Spectral Target Encoding

Alexandre Bouayad
University of Cambridge
United Kingdom

Martin Mihelich
École Normale Supérieure
Paris, France

Abstract

We introduce a new spectral technique for Bayesian inference in the case of the hierarchical beta-binomial model. Our inference method consists in an iterative algorithm, alternating between posterior expectation for the binomial parameters and a Bayesian form of the method of moments for the beta parameters. We mathematically establish efficiency and convergence properties for our algorithm. We empirically demonstrate that our inference method is more efficient than likelihood inference. The advantage is significant when considering statistical efficiency, and it is wide when considering computational efficiency. We apply our new inference method to target encoding in machine learning. We benchmark *spectral target encoding* against other known target encoding techniques on several supervised binary classification problems. We observe that the spectral target encoder is on par with, and regularly better than the best-in-class target encoders, while being much faster. We provide an implementation of the spectral target encoder in Python, compatible with the scikit-learn framework, together with the code for running the benchmarks in a reproducible way.

1 Introduction

Many datasets used for training and prediction by machine learning systems contain information of *categorical* nature. Such datasets record observations that can take on one of a limited, and usually fixed, number of possible values – the so-called *categories*. Examples of such situations are ubiquitous, within various application domains: fraud detection (Awoyemi et al., 2017), click prediction (McMahan et al., 2013), churn prediction (Vafeiadis et al., 2015), to cite a few.

There is most often no inherent way to order categories, nor there is a natural way to quantify differences between them. On the other hand, most prediction models in machine learning are built under the assumption that input variables are *continuous* ones. It is then necessary to add to those models some upstream layer, where categorical variables are *encoded*, that is, *represented*, into continuous ones. How categorical variables are encoded has substantial implications on the accuracy of the downstream prediction model, may it be linear or logistic regressions, decision trees (Lucena, 2020), neural networks (Hancock and Khoshgoftaar, 2020), or others, and one can directly measure such effects in diverse Kaggle[1] competitions.

There exists a full range of encoding methods and the literature on the subject is already vast. We refer the reader to comparative studies of Pargent et al. (2019, 2021) and to the survey of Hancock and Khoshgoftaar (2020). As a concrete illustration of such diversity, we mention the dedicated software library `category-encoders` (McGinnis et al., 2018), which proposes implementations within the scikit-learn framework of about twenty different encoders, as of today.

One of the most common encoding technique is *one-hot encoding*, which consists in replacing a category by a single binary variable. One-hot encoding has the advantage to retain all the information conveyed by the categorical variables. Its inconvenient however is to embed the categorical variables into a high-dimensional space (as high as the total number of categories minus 1) easily exposing the prediction models to the curse of dimensionality.

Another largely adopted solution in supervised learning is *target encoding*, which consists in replacing a category by the target's mean value for that category. Its comparative advantage to one-hot encoding is that each categorical variable is replaced by a one-dimensional continuous variable. At first glance, it may appear counter-intuitive to use the object of predictions to encode the explanatory variable on which those predictions are based. Although this indeed can bring

[1] `https://www.kaggle.com`

complications, e.g. target leakage, one can actually view target encoding as a first step towards prediction.

When there are few observations per category, estimation of the target's mean value for one category can suffers from high variance and results in overfitting for the global ML system. In order to mitigate such effects, target encoding often relies on a form of *regularisation*, which here amounts to interpolating between the *local means*, that is, the target's mean values per category, and, on the other side, the *global mean*, that is, the target's mean value based on all the observations. Such regularisation is very often – more or less explicitly – based on a Bayesian modelling of the target. The latter consists in postulating the existence of a common prior distribution for the local means. A category is then encoded by the mean of the posterior distribution, given the observations for that category.

The Bayesian approach supposes a statistical modelling for the local means. In the case where the target takes binary values, a natural model consists in assuming that the local means are beta distributed. In other words, this consists in assuming a *hierarchical beta-binomial model* (HBBM) for the target variable (Gelman et al., 2014). This brings, for each of the categorical variables, new hyper-parameters to the overall ML system, namely, the parameters of the beta distribution. Another possible and used model for the target variable is the *multilevel generalised linear model* with no predictors (Gelman and Hill, 2006). The latter can be viewed as a particular kind of *generalised linear mixed model* (GLMM).

The GLMM comes with inference methods for the whole set of its parameters. One is *likelihood inference*. Another, *Bayesian inference*, consists in a moment-based approach of the expectation-maximisation algorithm (Dempster et al., 1977), coupled to Gibbs sampling. By contrast, all known target encoding techniques that suppose for regularisation a prior beta distribution do not provide complete inference for the beta hyper-parameters. They perform partial inference instead, and they leave the rest of the beta hyper-parameters to the ML system.

One solution for inferring the complete set of beta parameters in the case of the HBBM would be using likelihood inference. However, to the best of the authors' knowledge, such inference is not used in practice for target encoding, nor has it been documented. One reason might be that likelihood inference is computationally expensive. One common appealing trait of the known HBBM-based target encoders is that they are indeed fast encoders. They are in particular much faster than GLMM-based target encoders. Another solution for complete inference in the case of the HBBM would be fine-tuning the remaining hyper-parameters, using direct cross-validation – or other more refined techniques, as the Swiss army infinitesimal jackknife (Giordano et al., 2019)). Once again, the downside of that latter solution would be the rapid increase of inference times. Nonetheless, complete inference usually improves the performance of target encoding.

Main contributions and results

We propose in the present paper to combine the best of two worlds, that is, fast and complete inference for target encoding. We introduce to this end a new *spectral technique* for Bayesian inference in the case of the HBBM. Our inference method consists in an iterative algorithm, alternating between posterior expectation for the binomial parameters and a Bayesian form of the method of moments for the beta parameters. Contrary to Bayesian inference for the GLMM, our inference method does not need Gibbs sampling to ensure good convergence properties, which we mathematically establish. We also empirically demonstrate that our inference method significantly outperforms likelihood inference when considering statistical efficiency. We do not provide a mathematical analysis of time complexity, but we empirically demonstrate that our inference method is considerably faster than likelihood inference. We propose to apply our inference method to target encoding; we call *spectral target encoding* the resulting new encoding technique. We demonstrate on several datasets that spectral target encoding is as fast as the other known HBBM-based target encoding techniques, while performing on par with, or better than, GLMM-based and fine-tuned ones. We provide an implementation of the spectral target encoder in `Python`, compatible with the `scikit-learn` framework, together with the code for running the benchmarks in a reproducible way; see `https://github.com/alexandrebouayad/spectral-target-encoding`.

Organisation of the paper

We recall in section 2 Bayesian target encoding based on the hierarchical beta-binomial model, and we introduce notations for the rest of the paper. We describe in section 3 the spectral inference method for the HBBM, and we state convergence theorems. We present in section 4 results for a benchmark of the spectral inference method, focused on statistical and computational efficiencies. We present in section 5 results for a benchmark of spectral target encoding on several datasets, and in combination with commonly used classifiers. The supplementary material contains proofs of the theorems presented in section 3, as well as more detailed results and a description of the datasets used for the benchmark of section 5. The GitHub repository accom-

panying this paper also contains detailed results for both the statistical and ML benchmarks.

General notations

We denote by $\mathbb{N}$ and $\mathbb{R}$ the set of non-negative integers and of real numbers, respectively. We denote by $|\mathbb{S}|$ the cardinality of a finite set $\mathbb{S}$.

For events A and B, we denote by $P(A)$ the probability of A, and by $P(A \,|\, B)$ the probability of A given B. For a random variable X, we denote by $\mathbb{E}(X)$ its expectation, and by $\mathrm{Var}(X)$ its variance. We denote by $\mathbb{E}(X \,|\, B)$ and $\mathrm{Var}(X \,|\, B)$ the conditional expectation and conditional variance of X given B.

For an estimator $\hat{\theta}$ of some quantity θ, we denote by $\mathrm{Bias}(\hat{\theta})$ its bias, and by $\mathrm{MSE}(\hat{\theta})$ its mean squared error. We denote by $\mathrm{Bias}(\hat{\theta} \,|\, B)$ and $\mathrm{MSE}(\hat{\theta} \,|\, B)$ the bias and mean squared error of $\hat{\theta}$ given B.

2 Bayesian target encoding

We propose in this section a recollection of Bayesian target encoding based on the hierarchical beta-binomial model. We thus here assume that the target variable takes binary values, and that target encoding is performed on each categorical variable separately and independently. As a consequence of the latter, we will consider in what follows only one categorical variable to be target encoded. The notations and assumptions introduced in this section we will be those in use throughout the rest of the paper.

We designate by X a categorical variable with values in a set $\mathbb{J}$, and we designate by Y the target variable with values in $\{0, 1\}$. Let us suppose that we have n observations $(x_1, y_1), \ldots, (x_n, y_n)$ for the pair (X, Y). We denote by Y_i the random variable from which the observation y_i has been drawn.

We designate by n_j the number of observations that we have for the category $j \in \mathbb{J}$, namely,

$$n_j = |\mathbb{I}(j)|, \quad \text{with } \mathbb{I}(j) = \{i : x_i = j\}. \tag{1}$$

We designate by a_j and b_j the numbers of observations for the category j and for which the target is 1 and 0, respectively, so that $a_j + b_j = n_j$. We write $\bar{a}_j$ and $\bar{b}_j$ for the numbers a_j and b_j in proportion to n_j, namely,

$$\bar{a}_j = a_j / n_j, \quad \bar{b}_j = b_j / n_j. \tag{2}$$

We assume that the Bernoulli variables Y_i for $i \in \mathbb{I}(j)$ are identically distributed, and we make the Bayesian assumption that their common parameter, which we denote by π_j, is random. We moreover assume that the variables π_j ($j \in \mathbb{J}$) are independent, and that, conditional on the latter, the variables Y_i ($1 \leq i \leq n$) are also independent. We denote by A_j the random variable from which a_j is drawn, that is, $A_j = \sum_{i \in \mathbb{I}(j)} Y_i$.

The *Bayes estimate* $\pi_j^* = \pi_j^*(a_j)$ is the expected value of π_j given our observations, namely,

$$\pi_j^* = \mathbb{E}\big[\pi_j \,|\, A_j = a_j\big]. \tag{3}$$

We recall that the Bayes estimator π_j^* is the minimum mean square error (MMSE) estimator of π_j, that is, $\mathrm{MSE}(\pi_j^*) \leq \mathrm{MSE}(\hat{\pi}_j)$ for any other estimator $\hat{\pi}_j$ of π_j.

Bayesian target encoding consists in replacing each category $j \in \mathbb{J}$ by the Bayes estimate π_j^*. In the presence of only one categorical variable, Bayesian target encoding thus yields the expected optimal predictor for Y given the observations, and with respect to the mean squared error. This illustrates our general remark made in the introduction about the prediction content of target encoding.

To complete the description of the hierarchical beta-binomial model, we add to the previous assumptions that the random Bernoulli parameters π_j ($j \in \mathbb{J}$) are identically beta distributed. We denote by α and β the shape parameters of the common beta distribution. We recall that α and β are positive real numbers, and that the support of the beta distribution is the open unit interval. We besides denote by ν the quantity $\alpha + \beta$, and we denote by μ the mean of the beta distribution, that is, α / ν.

The beta distribution is a *conjugate prior* for the binomial distribution; π_j given $A_j = a_j$ is again beta distributed, with shape parameters $\alpha + a_j$ and $\beta + b_j$. Hence, the Bayes estimate of π_j can be written as

$$\pi_j^* = \frac{a_j + \alpha}{n_j + \alpha + \beta} = \frac{n_j}{n_j + \nu}\, \bar{a}_j + \frac{\nu}{n_j + \nu}\, \mu. \tag{4}$$

The latter equality expresses π_j^* as an interpolation between the local mean $\bar{a}_j$ and the mean μ of the prior beta distribution. This illustrates our general remark made in the introduction about how the Bayesian approach naturally introduces regularisation for target encoding.

3 Spectral inference

We explain in this section the construction of the spectral inference algorithm for the hierarchical beta-binomial model. We use the same notations than in section 2. We first consider partial inference, before treating complete inference. The section contains two theorems. The supplementary material contains the proofs. As already evoked in the introduction, the

spectral inference algorithm iteratively bootstraps estimates of the HBBM parameters from initial estimates. Theorem 3.3 states that the algorithm converges to estimates that actually do not depend on the initial estimates. Theorem 3.1 states that in the case of partial inference, the algorithm converges to the best linear unbiased estimators. We view this last result as a strong indication that the algorithm converges to optimal estimates in the general case too.

As previously seen, the value of the Bayes estimate π_j^* depends on the values of the beta parameters μ and ν. In order to stress out such dependence, the expectation operator will be subscripted by μ and ν, as follows:

$$\pi_j^* = \mathbb{E}_{\mu,\nu}\big[\pi_j \,|\, A_j = a_j\big] = \frac{a_j + \nu\mu}{n_j + \nu}. \tag{5}$$

Let us first assume that μ is unknown and that ν is known. We then seek to estimate μ. If we are given an estimate $\hat{\mu}$ of μ, we can use it to construct an estimate $\hat{\pi}_j$ of π_j, which is defined as the Bayes estimate of π_j when μ is equal to its estimate $\hat{\mu}$, namely,

$$\hat{\pi}_j = \mathbb{E}_{\hat{\mu},\nu}\big[\pi_j \,|\, A_j = a_j\big] = \frac{a_j + \nu\hat{\mu}}{n_j + \nu}. \tag{6}$$

The estimates $\hat{\pi}_j$ ($j \in \mathbb{J}$) provide in turn information about the beta distribution, and we can average them in order to produce a new, updated, estimate $\check{\mu}$ of μ, namely,

$$\check{\mu} = \frac{1}{|\mathbb{J}|} \sum_{j \in \mathbb{J}} \hat{\pi}_j. \tag{7}$$

Armed with the new estimate $\check{\mu}$ of μ, we redo what we did with the initial estimate $\hat{\mu}$ of μ, which leads us to a new estimation of the Bernoulli parameters π_j ($j \in \mathbb{J}$). Repeating multiples times the process, we obtain a bootstrapping iterative algorithm for estimating the parameters of both the binomial and beta distributions. We summarise below the algorithm in pseudocode; see algorithm 1.

Data: $(a_j : j \in \mathbb{J})$, $(b_j : j \in \mathbb{J})$, $\hat{\mu}$
Result: $\hat{\mu}$, $(\hat{\pi}_j : j \in \mathbb{J})$
repeat until convergence
  foreach $j \in \mathbb{J}$ **do**
    $\hat{\pi}_j \leftarrow \dfrac{a_j + \nu\hat{\mu}}{a_j + b_j + \nu}$
  end
  $\hat{\mu} \leftarrow \texttt{mean}(\hat{\pi}_j : j \in \mathbb{J})$

Algorithm 1: μ unknown and ν known

The algorithm requires an initial estimate of μ. If we know a priori no more than ν about the beta distribution, we can sensibly start with $\hat{\mu} = 1/2$. A justification for such a choice is that the beta distribution with parameters $\mu = 1/2$ and $\nu = 1$ is the Jeffreys prior for the mean of a Bernoulli distribution; see Robert et al. (2009). That being said, we show that the asymptotic behaviour of our algorithm does not in fact depend on the choice of an initial estimate of μ.

Theorem 3.1. *For any initial estimate of μ in $(0, 1)$, algorithm 1 converges and yields new estimates of μ and $\boldsymbol{\pi}$ that are asymptotically independent on the initial estimate of μ. Moreover, the limit estimates, written $\hat{\mu}^*$ and $\hat{\boldsymbol{\pi}}^*$, are the best linear unbiased estimator (BLUE) of μ and $\boldsymbol{\pi}$, and can be expressed in terms of the observations as*

$$\hat{\mu}^* = \sum_{j \in \mathbb{J}} \frac{a_j}{n_j + \nu} \Bigg/ \sum_{j \in \mathbb{J}} \frac{n_j}{n_j + \nu}, \quad \hat{\pi}_j^* = \frac{a_j + \nu\hat{\mu}^*}{n_j + \nu}, \tag{8}$$

Remark 3.2. *The estimates of μ and $\boldsymbol{\pi}$ produced by algorithm 1 are sound, i.e., they remain after each iteration of the algorithm within the parameter space of the HBBM, namely,*

$$\hat{\mu} \in (0, 1), \quad \hat{\pi}_j \in [0, 1]. \tag{9}$$

We can view algorithm 1 as a moment-based version of the expectation-maximisation algorithm, where the expectation stage is preserved, while the method of maximum likelihood, used during the maximisation stage to estimate μ, is replaced by the method of moments. From this standpoint, and in light of theorem 3.1, we propose to adapt algorithm 1 to the case where both beta parameters μ and ν are unknown.

We now require initial estimates $\hat{\mu}$ and $\hat{\nu}$ of both beta parameters. Again, if we have no a-priori information about the beta distribution, sensible values are provided by the Jeffreys prior for the mean of a Bernoulli distribution, that is, $\hat{\mu} = 1/2$ and $\hat{\nu} = 1$. Similarly to what we did for algorithm 1, we construct an estimate $\hat{\pi}_j$ of π_j, which is defined as the Bayes estimate of π_j when μ and ν are equal to $\hat{\mu}$ and $\hat{\nu}$, namely,

$$\hat{\pi}_j = \mathbb{E}_{\hat{\mu},\hat{\nu}}\big[\pi_j \,|\, A_j = a_j\big] = \frac{a_j + \hat{\nu}\hat{\mu}}{n_j + \hat{\nu}}. \tag{10}$$

Let us suppose that we knew the value p_j that has been drawn from the latent variable π_j before a_j was drawn from A_j. The method of moments would then yield estimates $\tilde{\mu}$ and $\tilde{\mu}'_2$ of the first and second moments of the beta distribution, namely,

$$\tilde{\mu} = \frac{1}{|\mathbb{J}|} \sum_{j \in \mathbb{J}} p_j, \quad \tilde{\mu}'_2 = \frac{1}{|\mathbb{J}|} \sum_{j \in \mathbb{J}} p_j^2. \tag{11}$$

In reality, the estimates $\tilde{\mu}$ and $\tilde{\mu}'_2$ are not accessible to us, and the best replacements – with respect to the mean squared error – given the observation of A_j, and given the initial estimates $\hat{\mu}$ and $\hat{\nu}$, are provided by Bayes estimation, namely,

$$\check{\mu} = \frac{1}{|\mathbb{J}|} \sum_{j \in \mathbb{J}} \mathbb{E}_{\hat{\mu},\hat{\nu}} \big[\pi_j \,|\, A_j = a_j \big], \tag{12a}$$

$$\check{\mu}'_2 = \frac{1}{|\mathbb{J}|} \sum_{j \in \mathbb{J}} \mathbb{E}_{\hat{\mu},\hat{\nu}} \big[\pi_j^2 \,|\, A_j = a_j \big], \tag{12b}$$

that is,

$$\check{\mu} = \frac{1}{|\mathbb{J}|} \sum_{j \in \mathbb{J}} \frac{a_j + \hat{\nu}\hat{\mu}}{n_j + \hat{\nu}}, \tag{13a}$$

$$\check{\mu}'_2 = \frac{1}{|\mathbb{J}|} \sum_{j \in \mathbb{J}} \frac{(a_j + \hat{\nu}\hat{\mu})(a_j + \hat{\nu}\hat{\mu} + 1)}{(n_j + \hat{\nu})(n_j + \hat{\nu} + 1)}. \tag{13b}$$

Still following the method of moments, we obtain a new estimate $\check{\nu}$ of the parameter ν, defined as

$$\check{\nu} = \frac{\check{\mu} - \check{\mu}'_2}{\check{\mu}'_2 - \check{\mu}^2}. \tag{14}$$

Armed with the new estimates $\check{\mu}$ and $\check{\nu}$, we redo what we did with the initial estimates of μ and ν, which leads us to a new estimation of the Bernoulli parameters π_j ($j \in \mathbb{J}$). Repeating multiples times the process, we obtain a bootstrapping iterative algorithm for estimating the parameters of both the binomial and beta distributions. We summarise below the algorithm in pseudocode; see algorithm 2.

Data: $(a_j : j \in \mathbb{J})$, $(b_j : j \in \mathbb{J})$, $\hat{\mu}$, $\hat{\nu}$
Result: $\hat{\mu}$, $\hat{\nu}$, $(\hat{\pi}_j : j \in \mathbb{J})$
repeat until convergence
  foreach $j \in \mathbb{J}$ **do**
    $\hat{\pi}_j \leftarrow \dfrac{a_j + \hat{\nu}\hat{\mu}}{a_j + b_j + \hat{\nu}}$
    $\hat{\pi}'_j \leftarrow \dfrac{a_j + \hat{\nu}\hat{\mu} + 1}{a_j + b_j + \hat{\nu} + 1}$
  end
  $\hat{\mu} \leftarrow \texttt{mean}(\hat{\pi}_j : j \in \mathbb{J})$
  $\hat{\mu}'_2 \leftarrow \texttt{mean}(\hat{\pi}_j \hat{\pi}'_j : j \in \mathbb{J})$
  $\hat{\nu} \leftarrow \dfrac{\hat{\mu} - \hat{\mu}'_2}{\hat{\mu}'_2 - \hat{\mu}^2}$

Algorithm 2: μ and ν unknown

The algorithm requires initial estimates of μ and ν. Here again, we show that the asymptotic behaviour of our algorithm does not in fact depend on the choice of the initial estimates of μ and ν.

Theorem 3.3. *For any initial estimate of μ in $(0,1)$ and of ν in $(0,+\infty)$, algorithm 2 converges and yields new estimates of μ, ν and $\boldsymbol{\pi}$ that are asymptotically independent on the initial estimates of μ and ν. Moreover, the limit estimates, written $\hat{\mu}^*$, $\hat{\nu}^*$ and $\hat{\boldsymbol{\pi}}^*$, can be expressed as the unique solution to the following system of equations:*

$$\left.\begin{aligned} &\hat{\mu}^* = \sum_{j \in \mathbb{J}} \frac{a_j}{n_j + \nu} \Big/ \sum_{j \in \mathbb{J}} \frac{n_j}{n_j + \nu}, \\ &\hat{\pi}^*_j = \frac{a_j + \hat{\nu}^*\hat{\mu}^*}{n_j + \hat{\nu}^*}, \quad \hat{\pi}'^*_j = \frac{a_j + \hat{\nu}^*\hat{\mu}^* + 1}{n_j + \hat{\nu}^* + 1}, \\ &\frac{\hat{\mu}^*(\hat{\nu}^*\hat{\mu}^* + 1)}{\hat{\nu}^* + 1} = \frac{1}{|\mathbb{J}|} \sum_{j \in \mathbb{J}} \hat{\pi}^*_j \hat{\pi}'^*_j, \end{aligned}\right\} \tag{15}$$

Remark 3.4. *The estimates of μ, ν and $\boldsymbol{\pi}$ produced by algorithm 2 are sound, i.e., they remain after each iteration of the algorithm within the parameter space of the HBBM, namely,*

$$\hat{\mu} \in (0,1), \quad \hat{\nu} \in (0,+\infty), \quad \hat{\pi}_j \in [0,1]. \tag{16}$$

As already said, our algorithm can be viewed as a moment-based version of the expectation-maximisation algorithm. It should be however noted that our algorithm employs a Bayesian approach during the stage involving the method of moments. As it can be easily seen, a direct application of the method of moments may produce unsound estimates of the HBBM parameters, and would invalidate theorem 3.3. We besides empirically observed that the algorithm quickly diverges when using the usual method of moments.

4 Statistical benchmark

We consider in this section the statistical benchmarking of spectral inference against likelihood inference, where we compare the two methods in terms of both statistical and computational efficiency. We provide a description of the benchmark itself, together with a results summary.

Remark 4.1. *The likelihood inference algorithm that we consider here consists in running the L-BFGS-B optimisation algorithm (Byrd et al., 1995; Zhu et al., 1997) on the negative log-likelihood, with the linear constraint $\alpha, \beta \geq 0$. We compute the log-likelihood for the HBBM using the computation of the log-likelihood for the beta-binomial distribution as provided by the Python library SciPy(Virtanen et al., 2020). We use SciPy implementation of the L-BFGS-B algorithm, with default values for*

its parameters, including, in particular, the maximum number of iterations and the convergence threshold; see `https://docs.scipy.org/doc/scipy/reference/optimize.minimize-lbfgsb.html` *for more details.*

The benchmark consists in running the two inference methods on observations drawn from the hierarchical beta-binomial model for different values of the model parameters. Here are the parameters that we make vary: the shape parameters α and β of the beta distribution; the number of categories, that is, the number of binomial distributions; the number of draws for the binomial distributions, which we take constant across all the categories. We present in table 1 and table 2 a summary of the benchmark results for $\alpha = \beta = 10$ and for $\alpha = \beta = 0.1$. These two sets of parameters exemplify the unimodal and bimodal situations for the beta distribution.

In order to ensure statistical significance for the results, the HBBM is sampled 100 times (for each set of model parameters). We compute for each of the two inference methods the mean estimation error for the shape parameters of the beta distribution, namely,

$$\frac{1}{100}\sum_{i=1}^{100}\left[\left(\hat{\alpha}_i-\alpha\right)^2+\left(\hat{\beta}_i-\beta\right)^2\right]^{1/2}, \qquad (17)$$

where $\hat{\alpha}_i$ and $\hat{\beta}_i$ designate the estimate of α and β for the ith sampling.

We give in table 1 and table 2 the mean estimation error for likelihood inference in proportion to the mean estimation error for spectral inference. These two tables also contain the ratios of the inference runtimes. We have chosen to show here the results for 100 and 1000 categories, and for a number of draws per category varying from 1 to 1000; those numbers are commonly encountered in ML problems involving categorical data.

We observe that the error ratio is significantly larger than 1. This empirically demonstrates better statistical efficiency for spectral inference. We also observe a more pronounced difference in efficiency between spectral and likelihood inferences in the case of a bimodal beta distribution, as well as when the number of categories increases – which is precisely when categorical encoding in ML problems is the most needed.

Concerning inference runtimes, we observe a wide difference between the two inference methods. The spectral inference is faster than likelihood inference by several orders of magnitude. This empirically demonstrates better computational efficiency for spectral inference. Again, the differences between the two inference methods becomes more acute as the number of categories increases.

Remark 4.2. *The GitHub repository accompanying this paper contains the code of the benchmark, as well as the code of our implementation of likelihood inference for the HBBM. Both are in Python. The repository also contains all the benchmark results, together with additional statistics, as standard deviations.*

Table 1: Likelihood/spectral ($\alpha = \beta = 10$)

categories	draws per category	error ratio	runtime ratio
100	1	1.0	1378
100	10	2.9	695
100	100	1.0	729
100	1000	1.0	267
1000	1	1.0	8861
1000	10	1.2	3830
1000	100	1.0	1069
1000	1000	1.0	200

Table 2: Likelihood/spectral ($\alpha = \beta = 0.1$)

categories	draws per category	error ratio	runtime ratio
100	1	1.0	1275
100	10	2.1	732
100	100	2.6	354
100	1000	2.5	93
1000	1	1.0	8405
1000	10	7.9	2492
1000	100	9.8	478
1000	1000	5.8	78

5 ML benchmark

We consider in this section the benchmarking of spectral target encoding, that is, Bayesian target encoding based on the HBBM with spectral inference. We compare it against other known target encoding techniques, and against Bayesian target encoding based on HBBM with likelihood inference. We provide a description of the benchmark itself, together with a results summary.

We implemented the spectral target encoder in Python, within the scikit-learn framework. We also implemented Bayesian target encoding based on the HBBM with likelihood inference. The two encoders actually correspond in our implementation to two different `fit` methods of one same encoder, `HBBMEncoder`.

We compare the spectral target encoder to five other target encoders. Three of them, `TargetEncoder`, `GLMMEncoder` and `JamesSteinEncoder` are from the

Python library Category Encoders (McGinnis et al., 2018). `TargetEncoder` is the implementation of Bayesian target encoding as described by Micci-Barreca (2001). `GLMMEncoder` is a GLMM-based version of Bayesian target encoding. `JamesSteinEncoder` is based on the James-Stein estimator (Stein and James, 1961). We refer the reader to the library's documentation for more details on those three encoders. Another of the target encoders considered for the benchmark, `TargetRegressorEncoder`, has been recently submitted for incorporation to the Python's library scikit-learn; see `https://github.com/scikit-learn/scikit-learn/pull/17323`. This encoder is also based on the GLMM model, but without full inference of the model parameters; only the Bernoulli parameters are inferred, using Bayes estimation. We also consider `TargetEncoder` together with an optimisation by cross-validation of its `smoothing` hyper-parameter. The last encoder against which we compare the spectral target encoder is `HBBMEncoder` with likelihood inference.

The comparison is conducted on five binary classification datasets. Four of them, `Adult` `Amazon`, `Churn` and `Mortgage` are real datasets commonly used in ML benchmarking. The other one, `ToyDataset`, is a synthetic dataset. All these datasets contain categorical variables with a large number of categories. In order not to introduce any bias, we use as little preprocessing as possible, retaining all columns except those corresponding to ids.

Remark 5.1. *We give in the supplementary material a detailed description of the datasets.*

In addition to the varying datasets, we also consider in our comparisons four prominent classification algorithms, to be used in combination with the encoders: logistic regression, random forest (Breiman, 2001), multilayer perceptron (Friedman, 2001) and gradient boosting (Glorot and Bengio, 2010). We have used the scikit-learn implementations of the four algorithms (Pedregosa et al., 2011), with the default values for all of the parameters.

We perform a random split on each dataset, with 80% of the dataset's samples allocated to training. The random splitting is repeated 10 times, and we consider the mean of the corresponding 10 ROC AUC scores for each dataset, and for each combination of an encoder and a classifier.

We give in table 3 the average ranking of the encoders with respect to the ROC AUC scores. The average is first taken over the datasets, then over both the classifiers and the datasets. We observe that the spectral target encoder is the highest ranked encoder, with an average rank of 1.3. The second and third best encoders are `HBBMEncoder` with likelihood inference, and `GLMMEncoder`, respectively. Those are however much slower than the spectral target encoder (see below). The remaining encoders are in order `TargetEncoder`, `TargetRegressorEncoder`, and `JamesSteinEncoder`. They perform significantly less well than the spectral target encoder. We also observe that optimising by cross-validation the `smoothing` parameter of `TargetEncoder` slightly improves the performance of the latter, pulling it to the 4th position of the ranking table. Cross-validation is naturally more time expensive though. This empirically demonstrates that spectral inference is more efficient than cross-validation when applied to target encoding, considering either ML performance or computational time.

Concerning encoding runtimes, we observe in table 4 a wide difference between the spectral target encoder and the other two best-performing encoders, namely, `HBBMEncoder` with likelihood inference, and `GLMMEncoder`. The spectral inference is faster by several orders of magnitude. This naturally reflects the differences between spectral and likelihood inferences observed in section 4. On the other hand, we here have empirical evidence that spectral inference is mush faster than the inference method used by `GLMMEncoder`. The other encoders are on par with the spectral target encoder, although `TargetRegressorEncoder` is significantly slower on some of the datasets.

Remark 5.2. *Benchmark results for each dataset are summarised in the supplementary material. The GitHub repository accompanying this paper contains the code of the benchmark, as well as the code of* `HBBMEncoder`*. Both are in Python. The repository also contains all the benchmark results, together with additional statistics, as standard deviations.*

Table 3: Average rankings

	LR	GB	RF	MLP	total average
Spectral	1.6	1.4	1.0	1.4	1.4
HBBM-MLE	3.0	1.2	1.0	1.0	1.5
GLMM	1.6	1.6	1.6	1.6	1.6
CVTarget	3.6	3.4	1.6	2.4	2.8
Target	3.8	4.6	2.2	2.6	3.3
TargetReg	4.0	3.8	3.6	4.0	3.9
James-Stein	4.2	3.8	3.6	5.8	4.4

LR: linear regression
GB: gradient boosting
RF: random forest
MLP: multilayer perceptron

Table 4: Encoding runtimes

	AD	AM	CH	MG	TD
Spectral	1	1	1	1	1
HBBM-MLE	20	300	700	80	700
GLMM	300	80,000	5,000	900	3,000
Target	1	1	2	1	1
TargetReg	2	10	4	2	4
James-Stein	1	1	2	1	1

AD: Adult
AM: Amazon
CH: Churn
MG: Mortgages
TD: ToyDataset

Conclusion and further directions

We introduced in this paper spectral inference for the hierarchical beta-binomial model, and we established, both mathematically and empirically, that spectral inference is efficient and fast. We observed that the benefits of spectral inference were reflected when applied to target encoding. The latter fact suggests that the HBBM is a pertinent choice for modelling a binary target in the presence of categorical training data. It appears that spectral target encoding possesses the same prediction power, if not more, than target encoding based on the GLMM model, while being faster by several orders of magnitude. The authors believe that such advantage may be particularly appreciated when training time is of critical importance. Fast and powerful encoding also means more time for the selection of the downstream model in a ML system.

The present paper was exclusively concerned with binary targets. One natural and immediate extension of our work would be to adapt the spectral technique presented here to the case of a multinomial target and of a continuous target. On the mathematical side, it is remarkable that spectral inference does not rely on Gibbs sampling to achieve good convergence properties. A better mathematical qualification of those properties could help generalise the spectral technique of this paper to other models. It would be equally worth trying and generalising the optimality result of partial spectral inference (theorem 3.1) to the case of complete spectral inference.

References

J. O. Awoyemi, A. O. Adetunmbi, and S. A. Oluwadare. Credit card fraud detection using machine learning techniques: A comparative analysis. In *2017 International Conference on Computing Networking and Informatics (ICCNI)*, pages 1–9. IEEE, 2017.

L. Breiman. Random forests. *Machine learning*, 45(1): 5–32, 2001.

R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM Journal on scientific computing*, 16(5): 1190–1208, 1995.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

A. Gelman and J. Hill. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Analytical Methods for Social Research. Cambridge University Press, 2006. doi: 10.1017/CBO9780511790942.

A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian data analysis*. Boca Raton, FL: CRC Press, 2014. ISBN 978-1-4398-4095-5.

R. Giordano, W. Stephenson, R. Liu, M. Jordan, and T. Broderick. A swiss army infinitesimal jackknife. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1139–1147. PMLR, 2019.

X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

J. T. Hancock and T. M. Khoshgoftaar. Survey on categorical data for neural networks. *Journal of Big Data*, 7:1–41, 2020.

B. Lucena. Exploiting categorical structure using tree-based methods. In *International Conference on Artificial Intelligence and Statistics*, pages 2949–2958. PMLR, 2020.

W. D. McGinnis, C. Siu, S. Andre, and H. Huang. Category encoders: a scikit-learn-contrib package of transformers for encoding categorical data. *Journal of Open Source Software*, 3(21):501, 2018.

H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230, 2013.

D. Micci-Barreca. A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. *ACM SIGKDD Explorations Newsletter*, 3(1):27–32, 2001.

F. Pargent, B. Bischl, and J. Thomas. A benchmark experiment on how to encode categorical features in predictive modeling. Master's thesis, Ludwig-Maximilians-Universität München, Germany, 2019.

F. Pargent, F. Pfisterer, J. Thomas, and B. Bischl. Regularized target encoding outperforms traditional methods in supervised machine learning with high cardinality features. *arXiv preprint arXiv:2104.00629*, 2021.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

C. P. Robert, N. Chopin, and J. Rousseau. Harold Jeffreys's *Theory of probability* revisited. *Stat. Sci.*, 24(2):141–172, 2009.

C. Stein and W. James. Estimation with quadratic loss. In *Proc. 4th Berkeley Symp. Mathematical Statistics Probability*, volume 1, pages 361–379, 1961.

T. Vafeiadis, K. I. Diamantaras, G. Sarigiannidis, and K. C. Chatzisavvas. A comparison of machine learning techniques for customer churn prediction. *Simulation Modelling Practice and Theory*, 55:1–9, 2015.

P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on mathematical software (TOMS)*, 23(4):550–560, 1997.