Stochastic Gradient Descent with **gradient estimator for symbolical features**

Stochastic Gradient Descent with **gradient estimator for symbolical features**

Paul Peseux[1,1,1], Maxime Berar[1], Thierry Paquet[1], Victor Nicollet[1],

[a]*Lokad, Paris, FRANCE*
[b]*Litis, Rouen, FRANCE*

Abstract

Symbolic 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 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 exists. After a survey on symbolic data in public datasets, a novel gradient estimator is introduced. We show what this estimator minimizes in theory and show its efficiency on 6 different datasets with multiple model architectures.

This new estimator performs better than common estimators under similar settings. Overall, the aim of this paper is to convince researchers to thoroughly consider symbolic data and adapt their models, optimizers, benchmarks ... to these key features.

Keywords: symbolic data, gradient descent, online learning, gradient estimation

1. Introduction

Symbolic data are present in many application domains, some on them with critical constraints such as health **?** or supply chain **?**. They are opposed to numerical data as they are not represented by numbers, but by symbols. As a symbolic feature comes with an alphabet and conveys a specific meaning, it is fundamentally different from what we get with a numerical one. Recent gradient-based machine learning models have shown outstanding results on homogeneous data in areas such as speech recognition or computer vision **? ?**, and it is very tempting to use those models on symbolic data as well. In order to do so, symbolic data need a specific encoding

[1]corresponding author, paul.peseux@lokad.com

such that they can comply with the requirements of most of the machine learning models that rely on numerical input representations, one of the most common being one-hot-encoding **?**.

While learning representations for the natural language processing has allowed for symbolic representations to be embedded into numerical ones, in a very efficient way, relying on distributional hypothesis of the language **?**, other application domains more related to databases cannot benefit from the same hypothesis to train efficient embedding. Moreover, learning a representation frequently comes at the cost of losing the interpretability of the representation by domain experts. Model interpretability is fundamental in critical areas with high stake decisions such as health or supply chain **?**. As it relies on a bijection between the symbolic alphabets and the encoded vector components, one-hot-encoding provides a very simple way to build interpretable models. Its main consequence is that the encoded vector is binary, high dimensional and sparse. One common solution is to use dimensionality reduction methods **? ?**, but again it comes at the expense of a loss of model interpretability.

Recent works on tabular data **? ?** have shown that stochastic gradient-based machine learning models using a simple one-hot-encoding can perform well also for symbolic data compared to other models, despite the known structural problems of one-hot-encoding . Indeed, in this context, we face the issue that state-of-the-art stochastic gradient descent methods are not suited for one-hot encoded data. By default these methods assume the encoded vector as real valued vector which translates non-observed examples or features as numerical zeros, whereas they should be considered as a non-existing configurations. For example, the Adam optimizer **?** relies on the definition of momentum, which in this context updates parts of the gradient for the non-existing configurations. In this paper, a novel gradient estimator is introduced: **gradient estimator for symbolical features** (**GSE**), which takes into account the structural properties of the encoded symbolic data and scale the gradient accordingly.

The paper is organized as follow: first we present a review of public datasets that contains symbolic features. Then, we present different encodings and focus on one-hot-encoding that allows us to build *symbolic models* . Then, we tackle the problem of applying *stochastic* gradient descent on these models that accepts input symbolic data by design. We solve this issue with a new gradient estimator and exhibit the actual loss it minimizes in the convex case. This work ends with multiple experiments that show the robustness of the new gradient estimator.

Table 1: symbolic data

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

2. Symbolic models and feature encoding

Let us denote " *symbolic models* " the set of models that accept symbolic features by design and are numerical , i.e. their parameters can be updated through gradient descent. By symbolic features, we denote a feature s belonging to an alphabet of n_s symbols $\{s_1, \cdots, s_{n_s}\}$, whose cardinality depends on the data. In Table **??**, color and stores are symbolic features, and pink and blue are symbols, and forms the color alphabet. For example logistic regression introduced by **?** belongs to this set of models. Wide models described in **?** also correspond to this. However, Decision Trees do not as they cannot be considered numerical models, but rather ensemble models. Regular neural networks are not *symbolic models* either because they cannot use raw symbolic input data and need to be encoded.

A straightforward *symbolic models* for inputs taken from Table **??** could be **??** and is represented through the graphical representation depicted in Figure **??**

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

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

We stress the fact that the *symbolic* aspect applies to the input features: a *symbolic models* can be used for regression, multi-class classification ...

In this simple model **??**, the parameter μ_{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 *symbolic models* raises an issue as common gradient update techniques are not designed for symbolic features: not every symbol of a symbolic feature is present in every observation of a dataset while

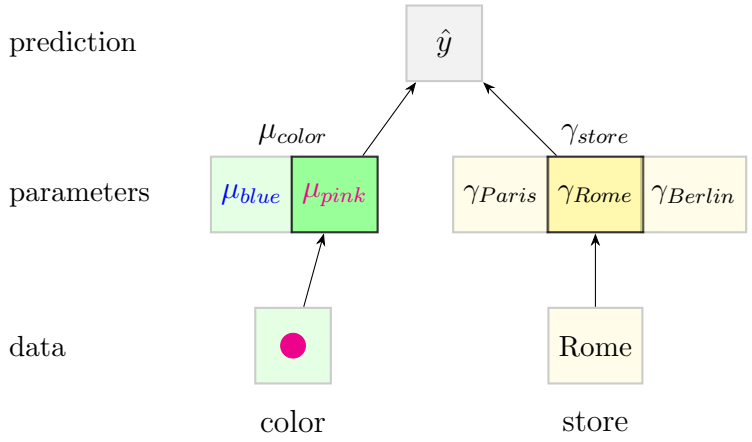


Figure 1: *symbolic models* accessing parameters

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 symbolic features. This paper links work on sparse gradient estimator and one-hot-encoding symbolic data that leads to a sparse but structured gradient.

symbolic models have to deal with numbers as parameters, not symbols. Symbolic data require a numerical embedding before they can be given as input.

Many possible encodings exist:

- ordinal encoding
- one-hot-encoding
- binary encoding
- target encoding
- leave-one-out encoding
- positional encoding
- ...

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 *symbolic models* . Model **??** one-hot-encoding is depicted in Equation **??** and Figure **??**.

one-hot-encoding a symbolic 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 **??**, 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 **?**. 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 **?** or **?** 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 symbolic 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 **?** especially in high-stake contexts such as disease diagnostics, where explanability 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 **??**, μ_{blue} (μ_{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 **??**, $\mu_{blue} > \mu_{pink}$ means that the blue color sells better than the pink one. Not only is the prediction of the model explainable, but the model itself conveys meaning, even without inputs.

Leave-one-out encoding turns the symbolic 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: **? ?**; **?**; **?** ... The following also applies to sparse but structured data. As one-hot-encoding is not suited for high-cardinality symbolic "sparse" features we exclude such feature from the following, which is not restrictive for domains such as health or supply chain.

3. Problem description

In this section we present the problems encountered with using one hot encoding and then illustrate it in the online learning set up.

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 $\{s_k\}_{k \leq p}$:

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

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

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

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

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

Figure **??** illustrates the formal definition. In Figure **??**, $\{\theta_1; \theta_2; \theta_3; \theta_4; \theta_5; \theta_6; \}$ are related to the first feature $feat_1$. On Model **??**, such notations give Figure **??**.

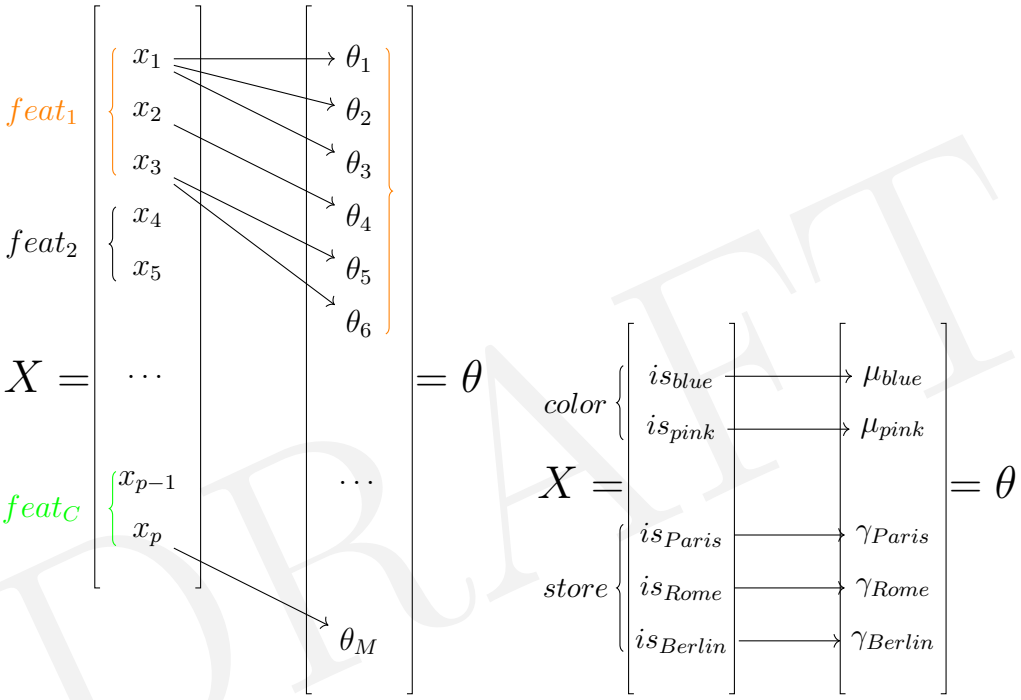
3.2. Gradient descent with symbolic variables

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:

$$\nabla F = \frac{1}{n} \sum_{obs} \nabla f_{obs} = \frac{1}{n} \sum_{i \leq n} \nabla_\theta f(X_i, y_i) \tag{3}$$

$$= \frac{1}{n} \sum_{batch} \quad \sum_{obs \in batch} \nabla f_{obs} \tag{4}$$

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 mean. Gradient descent techniques succeed in finding a minimum, notably because this the gradient estimator is unbiased.



(a) Generic n otations

(b) Notations applied to Model **??**

Figure 2: Notations

Regarding *symbolic models* , we can highlight the fact that not every symbolic parameter is affected by every observation. For example in model **??** μ_{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 symbolic 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.

$$\nabla_{\mu_{blue}} F = \frac{1}{n} \sum_{obs} \nabla_{\mu_{blue}} f_{obs} \tag{5}$$

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

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 μ_{red} in Model **??** with no red products? $\{obs \in batch | color(obs) = blue\}$ from Equation **??** might be empty, and in this case 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 concerns 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 bring any information, it should not be used for parameters updates.

$$\mu_{color} = \mu_{blue} \times is_{blue} + \mu_{pink} \times is_{pink} \tag{7}$$

$$\frac{\partial \mu_{color}}{\partial \mu_{blue}} |_{color=pink} = 0$$

$$\frac{\partial \mu_{color}}{\partial \mu_{pink}} |_{color=blue} = 0$$

This issue especially concerns under-represented symbols and small batches. The smallest the symbol cardinality and the smallest the batch, the highest probability to not be 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 online learning (i.e. with batch size of 1) example is presented in Section **??**. Many descent techniques[2] **?** such as **?** or **?** that rely on an estimator of the gradient use

[2]not all, see **?** for example

an unbiased estimator of the gradient. **?** proves that it is a sufficient condition for convergence in the convex setting. Here the gradients of the symbolic parameters are somehow biased.

one-hot-encoding should not be part of the model (see Equation **??**). It is a way to deal with symbolic data as positional encoding is not integrated to Transformers **?**. As it is not part of the model itself, it should not infer on model's parameters updates.

Encoding symbolic data is not part of the gradient-exposed part of the model..

3.3. Online Gradient descent

Let's consider the case of batches of size 1 also called online learning. It means that observations are taken one by one. In Model **??** Each observation concerns one and only one *color* and one and only one *store*.

Table 2: symbolic data

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

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 online learning should be generalized to bigger batches.

3.4. Comparison with common optimizer

Adagrad introduced by **?** is an optimizer to update parameters through gradient descent; the learning rate is parameter's history dependant:

$$\theta_{t,j} = \theta_{t-1,j} - \frac{\alpha}{\sqrt{\sum_{\tau=1}^{t} \nabla_{\theta_{j,\tau}} f^2}} \nabla_{\theta_{j,t}} f$$

Adagrad performs smaller updates for parameters associated with frequently occurring symbols, and larger updates for parameters associated with infrequent symbols. It has been designed to support sparse input such as one encoded data. Indeed an infrequent symbol (associated to $\theta_{j,t}$) will have a relatively small gradient history $\sum_{\tau=1}^{t} \nabla_{\theta_{j,\tau}} f_{\tau,j}^2$ as $\nabla_{\theta_{j,\tau}} f$ is often zero, which leads to a relatively big learning rate. To our knowledge it is one of the very few optimizer widely used that intrinsically take into account sparse data.

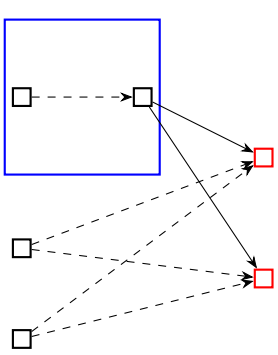
Many gradient-based models as **?** updates their parameters with Adam **?** or modified versions of it **? ?** as an optimizer. We can state that Adam is one of the default optimizer in Machine Learning in general. To simplify, with Adam, parameters are updated with their *momentum* taken into account. Using batches on symbolic data leads to "slow down" parameters' *momentum* if not present in the batch and follow their *momentum* for many batches if not present in consecutive ones. With the original Adam notation, a zero gradient gives:

$$\begin{aligned} m_t &= \beta_1 \times m_{t-1} \\ v_t &= \beta_2 \times v_{t-1} \\ \theta_t &= \theta_{t-1} - \frac{\alpha}{1-\beta_1^t} \frac{m_t}{\epsilon + \sqrt{\frac{v_t}{1-\beta_2^t}}} \end{aligned}$$

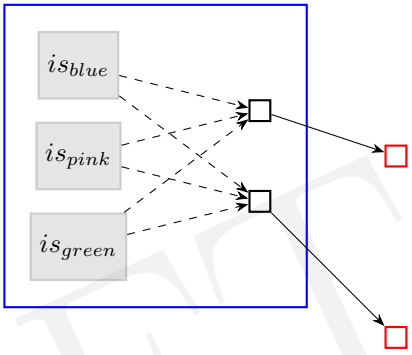
If a symbol is rarely present in batches, its related parameter will *follow* its *momentum* (created on batches where it is present) on batches where it is absent. This update does not make any sense for parameters that have been unused during the batch.

4. Deep Learning

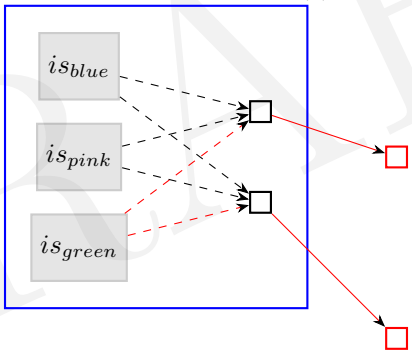
Most deep architectures are built to support numerical data such as image or audio but one-hot-encoding approach has already been applied by **?** for example and is called *Feature Tokenizer*. Applying **one-hot-encoding** to feed neural networks is theoretically equivalent to use a different network for each symbol. We illustrate that on a dense linear layer as depicted in Figures **??**. This use of different networks for each symbols raises the issue depicted in Section **??** when used with *batch* during training. Gradient estimation done by *batch* implies that the parameters are used on every observation, which totally makes sense for classical Deep Learning (i.e. with no one-hot encoded data). If a symbol is not present in a *batch*, it means that there is no accumulated gradient, which is different than accumulated gradient is zero.



(a) Dense Layer



(b) one-hot-encoding of a dense layer



(c) dense layer relative to she *green* symbol

Figure 3: one-hot-encoding for neural networks

5. Solution: gradient estimator for symbolical 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 make 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:

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

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

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 **??**.

With the current notations:

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

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_θ where m observations (over the $\mathcal{Z}$) are uniformly drawn. It is the estimator used by Algorithm **??**. This estimator is unbiased, proof can be found in appendices **??**:

Algorithm 1 gradient estimator for symbolical features

Require: $\mathcal{Z}$: data
Require: $update(\cdot, \cdot)$: chosen optimizer
Require: θ_0: Initial parameter vector
$t \leftarrow 0$
while θ_t not converged **do** $t \leftarrow t + 1$
Divide Z in $Batches$
for batch $\in$ Batches **do**
for symbol $\in$ Alphabet **do**
$c_{symbol} \leftarrow 0$
end for
$\mathbf{g} \leftarrow \vec{0}$
for X, y $\in$ batch **do**
$c_{symbol(X)} \leftarrow c_{symbol(X)} + 1$
Compute $\nabla_{\theta_{t-1}} f_{\theta_{t-1}}(X)$ thanks to y
$\mathbf{g} \leftarrow \mathbf{g} + \nabla_{\theta_{t-1}} f_{\theta_{t-1}}(X)$ ▷ **accumulate gradient**
end for
$\theta_t \leftarrow \theta_{t-1}$
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
end for
end while

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

Which is a sufficient condition for convergence in the convex setting **?**.

The loss function depicted in equation **??** seems similar to loss used for classification with unbalanced **output** categories. Let recall that what we propose here is different as we consider unbalanced **input** symbols.

In the case where symbol groups have same size C then the objective function resumes to:

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

as the p symbol groups form a partition of $\mathcal{Z}$.

In this case, our proposed gradient estimator is proportional to the classic one. If one uses the vanilla optimizer, **gradient estimator for symbolical features** 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 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 small learning rate and big gradient thanks to **gradient estimator for symbolical features** rather than big learning rate and small gradient. It is proven in the results Section **??**.

5.1. Comparison with other work

We completely agree on the following "not every update is informative about every feature due to the scarcity of nonzero entries" from **?** (AdaGrad). Our proposed approach coincides on the report but differs on the way to tackle the issue. Our

approach moves the encoding outside the gradient updates and does not make any assumption on the chosen optimizer. Thus our solution is totally compatible with AdaGrad. It has been tested and results are available in Section **??**.

Our proposed solution, i.e. **gradient estimator for symbolical features** on one-hot-encoded data, is very similar to vector embedding. The main difference here is the cardinality of the symbolic features concerned.

gradient estimator for symbolical features aims to handle imbalanced input symbolic data. In order to handle imbalanced output data, data augmentation is often used **?** **?** ... But this augmentation techniques apply on numerical input data. For symbolic data as input, this seems quite impossible to augment it: if rotating an image does not change the objects present on it, switching a binary value completely change the concerned observation.

6. Experimental and Results

We have implemented Algorithm **??** in two different cases and with two different languages : symbolic models and deep learning models using encoded symbolic data. In both settings, we aim to measure the impact of the **gradient estimator for symbolical features** . As a consequence the chosen metric will be the efficiency difference with the current treatment of symbolic parameters in batch gradient descent. The public datasets presented in table **??** will be used for evaluation as well as a private dataset from the supply chain domain.

Table 3: Datasets characteristics

Dataset	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

6.1. symbolic models on public datasets

For the *symbolic models* the experiments are run on a supply chain Domain Specific Language: Envision. It is a Python·like implementation of SQL narrowed for supply chain problems. A differentiable programming layer on the relational programming language presented in **?** has been added which gives a direct access to the gradients of *symbolic models* . The used programming language is well

documented [3] but not open source. 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 symbolical 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 that can be found here [4]. For each ride, we use the taxi identifier, distance, payment type and the tips amount. We use modified version of linear regression to predict the tip based on the trip distance and the payment type. We used a symbolic 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 **??**:

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

There is one γ per taxi and also one μ per payment method, this is a *symbolic models* . As the intercept is shared among all taxis, the dataset is unsplitable. A model based on Equation **??**

$$\hat{tips} = \gamma_{\text{taxi}} \times distance + b_{taxi} \tag{9}$$

could be split in different dataset (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 presented by **?** 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 **??**.

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

As seen table **??**, the relational batch performed better with our proposition based on Algorithm **??** with the following setting: 30 epochs ; optimizer Adam with default setting ; batch size of 1. Experiment was reproduced 20 times. The used metric is the mean square error (MSE).

6.2. symbolic models on a real case

TODO validate it with Joannes/Victor

We have successfully deployed to production such *symbolic models* at Lokad, a french company specialized in supply chain optimization, in order to weekly forecast

[3] `https://docs.lokad.com/`

[4] `https://data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew`

Table 4: Results with *symbolic models*

Dataset	Adam	Adam & **GSE**
Chicago Ride	35.58 ± 1.11	9.45 ± 16.33
Used Cars	7.10 ± 2.45	**0.08** ± **0.01**

sales of a large retail company. The forecast is done at the item level. The dataset contains 3 years of history and concerns more than 13 millions different items.

The implemented *symbolic models* is similar[5] to:

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

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

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

We use Adam as optimizer with its default values with **gradient estimator for symbolical features** and a minibatch of 1 to update the parameters. It outstandingly outperforms the classical gradient estimator on this (very) *symbolic models* . The final loss (decayed RMSE) is an order of magnitude better with **gradient estimator for symbolical features** on the testing dataset.

6.3. Deep Learning

For the Deep Learning models, we applied our solution on **PyTorch**. On this framework, it very easy to update every parameter gradient according to Algorithm **??**. The code is available and experiments can be found here[6]. In order to evaluate the impact of our proposal, we worked on 4 different symbolic datasets :

the Adult Census Income (ACI) dataset presented in **?** 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 **?** contains symbolic characteristics on $30m^2$ forest cells. The objective is to predict the forest

[5] we do not disclose the actual model for confidentiality reasons.

[6] https://github.com/ppmdatix/RelationalBatch

cover type. The KDD99 dataset accessible by **?** and aims to predict cyber-attacks. Finally the used Cars datasets from Belarus presented in **??**.

In order to only measure the impact of the **gradient estimator for symbolical features** , we *only* use those symbolic variables in our experiments. Those datasets tasks are quiet easy. As a consequence we use a very small network to highlight our approach. Our network is made of 3 dense layers of sizes $[4, 8, 4]$ with a batch of size 128. We also perform experiments on a ResNet-like network that give same results. We use the l_2 loss. We also used the same ResNet-like network than **?**. We have tested three different optimizer with their default settings: SGD (vanilla), Adagrad and Adam.

Results are reproducible in the repository and are recorded in Tables **?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ?? ??** in appendices. On the different datasets we have worked on, we always see an improvement of the loss on the testing dataset using the **gradient estimator for symbolical features** . A specific focus on the ACI dataset is done in Figure **??**. This proves the need to specifically handle stochastic gradient on symbolic data. Results in different settings demonstrate the advantage to use **gradient estimator for symbolical features** whatever the optimizer. Among other things, AdaGrad tries to handle gradient on sparse data (which includes one-hot encoded data) but we see a clear improvement on that task.

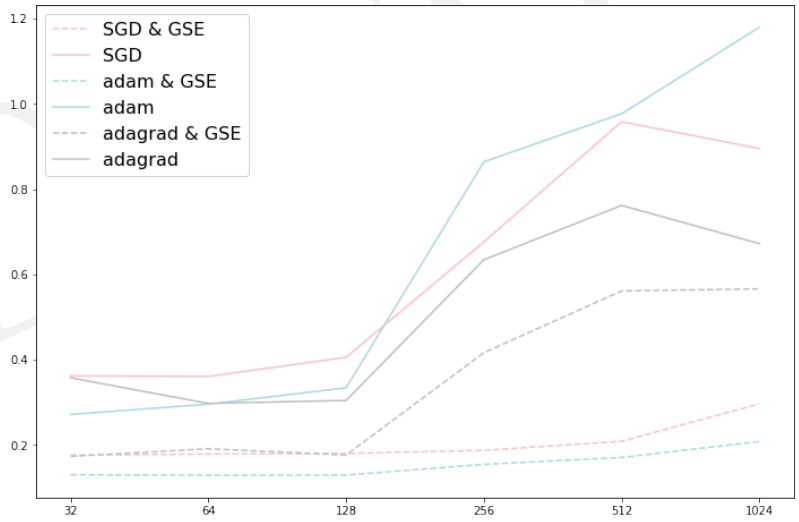


Figure 4: Results on the ACI dataset. The dashed curves represents experiments wit **GSE** and shows an improvement on the loss for every optimizer used.

7. Conclusions

This work addresses the issue of using stochastic gradient descent on symbolic data. We have shown how one-hot-encoding is needed for interpretable models. **Encoding symbolic data is not part of the gradient-exposed part of the model.**otherwise it leads to incoherent gradient. The proposed gradient estimator solves this problem and rely on the observation that **a non-present gradient is not a zero-gradient**. This thus unlocks correct treatment of symbolic data for all gradient-based models. The code and all the details of the study is mainly open-sourced[7] and demonstrates its utility on several datasets.

The main significance of this work is to highlight the lack of special treatment for symbolic data. They are unfairly underrepresented on public datasets and machine learning in general. We hope that this work will make researchers to take them more into considerations.

Acknowledgment

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

Dataset	SGD	SGD & **GSE**	Adagrad	Adagrad & **GSE**	Adam	Adam & **GSE**
ACI	0.36 ± 0.16	**0.18 ± 0.01**	0.27 ± 0.07	**0.13 ± 0.01**	0.36 ± 0.16	0.17 ± 0.02
compas	0.47 ± 0.28	0.24 ± 0.01	0.60 ± 0.26	**0.08 ± 0.02**	0.60 ± 0.19	0.38 ± 0.12
DGK						
Forest Cover	1.98 ± 0.01	**1.95 ± 0.01**	1.98 ± 0.02	**1.40 ± 0.05**	1.98 ± 0.01	1.96 ± 0.01
KDD99	3.22 ± 0.12	**0.30 ± 0.07**	3.10 ± 0.13	**0.17 ± 0.02**	3.15 ± 0.19	**0.55 ± 0.07**

Table .5: Results with mlp and batch of 32

Appendix .1. Performance impact

To implement this solution, we need to count each symbol occurrence in the batch. If well implemented, this is negligible. Especially when the data is column-oriented, you only need to *group by* symbol and count them. Moreover this has no relationship with the θ's value, it can be pre-computed and reused at every epoch if the training dataset stay static through steps.

[7] https://github.com/ppmdatix/RelationalBatch

Dataset	SGD	SGD & **GSE**	Adagrad	Adagrad & **GSE**	Adam	Adam & **GSE**
ACI	0.37 ± 0.07	**0.13 ± 0.01**	0.42 ± 0.07	**0.13 ± 0.01**	0.42 ± 0.07	**0.13 ± 0.01**
compas	0.69 ± 0.27	**0.06 ± 0.01**	0.65 ± 0.24	**0.01 ± 0.01**	0.87 ± 0.47	**0.05 ± 0.01**
DGK						
Forest Cover	2.03 ± 0.03	**1.14 ± 0.01**	2.02 ± 0.03	**1.04 ± 0.01**	2.01 ± 0.04	**1.06 ± 0.01**
KDD99	3.24 ± 0.06	**0.06 ± 0.01**	3.19 ± 0.05	**0.05 ± 0.01**	3.22 ± 0.06	**0.06 ± 0.01**

Table .6: Results with resnet and batch of 32

Dataset	SGD	SGD & **GSE**	Adagrad	Adagrad & **GSE**	Adam	Adam & **GSE**
ACI	0.36 ± 0.17	**0.18 ± 0.01**	0.30 ± 0.09	**0.13 ± 0.01**	0.30 ± 0.14	0.19 ± 0.08
compas	0.52 ± 0.46	0.25 ± 0.01	0.57 ± 0.27	**0.12 ± 0.05**	0.67 ± 0.36	0.49 ± 0.25
DGK	0.21 ± 0.06	**0.11 ± 0.01**	0.19 ± 0.06	**0.11 ± 0.01**	0.20 ± 0.08	0.14 ± 0.04
Forest Cover	1.99 ± 0.02	1.97 ± 0.01	1.97 ± 0.01	**1.52 ± 0.09**	1.98 ± 0.01	1.96 ± 0.01
KDD99	3.09 ± 0.18	**0.38 ± 0.08**	3.12 ± 0.21	**0.17 ± 0.03**	3.36 ± 0.12	**0.68 ± 0.16**

Table .7: Results with mlp and batch of 64

Dataset	SGD	SGD & **GSE**	Adagrad	Adagrad & **GSE**	Adam	Adam & **GSE**
ACI	0.35 ± 0.02	**0.14 ± 0.01**	0.40 ± 0.11	**0.13 ± 0.01**	0.47 ± 0.11	**0.13 ± 0.01**
compas	0.65 ± 0.23	**0.08 ± 0.01**	0.65 ± 0.31	**0.02 ± 0.01**	0.59 ± 0.20	**0.06 ± 0.01**
DGK	0.27 ± 0.11	**0.12 ± 0.01**	0.28 ± 0.04	**0.10 ± 0.01**	0.30 ± 0.13	**0.12 ± 0.01**
Forest Cover	2.02 ± 0.03	**1.29 ± 0.02**	2.02 ± 0.03	**1.05 ± 0.01**	2.02 ± 0.03	**1.07 ± 0.01**
KDD99	3.19 ± 0.10	**0.06 ± 0.01**	3.16 ± 0.08	**0.05 ± 0.01**	3.23 ± 0.06	**0.06 ± 0.01**

Table .8: Results with resnet and batch of 64

Dataset	SGD	SGD & **GSE**	Adagrad	Adagrad & **GSE**	Adam	Adam & **GSE**
ACI	0.41 ± 0.20	**0.18 ± 0.01**	0.33 ± 0.18	**0.13 ± 0.01**	0.30 ± 0.14	0.18 ± 0.02
compas	0.65 ± 0.19	**0.31 ± 0.03**	0.56 ± 0.35	**0.12 ± 0.05**	0.51 ± 0.24	0.43 ± 0.19
DGK						
Forest Cover	2.00 ± 0.02	1.98 ± 0.01	1.99 ± 0.01	**1.57 ± 0.09**	1.99 ± 0.02	1.97 ± 0.02
KDD99	3.19 ± 0.14	**0.60 ± 0.19**	3.27 ± 0.17	**0.16 ± 0.03**	3.25 ± 0.16	**0.96 ± 0.29**

Table .9: Results with mlp and batch of 128

Dataset	SGD	SGD & **GSE**	Adagrad	Adagrad & **GSE**	Adam	Adam & **GSE**
ACI	0.78 ± 0.20	**0.14 ± 0.01**	0.99 ± 0.33	**0.13 ± 0.01**	0.90 ± 0.22	**0.13 ± 0.01**
compas	0.69 ± 0.34	**0.11 ± 0.01**	0.75 ± 0.28	**0.02 ± 0.01**	0.75 ± 0.53	**0.06 ± 0.01**
DGK						
Forest Cover	2.02 ± 0.03	**1.52 ± 0.03**	2.01 ± 0.03	**1.03 ± 0.01**	2.01 ± 0.02	**1.06 ± 0.01**
KDD99	3.18 ± 0.03	**0.07 ± 0.01**	3.21 ± 0.08	**0.05 ± 0.01**	3.18 ± 0.07	**0.06 ± 0.01**

Table .10: Results with resnet and batch of 128

Dataset	SGD	SGD & **GSE**	Adagrad	Adagrad & **GSE**	Adam	Adam & **GSE**
ACI	0.68 ± 0.36	**0.19 ± 0.01**	0.86 ± 0.43	**0.15 ± 0.01**	0.63 ± 0.43	0.42 ± 0.30
compas	0.86 ± 0.40	0.49 ± 0.15	0.57 ± 0.30	**0.20 ± 0.04**	0.53 ± 0.29	0.47 ± 0.24
DGK						
Forest Cover	1.98 ± 0.02	1.97 ± 0.01	1.99 ± 0.02	**1.74 ± 0.06**	1.99 ± 0.02	1.97 ± 0.02
KDD99	3.12 ± 0.18	**0.96 ± 0.22**	3.11 ± 0.20	**0.19 ± 0.03**	3.14 ± 0.14	**1.47 ± 0.43**

Table .11: Results with mlp and batch of 256

Dataset	SGD	SGD & **GSE**	Adagrad	Adagrad & **GSE**	Adam	Adam & **GSE**
ACI	1.00 ± 0.28	**0.14 ± 0.01**	1.10 ± 0.43	**0.12 ± 0.01**	0.79 ± 0.18	**0.13 ± 0.01**
compas	0.73 ± 0.31	**0.17 ± 0.01**	0.69 ± 0.18	**0.03 ± 0.01**	0.57 ± 0.17	**0.07 ± 0.01**
DGK						
Forest Cover	2.01 ± 0.03	**1.73 ± 0.02**	2.01 ± 0.03	**1.05 ± 0.01**	2.03 ± 0.03	**1.09 ± 0.01**
KDD99	3.14 ± 0.08	**0.08 ± 0.01**	3.20 ± 0.05	**0.05 ± 0.01**	3.18 ± 0.07	**0.06 ± 0.01**

Table .12: Results with resnet and batch of 256

Dataset	SGD	SGD & **GSE**	Adagrad	Adagrad & **GSE**	Adam	Adam & **GSE**
ACI	0.96 ± 0.58	**0.21 ± 0.02**	0.98 ± 0.56	**0.17 ± 0.03**	0.76 ± 0.44	0.56 ± 0.35
compas	0.64 ± 0.36	0.46 ± 0.20	0.51 ± 0.36	0.28 ± 0.15	0.68 ± 0.35	0.63 ± 0.32
DGK	0.19 ± 0.06	0.15 ± 0.04	0.16 ± 0.06	0.11 ± 0.01	0.21 ± 0.12	0.18 ± 0.09
Forest Cover	1.98 ± 0.02	1.98 ± 0.01	1.98 ± 0.02	**1.85 ± 0.05**	1.99 ± 0.02	1.98 ± 0.02
KDD99	3.17 ± 0.19	**1.11 ± 0.24**	3.10 ± 0.10	**0.21 ± 0.02**	3.11 ± 0.18	**1.93 ± 0.41**

Table .13: Results with mlp and batch of 512

Dataset	SGD	SGD & **GSE**	Adagrad	Adagrad & **GSE**	Adam	Adam & **GSE**
ACI	0.40 ± 0.08	**0.15 ± 0.01**	0.34 ± 0.04	**0.12 ± 0.01**	0.38 ± 0.08	**0.13 ± 0.01**
compas	0.58 ± 0.14	**0.23 ± 0.02**	0.51 ± 0.12	**0.04 ± 0.01**	0.62 ± 0.20	**0.07 ± 0.01**
DGK	0.26 ± 0.04	**0.17 ± 0.01**	0.27 ± 0.06	**0.12 ± 0.01**	0.29 ± 0.08	**0.13 ± 0.01**
Forest Cover	2.02 ± 0.03	**1.86 ± 0.03**	2.02 ± 0.04	**1.04 ± 0.01**	1.99 ± 0.04	**1.08 ± 0.01**
KDD99	3.23 ± 0.09	**0.10 ± 0.01**	3.20 ± 0.08	**0.06 ± 0.01**	3.23 ± 0.06	**0.07 ± 0.01**

Table .14: Results with resnet and batch of 512

Dataset	SGD	SGD & **GSE**	Adagrad	Adagrad & **GSE**	Adam	Adam & **GSE**
ACI	0.90 ± 0.57	0.30 ± 0.09	1.18 ± 0.54	**0.21 ± 0.04**	0.67 ± 0.49	0.57 ± 0.44
compas	0.56 ± 0.31	0.47 ± 0.22	0.56 ± 0.21	0.42 ± 0.20	0.68 ± 0.33	0.64 ± 0.31
DGK	0.18 ± 0.09	0.15 ± 0.05	0.25 ± 0.14	0.16 ± 0.07	0.18 ± 0.07	0.16 ± 0.05
Forest Cover	1.98 ± 0.02	1.98 ± 0.02	1.99 ± 0.02	**1.93 ± 0.02**	1.99 ± 0.02	1.98 ± 0.02
KDD99	3.29 ± 0.13	**1.66 ± 0.34**	3.16 ± 0.15	**0.21 ± 0.02**	3.09 ± 0.11	**2.40 ± 0.21**

Table .15: Results with mlp and batch of 1024

Dataset	SGD	SGD & **GSE**	Adagrad	Adagrad & **GSE**	Adam	Adam & **GSE**
ACI	0.48 ± 0.14	**0.18 ± 0.01**	0.39 ± 0.05	**0.13 ± 0.01**	0.42 ± 0.11	**0.13 ± 0.01**
compas	0.59 ± 0.16	**0.27 ± 0.01**	0.60 ± 0.19	**0.05 ± 0.01**	0.72 ± 0.30	**0.09 ± 0.02**
DGK	0.29 ± 0.07	**0.18 ± 0.01**	0.25 ± 0.05	**0.12 ± 0.01**	0.30 ± 0.13	**0.13 ± 0.01**
Forest Cover	2.00 ± 0.03	**1.91 ± 0.03**	2.02 ± 0.03	**1.09 ± 0.01**	2.00 ± 0.03	**1.14 ± 0.01**
KDD99	3.19 ± 0.09	**0.13 ± 0.01**	3.17 ± 0.05	**0.05 ± 0.01**	3.18 ± 0.05	**0.07 ± 0.01**

Table .16: Results with resnet and batch of 1024

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:

$$\mathbb{P}(\tilde{K} = 1) = 1 - (\frac{\mid Z \mid - \mid T \mid}{\mid Z \mid})^m = P_1 \tag{A.1}$$

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

Theorem Appendix A.1 (Stoping 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:

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

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:

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

We aim to estimate

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

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

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

One notes

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

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

and

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

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

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

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

Proof.

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

□