

# Stochastic gradient descent with gradient estimator for categorical features

Paul Peseux<sup>1,2</sup>, Maxime Berar<sup>1</sup>, Thierry Paquet<sup>1</sup>, Victor Nicollet<sup>2</sup>

<sup>1</sup>Litis, Rouen FRANCE - <sup>2</sup>Lokad, Paris FRANCE

---

## Abstract

Categorical data are present in key areas such as health or supply chain, and this data require specific treatment. In order to apply recent machine learning models on such data, encoding is needed. In order to build interpretable models, one-hot encoding is still a very good solution, but such encoding creates sparse data. Gradient estimators are not suited for sparse data: the gradient is mainly considered as zero while it simply does not always exist, thus a novel gradient estimator is introduced. We show what this estimator minimizes in theory and show its efficiency on different datasets with multiple model architectures. This new estimator performs better than common estimators under similar settings. A real world retail dataset is also released after anonymization. Overall, the aim of this paper is to thoroughly consider categorical data and adapt models and optimizers to these key features.

*Keywords:* categorical data, gradient descent, gradient estimation, dataset release

---

## 1. Introduction

Tabular data represents a considerable amount of modern data especially in health and in the industry. Machine Learning has been applied to those tabular data for decades for different tasks such as regression or classification. Boosting methods Chen and Guestrin (2016) Ostroumova et al. (2018) are widely spread on these data and are still state of the art. After outstanding results on image, speech recognition, text . . . , some attempts to apply deep learning models on tabular data have been published recently but with a limited impact on the state of the art as presented by Borisov et al. (2021) or Gorishniy et al. (2021). Some deep learning approach have been agnostic on the specificity of tabular data and have tried to import their successful recipes from other application domains while other tried to adapt their architecture. Nevertheless no deep learning attempt did not succeed to clearly overtake classical machine learning models such as gradient-boosted tree ensembles Borisov et al. (2021).

There are different possible explanations for that. First deep learning needs a lot of training data to perform well and tabular data are often small. Second deep learning performs well on homogeneous data where embeddings can be learnt Rehman (2021) by exploiting the underlying nature of the data (sequence of words, or 2D spatial pixel neighborhood). In contrast, there is no such underlying structure on tabular data, and they show many local irregularities. As tabular data often contains categorical data, an encoding is needed and this encoding does not interact well with modern stochastic gradient-based parameter updates. Encoding increases dimension and scatters data but with modern stochastic methods, this data scattering creates empty batches where the last step of back-propagation continuously updates parameters agnostically of the data structured concerned by the batch Kingma and Ba (2014). This observation applies also on non deep models whose training rely on stochastic gradient descent.

Our main contributions are the following. First we propose a modification of training loss on categorical data with an unbiased estimator of this proposed loss. Second we compared performance on state deep learning models on categorical data with this new gradient estimator and shows that it outperforms regular estimators on different datasets. Third we applied this technique on in-production model on a private dataset that we open source.

The rest of the article is organized as follow. After an overview of the recent works on tabular data we point out the issue of applying modern stochastic gradient descent on categorical data. Then we propose a novel gradient estimator and show that he is unbiased for a relevant loss on categorical data. We perform many experiments on public datasets that show how it outperforms classical gradient estimator. We also release a large private dataset of the french retailer Celio.

## 2. Related works

As presented in Borisov et al. (2021) or Gorishniy et al. (2021), tabular data, as opposed to images, are heterogeneous, i.e. they present high variability of data types and formats, in their underlying structure. Then often contain categorical input features and are strongly structured. This structure is thus specific to each tabular dataset and a modification of an input categorical feature might completely change the meaning of the concerned data while a pixel modification does not fundamentally change an image. For many tasks as binary or multiclass classification and regression, deep learning did not surpass tree models on tabular data yet as presented in Borisov et al. (2021) and is presented as its last “unconquered castle” by Kadra et al. (2021). Various architectures such as MLP, ResNet, Transformer, NET-DNF... have been

applied to tabular data. We can split them into two categories: the raw deep learning Models and the adapted deep learning Models. The first rely on some known deep learning architectures that are directly applied on tabular data, without any modification of their architecture. The second one adapts deep learning architectures in order to better fit the tabular data specificity Arik and Pfister (2021) Song et al. (2019) Popov et al. (2019). These attempts did not outperform the standards models such as XGBoost from Chen and Guestrin (2016) or CatBoost from Ostroumova et al. (2018) which are still the state-of-the-art in this domain, as presented in Figure 1 from Borisov et al. (2021).

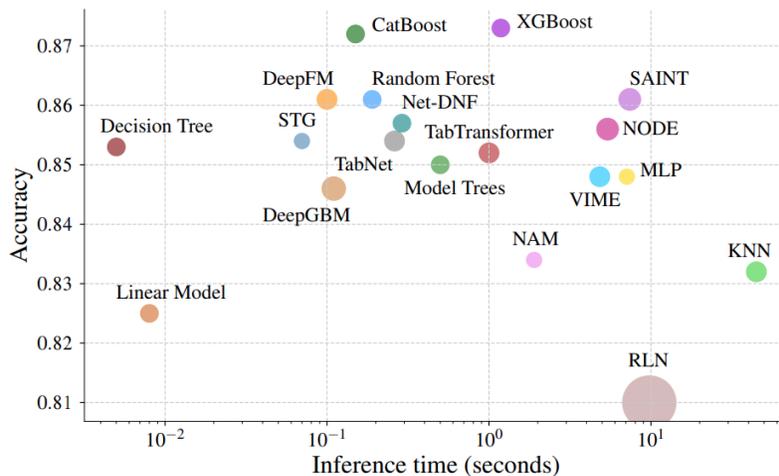


Figure 1: Directly taken from Fig.3 of Borisov et al. (2021), "inference time benchmarks for selected methods on the Adult data set with 32.561 samples. The circle size reflects the accuracy standard deviation".

Contrary to these tree-based approaches, one of the common point of all deep learning approaches applied on tabular data is to work with a training loss that assumes that every input category is concerned by every observation.

### 3. Training with categorical data

Let us denote "categorical models" the set of models that accept categorical features by design and are numerical, i.e. their parameters can be updated through gradient descent. By categorical features, we denote a feature  $s$  belonging to an alphabet of  $n_s$  symbols  $\{s_1, \dots, s_{n_s}\}$ , whose cardinality depends on the application. In Table 1, color and stores are categorical features, and pink and blue are symbols, and forms the color alphabet. Wide models described in Cheng et al. (2016) correspond to

Table 1: categorical data

Color	Store	Sales
blue	Paris	14
pink	Rome	12
pink	Rome	13
...	...	...
blue	Berlin	17
pink	Paris	8

this definition. However, decision trees do not as they cannot be considered numerical models, but rather ensemble models. Regular neural networks are not categorical models either because they cannot use raw categorical input data and require an additional encoding step of the categorical data. We stress the fact that the *categorical* aspect applies to the input features: a categorical model can be used for regression, multi-class classification ...

A straightforward categorical model for inputs taken from Table 1 could be defined by Equation 1 and is represented through the graphical representation depicted in Figure 2:

$$\hat{y} = \mu_{color} \times \gamma_{store} \tag{1}$$

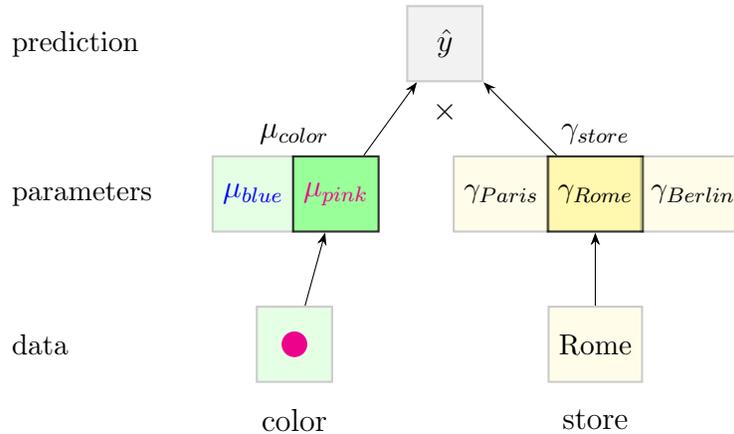


Figure 2: categorical model accessing parameters

with  $\hat{y}$  the estimated sales. This toy model will be used as an example throughout

the paper. In this simple Model 1, the parameter  $\mu_{color}$  has thus a value for each color, and we aim to find the best ones in order to have a good predictive model. One of the main techniques to do that is gradient descent. Partly due to the very large amount of data often encountered in practice, *stochastic* gradient descent is used. However, applying *stochastic* gradient descent on categorical models raises an issue as common gradient update techniques are not designed for categorical features: not every symbol of a categorical feature is present in every observation of a dataset while regular numerical models assume that every feature is present on every observation. We propose an updated version of gradient estimation used to update parameters. Its specificity is to take into account the categorical features. This paper links work on sparse gradient estimator and one-hot-encoding categorical data that leads to a sparse but structured gradient.

No universally good method of encoding exists and choice should always rely on data (alphabet cardinality, relationships between them ...). In the following we will focus on one-hot-encoding because it is precisely what is done in a categorical model. Model 1 one-hot-encoding is depicted in Equation 7 and Figure 2.

One-hot-encoding a categorical variable with cardinality  $n$  is performed by creating  $n$  binary vectors for each occurrence of the symbol. If there are few symbols, there are only a few newly created columns. On data stored in Table 1, one-hot-encoding the feature *Color* creates the features  $is_{blue}$  and  $is_{pink}$ . Thus one-hot-encoding is adapted to low cardinality features. Otherwise one might face the curse of dimensionality as described in Chen (2014). Moreover, symbols present in the testing dataset but unseen in the training dataset are incompatible with one-hot-encoding as it makes the assumption that all symbols are present in the training dataset; the data is pre-processed according to them. Text encoding as presented in Savinov et al. (2021) or Ha and Gao (2021) represents sentences in a latent space, as two different ones might have a very similar meaning and thus a very close representation in the latent space. This does not apply on categorical data where each symbol has a very specific meaning. When those adequate conditions are not met, other encodings (such as leave-one-out) should be preferred. Interpretability of the model is fundamental as explained by Rudin (2019) especially in high-stake contexts such as disease diagnostics, where explainability of the result is expected for the human expert to take his final decision. In this direction, using one-hot-encoding is crucial. Having parameters directly related to the application semantic by giving access to their relation with the input symbols is a requirement for the design of white-box models. In Model 1,  $\mu_{blue}$  ( $\mu_{pink}$  respectively) has a valuable meaning: this is the impact of the blue (pink respectively) color on the sales. Parameters value not only serve model prediction quality, they are also *interpretable*. On Model 1,  $\mu_{blue} > \mu_{pink}$  means that the blue color sells better than

the pink one. Not only is the prediction of the model explainable, but the trained model itself conveys meaning, even without inputs.

Leave-one-out encoding turns the categorical feature into one numerical feature. This has several advantages (no curse of dimensionality for example) but gives no directly interpretable parameters. One-hot-encoding leads to sparse but structured data by construction. Gradient descent on sparse data is an extensively studied subject: Ma and Zheng (2016) Chickering and Heckerman (2013); Dekel (2016); Zhao et al. . . . The following also applies to sparse but structured data. As one-hot-encoding is not suited for categorical “sparse” features with high-cardinality, we exclude such feature from the following, which is not restrictive for domains such as health or supply chain.

### 3.1. Notation

We consider the supervised learning set up with a given set of training labeled data  $\mathcal{Z} = \{z_i = (X_i; y_i); i = 1 \dots n\}$ , with the feature vectors  $X_i \in \mathbb{R}^p$  and the scalar targets  $y_i \in \mathbb{R}$ . Each one of the  $p$  components of the feature vectors is related to one of the  $p$  symbols  $\{s_k\}_{k \leq p}$ :

$$\forall (X_i, \cdot) \in \mathcal{Z} \quad \forall k \leq p \quad X_i \text{ corresponds to } s_k \Leftrightarrow X_i^k \neq 0 \quad (2)$$

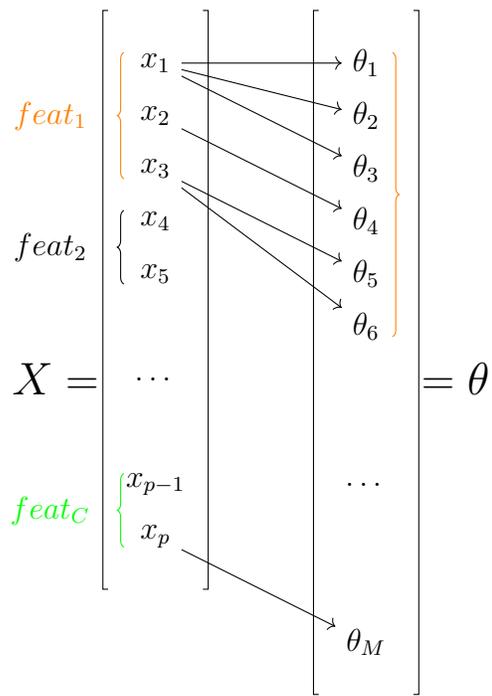
We aim to find the best parameter  $\theta^* \in \mathbb{R}^m$  ( $m \geq p$ ) to minimize the loss  $F_{\theta^*}$  on the whole dataset:

$$\begin{aligned} f : \Theta \times \mathcal{Z} &\longrightarrow \mathbb{R} \\ \theta, (X, y) &\longrightarrow f_{\theta}(X, y) \end{aligned}$$

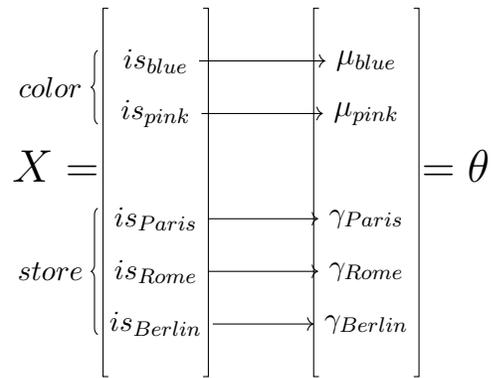
$$\begin{aligned} \theta^* &= \underset{\theta}{\operatorname{arg\,min}} \quad F_{\theta} \\ &= \underset{\theta}{\operatorname{arg\,min}} \sum_{X, y \in \mathcal{Z}} f_{\theta}(X, y) \\ &= \underset{\theta}{\operatorname{arg\,min}} \sum_{i=1 \dots n} f_{\theta}(X_i, y_i) \end{aligned}$$

This notation is generic and includes sparse data in general. One-hot encoded data is a specific case of sparse data.

Figure 3 illustrates the formal definition. In Figure 3a,  $\{\theta_1; \theta_2; \theta_3; \theta_4; \theta_5; \theta_6; \}$  are related to the first feature *feat*<sub>1</sub>. On Model 1, such notations give Figure 3b.



(a) Generic notations



(b) Notations applied to Model 1

Figure 3: Notations

### 3.2. Gradient descent with categorical features

Classical stochastic gradient descent relies on an unbiased gradient’s estimator. Rather than using the complete gradient on all observations, observations are divided into *batches* and the gradient is estimated on those:

$$\psi F = \frac{1}{n} \sum_{obs} \psi f_{obs} = \frac{1}{n} \sum_{i \leq n} \psi_{\theta} f(X_i, y_i) \quad (3)$$

$$= \frac{1}{n} \sum_{batch} \sum_{obs \in batch} \psi f_{obs} \quad (4)$$

In particular, this makes sense on large datasets where computing the exact full (i.e. on all observations) gradient might be very expensive. The accumulated gradient is then divided by the size of the batch to get the gradient estimation. Gradient descent techniques succeed in finding a minimum, notably because this the gradient estimator is unbiased, as proven by Defossez et al. (2020). Regarding categorical models, we can highlight the fact that not every categorical parameter is affected by every observation. For example in Model 1  $\mu_{blue}$  is only used on observations that concern a blue product. By construction via one-hot-encoding each observation concerns one and only one symbol for each categorical feature. Applying gradient descent in a batch mode to estimate the gradient gives a biased estimator. This issue is exacerbated on a batch that does not include a symbol. In this instance, the gradient does not even make sense.

$$\psi_{\mu_{blue}} F = \frac{1}{n} \sum_{obs} \psi_{\mu_{blue}} f_{obs} \quad (5)$$

$$= \frac{1}{n} \sum_{batch} \sum_{\substack{obs \in batch \\ color(obs)=blue}} \psi_{\mu_{blue}} f_{obs} \quad (6)$$

What would be the parameter’s gradient of a symbol that is not present at all in the dataset? What would be the gradient of  $\mu_{red}$  in Model 1 with no red products?  $\{obs \in batch | color(obs) = blue\}$  from Equation 6 might be empty, and in this case, the parameters related to the *blue* symbol are not present. And a non-present gradient is not a zero-gradient. Thanks to one-hot-encoding, we have prior info on the data and the gradient. With an observation that does not concern the symbol  $s_k$ , we know for a fact that the gradient of its related parameters will be numerically zero. Thus this numerically zero gradient does not provide any information, it should not be used for parameters updates.

$$\begin{aligned} \mu_{color} &= \mu_{blue} \times iS_{blue} + \mu_{pink} \times iS_{pink} \\ \frac{\partial \mu_{color}}{\partial \mu_{blue}} \Big|_{color=pink} &= 0 \\ \frac{\partial \mu_{color}}{\partial \mu_{pink}} \Big|_{color=blue} &= 0 \end{aligned} \tag{7}$$

This issue especially concerns under-represented symbols and small batches. The smaller the symbol cardinality and the smaller the batch, the higher the probability that it's not included in the batch. When a symbol is not present in the batch, we state that its related parameters should not be modified. To support this idea, an atomic stochastic gradient descent (i.e. with batch size of 1) example is presented in Section 3.3. Many descent techniques (not all, see Duncan et al. (2014)) such as Oktay et al. (2020), Tucker et al. (2017) or Kool et al. (2021) that rely on an estimator of the gradient use an unbiased estimator of it. Defossez et al. (2020) proves that it is a sufficient condition for convergence in the convex setting. Here the gradients of the categorical parameters are somehow biased. One-hot-encoding should not be part of the model (see Equation 7). It is a way to deal with categorical data as positional encoding is not integrated to Transformers Vaswani et al. (2017). As it is not part of the model itself, it should not infer on model's parameters updates. Encoding categorical data is not part of the gradient-exposed part of the model.

### 3.3. Stochastic Gradient Descent

Let's consider stochastic gradient descent, i.e. the case of batches of size 1. It means that observations are taken one by one. In Model 1 each observation concerns one and only one *color*, and one and only one *store*. In this case it is logical to only update the parameters of the concerned symbol. The gradients of the parameters of the unconcerned symbols do not exist. And a non-present gradient is not a zero-gradient. This logical behaviour for atomic stochastic gradient descent should be generalized to bigger batches.

## 4. Solution: gradient estimator for categorical features

With all we've discussed, the solution is straight forward and very easy to implement.

If  $\{symbol(obs) = s_k / obs \in batch\}$  is empty, parameters related to the  $s_k$  symbol should **not** be updated. Indeed a non-present gradient is not a zero-gradient. The proposed gradient estimator thus makes the difference between a zero gradient and a

non-present gradient. Then one needs to count each symbol uses and to apply the unbiased gradient. Counting symbol uses allow us to compute unbiased gradient. Then:

$$\begin{aligned}\psi_{\theta_m} F &= \frac{\partial F}{\partial \theta_m} \\ &= \sum_{batch} \sum_{obs \in batch} \psi_{\theta_m} f(obs) \\ &= \sum_{batch} \sum_{\substack{obs \in batch \\ obs \in s_k}} \psi_{\theta_m} f(obs)\end{aligned}$$

with  $S_k = \{obs \in batch / symbol(obs) = s_k\}$

So one needs to divide the accumulated gradient by the cardinality of  $S_k = \{obs \in batch / symbol(obs) = s_k\}$ . If this set is empty, parameters related to the  $s_k$  symbol should not be updated. Again a non-present gradient is not a zero-gradient. This is presented in Algorithm 1.

With the current notations:

$$F_\theta = \frac{1}{p} \sum_{k=1}^p \sum_{X, y \in S_k} \frac{1}{\#S_k} f_\theta(X, y) \quad (8)$$

With such loss objective function, drawing random observations in  $\mathcal{Z}$  does not give an unbiased gradient estimator.

$$F_{\binom{\mathcal{Z}}{m}\theta} = \frac{1}{p} \sum_{k=1}^p \sum_{X, y \in S_k \cap \binom{\mathcal{Z}}{m}} \frac{1}{\#[S_k \cap \binom{\mathcal{Z}}{m}]} f_\theta(X, y)$$

$F_{\binom{\mathcal{Z}}{m}\theta}$  is a random estimator of  $F_\theta$  where  $m$  observations (over the  $\mathcal{Z}$ ) are uniformly drawn. It is the gradient estimator for categorical features (GCE) used by Algorithm 1. This estimator is unbiased, proof can be found in appendices Appendix B.1:

$$\mathbb{E}[\psi F_{\binom{\mathcal{Z}}{m}}] = \psi F_\theta$$

This is a sufficient condition for convergence in the convex setting Defossez et al. (2020). The loss function depicted in Equation 8 seems similar to loss used for classification with unbalanced *output* categories. Let's recall that what we propose here is different as we consider unbalanced *input* symbols. In the case where symbol groups have the same size  $C$  then the objective function resumes to  $F_{\theta, C}$ :

---

**Algorithm 1** gradient estimator for categorical features

---

**Require:**  $\mathcal{Z}$ : data

**Require:**  $update(\cdot, \cdot)$ : chosen optimizer

**Require:**  $\theta_0$ : Initial parameter vector

```
 $t \leftarrow 0$ 
while  $\theta_t$  not converged do  $t \leftarrow t + 1$ 
  Divide  $Z$  in Batches
  for batch  $\in$  Batches do
5:   for symbol  $\in$  Alphabet do
      $c_{symbol} \leftarrow 0$ 
   end for
    $\mathbf{g} \leftarrow \vec{0}$ 
   for  $X, y \in$  batch do
10:     $c_{symbol(X)} \leftarrow c_{symbol(X)} + 1$ 
     Compute  $\psi_{\theta_{t-1}} f_{\theta_{t-1}}(X)$  thanks to  $y$ 
      $\mathbf{g} \leftarrow \mathbf{g} + \psi_{\theta_{t-1}} f_{\theta_{t-1}}(X)$  ▷ accumulate gradient
   end for
    $\theta_t \leftarrow \theta_{t-1}$ 
15:   for  $symbol \in$  Alphabet do
     if  $c_{symbol} > 0$  then ▷ a non-present gradient is not a zero-gradient
        $\theta_{t,symbol} \leftarrow update(\theta_{t-1,symbol}, \frac{1}{c_{symbol}} \mathbf{g}_{symbol})$  ▷ scaled gradient
     end if
   end for
20: end for
end while
```

---

$$\begin{aligned}
F_{\theta,C} &= \frac{1}{p} \sum_{k=1}^p \sum_{X,y \in s_k} \frac{1}{\#s_k} f_{\theta}(X, y) \\
&= \frac{1}{p} \sum_{k=1}^p \frac{1}{C} \sum_{X,y \in s_k} f_{\theta}(X, y) \\
&= \frac{1}{p \times C} \sum_{X,y \in Z} f_{\theta}(X, y) \\
&= \frac{1}{Z} \sum_{X,y \in Z} f_{\theta}(X, y)
\end{aligned}$$

as the  $p$  symbol groups form a partition of  $Z$ .

In this case, our proposed gradient estimator is proportional to the classic one. If one uses the vanilla optimizer, GCE is equivalent to the classic one with a bigger learning rate:

$$\theta_t = \theta_{t-1} - \alpha g_t$$

But with other optimizers such as Adam or AdaGrad which are very sensitive to the learning rate, if too high, one might not converge. So even in the balanced case, it is better to have a small learning rate and a big gradient thanks to GCE rather than a big learning rate and a small gradient. Results are presented in Section 5.

## 5. Experimental and Results

We have implemented Algorithm 1 in two different cases and with two different languages: deep learning models and categorical models using encoded categorical data. In both settings, we aim to measure the impact of GCE. As a consequence, the chosen metric will be the efficiency difference with the current treatment of categorical parameters in batch gradient descent. The public datasets presented in Table 2 will be used for evaluation as well as a private dataset from the supply chain domain. The used metric is the mean square error (MSE) for regression and error rate (i.e.  $1 - Accuracy$ ) for classification tasks.

### 5.1. Deep Learning

For the deep learning models, we applied our solution on PyTorch Paszke et al.. On this framework, it is very easy to update every parameter gradient according to

Table 2: Datasets characteristics

<b>Dataset</b>	Chicago	ACI	compas	DGK	Forest Cover	KDD99	UsedCars
instances	194m	48k	7.2k	72k	15k	494k	38k
max cardinality	7.9k	42	341	1k	40	66	1.1k

Algorithm 1. The code is available and experiments can be found here<sup>1</sup>. In order to evaluate the impact of our proposal, we worked on six different categorical datasets:

The Adult Census Income (ACI) dataset presented in Kohavi (1997) that aims to predict wealth category of individuals. The Compas dataset contains information on criminal defendant’s likelihood of re-offending. The Forest Cover dataset presented Blackard and Dean (1999) contains categorical characteristics on  $30m^2$  forest cells. The objective is to predict the forest cover type. The KDD99 dataset accessible by Archive (1999) aims to predict cyber-attacks. The Don’t Get Kicked (DGK) dataset introduced by Kaggle (2012). The objective is to predict if the car purchased at the Auction is a good or a bad buy. Finally the Used Cars datasets from Belarus are presented in 5.2.

In order to only measure the impact of GCE, we *only* use those categorical variables in our experiments. Those datasets tasks are quite easy. As a consequence we use small networks to highlight our approach. The MLP network is made up of 3 dense layers of sizes [4, 8, 4]. We also perform experiments on a ResNet-like network very similar to Gorishniy et al. (2021). We have tested three different optimizers with their default settings: SGD (vanilla), AdaGrad and Adam. Tests have been run on several batch sizes:  $2^5 \dots 10$ . To record results, each experiment has been run 10 times. Results are reproducible in the repository and are recorded in Tables B.4 B.5 B.6 B.7 B.8 B.9 B.10 B.11 B.12 B.13 B.14 B.15 in the appendices. On the different datasets we have worked on, we see an improvement of the loss on the testing dataset using GCE. A specific focus on the ACI dataset is done in Figure 4 as this dataset is also used in many other works Borisov et al. (2021). The bigger the batch, the less GCE outperforms classical estimator. This is logical as in big batches, more symbols are concerned. This proves the need to specifically handle stochastic gradient on categorical data. Results in different settings demonstrate the advantage to use GCE whatever the optimizer. Among other things, AdaGrad tries to handle gradient on

<sup>1</sup><https://github.com/ppmdatix/GCE>

sparse data (which includes one-hot encoded data) but we see a clear improvement on that task. Note that we use small architecture and only the categorical features of the dataset in order to isolate the impact of GCE.

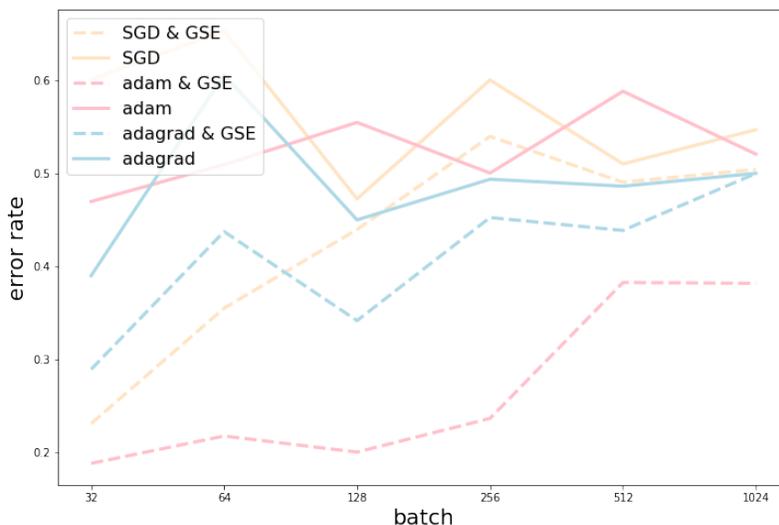


Figure 4: Results (error rate) on the ACI dataset with the ResNet-like network. The dashed curves represents experiments with **GCE** and shows an improvement on the loss for every optimizer used.

### 5.2. categorical models on public datasets

For the categorical models, the experiments are run on a supply chain Domain Specific Language: Envision. It is a Python-like implementation of SQL narrowed down for supply chain problems. A differentiable programming layer on the relational programming language presented in Peseux (2021) has been added which gives a direct access to the gradients of categorical models. The used programming language is well documented <sup>2</sup> but not open source yet. For both datasets, we will compare the resulting models resulting from stochastic optimization based on Adam or our algorithm for the determination of the parameters of a linear regression model, slightly modified to take into account categorical variables. To implement this idea, we worked on two different public datasets: the Chicago taxi ride dataset and the Belarus used car dataset.

The Chicago taxi rides dataset can be found here <sup>3</sup>. For each ride, we use the taxi

<sup>2</sup><https://docs.lokad.com/>

<sup>3</sup><https://data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew>

identifier, distance, payment type and the tips amount. We use a modified version of linear regression to predict the tip based on the trip distance and the payment type. We used a categorical version of this model where the slope depends on the taxi and the payment method, the intercept remains shared among all the trips, presented in Equation 9:

$$\hat{tips} = (\gamma_{\text{taxi}} \times \mu_{\text{payment}}) \times \text{distance} + b \quad (9)$$

There is one  $\gamma$  per taxi and also one  $\mu$  per payment method, this is a categorical model. As the intercept is shared among all taxis, the dataset is unsplittable. A model based on Equation 10

$$\hat{tips} = \gamma_{\text{taxi}} \times \text{distance} + b_{\text{taxi}} \quad (10)$$

could be split into different datasets (one per taxi) and thus we would be in the classical setting of a linear regression.

We also worked on the Belarus used cars dataset. It is presented by Lepchenkov (2019) and contains vehicle features. We take into account the car manufacturer, the production year, the origin region of the car to predict the selling price of the car as presented in Equation 11.

$$\hat{price} = (\gamma_{\text{manufacturer}} \times \mu_{\text{region}}) \times \text{year} + b \quad (11)$$

As seen in Table 3, the relational batch performed better with our proposition based on Algorithm 1 with the following setting: 30 epochs ; optimizer Adam with default setting ; batch size of 1. Experiment was reproduced 20 times.

Table 3: Results with categorical models

Dataset	Adam	Adam & GCE
Chicago Ride	35.58 ± 1.11	<b>9.45 ± 16.33</b>
Used Cars	7.10 ± 2.45	<b>0.08 ± 0.01</b>

### 5.3. categorical models on a real case

We have successfully deployed to production such categorical model at Lokad, a French company specialized in supply chain optimization, in order to weekly forecast sales of Celio, a large retail company. The dataset is open sourced (with anonymization) and presented <sup>4</sup>. The dataset contains 3 years of history and concerns

---

<sup>4</sup>soon

100k different items. The dataset contains multiple categorical features for each item. The objective is to forecast sales at the item level. The implemented categorical model is similar<sup>5</sup> to the following:

$$\hat{y}(item, week) = \theta_{store(item)} \times \theta_{color(item)} \times \theta_{size(item)} \times \Theta[group(item), WeekNumber(week)]$$

$\Theta[group(item), WeekNumber(week)]$  is a parameter vector that can be seen as a function that aims to capture the annual seasonality for a given group of items:

$$\Theta : Groups \times [[1, 52]] \longrightarrow \mathbb{R}$$

This model could be formalized as a small neural network. We use Adam as optimizer with its default values with GCE and stochastic gradient descent to update the parameters. It outstandingly outperforms the classical gradient estimator on this categorical model. The final loss (decayed RMSE) is an order of magnitude better with GCE on the testing dataset.

## 6. Conclusions

This work addresses the issue of using stochastic gradient descent on categorical data. We have shown how one-hot-encoding is needed for interpretable models. Encoding categorical data is not part of the gradient-exposed part of the model otherwise it leads to incoherent gradients. The proposed gradient estimator solves this problem and relies on the observation that a non-present gradient is not a zero-gradient. This therefore unlocks correct treatment of categorical data for all gradient-based models. The code and all the details of the study are mainly open-sourced<sup>6</sup> and demonstrates its utility on several datasets, including on a real world supply chain dataset open sourced for this work.

The main significance of this work is to highlight the lack of special treatment for categorical data. They are unfairly underrepresented on public datasets and machine learning in general. We hope that this work will encourage researchers to take them into consideration, for example by using GCE.

---

<sup>5</sup>we do not disclose the actual model for confidentiality reasons.

<sup>6</sup><https://github.com/ppmdatix/GCE>

## Acknowledgment

This work was supported by the University of Rouen and the French company Lokad We would like to thank Gaëtan Delétoille, Joannes Vermorel and Antonio Cifonelli for interesting discussions on the topic.

## References

- Archive, U.K., 1999. Kdd99 dataset. URL: <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>.
- Arik, S.Ö., Pfister, T., 2021. Tabnet: Attentive interpretable tabular learning. ArXiv abs/1908.07442.
- Blackard, J.A., Dean, D.J., 1999. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and Electronics in Agriculture* 24, 131–151.
- Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., Kasneci, G., 2021. Deep neural networks and tabular data: A survey.
- Chen, L., 2014. Curse of Dimensionality. pp. 1–1. doi:10.1007/978-1-4899-7993-3\_133-2.
- Chen, T., Guestrin, C., 2016. Xgboost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* .
- Cheng, H.T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhya, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., Anil, R., Haque, Z., Hong, L., Jain, V., Liu, X., Shah, H., 2016. Wide and deep learning for recommender systems, 7–10. doi:10.1145/2988450.2988454.
- Chickering, D., Heckerman, D., 2013. Fast learning from sparse data .
- Defossez, A., Bottou, L., Bach, F., Usunier, N., 2020. On the convergence of adam and adagrad.
- Dekel, O., 2016. Linear learning with sparse data .
- Duncan, J., Wu, Q., Promislow, K., 2014. Biased gradient squared descent saddle point finding method. *The Journal of chemical physics* 140, 194102. doi:10.1063/1.4875477.

- Gorishniy, Y., Rubachev, I., Khrulkov, V., Babenko, A., 2021. Revisiting deep learning models for tabular data.
- Ha, T.T., Gao, X., 2021. Evolving multi-view autoencoders for text classification, pp. 270–276. doi:10.1145/3486622.3493969.
- Kadra, A., Lindauer, M.T., Hutter, F., Grabocka, J., 2021. Well-tuned simple nets excel on tabular datasets, in: NeurIPS.
- Kaggle, 2012. Don't get kicked competitions. URL: <https://www.kaggle.com/c/DontGetKicked/overview>.
- Kingma, D., Ba, J., 2014. Adam: A method for stochastic optimization. International Conference on Learning Representations .
- Kohavi, R., 1997. Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. KDD .
- Kool, W., Maddison, C., Mnih, A., 2021. Unbiased gradient estimation with balanced assignments for mixtures of experts.
- Lepchenkov, K., 2019. Used-cars-catalog. URL: <https://www.kaggle.com/lepchenkov/usedcarscatalog>.
- Ma, Y., Zheng, T., 2016. Stabilized sparse online learning for sparse data. Journal of Machine Learning Research 18.
- Oktay, D., McGreivy, N., Aduol, J., Beatson, A., Adams, R., 2020. Randomized automatic differentiation .
- Ostroumova, L., Gusev, G., Vorobev, A., Dorogush, A.V., Gulin, A., 2018. Catboost: unbiased boosting with categorical features, in: NeurIPS.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., . Pytorch: An imperative style, high-performance deep learning library.
- Peseux, P., 2021. Differentiating relational queries, in: PhD@VLDB.
- Popov, S., Morozov, S., Babenko, A., 2019. Neural oblivious decision ensembles for deep learning on tabular data.

- Rehman, U., 2021. Relation on nlp with machine and language. *Global Sci-Tech* 13, 39–42. doi:10.5958/2455-7110.2021.00011.2.
- Rudin, C., 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence* 1, 206–215. doi:10.1038/s42256-019-0048-x.
- Savinov, N., Chung, J., Binkowski, M., Elsen, E., Oord, A., 2021. Step-unrolled denoising autoencoders for text generation.
- Song, W., Shi, C., Xiao, Z., Duan, Z., Xu, Y., Zhang, M., Tang, J., 2019. Autoint: Automatic feature interaction learning via self-attentive neural networks. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* .
- Tucker, G., Mnih, A., Maddison, C., Sohl-Dickstein, J., 2017. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models .
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., Polosukhin, I., 2017. Attention is all you need .
- Zhao, P., Wang, D., Wu, P., Hoi, S., . A unified framework for sparse online learning. *ACM Transactions on Knowledge Discovery from Data* 14, 1–20. doi:10.1145/3361559.

## Appendix A. Uniform draw

Let  $Z$  be a non-empty finite set and  $T \subset Z$  also non-empty.

We uniformly draw  $m > 0$  elements in  $Z$  with replacement. We focus on the first drawing where at least one of the  $m$  drawn elements belongs to  $T$ . We note  $\tilde{K}$  this drawing. Thus:

$$\mathbb{P}(\tilde{K} = 1) = 1 - \left(\frac{|Z| - |T|}{|Z|}\right)^m = P_1 \quad (\text{A.1})$$

$$\mathbb{P}(\tilde{K} = n) = (1 - P_1)^{n-1} P_1 \quad (\text{A.2})$$

**Theorem Appendix A.1** (Stopping time).  $\mathbb{E}[\tilde{K}] = \frac{1}{P_1}$

*Proof.*

$$\begin{aligned} \mathbb{E}[\tilde{K}] &= \sum_{n=1}^N n \mathbb{P}(\tilde{K} = n) = \sum_{n=1}^N n (1 - P_1)^{n-1} P_1 \\ &= \frac{P_1}{1 - P_1} \sum_{n=1}^N n (1 - P_1)^n \end{aligned}$$

For  $0 < x < 1$  we get:

$$\begin{aligned} \sum_{n=1}^{\infty} n x^n &= \sum_{n=1}^{\infty} x \frac{\partial x^n}{\partial x} \\ &= x \frac{\partial}{\partial x} \sum_{n=1}^{\infty} x^n \\ &= x \frac{\partial}{\partial x} \sum_{n=0}^{\infty} x^n \\ &= x \frac{\partial}{\partial x} \frac{1}{1 - x} \\ &= \frac{x}{(1 - x)^2} \end{aligned}$$

Then

$$\begin{aligned}\sum_{n=1}^{\infty} n\mathbb{P}(\tilde{K} = n) &= \frac{P_1}{1 - P_1} \frac{1 - P_1}{P_1^2} \\ &= \frac{1}{P_1}\end{aligned}$$

□

**Remark 1.** *It is the same result if the drawings are done without replacement. The only difference is a highest  $P_1$ .*

## Appendix B. Estimator

Let  $Z$  be a non-empty finite set and  $T \subset Z$  also non-empty. We have a score function  $s$  on  $T$ :

$$\begin{aligned}s &: T \longrightarrow \mathbb{R} \\ t &\longrightarrow s(t)\end{aligned}$$

We aim to estimate

$$s_T = \frac{1}{|T|} \sum_{x \in T} s(x)$$

Let  $(M_k)_{k \leq K}$  a serie of  $K$  draws uniform with replacement of  $m$  elements of  $Z$ .

**Remark 2.** *Thanks to Theorem Appendix A.1 we can ignore the first draws  $M_0$  such as  $M_0 \cap T = \emptyset$*

One notes

$$\begin{aligned}M_k &= (M_k \cap T) \sqcup (M_k \cap (Z \setminus T)) \\ &= (M_k^T) \sqcup (M_k \cap (Z \setminus T))\end{aligned}$$

$$avg(M_k^T) = \begin{cases} 0 & \text{if } M_k^T = \emptyset \\ \frac{1}{|M_k^T|} \sum_{x \in M_k^T} s(x) & \text{otherwise} \end{cases}$$

and

$$\bar{K} = |\{k \leq K | M_k^T \neq \emptyset\}|$$

Thanks to Remark 2 we have  $\bar{K} \geq 1$ . Then the proposed estimator is  $\hat{a}$ :

$$\hat{a} = \frac{1}{\bar{K}} \sum_{k=1}^K \text{avg}(M_k^T)$$

**Theorem Appendix B.1** (Unbiased estimator).  $\hat{a}$  is an unbiased estimator of  $s_T$

*Proof.*

$$\begin{aligned} \mathbb{E}[\tilde{a}] &= \frac{1}{\hat{K}} \sum_{\substack{M_k^T \neq \emptyset \\ k=1}}^K \frac{1}{|M_k^T|} \sum_{x \in M_k^T} \mathbb{E}[s(x)] \\ &= \frac{\bar{K}}{\bar{K}} \frac{|M_k^T|}{|M_k^T|} \mathbb{E}[s_T] \\ &= s_T \end{aligned}$$

□

Dataset	SGD	SGD & GCE	Adagrad	Adagrad & GCE	Adam	Adam & GCE
<b>ACI</b>	0.48 ± 0.24	0.24 ± 0.01	0.48 ± 0.23	<b>0.21 ± 0.01</b>	0.60 ± 0.24	0.46 ± 0.22
<b>DGK</b>	0.56 ± 0.35	<b>0.12 ± 0.01</b>	0.60 ± 0.35	<b>0.12 ± 0.01</b>	0.41 ± 0.36	0.35 ± 0.35
<b>Forest Cover</b>	1.98 ± 0.02	1.95 ± 0.01	1.98 ± 0.02	<b>1.44 ± 0.04</b>	1.98 ± 0.04	1.95 ± 0.03
<b>KDD99</b>	1.75 ± 0.20	<b>0.93 ± 0.12</b>	1.80 ± 0.17	<b>0.07 ± 0.03</b>	1.94 ± 0.19	1.63 ± 0.19
<b>Used Cars</b>	1.07 ± 0.06	<b>0.98 ± 0.01</b>	1.08 ± 0.07	<b>0.99 ± 0.01</b>	1.10 ± 0.08	1.02 ± 0.04

Table B.4: Results with mlp and batch of 32

Dataset	SGD	SGD & GCE	Adagrad	Adagrad & GCE	Adam	Adam & GCE
<b>ACI</b>	0.49 ± 0.10	<b>0.20 ± 0.01</b>	0.48 ± 0.14	<b>0.19 ± 0.01</b>	0.49 ± 0.14	<b>0.19 ± 0.01</b>
<b>DGK</b>	0.45 ± 0.19	<b>0.12 ± 0.01</b>	0.50 ± 0.20	<b>0.12 ± 0.01</b>	0.52 ± 0.24	<b>0.14 ± 0.01</b>
<b>Forest Cover</b>	2.01 ± 0.04	<b>1.18 ± 0.01</b>	2.01 ± 0.04	<b>1.03 ± 0.01</b>	2.00 ± 0.04	<b>1.05 ± 0.01</b>
<b>KDD99</b>	1.81 ± 0.10	<b>0.07 ± 0.01</b>	1.84 ± 0.08	<b>0.01 ± 0.01</b>	1.82 ± 0.12	<b>0.04 ± 0.01</b>
<b>Used Cars</b>	1.23 ± 0.10	<b>1.02 ± 0.01</b>	1.20 ± 0.09	<b>1.04 ± 0.01</b>	1.18 ± 0.05	<b>1.03 ± 0.01</b>

Table B.5: Results with resnet and batch of 32

Dataset	SGD	SGD & GCE	Adagrad	Adagrad & GCE	Adam	Adam & GCE
<b>ACI</b>	0.45 ± 0.25	0.25 ± 0.02	0.44 ± 0.23	0.20 ± 0.03	0.53 ± 0.23	0.41 ± 0.22
<b>DGK</b>	0.32 ± 0.32	0.11 ± 0.01	0.43 ± 0.38	0.12 ± 0.01	0.43 ± 0.38	0.29 ± 0.30
<b>Forest Cover</b>	2.00 ± 0.03	1.98 ± 0.02	1.99 ± 0.03	<b>1.54 ± 0.06</b>	1.99 ± 0.03	1.97 ± 0.02
<b>KDD99</b>	1.84 ± 0.14	<b>1.32 ± 0.11</b>	1.85 ± 0.23	<b>0.13 ± 0.08</b>	1.95 ± 0.19	1.74 ± 0.20
<b>Used Cars</b>	1.10 ± 0.06	<b>1.01 ± 0.01</b>	1.11 ± 0.09	1.01 ± 0.01	1.11 ± 0.10	1.05 ± 0.06

Table B.6: Results with mlp and batch of 64

Dataset	SGD	SGD & GCE	Adagrad	Adagrad & GCE	Adam	Adam & GCE
<b>ACI</b>	0.54 ± 0.12	<b>0.21 ± 0.01</b>	0.55 ± 0.11	<b>0.17 ± 0.01</b>	0.57 ± 0.14	<b>0.16 ± 0.01</b>
<b>DGK</b>	0.45 ± 0.12	<b>0.11 ± 0.01</b>	0.52 ± 0.17	<b>0.12 ± 0.01</b>	0.44 ± 0.16	<b>0.13 ± 0.01</b>
<b>Forest Cover</b>	2.02 ± 0.05	<b>1.37 ± 0.03</b>	1.99 ± 0.03	<b>1.02 ± 0.01</b>	2.02 ± 0.04	<b>1.06 ± 0.01</b>
<b>KDD99</b>	1.81 ± 0.15	<b>0.12 ± 0.01</b>	1.87 ± 0.07	<b>0.02 ± 0.01</b>	1.89 ± 0.09	<b>0.06 ± 0.01</b>
<b>Used Cars</b>	1.13 ± 0.06	<b>0.99 ± 0.01</b>	1.15 ± 0.07	<b>1.02 ± 0.01</b>	1.20 ± 0.17	<b>1.01 ± 0.01</b>

Table B.7: Results with resnet and batch of 64

Dataset	SGD	SGD & GCE	Adagrad	Adagrad & GCE	Adam	Adam & GCE
<b>ACI</b>	0.45 ± 0.22	0.32 ± 0.16	0.46 ± 0.23	0.25 ± 0.02	0.48 ± 0.23	0.42 ± 0.22
<b>DGK</b>	0.58 ± 0.37	0.30 ± 0.30	0.58 ± 0.35	<b>0.12 ± 0.01</b>	0.74 ± 0.29	0.49 ± 0.30
<b>Forest Cover</b>	1.98 ± 0.03	1.97 ± 0.02	1.99 ± 0.03	<b>1.60 ± 0.07</b>	1.99 ± 0.02	1.97 ± 0.02
<b>KDD99</b>	1.80 ± 0.19	1.50 ± 0.15	1.89 ± 0.19	<b>0.45 ± 0.47</b>	1.80 ± 0.18	1.69 ± 0.18
<b>Used Cars</b>	1.16 ± 0.09	<b>1.03 ± 0.02</b>	1.12 ± 0.08	1.02 ± 0.01	1.13 ± 0.11	1.08 ± 0.09

Table B.8: Results with mlp and batch of 128

Dataset	SGD	SGD & GCE	Adagrad	Adagrad & GCE	Adam	Adam & GCE
<b>ACI</b>	0.47 ± 0.12	<b>0.24 ± 0.01</b>	0.48 ± 0.15	<b>0.19 ± 0.01</b>	0.56 ± 0.09	<b>0.19 ± 0.01</b>
<b>DGK</b>	0.50 ± 0.19	<b>0.11 ± 0.01</b>	0.47 ± 0.18	<b>0.12 ± 0.01</b>	0.39 ± 0.13	<b>0.13 ± 0.01</b>
<b>Forest Cover</b>	2.00 ± 0.03	<b>1.59 ± 0.02</b>	2.00 ± 0.03	<b>1.01 ± 0.01</b>	2.00 ± 0.02	<b>1.04 ± 0.01</b>
<b>KDD99</b>	1.85 ± 0.15	<b>0.22 ± 0.01</b>	1.90 ± 0.11	<b>0.01 ± 0.01</b>	1.82 ± 0.13	<b>0.08 ± 0.01</b>
<b>Used Cars</b>	1.17 ± 0.05	<b>1.02 ± 0.01</b>	1.17 ± 0.04	<b>1.06 ± 0.02</b>	1.16 ± 0.07	<b>1.03 ± 0.01</b>

Table B.9: Results with resnet and batch of 128

Dataset	SGD	SGD & GCE	Adagrad	Adagrad & GCE	Adam	Adam & GCE
<b>ACI</b>	0.50 ± 0.26	0.49 ± 0.25	0.50 ± 0.23	<b>0.23 ± 0.01</b>	0.50 ± 0.26	0.43 ± 0.23
<b>DGK</b>	0.34 ± 0.36	0.30 ± 0.31	0.53 ± 0.36	<b>0.11 ± 0.01</b>	0.50 ± 0.36	0.28 ± 0.29
<b>Forest Cover</b>	1.98 ± 0.03	1.98 ± 0.02	1.98 ± 0.02	<b>1.79 ± 0.06</b>	2.00 ± 0.03	1.97 ± 0.02
<b>KDD99</b>	1.84 ± 0.24	1.70 ± 0.23	1.74 ± 0.10	<b>0.57 ± 0.33</b>	1.88 ± 0.23	1.78 ± 0.23
<b>Used Cars</b>	1.08 ± 0.07	1.04 ± 0.02	1.10 ± 0.06	<b>1.02 ± 0.01</b>	1.11 ± 0.07	1.07 ± 0.05

Table B.10: Results with mlp and batch of 256

Dataset	SGD	SGD & GCE	Adagrad	Adagrad & GCE	Adam	Adam & GCE
<b>ACI</b>	0.54 ± 0.14	<b>0.24 ± 0.01</b>	0.53 ± 0.13	<b>0.19 ± 0.01</b>	0.52 ± 0.10	<b>0.19 ± 0.01</b>
<b>DGK</b>	0.59 ± 0.16	<b>0.11 ± 0.01</b>	0.50 ± 0.16	<b>0.12 ± 0.01</b>	0.48 ± 0.19	<b>0.14 ± 0.01</b>
<b>Forest Cover</b>	1.99 ± 0.04	<b>1.73 ± 0.03</b>	2.01 ± 0.02	<b>1.03 ± 0.01</b>	2.03 ± 0.02	<b>1.07 ± 0.01</b>
<b>KDD99</b>	1.76 ± 0.07	<b>0.47 ± 0.04</b>	1.80 ± 0.05	<b>0.02 ± 0.01</b>	1.86 ± 0.09	<b>0.16 ± 0.02</b>
<b>Used Cars</b>	1.17 ± 0.11	<b>1.00 ± 0.01</b>	1.17 ± 0.11	1.06 ± 0.01	1.13 ± 0.06	<b>1.01 ± 0.01</b>

Table B.11: Results with resnet and batch of 256

Dataset	SGD	SGD & GCE	Adagrad	Adagrad & GCE	Adam	Adam & GCE
<b>ACI</b>	0.56 ± 0.24	0.50 ± 0.24	0.55 ± 0.26	0.32 ± 0.17	0.45 ± 0.26	0.41 ± 0.24
<b>DGK</b>	0.54 ± 0.35	0.47 ± 0.36	0.80 ± 0.23	<b>0.22 ± 0.11</b>	0.51 ± 0.38	0.50 ± 0.38
<b>Forest Cover</b>	1.97 ± 0.02	1.97 ± 0.02	2.01 ± 0.03	<b>1.92 ± 0.02</b>	2.01 ± 0.03	1.99 ± 0.02
<b>KDD99</b>	1.82 ± 0.17	1.74 ± 0.16	1.89 ± 0.18	<b>1.41 ± 0.17</b>	1.85 ± 0.20	1.79 ± 0.20
<b>Used Cars</b>	1.10 ± 0.08	1.05 ± 0.04	1.10 ± 0.07	0.99 ± 0.03	1.11 ± 0.11	1.08 ± 0.08

Table B.12: Results with mlp and batch of 512

Dataset	SGD	SGD & GCE	Adagrad	Adagrad & GCE	Adam	Adam & GCE
<b>ACI</b>	0.47 ± 0.15	<b>0.25 ± 0.02</b>	0.49 ± 0.11	<b>0.18 ± 0.01</b>	0.54 ± 0.15	<b>0.19 ± 0.01</b>
<b>DGK</b>	0.52 ± 0.21	<b>0.13 ± 0.01</b>	0.60 ± 0.21	<b>0.12 ± 0.01</b>	0.46 ± 0.17	<b>0.13 ± 0.01</b>
<b>Forest Cover</b>	2.03 ± 0.04	<b>1.86 ± 0.03</b>	2.01 ± 0.04	<b>1.02 ± 0.01</b>	2.00 ± 0.05	<b>1.08 ± 0.01</b>
<b>KDD99</b>	1.79 ± 0.07	<b>0.88 ± 0.03</b>	1.84 ± 0.10	<b>0.02 ± 0.01</b>	1.84 ± 0.08	<b>0.22 ± 0.04</b>
<b>Used Cars</b>	1.18 ± 0.07	<b>1.03 ± 0.01</b>	1.15 ± 0.08	<b>1.04 ± 0.01</b>	1.14 ± 0.05	<b>1.02 ± 0.01</b>

Table B.13: Results with resnet and batch of 512

Dataset	SGD	SGD & GCE	Adagrad	Adagrad & GCE	Adam	Adam & GCE
<b>ACI</b>	0.51 ± 0.26	0.50 ± 0.26	0.46 ± 0.26	0.36 ± 0.21	0.48 ± 0.25	0.45 ± 0.25
<b>DGK</b>	0.42 ± 0.37	0.42 ± 0.37	0.49 ± 0.37	0.16 ± 0.06	0.59 ± 0.36	0.58 ± 0.37
<b>Forest Cover</b>	1.99 ± 0.03	1.99 ± 0.03	1.99 ± 0.02	1.95 ± 0.01	1.99 ± 0.03	1.98 ± 0.03
<b>KDD99</b>	1.83 ± 0.27	1.77 ± 0.26	1.91 ± 0.29	1.67 ± 0.30	1.82 ± 0.21	1.78 ± 0.20
<b>Used Cars</b>	1.08 ± 0.08	1.06 ± 0.06	1.19 ± 0.13	1.08 ± 0.08	1.09 ± 0.07	1.07 ± 0.06

Table B.14: Results with mlp and batch of 1024

Dataset	SGD	SGD & GCE	Adagrad	Adagrad & GCE	Adam	Adam & GCE
<b>ACI</b>	0.53 ± 0.13	<b>0.32 ± 0.07</b>	0.51 ± 0.10	<b>0.18 ± 0.01</b>	0.54 ± 0.15	<b>0.19 ± 0.01</b>
<b>DGK</b>	0.48 ± 0.20	<b>0.18 ± 0.04</b>	0.44 ± 0.14	<b>0.13 ± 0.01</b>	0.48 ± 0.19	<b>0.15 ± 0.01</b>
<b>Forest Cover</b>	2.00 ± 0.04	<b>1.88 ± 0.04</b>	2.01 ± 0.04	<b>1.04 ± 0.01</b>	2.03 ± 0.04	<b>1.13 ± 0.01</b>
<b>KDD99</b>	1.80 ± 0.10	<b>1.12 ± 0.09</b>	1.86 ± 0.10	<b>0.05 ± 0.01</b>	1.81 ± 0.06	<b>0.32 ± 0.07</b>
<b>Used Cars</b>	1.11 ± 0.02	<b>1.05 ± 0.01</b>	1.19 ± 0.09	<b>1.03 ± 0.01</b>	1.12 ± 0.03	<b>1.01 ± 0.01</b>

Table B.15: Results with resnet and batch of 1024