Learning a logistic model from aggregated data

Alexandre Gilotte
a.gilotte@criteo.com
Criteo
Paris, France

David Rohde
d.rohde@criteo.com
Criteo
Paris, France

ABSTRACT

Recent moves towards a privacy aware internet have increased the need for methods to learn from aggregated data, where both the labels and features are only observed through aggregated queries. Aggregated data can easily be made differentially private by injecting some noise, and then shared with a third party, with the guarantee that no personal information is leaked. However, it is not clear how a third party should use such data to learn a model, as classical supervised learning methods do not apply here. In this paper we explain how existing methods on Markov Random Fields may be applied to fit a model to the joint distribution of labels and features from aggregated data (exploiting the presence of sufficient statistics), and use the conditional distribution of the obtained model to predict the labels. We then show how to modify the training objective to improve the quality of the learned conditional distribution. We further show experimentally on a public online advertising dataset that our method can perform close to a logistic regression with full access to the dis-aggregated data set.

CCS CONCEPTS

• **Information systems → Computational advertising**.

KEYWORDS

aggregated data, differential privacy, Markov random field, logistic regression, ad click prediction

ACM Reference Format:
Alexandre Gilotte and David Rohde. 2021. Learning a logistic model from aggregated data. In ,. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

1.1 Learning from aggregated data

In this work, we investigate a method to learn a discriminative model predicting a binary label, from aggregated data only. Our aggregated data consists in a set of contingency tables, counting the number of examples on several overlapping projections. An example of a toy dataset can be seen in Table 1, with 5 examples consisting of 3 categorical features and a binary labels. Table 2 shows the resulting contingency tables, when we aggregate this toy dataset on each pair of features. Usual Machine Learning methods requires access to the full dataset such as Table 1. Our goal here here is to learn a model predicting the label as a function of all available features, using only aggregated data such those in Table 2.

Table 1: Toy example dataset

	Feature 1	Feature 2	Feature 3	label
Example 1	"1"	B	a	1
Example 2	"2"	A	b	1
Example 3	"1"	B	b	0
Example 4	"2"	B	a	1
Example 5	"1"	A	b	0

Table 2: Example of aggregated data

Feature 1	Feature 2	Examples count	Sum(label)
"1"	A	1	0
"1"	B	1	1
"2"	A	1	1
"2"	B	2	1
Feature 1	**Feature 3**	**Examples count**	**Sum(label)**
"1"	a	1	1
"1"	b	2	0
"2"	a	1	0
"2"	b	1	1
Feature 2	**Feature 3**	**Examples count**	**Sum(label)**
A	b	2	1
B	a	2	2
B	b	1	0

We also investigate how to learn the model when some noise from a known distribution has been added to the aggregated data. The addition of noise into the data is a well-known practice to obtain differential privacy, and our method may then be the basis of a practical solution to allow a third party to learn a differentially private model on a dataset, by simply releasing differential private aggregated data, and letting the third party use our method to learn directly from these aggregated data.

1.2 Motivation: Online advertising, Chrome privacy sandbox and Aggregation API

One specific motivation for this work comes from online advertising, and the changes proposed by Google Chrome to this industry [11].

Online advertising today heavily relies on models predicting the probability that a user clicks on a banner, given a list of features

,
© 2021 Association for Computing Machinery.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

describing the user and the banner. These models are trained using classical supervised learning methods, from examples of past displayed banners.

The market of online advertising however is now moving towards providing more privacy to users. In particular, Google Chrome is considering, in its 'Privacy Sandbox', to prevent marketers from accessing datasets of past banners which may contain private information about the users. Instead, marketers would only have access to noisy aggregated data, queried through an *Aggregation API* [4]. While the exact specifications of these reports are not yet fully defined at the time of writing of this article, they would likely consist on a set of (noisy) contingency tables similar to those shown in Table 2. It will be critical for marketers to find methods to train their click-predictions models from these aggregated reports; and the algorithm we propose in this work might be applied in this setting.[1]

Criteo AdKDD challenge. Because of the importance if this problem for the online advertising industry, we at Criteo recently (as of May 2021) launched a competition [2] whose goal is to learn a click-prediction model from aggregated data. The available dataset consists on contingency tables counting examples and clicks on a large dataset, with one table for each single feature, and one table for each pair of features, both with some additive Gaussian noise. The algorithm we propose here may be applied to this dataset.

1.3 An overview of our approach

Let X the features vector and Y the label. Our goal is to produce a model predicting $\mathbb{P}(Y = y|X = x)$, from the aggregated data only. We propose to approximate the *joined* distribution on X, Y by a parametric generative model $\mathbb{P}_\mu(X = x, Y = y)$. Such a joined model attributes a probability to the set of observed aggregated data, and we thus select the parameter μ by maximising the likelihood of these observed data. Solving this optimization problem for general classes of models may be intractable, but it happens to be possible when restricting the search to a well chosen family of Markov Random Fields, with features matching the available data. The predictions are then made by computing the conditional distribution $\mathbb{P}_\mu(Y = y|X = x) = \frac{\mathbb{P}_\mu(Y=y,X=x)}{\mathbb{P}_\mu(X=x)}$ inferred from the joined model. With the family of Markov Random Fields that we propose in section 4, the shape of the conditional distribution $\mathbb{P}_\mu(Y = y|X = x)$ coincides with a logistic model, and we will therefore compare the performances of our models trained from aggregated data with these of a logistic regression.

2 RELATED WORK

Differential privacy. Differential privacy [3] is a way to protect the privacy of individual records by ensuring that the published data are not significantly changed by the addition or removal of a single record.

More formally, we say that a randomized algorithm $\mathcal{A}$ is (ϵ, δ) - differential private, if for any pair of datasets X and X' differing by only one record, and any possible set of results $S \subset range(\mathcal{A})$,

$$\mathbb{P}(A(X) \in S) \leq \mathbb{P}(A(X') \in S) \cdot \exp(\epsilon) + \delta$$

[1] Of course, its applicability in practice could depend a lot on the yet unknown details of the aggregation API.

Ecological inference . Learning about individual behavior when only aggregated data are available has been long studied under the name *ecological inference* problem. It had given rise to a rich literature [5] of methods to get bounds and models that can use aggregated data. But it is typically applied to low dimensionality problems and it is not obvious how to generalize to problems with more features.

Markov Random Fields. Markov Random Fields (MRF, also known as undirected graphical models) are a class of probabilistic models on multidimensional data, where the joint model on a discrete multivariate X is defined as a product of *factors* involving subsets of the components of vector X:

$$\mathbb{P}_\mu(X = x) = \frac{1}{Z_\mu} \prod_C exp(f_C(x_C, \mu)) \tag{1}$$

where C are subsets of the components of X, f_c are parameterized functions of these components, and Z_μ is a normalisation constant. MRFs have been studied for their ability to model complicated distribution in a compact way. [6, 7]

Learning the parameters of an MRF is not straightforward, because exact computation of the likelihood requires evaluating $Z\mu = \sum_{x \in \mathcal{X}} \prod_C \exp(f_C(x_C, \mu))$ where the set $\mathcal{X}$ may be extremely large ($\mathcal{X}$ is usually a combinatorial object). Many approximate methods for estimating Z_μ or its derivative have been explored, such as using Markov Chains Monte Carlo methods such as Gibbs sampling or variational approximations [6–9, 14].

Collective Graphical Models. One important property of MRF is that they form an exponential family [6], making it possible to learn the MRF from aggregated data only. The special case where the observations consists of noisy aggregated data has also been defined in [13], and optimization methods refined in [10, 12]. Finally, [1, 8] built on those ideas to learn a differentially private MRF by adding Laplace noise to the sufficient statistics. While it is very similar to the method we propose, it focuses on learning accurately the joint model, while what matters for us is only the quality of the conditional distribution $\mathbb{P}(Y = y|X = x)$ on one specific component Y.

3 LEARNING FROM AGGREGATED DATA

3.1 Problem setup

Supervised learning loss. Let $(x_i, y_i)_{i \in 1...n}$ be a dataset of n feature vectors $x_i \in \mathcal{X}$ and binary labels $y_i \in \{0, 1\}$. As usual in supervised learning, we assume that examples (x_i, y_i) are independently sampled from an unknown distribution $\pi : \mathcal{X} \times \{0, 1\} \longrightarrow [0, 1]$.

We will also assume that examples x are made of D categorical features.

Our goal is the same as the supervised learning setup; we would like to find a function $f : \mathcal{X} \longrightarrow [0, 1]$ with a low expected loss

$$\mathbb{E}_{X,Y \sim \pi}\big(L(f(X), Y)\big)$$

where L is the negative log-likelihood loss

$$L(f(x), y) \equiv y \cdot \log(f(x)) + (1 - y) \cdot \log(1 - f(x))$$

Observed data. In our setup, however, the dataset $(x_i, y_i)_{i\in 1\dots n}$ is not observed directly. Instead, we observe the count of examples and sum of labels in the dataset in several contingency tables [2] such as Table 2.

Formally, let $\mathcal{K}$ a finite set of "projections" $k : \mathcal{X} \longrightarrow \{0, 1\}$, and K the size of this set.

While this formalization is a bit more general, we may think here of each function k as one row of a contingency table: an example x is either counted ($k(x) = 1$) or not ($k(x) = 0$) in this row.

We then define, for $x \in \mathcal{X}$, the binary vector:

$$\mathcal{K}(x) := (k(x))_{k\in\mathcal{K}} \in \{0, 1\}^K$$

($\mathcal{K}(x)$ thus encodes the list of rows where example x is counted.)

We may now define the aggregated data as the two vectors:

$$\mathbf{c} := \sum_i \mathcal{K}(x_i)$$

$$\mathbf{s} := \sum_i \mathcal{K}(x_i) \cdot y_i$$

Each coordinate of $\mathbf{c}$ (respectively $\mathbf{s}$) is thus the count of examples (respectively the sum of labels) on one row of a contingency table. To keep notations compact, we will also note $\mathbf{a}$ the concatenation of vectors $\mathbf{c}$ and $\mathbf{s}$.

The available dataset consists of the vector a of aggregated data. Also, the family of functions $\mathcal{K}$ is assumed to be known, ie we know how the data were aggregated. In practice, $\mathcal{K}$ is of course defined implicitly by the rows of the contingency tables. In the experiments we made, we used one contingency table for each pair of features of the dataset.

Finally, since the dataset (x_i, y_i) is a list of independent realizations of random variables X and Y, we will view the vector a as the realization of some random variables A.

4 LEARNING FROM AGGREGATED DATA BY FITTING A MODEL OF THE JOINT DISTRIBUTION

4.1 Modeling the joint distribution of X and Y

Let us emphasize again that usual supervised methods, which typically make a parametrised model of the conditional law of $P(Y|X = x)$ and select the parameters maximizing the likelihood of the data, are not applicable in this setup. Indeed a model on $P(Y|X = x)$ does not assign a probability to the observed event $A = \mathbf{a}$.

Instead, we propose to use a parametrised model of the joined distribution $P(X = x, Y = y)$. Such a model assigns a well defined probability to the event $A = \mathbf{a}$, and we will thus try to retrieve the parameters maximising the likelihood of this event. Prediction of the label Y may then be obtained from this joined model by applying Bayes' rule.

[2]Note that the information provided by each table may overlap, with each features appearing in several tables. In particular we ran our experiments in the case when there is one table for each pair of features. This overlap makes the available data more complicated to use, but also contains potentially important correlation information between the features. The formalisation we propose here is agnostic to those details, but the final performances of the model would certainly depend on the exact set of available tables.

4.2 Joined log linear model on X and Y

Let $\mu_c, \mu_s \in \mathcal{R}^K$ two vectors of parameters, and $\mu \in \mathcal{R}^{2K}$ the concatenation of μ_c and μ_s.

We define a distribution π_μ over the joined features and label space $\mathcal{X} \times \mathcal{Y}$ by:

$$\pi_\mu(x, y) \equiv \frac{1}{Z_\mu} \exp\left(\mathcal{K}(x) \cdot \mu_c + y\, \mathcal{K}(x) \cdot \mu_s\right) \tag{2}$$

Where we noted Z_μ the normalization constant:

$$Z_\mu \equiv \sum_{x',y' \in \mathcal{X}\times\mathcal{Y}} \exp(\mathcal{K}(x') \cdot \mu_c + y'\, \mathcal{K}(x') \cdot \mu_s) \tag{3}$$

Remark 1. *The sum which appears in equation 3 is usually intractable, and we could at best try to approximately estimate this normalization constant. But as we will see further, it won't be necessary in practice.*

The models we defined here belong to the family of "Markov Random Fields" (MRF).

4.3 Predicting with this model

Before we discuss how to select the parameters μ, let us quickly describe how to retrieve the conditional distribution $\pi_\mu(Y = 1|X = x)$ with this model. This is done simply by applying Bayes' rule to equation 2

$$\begin{aligned}\pi_\mu(Y = 1|X = x) &= \frac{\pi_\mu(Y = 1, X = x)}{\pi_\mu(Y = 0, X = x) + \pi_\mu(Y = 1, X = x)} \\ &= \sigma(\mathcal{K}(x) \cdot \mu_s)\end{aligned} \tag{4}$$

Where σ is the logistic function. We note here that this equation has exactly the shape of a logistic model on features $\mathcal{K}(x)$, which is why we may think of our method as *training a logistic model from aggregated data*. For this reason, it is natural to compare the performances of our model to those of a logistic regression using the same features. [3]

4.4 Maximizing the likelihood of the data

As mentioned before, the model of equation 2 assigns a well defined probability $\pi_\mu(A = \mathbf{a})$ to the observed event $A = \mathbf{a}$. We will note $\pi_\mu(A = \mathbf{a})$ this probability. [4] It is therefore natural to select the parameters μ^* maximising the (log) likelihood of this event:

$$\text{Find } \mu^* \in \text{argmax}_\mu \log \pi_\mu(A = \mathbf{a}) \tag{5}$$

While this problem might be very challenging to solve for arbitrary classes of models, in our case the gradient of this log-likelihood has a rather simple close formula.

Lemma 4.1 (Gradient of the log-likelihood).

$$\nabla_\mu \log \pi_\mu(A = \mathbf{a}) = \mathbf{a} - \mathbb{E}_\mu(A)$$

Where $\mathbb{E}_\mu(A)$ is the expectation of the aggregated vector when the n samples $(X_i, Y_i)_i$ of the dataset come from the model π_μ.

[3]Note that we certainly expect our model to perform worse than a logistic regression trained on the full dis-aggregated dataset. The whole point of our method of course is that it does not require this dis-aggregate dataset.

[4]The computation of $\pi_\mu(A = \mathbf{a})$ however requires to compute Z_μ and is thus intractable. But as we will see, we never need to compute it directly.

This lemma is the main motivation for choosing the model of equation 2. It is a direct consequence of the fact this equation defines an exponential family [6]. [5]

4.5 Model Regularization

In the case of logistic models, it is well known that a regularization of the parameters may increase the performances on the validation set when the number of parameters is large and inputs are correlated. It is especially the case when using the second (or larger) order kernel. Since our model generalizes this logistic regression, it certainly would also benefit from some kind of regularization. We used a L2 regularization in the experiments due to its simplicity, and because it makes the loss strongly convex. However, we noted that a single regularization parameter was not performing so well. Instead we penalized differently the components μ_c and μ_s of the parameters vector. We thus have two distinct parameters λ_c and λ_s :

$$penalty(\mu) \equiv \lambda_c \cdot \mu_c^2 + \lambda_s \cdot \mu_s^2 \tag{6}$$

We experimentally observed[6] that λ_s should be set to a value similar to the L2 regularisation of a "normal" logistic model, while λ_c should be kept much smaller.

4.6 Model optimization

Fitting the parameters of a MRF is a well studied problem, and we picked one method which only use the sufficient statistics **a** Let us describe it quickly: We used a Gibbs sampling method to generate samples of X, Y, and estimate the expectation of $E_\mu(A)$ by Monte Carlo from those Gibbs samples. This allow us to compute the gradient with lemma 4.1, and we thus run a gradient descent. To limit the time required by Gibbs sampling, we reuse the same samples from one iteration to another, following the "Persistent Contrastive Divergence" approach of [14]. We also reduced the variance of the Monte Carlo estimator by marginalising on Y.

For additional details, we refer to our repository where we will release the code to run our experiments.

5 NOISY AGGREGATED DATA

As we explained in the introduction, some noise may be injected to the aggregated data to obtain privacy guarantees. Typically this noise is additive and i.i.d., and follows either a Laplace or Gaussian distribution. The exact distribution of the noise is assumed to be known. Advanced methods to fit a MRF from noisy sufficient statistics have been proposed, in particular [1, 10]. It is unclear however how easily they would scale to the models with the millions of parameters found in computational advertising. The method we use in our experiments and describe in this section is easy to implement and does not increase significantly the cost of training the model. We leave scaling and testing alternative methods to future work.

[5] See the supplementary materials for an elementary proof.
[6] Experimental results are detailed in the supplementary materials.

5.1 Problem definition

In this section, the aggregated data A are no longer observed. Instead, we observe a realization **b** of the noisy counts:

$$B \equiv A + L$$

where $L \in \mathbb{R}^{2K}$ is a random vector of iid samples of some noise distribution. We assume this distribution to be known. We note $f_L(L = l)$ the density of this noise, and $f_\mu(A+L = b)$ the probability density of the event $A + L = b$ when examples are sampled from π_μ.

Applying the maximum likelihood principle, we seek to find μ^* maximising density of the observed noisy data.

$$\mu^* = \text{argmin}_\mu - \log f_\mu(A + L = \mathbf{b}) + \text{penalty}(\mu) \tag{7}$$

5.2 Gradient of the noisy aggregated data likelihood

The following lemma [7] tells us how the gradient of the log-likelihood is modified by the noise:

Lemma 5.1.

$$\nabla_\mu f_\mu(A + L = \mathbf{b}) = \mathbf{a} - \mathbb{E}_\mu(A) - \mathbb{E}_\mu(L|L + A = \mathbf{b})$$

Compared to the noiseless case (lemma 4.1), this gradient now has an additional term $\mathbb{E}_\mu(L|L + A = b)$. The next section describes how this additional term may be approximated, and we can thus amend the gradient descent of section 4.6 by simply adding this term to the gradient computation.

5.3 Approximating the expected noise

We thus need to compute $\mathbb{E}_\mu(L|A + L = \mathbf{b})$.

Independence approximation. Let j denote the index of one single component on the aggregated data. As exact estimation is intractable, we propose to estimate each component independently:

$$\mathbb{E}_\mu(L_j|L + A = \mathbf{b}) \approx \mathbb{E}_\mu(L_j|L_j + A_j = b_j)$$

Intuitively, this approximation is quite natural: one component l_j of the noise depends mostly on the associated noisy data b_j. The dependency to the other components comes only from the complicated correlations between the components of A, which the proposed approximation simply ignores.

Computing $\mathbb{E}_\mu(L_j|L_j + A_j = b_j)$ is a one-dimensional problem.

Noting that b_j is known, A_j follows a binomial of known parameters[8], and L_j follows a know distribution, this one dimensional problem may be solved for example by numeric integration.

In our experiments we compared the method we proposed here with the baseline of applying directly the algorithm of section 4.6 to the noisy data, without the additional term of lemma 5.1.

6 EXPERIMENTS

6.1 Comparing to logistic regression with a quadratic kernel

Our main point of comparison will be to logistic regression trained on the full dis-aggregated dataset . There are several reasons for this choice:

[7] The proof is included in the supplementary materials: https://github.com/criteo-research/ad_click_prediction_from_aggregated_data
[8] More precisely its expectation was already estimated, by Gibbs sampling, in section 4.

- While not state of the art, logistic regression models offer a good trade-off between simplicity and accuracy, especially when including a quadratic kernel.
- The marginal distribution $\mathbb{P}(Y|X)$ of our models has exactly the shape of a logistic model (equation 4, making it a natural comparison.

Our main comparison is thus with a logistic regression defined by:[9]

$$\text{logistic}(\theta, x) \equiv \sigma\Big(\mathcal{K}(x) \cdot \theta\Big) \tag{8}$$

Remember here that $\mathcal{K}(x)$ is a vector encoding on which rows of the contingency tables an example x is counted. We have in our experiments one contingency table for each pair of features, and in this case $\mathcal{K}(x)$ is exactly the composition of a one-hot encoding of the features with a quadratic kernel. We also included as a simpler baseline a logistic regression without the quadratic kernel.

6.2 Experiments on a public Criteo dataset

Dataset. We experimented [10] with this algorithm by training a model predicting clicks on a public dataset [11].

This dataset contains 16M examples, with 36% of clicks, and each example has 11 categorical features. Most of these features have between 10 and 100 modalities, but one has about 10k distinct modalities.

We also used smaller versions of this dataset :

- In *Dataset-small* we sampled 1% of the data, and kept only 5 features with smallest modalities count.
- In *Dataset-sampled* we also sampled 1% of the examples, but kept all 11 features.
- Finally in *Dataset-large* we kept all examples and features.

On each dataset, we used 30% samples for validation, and trained on the remaining 70%.

Compared models.

- *LR2* is the logistic regression with second order kernel described in equation 8.
- *LR* is a "vanilla" logistic regression without second order kernel.
- *MRF* is the joined model trained from aggregated data described in this paper.
- *NB* is a Naive Bayes classifier.

Reported metrics. We reported the *normalized log-likelihood* of $P(Y|X)$, on both train and validation sets, defined as:

$$NLLH \equiv \frac{LogLikelihood(prediction(X), Y)}{Entropy(Y)} - 1$$

.

[9]The shape of the prediction function is exactly the same as our model's, the only difference thus the way the parameters θ are fitted: this logistic regression is trained on the whole dis-aggregated dataset.
[10]Python code to re-run our experiments will be me made available here: https://github.com/criteo-research/ad_click_prediction_from_aggregated_data .
[11]https://ailab.criteo.com/criteo-attribution-modeling-bidding-dataset/

Table 3: Results on Dataset-sampled

Model	NLLH train set	NLLH validation	best L2 parameter	run time
LR2	0.1113	0.0748	80.0	2400s
LR	0.0876	0.0711	8.0	7s
MRF	0.0960	0.0734	160.0	13000s
NB	0.0431	0.0396	256	

Table 4: Results on Dataset-large

Model	NLLH train set	NLLH validation	best L2 parameter	run time
LR2	0.0991	0.0907	50.0	2h
LR	0.0769	0.0757	8.0	0.2h
MRF (50k samples)	0.0712	0.0681	400.0	6h
MRF (400k samples)	0.0899	0.0869	400.0	120h

Results summary. Results are shown on Tables 5 and 4.[12] On each dataset, our proposed model was able to outperform the logistic regression with single features. It was also able to get results very close to these of the logistic regression with second order kernel on the datasets *Small* and *Sampled*, and still reasonably close on the *dataset-Large*. However this was at the expense of a much longer training time.

For details on how the choice of meta-parameters and results on Dataset-small, we refer to the supplementary materials.

6.3 Training with noisy aggregated data

To test how the model performs on differentially private data, we added noise to the aggregated data using either the Laplace mechanism to achieve ϵ-differential privacy, or the Gaussian mechanism to achieve ϵ, δ-differential privacy.

Experimental results with the noise modelization of section 5.3. We tested the performances of our model trained on noisy aggregated data for different values of ϵ and δ:

- *MRF* is the model of section 4.6 with no change to the training procedure (i.e. approximating the noise by 0).
- *MRF-N* uses the explicit noisy likelihood described in section 5.

We also tried different values[13] of the regularization parameter λ_s, and reported only the best run on validation data.

Results on noisy data. Results on *Dataset-Sampled* are reported in Table 5

Unsurprisingly, results are worse for smaller ϵ. [14]

We also found that modeling explicitly the noise helps significantly in the presence of a lot of noise, while for smaller amounts of noise it seems sufficient to increase the regularization.

[12]Confidence intervals were not included due to the cost of systematically rerunning the experiments.
[13]Here we benched values on a log(4) scale, i.e. 1,4,16,...
[14]We remind that a smaller ϵ means a higher privacy protection.

Table 5: Results on noisy Dataset-sampled

ϵ	δ	noise σ	model	LLH train	LLH validation	best L2
1.0	0	93.3	MRF	0.0230	0.0232	2560
			MRF-N	0.0490	0.0474	160
1.0	10^{-7}	44.8	MRF	0.0451	0.0430	2560
			MRF-N	0.0617	0.0599	160
1.0	10^{-4}	33.1	MRF	0.0436	0.0414	2560
			MRF-N	0.0661	0.0631	160
10.0	0	9.3	MRF	0.0732	0.0664	2560
			MRF-N	0.0788	0.0691	160
10.0	10^{-7}	5.0	MRF	0.0911	0.0701	160
			MRF-N	0.0832	0.0718	160

Table 6: Preliminary results on Criteo-AdKDD challenge.

Model	Normalized LLH
Logistic, full dataset	0.289
Logistic, 2 order kernel, full dataset	0.311
Logistic, small training set	0.236
Logistic, 2 order kernel, small training	0.238
MRF, 100k Gibbs samples	0.253
MRF, 1M Gibbs samples	0.262

On *Dataset-Sampled*, using the Gaussian mechanism (with $\delta \neq 0$) helps to retrieve better models than when using the Laplace mechanism (with $\delta = 0$), while it is not the case on Dataset-small. This was to be expected: we note here that the variance of the noise is $O(D^2)$ with the Gaussian mechanism, while it is $O(D^4)$ [15] with the Laplace mechanism. For this reason, we believe the Gaussian mechanism should be preferred whenever possible for large scale applications.

6.4 Preliminary results on Criteo AdKDD challenge

Table 6 shows some preliminary results of applying our method on the dataset from the Criteo AdKDD challenge, and compares it to logistic models trained either on the small train, or on the full dis-aggregated dataset (which is not available to competitors, but will be released after the competition). [16]

7 CONCLUSION AND COMMENTS

Aggregate noisy queries of user level data allow user privacy to be maintained yet provide useful if degraded information to marketers for performance advertising purposes. Traditional supervised learning algorithms do not apply in this setting and new approaches are needed. This paper presents some partial solutions to the modelling and computational challenges posed by privacy-aware machine learning in this setting. We show that on a medium-size dataset, it is possible to produce a discriminative model based on aggregated data which approaches the performances as a model trained on the whole dataset. Several points still need addressing and will require further work before this method may be really viable in production in the online advertising industry.

Selecting parameters from aggregated *data.* Learning from aggregate data has some intrinsic limitations that seem difficult to fully overcome. Predictive quality of the model is given by the 'discriminative likelihood' $P(Y|X, \mu)$ which simply cannot be evaluated in an aggregate setting - the lack of access to $P(Y|X, \mu)$ makes hyperparameter settings such as the amount of regularisation to apply difficult in the aggregate setting. In an industrial context, it would also make very difficult to further improve the models and to have an offline control of their quality.

Scaling to larger datasets. The method we developed performed well on the medium size dataset we experimented with, but we do not know yet if it can also be competitive on larger datasets such the one included in the adkdd challenge or on a production dataset.

Finally, there are intrinsic limitations in the aggregate setting that likely cannot be overcome by any algorithm. If aggregation is viable for the advertising industry remains an open question.

[15] Because the number of queries on pairs of features is $O(D^2)$.

[16] Note that to keep the challenge more accessible, two dis-aggregated datasets were also included: a small dis-aggregated labeled training set, and a medium sized dis-aggregated but un-labeled test set. Competitors are of course allowed to use those datasets to fit their models, and we thus expect that the winner of the challenge would be using an hybrid method leveraging all datasets. But we are eager to see how our model compare to other solutions using only the aggregated data.

REFERENCES

[1] Garrett Bernstein, Ryan McKenna, Tao Sun, Daniel Sheldon, Michael Hay, and Gerome Miklau. 2017. Differentially private learning of undirected graphical models using collective graphical models. In *International Conference on Machine Learning*. PMLR, 478–487.

[2] Eustache Diemert. 2021. Criteo AdKDD Competition. https://github.com/criteo-research/criteo-privacy-preserving-ml-competition

[3] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*. Springer, 265–284.

[4] Charlie Harrison. 2021. Measurement API. https://github.com/WICG/conversion-measurement-api

[5] Gary King, Martin A Tanner, and Ori Rosen. 2004. *Ecological inference: New methodological strategies*. Cambridge University Press.

[6] Daphne Koller and Nir Friedman. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

[7] Volodymyr Kuleshov and Stefano Ermon. [n.d.]. *Notes on probabilistic graphical models*. https://ermongroup.github.io/cs228-notes/

[8] Ryan McKenna, Daniel Sheldon, and Gerome Miklau. 2019. Graphical-model based estimation and inference for differential privacy. In *International Conference on Machine Learning*. PMLR, 4435–4444.

[9] Iain Murray and Zoubin Ghahramani. 2012. Bayesian Learning in Undirected Graphical Models: Approximate MCMC algorithms. arXiv:1207.4134 [cs.LG]

[10] Thien Nguyen, Akshat Kumar, Hoong Chuin Lau, and Daniel Sheldon. 2016. Approximate inference using DC programming for collective graphical models. In *Artificial Intelligence and Statistics*. PMLR, 685–693.

[11] privacy-sandbox [n.d.]. *Privacy-Sandbox*. https://www.chromium.org/Home/chromium-privacy/privacy-sandbox

[12] Daniel Sheldon, Tao Sun, Akshat Kumar, and Tom Dietterich. 2013. Approximate inference in collective graphical models. In *International Conference on Machine Learning*. PMLR, 1004–1012.

[13] Daniel R Sheldon and Thomas Dietterich. 2011. Collective Graphical Models. In *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger (Eds.), Vol. 24. Curran Associates, Inc., 1161–1169.

[14] Tijmen Tieleman. 2008. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*. 1064–1071.