COMPACT EMBEDDING OF BINARY-CODED INPUTS AND OUTPUTS USING BLOOM FILTERS

Joan Serrà & Alexandros Karatzoglou *
Telefónica Research
Pl. Ernest Lluch i Martín, 5
Barcelona, 08019, Spain
firstname.lastname@telefonica.com

ABSTRACT

The size of neural network models that deal with sparse inputs and outputs is often dominated by the dimensionality of those inputs and outputs. Large models with high-dimensional inputs and outputs are difficult to train due to the limited memory of graphical processing units, and difficult to deploy on mobile devices with limited hardware. To address these difficulties, we propose Bloom embeddings, a compression technique that can be applied to the input and output of neural network models dealing with sparse high-dimensional binary-coded instances. Bloom embeddings are computationally efficient, and do not seriously compromise the accuracy of the model up to 1/5 compression ratios. In some cases, they even improve over the original accuracy, with relative increases up to 12%. We evaluate Bloom embeddings on 7 data sets and compare it against 4 alternative methods, obtaining favorable results.

1 INTRODUCTION

The size of neural network models that deal with sparse inputs and outputs is often dominated by the dimensionality of such inputs and outputs. This is the case, for instance, with recommender systems, where high-dimensional sparse vectors, typically in the order from tens of thousands to hundreds of millions, constitute both the input and the output of the model (e.g., Wu et al., 2016; Hidasi et al., 2016; Cheng et al., 2016). This results in large models that present a number of difficulties, both at training and prediction stages. Apart from training and prediction times, an obvious bottleneck of such models is space: their size (and even performance) is hampered by the physical memory of graphical processing units, and they are difficult to deploy on mobile devices with limited hardware (cf. Han et al., 2016).

In this abstract, we introduce Bloom embeddings (BEs), an unsupervised embedding technique that can be applied to both input and output layers of neural network models leveraging binary (one-hot encoded) data. BEs are based on the idea of Bloom filters (Bloom, 1970), and therefore inherit part of the theory developed around that idea (Blustein & El-Maazawi, 2002; Mitzenmacher & Upfal, 2005). A BE produces a lower-dimensionality binary embedding that can be easily mapped back to the original instance. An interesting feature of BE is that the accuracy of the original model is not compromised, provided that the embedding dimension is not too low. Furthermore, in some cases, we show that training with embedded vectors can even increase prediction accuracy. BEs require no changes to the core network structure nor to the model configuration, and work with a softmax output, the most common output activation for binary-coded instances. As it is unsupervised, a BE does not require any preliminary training. Moreover, it is a constant-time operation that can be either performed on-the-fly, requiring no disk or memory space, or can be cached in random access memory, occupying orders of magnitude less space than a typical embedding matrix. Lower dimensionality of input/output vectors can result in faster training, and the mapping from the embedded space to the original one should not add an overwhelming amount of time to the prediction stage.

*An extended version of the paper was submitted to the main coference track: https://openreview.net/forum?id=rkKCdAdgx.

2 RELATED WORK

A common unsupervised approach to embed high-dimensional inputs is the hashing trick (HT; Langford et al., 2007; Shi et al., 2009). A more elementary version of it (Ganchev & Dredze, 2008) can be used at the outputs too by considering it as a special case of the proposed BE. A framework providing both encoding and decoding strategies is error-correcting output codes (ECOC; Dietterich & Bakiri, 1995), which can be adapted to class sets (Armano et al., 2012). The compressed sensing approach of Hsu et al. (2009) builds on top of ECOC to reduce multi-label regression to binary regression problems. Cissé et al. (2013) use Bloom filters for the same purpose, avoiding to tackle the full output at once. Other data-dependent approaches rely on variants of singular value decomposition (SVD) or canonical correlation analysis (CCA). For instance, Chollet (2016) has successfully applied an SVD approach to deal with binary-coded outputs for image classification.

From a more general perspective, reducing the space of (or compressing) neural network models is an active research topic (e.g. Courbariaux et al., 2015; Han et al., 2016). However, these methods typically do not focus on input layers and, to the best of our knowledge, none of them deals with high-dimensional outputs. It is also worth noting that a number of techniques have been proposed to efficiently deal with high-dimensional outputs, specially in the natural language processing domain (e.g., hierarchical or adaptive softmax). Yet, as mentioned, the focus of these works is on speed, not on space. The work of Vincent et al. (2015) focuses on both aspects of very large sparse outputs but, to the best of our knowledge, cannot be applied to traditional softmax outputs.

3 APPROACH

Given an instance $\mathbf{x}$ with dimensionality d that is binary-coded, that is $x_i \in \{0, 1\}$, and sparse, such that $\sum_{i=1}^{d} x_i = c \ll d$, we represent it as a set $\mathsf{p} = \{p_i\}_{i=1}^{c}$, where $p_i \in \mathbb{N}_{\leq d}$ is the position of such elements in $\mathbf{x}$. For every set p, we generate an embedded instance $\mathbf{u}$ of dimensionality $m < d$ by first setting all m components to 0 and then iteratively assigning

$$u_{H_j(p_i)} = 1 \tag{1}$$

for every element p_i, $i = 1, \dots c$, and every projection H_j, $j = 1, \dots k$. Projections H_j correspond to a set of k independent hash functions $\mathsf{H} = \{H_i\}_{i=1}^{k}$, each of which with a range from 1 to m, ideally distributing the projected elements uniformly at random (Mitzenmacher & Upfal, 2005).

To recover a probability-based ranking of the d elements at the output of the model, we assume a softmax activation and compute a probability vector $\hat{\mathbf{v}} = [\hat{v}_1, \dots \hat{v}_m]$ that, at training time, is compared to the binary embedding $\mathbf{v}$ of the ground truth $\mathbf{y}$. At prediction time, we can understand $\hat{\mathbf{v}}$ as a k-way factorization of every element $\hat{y}_i$ of our prediction $\hat{\mathbf{y}}$. Then, following the idea of Bloom filters, if y_i maps to v_i and $\hat{v}_i = 0$, we can confirm that that element is definitely not in the output of the model (Blustein & El-Maazawi, 2002). Otherwise, if $\hat{v}_i$ is relatively large, we want the likelihood of that element to reflect that. Specifically, given an element position q_i from q representing $\mathbf{y}$ (analogously to p and $\mathbf{x}$), we can compute the negative log-likelihood

$$L(q_i) = -\sum_{j=1}^{k} \log\left(\hat{v}_{H_j(q_i)}\right), \tag{2}$$

and assign outputs $\hat{y}_i = L(q_i)$. Note that iterating Eq. 2 for $i = 1, \dots d$ defines a ranking over the elements in $\hat{\mathbf{y}}$, which is the most common way to define (and evaluate) sparse high-dimensional outputs. From there, if needed, a probability distribution can be recovered by re-normalization.

4 RESULTS

We study the performance of BEs on 7 different publicly-available data sets which comprise a mix of recommender systems and language modeling tasks: Movielens (ML), Penn Treebank (PTB), CADE web directory, Million song data set (MSD), Amazon book reviews (AMZ), Book crossing (BC), and YooChoose RecSys challenge (YC). For each data set, we select an appropriate baseline neural network architecture. We experiment with both feed-forward (autoencoder-like) and recurrent networks, carefully selecting their parameters and configuration to match the literature results.

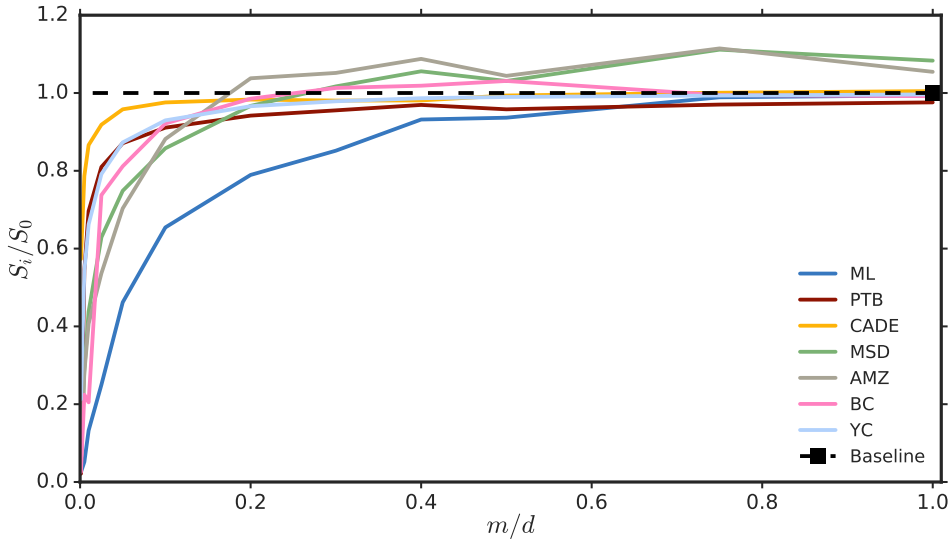

Figure 1: Score ratios S_i/S_0 as a function of dimensionality ratio m/d using $k = 4$. Qualitatively similar plots are observed for other values of k.

For every task, we compute a baseline score S_0, corresponding to running the plain neural network model without any embedding. To compare the performance across different tasks using different evaluation measures, we report the performance of the i-th task with respect to its baseline score, S_i/S_0. Similarly, to compare across different data dimensionalities, we report the ratio of embedding dimensionality m with respect to the original dimensionality d of the task, m/d.

To assess the performance of BE, we plot the performance ratio S_i/S_0 as a function of the embedding ratio m/d (Fig. 1). Firstly, we observe that score ratios approach 1 as m approaches d, indicating that the introduction of BE does not degrade the original score of the Baseline when the embedding dimension m is comparable to the original dimension d. Secondly, we observe that the lower the dimensionality ratio, the lower the score ratio. This is to be expected, as one cannot embed sets of elements with their intrinsic dimensionality to an infinitesimally small m. Importantly, the reduction of S_i/S_0 should not be linear with m/d, but should maximize S_i for low m (thus getting curves close to the top left corner of Fig. 1). We see that BE fulfills this requirement. In general, we can reduce the size of inputs and outputs 5 times ($m/d = 0.2$) and still maintain more than 92% of the value of the original score.

An additional observation is worth noting: we find that BE can improve the scores over the Baseline for a number of tasks. That is the case for 3 out of the 7 considered tasks: MSD with $m/d \geq 0.3$, AMZ with $m/d \geq 0.2$, and BC with $0.3 \leq m/d \leq 0.6$. The fact that an embedding performs better than the original Baseline has been observed for a few other methods in some specific tasks (Langford et al., 2007; Chollet, 2016). Here, depending on the task and the embedding dimension, relative increases go from 1 to 12%. We hypothesize that, in the case of BE, such increases come from having k times more active elements in the ground truth output (Sec. 3). With that, a better estimation of the gradient may be computed (larger errors that propagate back to the rest of the network).

We also compared the performance of BE against 4 different state-of-the-art methods (Sec. 2): HT, ECOC, SVD, and CCA. To do so, we selected two different embedding ratios m/d for each of the 7 tasks and computed the performance ratio S_i/S_0, resulting in 14 'test points' for every method. The results of BE were very encouraging, as it statistically significantly outperformed all the state-of-the-art methods in 10 of the test points (5 out of the 7 tasks). SVD and CCA each won on one of the remaining tasks. Nonetheless, we should notice that these approaches introduce a separate degree of supervised learning by exploiting pairwise element co-occurrences. In contrast, BE does not require any learning. It is also faster, as we only have to compute k hashes, instead of performing some kind of SVD decomposition on a potentially high-dimensional matrix. Computing a BE is an $O(ck)$ operation, where $c \ll d$ (see above) and $k \leq 5$ in all our experiments. Recovering the original ranking is an $O(dk)$ operation, thus only adding $O(k)$ overhead over non-BE operation. Overall, BEs are a fast operation that scales to large sparse data sets.

REFERENCES

G. Armano, C. Chira, and N. Hatami. Error-correcting output codes for multi-label text categorization. In *Proc. of the Italian Information Retrieval Conf. (IIR)*, pp. 26–37, 2012.

B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

J. Blustein and A. El-Maazawi. Bloom filters – a tutorial, analysis, and survey. Technical report, Faculty of Computer Science, Dalhousie University, Halifax, Canada, 2002. URL `https://www.cs.dal.ca/research/techreports/cs-2002-10`.

H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah. Wide & deep learning for recommender systems. In *Proc. of the Workshop on Deep Learning for Recommender Systems (DLRS)*, pp. 7–10, 2016.

F. Chollet. Information-theoretic label embeddings for large-scale image classification. ArXiv: 1607.05691, 2016.

M. Cissé, N. Usunier, T. Artières, and P. Gallinari. Robust Bloom filters for large multilabel classification tasks. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1851–1859. 2013.

M. Courbariaux, Y. Bengio, and J.-P. David. BinaryConnect: training deep neural networks with binary weights during propagations. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems (NIPS)*, pp. 3123–3131. 2015.

T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

K. Ganchev and M. Dredze. Small statistical models by random feature mixing. In *ACL Workshop on Mobile Language Processing (MLP)*, pp. 19–20, 2008.

S. Han, H. Mao, and W. J. Dally. Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2016.

B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. Session-based recommendations with recurrent neural networks. In *Proc. of the Int. Conf. on Learning Representations (ICLR)*, 2016. URL `https://arxiv.org/abs/1511.06939`.

D. J. Hsu, S. M. Kakade, J. Langford, and T. Zhang. Multi-label prediction via compressed sensing. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta (eds.), *Advances in Neural Information Processing Systems (NIPS)*, volume 22, pp. 772–780. 2009.

J. Langford, L. Li, and A. Strehl. Vowpal wabbit online learning project. Technical report, 2007. URL `http://hunch.net/?p=309`.

M. Mitzenmacher and E. Upfal. *Probability and computing: randomized algorithms and probabilistic analysis*. Cambridge University Press, Cambridge, UK, 2005.

Q. Shi, J. Petterson, G. Dror, J. Langford, A. Smola, and S. V. N. Vishwanathan. Hash kernels for structured data. *Journal of Machine Learning Research*, 10:2615–2637, 2009.

P. Vincent, A. Brébisson, and X. Bouthilier. Efficient exact gradient update for training deep networks with very large sparse targets. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1108–1116. 2015.

Y. Wu, C. DuBois, A. X. Zheng, and M. Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *Proc. of the ACM Int. Conf. on Web Search and Data Mining (WSDM)*, pp. 153–162, 2016.