Subpopulation Data Poisoning Attacks

Matthew Jagielski
Northeastern University
jagielski.m@husky.neu.edu

Paul Hand
Northeastern University
p.hand@northeastern.edu

Alina Oprea
Northeastern University
a.oprea@northeastern.edu

Abstract

Machine learning applications increasingly leverage large, diverse datasets. It is difficult to guarantee the trustworthiness of data as the size of the dataset grows, introducing the possibility of data poisoning attacks. In this work, we introduce a novel data poisoning attack called a *subpopulation attack*, which is particularly relevant when datasets are large and diverse. As a result, we find that our attacks are relevant for finance applications and tasks with fairness concerns.

1 Introduction

Modern machine learning applications leverage large, diverse datasets. Their size make it difficult to guarantee the trustworthiness of the data—it is intractible to inspect every record in a dataset. In very diverse datasets, the performance of a sufficiently complex model on one subpopulation can become uncorrelated with the performance on a different subpopulation. This property arises frequently in research on algorithmic fairness, where fair classifiers act differently for different subpopulations (Hardt et al. [2016]), or where good performance on one subpopulation does not imply good performance on another (Buolamwini and Gebru [2018]). We also claim that this concern is of additional interest to the finance community—consumer behavior has tremendous diversity, and financial datasets are very large.

We propose a novel form of data poisoning attack which is particularly relevant for large, diverse datasets, which we call *subpopulation attacks*. In this attack, an adversary's goal is to compromise the performance of a classifier on a particular subpopulation, while maintaining its performance for the rest of the population. We propose a simple algorithm for conducting subpopulation attacks, then run experiments to explore their efficacy based on the subpopulation under attack. Our algorithm is black-box in the target's dataset and model, and the attack does not require modifying points at test time, which has been a common thread in prior work. Additionally, the size of the attack is small relative to the overall dataset, making it stealthy. We find that the structure of a dataset itself can be used to identify vulnerable subpopulations, and that our attack can effectively exploit existing weaknesses of machine learning, such as dataset bias.

2 Background

2.1 Machine Learning Basics

Consider a training set of examples $D = \{x_i, y_i\}_{i=1}^n$, with each $x_i \in \mathcal{X}, y_i \in \mathcal{Y}$ drawn from some distribution $\mathcal{D}$. We consider binary classification tasks (although these attacks can be extended to more general tasks), where $\mathcal{Y} = \{0, 1\}$. The goal of a learning algorithm A, when given dataset D

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

is to return a function f maximizing $\mathbb{E}_{x,y\sim\mathcal{D}}\left[f(x)=y\right]$.

2.2 Poisoning Attacks

In settings with large training sets, machine learning is vulnerable to *poisoning attacks*, where an adversary is capable of adding data into the training set. This is typically because data is collected from a large number of sources which cannot all be trusted. For example, OpenAI trained their GPT-2 model on all webpages where at least 3 users of the social media site Reddit had interacted with the link (Radford et al. [2011]). Google also trains word prediction models from data collected from Android phones (Hard et al. [2018]).

More formally, the adversary adds m *contaminants* $D_p = \{x_i^c, y_i^c\}_{i=1}^m$ to the training set, so that the learner minimizes the poisoned objective $L(f, D \cup D_p)$ rather than $L(f, D)$. The set D_p is constructed to achieve some objective. Prior work has considered the following objectives: (1) misclassifying as many points as possible (called an availability attack, see Biggio et al. [2012], Mei and Zhu [2015], Xiao et al. [2015], Jagielski et al. [2018]), (2) misclassifying a single target point (called a targeted attack, see Koh and Liang [2017], Suciu et al. [2018], Shafahi et al. [2018]), and (3) predictably misclassifying out-of-distribution points (called backdoor attacks, see Chen et al. [2017], Gu et al. [2017]).

3 Subpopulation Attacks

3.1 Threat Model

We consider a black-box adversary who has no information on the learning algorithm (model details, training procedure) and does not have access to the training dataset D. We do, however, allow the adversary a large auxillary dataset D_{aux} which is distinct from D, but sampled from the same distribution. While this assumption could in theory be removed with a good generative model, we leave exploring this possibility to future work. The adversary cannot modify points at test time, as required by prior work on backdoor attacks (Chen et al. [2017] and Gu et al. [2017]). The adversary must remain stealthy and practical by adding a small number of contaminants relative to the total dataset size $|D| = n$ (our experiments use attacks with at most $0.023n$ contaminants).

3.2 Definition

A subpopulation attack is an interpolation between a targeted attack (misclassifying a single point) and an availability attack (misclassifying as many points as possible). The adversary's goal is twofold—impact the predictions on inputs coming from a subpopulation in the data, but do not impact the performance of the model on points outside this subpopulation. Crucially, this subpopulation consists of natural data, and does not require modifying points to observe the attack, as is the case for backdoor attacks. We allow the adversary to pick a subpopulation by selecting a *filter function*, which partitions the population into the subpopulation to impact and the remainder of the data, whose performance should not change. Formally, we write:

Definition 3.1. *Subpopulation Attacks.* Fix some learning algorithm A and training dataset D (which may or may not be known to the adversary). A subpopulation attack consists of a dataset of contaminants D_p and a *filter function* $\mathcal{F} : \mathcal{X} \to \{0, 1\}$. D_p is constructed to minimize the collateral damage (on points not in the subpopulation) and maximize the target damage (on points in the subpopulation) when appended to the training set:

$$\textsc{Collat}(\mathcal{F}, D_p) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\mathbb{1}\left(A(D \cup D_p)(x) \neq y\right) - \mathbb{1}\left(A(D)(x) \neq y\right) \mid \mathcal{F}(x) = 0\right] \quad (1)$$

$$\textsc{Target}(\mathcal{F}, D_p) = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[\mathbb{1}\left(A(D \cup D_p)(x) \neq y\right) - \mathbb{1}\left(A(D)(x) \neq y\right) \mid \mathcal{F}(x) = 1\right] \quad (2)$$

We will evaluate subpopulation attacks by reporting the collateral damage (Equation 1) and the target damage (Equation 2) on the test set D_{test}. A good attack will have small collateral damage and large target damage.

The choice of filter function is as important to the adversary as the selection of contaminants. There may be some choices of filter function which are hard to learn, as shown in Figure 1. For this

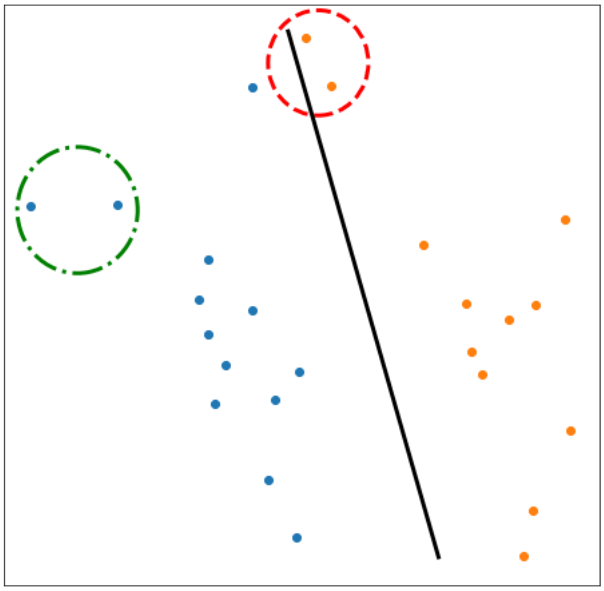

Figure 1: The red dashed circle is a good filter function—points in the circle could be easily misclassified without harming any other predictions. The green dash-dot circle is a bad filter function—it is impossible to misclassify points in that circle with a linear classifier without modifying predictions of a significant fraction of points.

reason, we will propose two techniques to construct filter functions, called FEATUREMATCH and CLUSTERMATCH, and a simple poisoning algorithm. In the experimental section, we will investigate properties of effective filter functions.

3.3 Filter Function Selection

3.3.1 FEATUREMATCH

In FEATUREMATCH, the adversary has specific features and values in mind to target. For example, if a dataset has a race and gender information, an adversary may want to harm the performance specifically for black men. The following is a filter function for this goal:

$$\mathcal{F} = \mathbb{1}(\text{"race"} = \text{"black"} \wedge \text{"gender"} = \text{"male"}).$$

Given a list of feature indices f and corresponding list of values V, we can produce a filter function $\mathcal{F} = \text{FEATUREMATCH}(f, V)$ as follows:

$$\text{FEATUREMATCH}(f, V)(x) = \bigwedge_{i \in f} (x_i \in V_i)\,.$$

3.3.2 CLUSTERMATCH

If we can identify a natural cluster in the data, attacking one may allow us to compromise the model for that cluster but not elsewhere. Following this intuition, we propose CLUSTERMATCH (see Algorithm 1), in which the adversary produces clusters and targets only a specific cluster. In our experiments, we use KMeans for clustering, but any algorithm which generates meaningful clusters for a given dataset should work.

In CLUSTERMATCH, the adversary does not have direct control over the subpopulations to attack, which limits but does not remove motivation for subpopulation attacks. For example, consider an adversary who wishes to disrupt street sign detection in a self-driving car through a subpopulation attack—they could run clustering to identify vulnerable street signs which will be easiest to target, and attack those, increasing the impact and stealth of their attack. In general, an adversary can generate clusterings and identify a cluster that is both aligned with their goals and will be easy to attack.

Algorithm 1 CLUSTERMATCH

Input: Dataset D, cluster count k, target cluster t
CLUSTERFUNC = FINDCLUSTERS(D, k) ▷ classifies a point into one of k clusters
return $\mathcal{F} = \mathbb{1}(\text{CLUSTERFUNC}(x) = t)$

3.3.3 Selecting a filter function

Both of these techniques are capable of making many different filter functions. For example, if clustering returns 100 clusters in CLUSTERMATCH, the adversary has 100 choices of filter function, but must select one for the poisoning attack. One way to do this is clear—if the adversary only cares about attacking members of one subpopulation, they should only target that subpopulation. However, if there are multiple subpopulations of interest (or the adversary wants to optimize the chance the attack will be successful), an adversary may be interested in knowing which one will be most susceptible to attack. In the experimental section, we explore approaches to selecting the most effective filter functions to target. We also evaluate both FEATUREMATCH and CLUSTERMATCH on the best filter functions the attack produces.

3.4 A Simple Poisoning Algorithm

We adapt a common baseline algorithm, random flipping, to our setting. This algorithm is presented in Algorithm 2. The adversary needs to pick an attack size m, which should be comparable to the size of the subpopulation itself. The subpopulation should be small relative to the full dataset, making the attack require few contaminants. To add m contaminants, they sample m points satisfying the filter function from D_{aux} and add these to the training set with flipped labels. This achieves the first goal—misclassify points satisfying the filter function. The filter function itself is what guarantees we do not misclassify other points—if it represents a good enough separation, then the learning algorithm will be able to learn the poisoned function for the targeted subpopulation.

Algorithm 2 Random flipping subpopulation attack.

Input: Auxillary dataset D_{aux}, filter function $\mathcal{F}$, attack size m
$D_p = \{\}$
while $|D_p| < m$ **do**
  for $(x, y) \in D_{aux}$ **do**
    if $\mathcal{F}(x)$ **then**
      $D_p = D_p \cup \{(x, 1 - y)\}$
return D_p

3.5 Putting It All Together

All we require for a subpopulation attack is a subpopulation to target (i.e., a filter function) and an algorithm for generating the poison set given the filter function. Here, we consider creating subpopulations with FEATUREMATCH and CLUSTERMATCH, selecting these subpopulations arbitrarily, and using a random flipping poisoning attack to generate the poison set. Each component of the attack could be replaced and improved by other techniques, a task we leave to future work. In Algorithm 3, we present an algorithm for subpopulation attacks in full generality.

Algorithm 3 Generic Subpopulation Attack. In this work, we consider K_{adv} to consist only of an auxillary dataset D_{aux}.

Input: Adversarial knowledge K_{adv}, attack size m
FilterFuncs = MAKEFILTERFUNCS(K_{adv}) ▷ e.g., FEATUREMATCH or CLUSTERMATCH
$\mathcal{F}$ = SELECTFILTERFUNC(FilterFuncs, K_{adv})
return GENERATEATTACK$(\mathcal{F}, m, K_{adv})$ ▷ e.g., random flipping

4 Experiments

To validate the threat of subpopulation attacks, we run our attacks on the Adult dataset from the UCI machine learning repository. We ask the following research questions: (1) Which of FEATUREMATCH and CLUSTERMATCH produces the best attacks? (2) What properties of a filter function make it effective? (3) What is the concrete impact of our attacks?

4.1 Experiment Setup

We use the UCI Adult dataset (Dua and Graff [2017]), where the goal is to use demographic information (race, gender, education, work type, etc.) to predict whether a person has an income of $\geq$50K USD. This is a representative task which may have fairness concerns (Kearns et al. [2019]). This is an unbalanced dataset—we downsample both the training and testing sets to make a separate balanced task. We drop the education (duplicated), fnlwgt (an aggregate metric), and native-country (very large and unbalanced feature) columns, and one-hot encode all remaining categorical features. We split the training set into two parts, the training set D, and the adversary's auxillary dataset D_{aux}. After these steps, D has 7841 samples, D_{aux} has 7841 samples, and D_{test} has 7692 samples, all with 57 features. We run our experiments with a neural network model with 10 hidden units trained with scikit-learn (Pedregosa et al. [2011]), reaching 83.5% accuracy on the unpoisoned data. All other parameters use the scikit-learn defaults, and experiment results are averaged over 3 trials.

For FEATUREMATCH, we use every combination of the (race, gender, education level) features present in D_{aux}. For CLUSTERMATCH, we run the KMeans clustering algorithm on D_{aux} with 100 clusters. For both algorithms, we exclude filter functions with fewer than 10 samples (these are too small to draw conclusions from), and with more than 100 samples from D_{aux} (poisoning these would require more samples, causing a more noticeable attack). This results in 31 filter functions from CLUSTERMATCH, and 35 filter functions from FEATUREMATCH. In order to evaluate the impact of the attack as a function of the number of samples, we first measure for each filter function $\mathcal{F}$ the number of elements in D satisfying $\mathcal{F}$: $n_{\mathcal{F}} = |\{x | x \in D, \mathcal{F}(x) = 1\}|$. For each filter function, we measure our two metrics when attacking with $\{n_{\mathcal{F}}/3, 2n_{\mathcal{F}}/3, n_{\mathcal{F}}, 4n_{\mathcal{F}}/3, 5n_{\mathcal{F}}/3, 2n_{\mathcal{F}}\}$ contaminants. The attack is black box—measuring $n_{\mathcal{F}}$ is the only time the attack depends on D, and this is only to fairly evaluate each attack's strength. The attacks are generated with the random flipping attack in Algorithm 2, repeating samples multiple times if D_{aux} does not have enough samples for the given filter function. We report results for the 1, 3, and 6 filter functions with the highest TARGET for both techniques. Code is available on Google Colab here: Jagielski et al. [2019].

The size of the attacks are small relative to the size of the dataset. Each attack size is relative to $n_{\mathcal{F}}$. Over the top 6 clusters in both FEATUREMATCH and CLUSTERMATCH, the minimum value of $n_{\mathcal{F}}$ is 11, the median value is 39, and the maximum value is 118. This means the smallest attack we run (that is effective!) uses only 2 contaminants, and the largest attack uses 177 contaminants, which is still only 2.3% of the dataset. This small sample requirement highlights the practicality of these attacks.

4.2 Results

4.2.1 Comparing FEATUREMATCH and CLUSTERMATCH

We compare the performance of FEATUREMATCH and CLUSTERMATCH in Table 1. Most notably, CLUSTERMATCH achieves significantly better target damage than FEATUREMATCH, often doubling the target damage of the top 1, 3, or 6 filter functions. CLUSTERMATCHhas slightly higher collateral damage. This supports the hypothesis that distinct clusters can be learned somewhat independently of the rest of the dataset, and offers the possibility of improving subpopulation attacks with more sophisticated unsupervised learning techniques.

4.2.2 Filter Function Strength

By sorting the TARGET metric over filter functions created by FEATUREMATCH, we can see which population subgroups the learning algorithm is the easiest to trick on. Five of the top 6 clusters consist of low education white men (on a scale of 2 to 16, the education levels are 2, 3, 7, 8, 5 in decreasing order of TARGET). The only other cluster contains highly educated (education level

Experiment	TARGET by Number of Contaminants						Maximum
	$n_{\mathcal{F}}/3$	$2n_{\mathcal{F}}/3$	$n_{\mathcal{F}}$	$4n_{\mathcal{F}}/3$	$5n_{\mathcal{F}}/3$	$2n_{\mathcal{F}}$	COLLAT
FEATUREMATCH Top 1	0.137	0.143	0.244	0.467	0.467	0.444	0.009
CLUSTERMATCH Top 1	0.364	0.542	0.667	0.661	0.923	0.667	0.031
FEATUREMATCH Top 3	0.119	0.13	0.201	0.297	0.292	0.32	0.006
CLUSTERMATCH Top 3	0.343	0.442	0.506	0.526	0.53	0.462	0.03
FEATUREMATCH Top 6	0.094	0.113	0.169	0.224	0.225	0.245	0.014
CLUSTERMATCH Top 6	0.31	0.388	0.427	0.388	0.425	0.398	0.024

Table 1: Mean TARGET and COLLAT for neural networks when the top 1, 3, and 6 clusters for the FEATUREMATCH and CLUSTERMATCH filter function selection algorithms are under attack at various numbers of contaminants (as a function of the size of the subpopulation $n_{\mathcal{F}}$). For all cluster counts, CLUSTERMATCH produces more effective filter functions than FEATUREMATCH.

Cluster	Original Accuracy	Poisoned Accuracy	# Poisoned Points
C1: Size 47	100%	7.74%	78
C2: Size 16	82.35%	39.22%	10
C3: Size 30	100%	63.63%	10
C4: Size 26	100%	56.67%	34
C5: Size 59	85.29%	36.76%	78

Table 2: CLUSTERMATCH can successfully degrade the performance of several subpopulations.

16) white women. Under attack, the model will predict low educated white men to have a higher salary, and the highly educated white women to have a lower salary. The former indicates some bias towards white men in the dataset, while the latter indicates bias against women.

4.2.3 Concrete Effectiveness of CLUSTERMATCH

In Table 2, we investigate the concrete effectiveness of CLUSTERMATCH. We find that there is a subpopulation where adding 78 poisoning points ($<$1% of the training dataset) can degrade the performance on the subpopulation from 100% accuracy to 7.74% accuracy, completely compromising the performance for this subpopulation. In another example, we find $<$0.2% of the training dataset (10 poison points) decreases the performance on the subpopulation from 100% accuracy to 39.22% accuracy. These constitute meaningful performance degradations for these subpopulations.

5 Conclusion

We proposed a novel form of poisoning attack, called subpopulation attacks. These attacks are effective when datasets are large and diverse, such as in fairness-sensitive or financial applications. They are also stealthy, can be conducted with small black-box attacks, and do not require modifying inputs at test time. We propose and evaluate two subpopulation generating algorithms, called FEATUREMATCH and CLUSTERMATCH, and adapt a standard baseline attack generation algorithm to generate subpopulation attacks. We show that subpopulation attacks are especially effective at exploiting existing biases in data. We believe there is ample opportunity for future work to improve upon our subpopulation generation, subpopulation selection, and attack generation algorithms.

References

B. Biggio, B. Nelson, and P. Laskov. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.

J. Buolamwini and T. Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91, 2018.

X. Chen, C. Liu, B. Li, K. Lu, and D. Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

D. Dua and C. Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

T. Gu, B. Dolan-Gavitt, and S. Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, 2018.

M. Hardt, E. Price, N. Srebro, et al. Equality of opportunity in supervised learning. In *Advances in neural information processing systems*, pages 3315–3323, 2016.

M. Jagielski, A. Oprea, B. Biggio, C. Liu, C. Nita-Rotaru, and B. Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 19–35. IEEE, 2018.

M. Jagielski, P. Hand, and A. Oprea. Subpopulation Data Poisoning Attacks Colab Notebook, Dec. 2019. URL `https://colab.research.google.com/drive/1qWZeEnzxO9P9lLpjIsvBSdTBZwNDK02I`.

M. Kearns, S. Neel, A. Roth, and Z. S. Wu. An empirical study of rich subgroup fairness for machine learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 100–109. ACM, 2019.

P. W. Koh and P. Liang. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1885–1894. JMLR. org, 2017.

S. Mei and X. Zhu. Using machine teaching to identify optimal training-set attacks on machine learners. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

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*, 12:2825–2830, 2011.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2011. URL `https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf`.

A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein. Poison frogs! targeted clean-label poisoning attacks on neural networks. In *Advances in Neural Information Processing Systems*, pages 6103–6113, 2018.

O. Suciu, R. Marginean, Y. Kaya, H. Daume III, and T. Dumitras. When does machine learning {FAIL}? generalized transferability for evasion and poisoning attacks. In *27th {USENIX} Security Symposium ({USENIX} Security 18)*, pages 1299–1316, 2018.

H. Xiao, B. Biggio, G. Brown, G. Fumera, C. Eckert, and F. Roli. Is feature selection secure against training data poisoning? In *International Conference on Machine Learning*, pages 1689–1698, 2015.