Enhancing the Privacy of Predictors

Ke Xu,[*] Swair Shah,[*] Tongyi Cao,[†] Crystal Maung,[*] Haim Schweitzer[*]
{ke.xu5,swair,hschweitzer}@utdallas.edu, tcao@umass.edu, Crystal.Maung@gmail.com

Abstract

The privacy challenge considered here is to prevent an adversary from using available feature values to predict confidential information. We propose an algorithm providing such privacy for predictors that have a linear operator in the first stage. Privacy is achieved by zeroing out feature components in the approximate null space of the linear operator. We show that this has little effect on predicting desired information.

The Problem Being Addressed. Consider a company that develops technology for predicting desired information from raw data. The privacy concern is the potential inappropriate use of the client data to predict confidential information that the client does not wish to expose. Let x be the feature vector, y_d the desired labels, and y_c the confidential labels. The goal is to "clean" x so that the cleaned feature vector $\tilde{x}$ can be used to accurately predict y_d, but will typically fail if used to predict y_c. Let f_d, f_c denote predictors for y_d, y_c respectively. We wish the following holds:

$$f_d(\tilde{x}) = \tilde{y}_d \approx y_d, \quad e_{\text{utility}} = |\tilde{y}_d - y_d|^2$$
$$f_c(\tilde{x}) = \tilde{y}_c \not\approx y_c, \quad e_{\text{privacy}} = |\tilde{y}_c - y_c|^2$$

Our goal is to maximize e_{privacy} while minimizing e_{utility}. We show how to achieve this for predictors with a linear operator in the first stage. These include, among others, linear regression, neural nets, and algorithms that use PCA.

Related Work. Adapting machine learning terminology of learning/testing our goal is protecting the privacy of information during testing. It is different from studies concerned with the privacy of the training data, such as differential privacy, e.g., (Dwork and Roth 2014; Sarwate and Chaudhuri 2013), where noise is added to blur the distinction between individual items in the training data. Unlike work on differential privacy we distinguish between desired and confidential information. Recent studies (Enev et al. 2012; Hamm 2015) propose similar models with different (or no) assumptions about the predictors. By contrast, we assume knowledge of these predictors, yielding different algorithms.

[*]Department of Computer Science, Univ. of Texas at Dallas.
[†]CICS, UMass Amherst.

The Main Idea. We consider a predictor f_d that starts with a known linear operator A_d. Zeroing out feature components in the approximate null space of A_d^T will not change the prediction of y_d, and will typically increase the error of predicting y_c. Consider the following toy example:

(x_1, x_2)	cleaned (x_1, x_2)	y_d	y_c
(3, 1)	(1, -1)	2	5
(4, 2)	(1, -1)	2	8
(5, 1)	(2, -2)	4	7

Both y_d and y_c can be calculated exactly from x_1, x_2 by: $y_d = x_1 - x_2$, $y_c = x_1 + 2x_2$. Here $A_d^T = (1, -1)$, with the vector $(1, 1)$ in its null space. Thus, the vector (x_1, x_2) can be cleaned by zeroing out projections on that direction.

After cleaning, the same linear model can still be used to predict y_d from the cleaned features. But it is impossible to compute y_c exactly from the cleaned features, since two different values of y_c must be inferred from two identical cleaned feature vectors (lines 1, 2).

Cleaning. We have developed several algorithms that clean feature vectors (Xu et al. 2017), making different assumptions on f_c. The simplest one cleans projections on the exact null space as shown in the above example. We describe in detail an algorithm that assumes f_c to be linear, and estimates it from training data. It is shown below.

```
   Input: $x$ (feature vector), $A_d$ (from the known predictor),
   $\epsilon$ (desired value of $e_{\text{utility}}$), training data $(x^j, y_c^j)$.
   Output: the cleaned feature vector $\tilde{x}$.
1. Use training data to compute $A_c$ so that $y_c^j \approx A_c^T x^j$.
2. For $B_d = A_d A_d^T$, $B_c = A_c A_c^T$, solve the generalized
   eigenvalue problem $B_d v = \gamma B_c v$.
3. Sort the generalized eigenvalues/eigenvectors in increasing
   order of $\gamma$. The result is denoted by: $(v_1, \gamma_1), \ldots, (v_n, \gamma_n)$.
4. $\forall i$: $a_i = v_i^T x$, $\lambda_i^d = v_i^T B_d v_i$, $\delta_i = \lambda_i^d a_i^2$
5. Find the index $t$ such that: $\sum_{i=1}^{t} \delta_i < \epsilon$, $\sum_{i=1}^{t+1} \delta_i \geq \epsilon$
6. Set $\alpha_i = \begin{cases} 1 & 1 \leq i \leq t \\ \sqrt{(\epsilon - \sum_{k=1}^{t} \delta_k)/\delta_{t+1}} & i = t+1 \end{cases}$
7. Output $\tilde{x} = x - \sum_{i=1}^{t+1} \alpha_i a_i v_i$.
```

Algorithm 1 for cleaning a feature vector x.

Theorem 1: Let $\tilde{x}$ be the result of cleaning the feature vector x with a user defined parameter ϵ. Then $e_{\text{utility}} = \epsilon$.

Proof: $y_d - \tilde{y}_d = A_d^T(x - \tilde{x}) = A_d^T \sum_{i=1}^{t+1} \alpha_i a_i v_i$

$$e_{\text{utility}} = |y_d - \tilde{y}_d|^2 = (\sum_{i=1}^{t+1} \alpha_i a_i v_i^T) B_d (\sum_{j=1}^{t+1} \alpha_j a_j v_j)$$

$$= \sum_{i=1}^{t+1} \lambda_i^d \alpha_i^2 a_i^2 = \sum_{i=1}^{t} \lambda_i^d a_i^2 + \lambda_{t+1}^d a_{t+1}^2 r^2 = \epsilon \quad \blacksquare$$

We proceed to analyze the privacy properties of Algorithm 1. The generalized eigenvalues γ_i satisfy:

$$\gamma_i = \lambda_i^d / \lambda_i^c, \quad \text{where } \lambda_i^d = v_i^T B_d v_i,\ \lambda_i^c = v_i^T B_c v_i$$

Following the same derivation as in Theorem 1 we have:

$$e_{\text{utility}} = \textstyle\sum_{i=1}^{t+1} \lambda_i^d \alpha_i^2 a_i^2 = \epsilon, \quad e_{\text{privacy}} = \textstyle\sum_{i=1}^{t+1} \lambda_i^c \alpha_i^2 a_i^2$$

Therefore we should select generalized eigenvectors with small λ^d values and large λ^c values. This suggests that the selection order should be according to increasing values of γ_j. The value of t is determined so that the $e_{\text{utility}} = \epsilon$. Thus, the algorithm solves the following optimization problem:

$$\min_{\alpha_i} \sum_{i=1}^{t+1} \alpha_i^2 \frac{\lambda_i^d a_i^2}{\lambda_i^c a_i^2} = \sum_{i=1}^{t+1} \alpha_i^2 \frac{\lambda_i^d}{\lambda_i^c} = \sum_{i=1}^{t+1} \gamma_i \alpha_i^2$$

$$\text{subject to: } 0 < \alpha_i \le 1, \quad \sum_{i=1}^{t+1} \alpha_i \lambda_i^d a_i^2 = \epsilon$$

Experimental Results. We evaluated the proposed algorithm on datasets from the Mulan (Tsoumakas et al. 2011) repository. Partial results for Algorithm 1 are shown here. The "scene" dataset has 2407 instances, each described in terms of 294 features and 6 labels. Similarly, the "wq" dataset has 1060 instances, 16 features, and 14 labels.

The results are averaged over 10 runs. In each run 90% of the dataset is chosen randomly to compute the model and the rest is used as testing data. N_d and N_c denote the number of desired and confidential labels respectively. The results are shown in Fig 1. We observe that e_{utility} is exactly ϵ, and e_{privacy} is much bigger than e_{utility}. To provide further insight into the cleaning performance we count the test cases for which the cleaning achieves "complete privacy":

If the error of predicting y_c using $\tilde{x}$ is greater than the error of predicting y_c from the mean feature vector, then ***complete privacy*** has been achieved.

The Complete Privacy results in the tables (specified as CP), give the percentage of test cases for which the cleaner achieves *complete privacy*. As the results show, typically the adversary would be no more advantageous using the cleaned feature vector $\tilde{x}$ than using the mean feature vector.

Dataset	N_d	N_c	e_{utility}	e_{privacy}	CP
scene	1	5	0.01	0.796	81.8%
	3	3	0.01	1.296	80.7%
	5	1	0.01	3.659	82.3%
wq	1	13	0.01	2.654	52.6%
	7	7	0.01	1.956	57.8%
	13	1	0.01	1.403	45.5%

Figure 1: Results with no attack, $\epsilon = 0.01$.

Dataset	N_d	N_c	e_{utility}	e_{privacy}	CP
scene	1	5	0.01	0.289	51.0%
	3	3	0.01	0.233	44.5%
	5	1	0.01	0.043	41.8%
wq	1	13	0.01	0.308	48.9%
	7	7	0.01	0.164	38.7%
	13	1	0.01	0.086	30.9%

Figure 2: Results with the proposed attack. $\epsilon = 0.01$.

e_{utility}	Laplace Mechanism		Alg 1 with attack	
	e_{privacy}	CP	e_{privacy}	CP
0.01	0.009	31.5%	0.289	51.0%
0.02	0.019	31.4%	0.289	50.3%
0.03	0.027	31.3%	0.289	51.1%

Figure 3: Comparison with the Laplace Mechanism on dataset scene. N_d is 1, N_c is 5.

A Proposed Attack. In our model the adversary knows the cleaning algorithm. Therefore, the adversary can clean training data and learn to predict y_c from the cleaned data. Fig. 2 shows the results with this attack. Observe that e_{privacy} is considerably higher than the e_{utility}. If needed, a higher privacy can be obtained by increasing ϵ.

Comparison with a Differential Privacy Mechanism. A standard way to achieve differential privacy is the Laplace mechanism (Dwork and Roth 2014). We adjust the Laplace noise parameter to produce the same e_{utility} as Algorithm 1. The results are shown in Fig. 3. Observe that even with the proposed attack, our approach achieves better complete privacy rate compared to the Laplace mechanism.

Concluding Remarks. We describe a mechanism for enhancing the privacy of predictors. Cleaning the data by zeroing out the null-space of the predictor will have a negligible effect on the prediction accuracy but will increase the privacy. The trade-off between utility and privacy can be controlled by adjusting the ϵ parameter. The proposed scheme can be applied to models that start with a linear operator.

References

Dwork, C., and Roth, A. 2014. The algorithmic foundations of differential privacy. *FTTCS* 9(3 & 4):211–407.

Enev, M., Jung, J., Bo, L., Ren, X., and Kohno, T. 2012. SensorSift: balancing sensor data privacy and utility in automated face understanding. *ACSAC*, 149–158.

Hamm, J. 2015. Preserving Privacy of Continuous High-dimensional Data with Minimax Filters. *AISTATS*, 324-332.

Sarwate, A., and Chaudhuri, K. 2013. Signal processing and machine learning with differential privacy: theory, algorithms, and challenges. *IEEE Signal Processing Magazine* 30(5):86–94.

Xu, K., Cao, T., Shah, S., Maung, C., and Schweitzer, H. 2017. Cleaning the Null Space: A Privacy Mechanism for Predictors. *AAAI* 17.

Tsoumakas, G., Xioufis, E.S., Vilcek, J., and Vlahavas, I. 2011. Mulan: A Java Library for Multi-Label Learning. *JMLR* 12.