SAFE – Secure Aggregated Frequency Estimates

Eric Bax
Verizon Media
Los Angeles, CA
ebax@verizonmedia.com

Charlotte Bax
Flintridge Prep
Los Angeles, CA
baxhome@yahoo.com

***Abstract*—We propose a system for privacy-aware machine learning. The data provider encodes each record in way that avoids revealing information about the record's field values or about the ordering of values from different records. A data center stores the encoded records and uses them to perform classification on queries consisting of encoded input field values. The encoding provides privacy for the data provider from the data center and from a third party issuing unauthorized queries. But the encoding makes regression-based and many tree-based classifiers impossible to implement. It does allow histogram-type classifiers that are based on category membership, and we present one such classification method that ensures data sufficiency on a per-classification basis.**

***Index Terms*—machine learning, privacy, big data, local classifier**

I. Privacy-Preserving Machine Learning

There are increasing concerns about data privacy [1], [2]. When a client uses a remote service to provide machine learning, the client provides data (machine learning examples), and the remote service provider uses the data to classify example inputs received from clients as queries. Risks to privacy include interception of client-server communications, data theft from or unauthorized analysis by the service provider, and information leaks through responses to queries from unauthorized clients.

Privacy-preserving machine learning techniques [3]–[5] can mitigate these risks. In many privacy-preserving machine learning schemes, instead of using data examples directly, the service provider receives encoded versions of the data, and must use the encoded data for classification tasks. The type of encoding determines which operations can be used and the speed of those operations.

Fully homomorphic encoding [6] maps values to encoded values in a way that allows addition and multiplication to be performed on the encoded values to produce the encoded sums and products. This allows classification methods that use arithmetic operations, such as regression and curve-fitting. The drawback is very slow computation, making it inappropriate for big-data settings.

Order-preserving encoding [7] allows classification using decision trees, since each split depends on determining whether a field value is above a threshold. Order-preserving encoding preserves less information about the data than fully homomorphic encoding. It enables faster operations and withholds more information from the service provider. However, order-preserving encoding still releases information about ordering (as it must), and some methods can also leak other information over time [8].

This paper outlines the potential for encodings that preserve neither arithmetic value nor ordering. Each value is mapped to an encoded value in a manner that allows neither arithmetic operations nor order comparison. This withholds more information from the service provider, and it limits classification methods. It allows for equality comparison, at full speed. This enables histogram-type classification, such as voting over in-sample examples that share all field values with the input example being classified.

Due to a combinatorial explosion, with a moderate or large number of fields per record, even a huge data set is likely to have no examples that match the inputs of many examples to be classified. So we propose a classification method that iteratively decreases the number of field-value matches needed to be a voting example until a pre-specified number of voting examples are found. The method is a local classifier, like a k-nearest neighbors classifier [9]–[11], and it allows modified forms of k-nn metric learning [12], [13] and validation [14].

II. Encoding

The data provider encodes data record-by-record before sending it to the data center. Let a record be a sequence of field values: $(x_1, \ldots, x_r)$. The values x_i include both the input values for machine learning and the output or outputs (also called labels). The encoding process for data insertion is:

1) Convert the values in each non-categorical field into categorical values. For example, if a field's values are real numbers, then partition the range of values into subranges, and replace the field value by a value representing its subrange. (***later convert a single field into a sequence of categorical fields – hierarchy, rolling windows.***)
2) Encode each field value x_i to an encoded value x'_i, using a mapping that is one-to-one from field and value to encoded value: $(i, x_i) -> x'_i$.
3) Randomly shuffle $(x'_1, \ldots, x'_r)$ to form the encoded record: set X'.

The encoded record is sent to the data center, which collects encoded records to answer queries. Queries are also encoded, but field values to be used as inputs are separated from field values that constitute a "positive" label. For example, if fields 1 to 25 are inputs and values a, b, and c of field 32 are considered

positive labels, then the encoded query has two components: a random shuffle of $(x'_1, \ldots, x_2 5')$ and a random shuffle of the encodings of (32, a), (32, b), and (32, c). Refer to the set of encoded inputs as Q' and the set of positive labels as Y'. (***later – multi-class, just give sets of y_i's that label each class***).

III. Classification

(***later – classify based on a random sample of the data set, and mention reservior sampling***) The machine learning service provider responds to each query with a frequency k of positive labels and a number n of samples used to find the frequency. For example, $k = 100$ and $n = 101$ indicates that the algorithm found 100 examples in the data set with a positive label, i.e. having some $x'_i \in Y'$, of the 101 examples used. The classification algorithm has a parameter $n_{\min}$, which is the minimum acceptable value for n. For query (Q', Y') and examples X', the algorithm attempts to use all examples with all members of Q' in X' (i.e., $Q' \subseteq X'$). If there are fewer than $n_{\min}$ such examples in the data set, then the algorithm also includes examples with all but one of the members of Q' in X' ($|Q' - X'| \leq 1$), and continues to expand the set of examples if there are fewer than $n_{\min}$:

1) $\forall X'$:
   a) $d = |Q' - X'|$ # "Distance" is number of elements of Q' not in X'.
   b) $n_d += 1$ # Add to number of distance d samples.
   c) if $|X' \cap Y'| \geq 1$ then $k_d += 1$ # If positive label, then add to successes.
2) $sk = 0$, $sn = 0$ # Sum of k and n over distances.
3) For d in $0, 1, 2, \ldots$:
   a) $sk += k_d$
   b) $sn += n_d$
   c) if $n_d \geq n_{\min}$ then return (sk, sn)

For additional privacy, k and n can be encoded (for example, using public-key cryptography [**?**]) before being sent as a response to the query. The value $\frac{k}{n}$ is an estimate of the probability that the query example has a positive label. The value $n_{\min}$ mediates a tradeoff: a higher value generally causes more examples to be used for classification; a lower value generally causes more-similar examples to be used.

HEREHow to use k and n to get a classification. (Can go with majority, and can also compare to overall k / n over whole data set for the positive labels – if it was stored while sending data, or if it is returned from data center.) What do k and n mean? confidence.

IV. Validation

Can validate overall error rate using histogram methods and/or k-nn methods. (Or something similar.) Cross-validation. (Use two holdout sets, bound the error rate over both, then use each to bound the probabilities that each set is changes a classification by being added to the non-holdout examples, and add those probabilities to the error bound.)

Can also say how much more likely the samples are to be positive than all samples.

Can weight different differences – not havind some q's in Q' in X' makes X' more "distant" from Q' than other q's. Weight by q. Learn a metric: Gert Lanckriet, UCSD.

Can use multiple labels and values to obfuscate. Need not all be real data.

Can use data with non-overlapping values to obfuscate which values are actually used. (Will never have values in common with "real" queries.)

References

[1] Regulation (EU) 2016/679 of the European parliament and of the council. *Official Journal of the European Union*, 2016.

[2] David Ramli. Apple's Tim Cook calls for more regulations on data privacy. *Bloomberg.com*, 2018.

[3] Jaideep Vaidya, Hwanjo Yu, and Xiaoqian Jiang. Privacy-preserving svm classification. *Knowledge and Information Systems*, 14(2):161–178, 2008.

[4] Irene Giacomelli, Somesh Jha, Ross Kleiman, David Page, and Kyonghwan Yoon. Privacy-preserving collaborative prediction using random forests. *AMIA Joint Summits on Translational Science proceedings. AMIA Joint Summits on Translational Science*, pages 248–257, 2019.

[5] Jonathan M. Bloom. Secure multi-party linear regression at plaintext speed, 2019.

[6] Shai Halevi and Victor Shoup. Bootstrapping for helib. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015*, pages 641–670, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[7] Shiyu Ji, Jinjin Shao, Daniel Agun, and Tao Yang. Privacy-aware ranking with tree ensembles on the cloud. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, page 315–324, New York, NY, USA, 2018. Association for Computing Machinery.

[8] R. A. Popa, F. H. Li, and N. Zeldovich. An ideal-security protocol for order-preserving encoding. In *2013 IEEE Symposium on Security and Privacy*, pages 463–477, 2013.

[9] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1967.

[10] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, 2001.

[11] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer, 2009.

[12] Brian McFee and Gert R. G. Lanckriet. Metric learning to rank. In *ICML*, pages 775–782, 2010.

[13] Dor Kedem, Stephen Tyree, Fei Sha, Gert R. Lanckriet, and Kilian Q Weinberger. Non-linear metric learning. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2573–2581. Curran Associates, Inc., 2012.

[14] Eric Bax, Lingjie Weng, and Xu Tian. Speculate-correct error bounds for k-nearest neighbor classifiers. *Machine Learning*, 108:2087–2111, 2019.