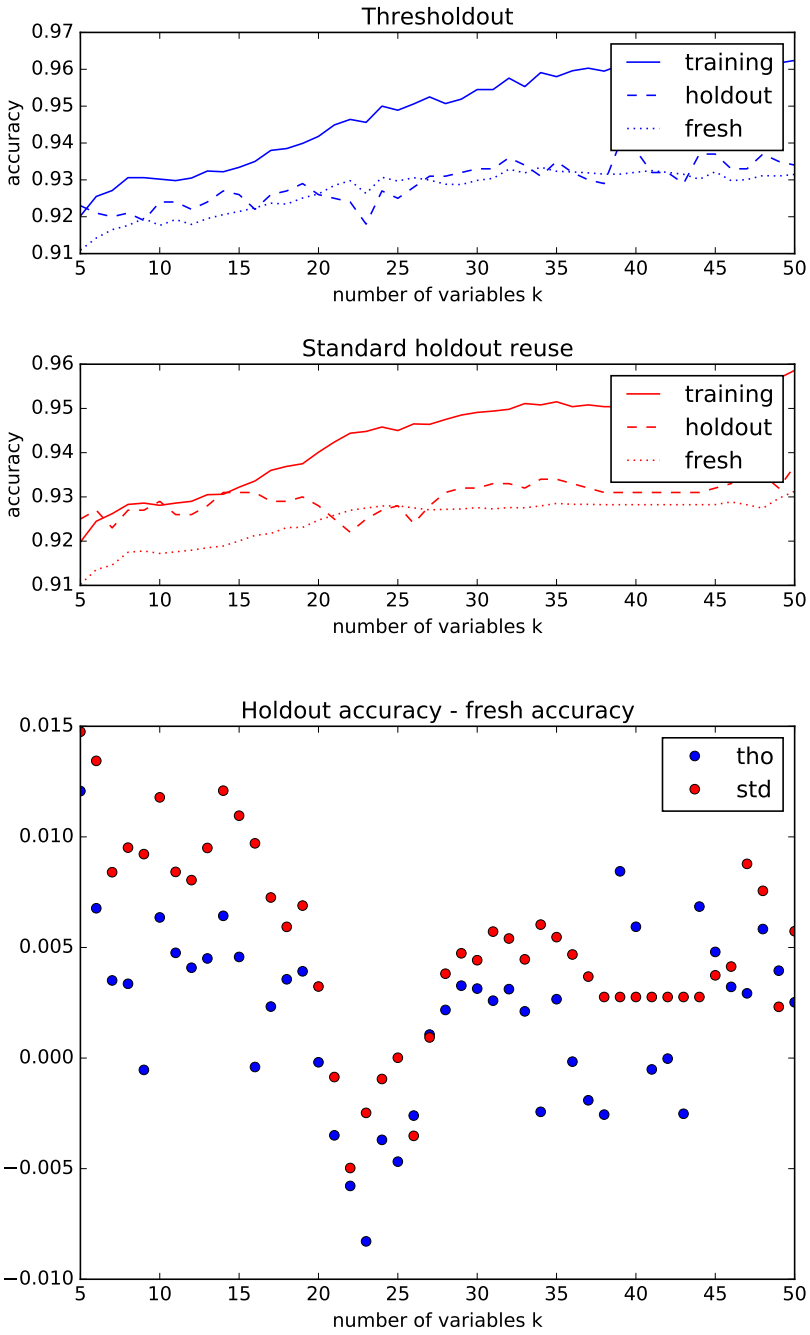

Figure 2: Average accuracy over 20 experiments on the 2016 election data. Here $|S_t| = 500$ but $|S_h| = 50$. We set $B = 25$, $\sigma = 0.1$, and $T = 0.4$. Thresholdout is denoted by "tho" and standard reuse is denoted by "std."

3.2 Experiments with real data

We now report the results of our experiments on real data, using the dataset [7] on county demographics and 2016 presidential election results. Relative to the data for the synthetic experiments, in this dataset the correlations between attributes and labels are much higher, and the total number of

attributes is rather small. When $|S_h|$ is close to the number of attributes ($d = 50$), though, we still see modest gains for Thresholdout over the standard holdout reuse.

Figures 2 illustrates the average reported accuracies on S_t, S_h, and fresh data over 20 experiments. Here different experiments correspond to different (random) splits of the dataset into training, holdout, and fresh data. Note that the discrepancy between reported accuracies on the holdout set and on the fresh data is consistently smaller when we use Thresholdout, although the difference is modest.

This difference disappears when we increase $|S_h|$, which is to be expected since the larger the holdout set is, the more similarly it should behave to the population. Thus the effects of overfitting to the holdout set are small: we cannot bias ourselves away from the truth on the population very much, even if we bias towards the holdout set. This can be seen when we increase to $|S_h| = 500$ in Figure 3, where for both the Thresholdout method and the standard reuse method, the discrepancy between accuracies reported on the holdout and fresh datasets is extremely small.

4 Conclusions

Empirically, the setting in which Thresholdout appears most useful for combatting overfitting to a holdout set is when the holdout set is small relative to the full number of features available in the data. This is a common setting in machine learning: in order to maximize the amount of data available to train the model, a relatively small number of samples (say, 10% of the original data) are reserved for testing the model's accuracy, while the rest (say 90%) are used for training. In this setting, the data analyst adaptively querying the holdout set to confirm the validity of model choices, e.g. checking whether the selected variables make sense in the holdout set as well as the training set, can easily cause overfitting to the holdout set, which is likely to be anomalous in some way because of its small size. When both the holdout set and the training set are large, on the other hand, results are more likely to be generalizable to the population.

When the holdout set *is* small, we have seen that in our results that Thresholdout improves the generalizability of reported accuracies on a holdout set, in both synthetic and real data. The gains were modest on real data, but Thresholdout was still consistently better than the standard reuse method.

5 Future work

In the future, it would be useful to test the performance of Thresholdout on a wider variety of datasets. This was outside the scope of this project, but would be a relatively straightforward extension.

It would also be interesting to investigate the performance of Thresholdout in settings other than variable selection. For example, one might consider model type selection (e.g. choosing from neural networks vs support vector machines vs. other types of machine learning models), choosing kernels or other variable transformations, or hyperparameter tuning. All of these choices about model *structure* are typically based on the observed structure of the data, and in practice may take into account the testing data or holdout set in addition to the training set. These choices are thus susceptible to causing the same issues with statistical validity as variable selection.

Finally, an implementation of Thresholdout integrated directly into a database management system is desirable. The simplicity of Thresholdout should make this a reasonable goal, especially because of its similarity to methods from differential privacy, which have already been put to use more widely. An implementation would allow data to be made available to analysts with limits on their interaction with the pre-specified holdout set, and should hopefully lead to more reproducible and generalizable results.

References

[1] R. Bassily, K. Nissim, A. Smith, T. Steinke, U. Stemmer, and J. Ullman, *Algorithmic stability for adaptive data analysis*, Proceedings of the forty-eighth annual acm symposium on theory of computing, 2016, pp. 1046–1059.

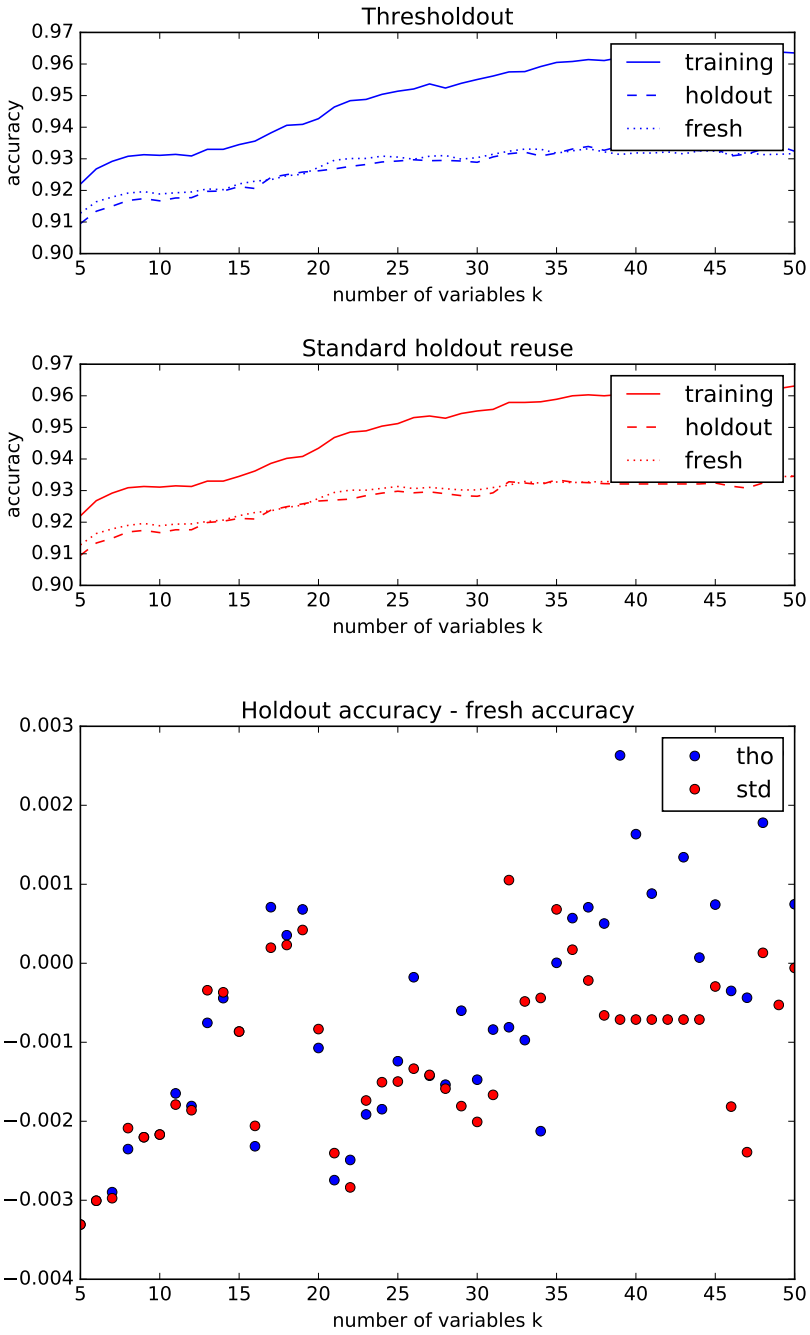

Figure 3: Average accuracy over 20 experiments on the 2016 election data, with a larger holdout set. Here $|S_t| = |S_h| = 500$. We set $B = 25$, $\sigma = 0.1$, and $T = 0.4$. Thresholdout is denoted by "tho" and standard reuse is denoted by "std."

[2] C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. Roth, *Guilt-free data reuse*, Communications of the ACM **60** (2017April), no. 4, 86–93.

[3] M. Hardt and G. N Rothblum, *A multiplicative weights mechanism for privacy-preserving data analysis*, Foundations of computer science (focs), 2010 51st annual ieee symposium on, 2010, pp. 61–70.

[4] M. Hardt and J. Ullman, *Preventing false discovery in interactive data analysis is hard*, Proceedings of the 2014 ieee 55th annual symposium on foundations of computer science, 2014, pp. 454–463.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine learning in Python*, Journal of Machine Learning Research **12** (2011), 2825–2830.

[6] T. Steinke and J. Ullman, *Interactive fingerprinting codes and the hardness of preventing false discovery*, Proceedings of the 28th conference on learning theory, 201503, pp. 1588–1628.

[7] J. Wilson, *2012 and 2016 presidential elections*. `https://www.kaggle.com/joelwilson/2012-2016-presidential-elections`.