Learning Distance Estimators from Pivoted Embeddings of Metric Objects

Fabio Carrara(✉), Claudio Gennaro, Fabrizio Falchi, and Giuseppe Amato

ISTI-CNR, via G. Moruzzi 1, 56124 Pisa, Italy
{fabio.carrara,claudio.gennaro,fabrizio.falchi,
giuseppe.amato}@isti.cnr.it

Abstract. Efficient indexing and retrieval in generic metric spaces often translate into the search for approximate methods that can retrieve relevant samples to a query performing the least amount of distance computations. To this end, when indexing and fulfilling queries, distances are computed and stored only against a small set of reference points (also referred to as pivots) and then adopted in geometrical rules to estimate real distances and include or exclude elements from the result set. In this paper, we propose to learn a regression model that estimates the distance between a pair of metric objects starting from their distances to a set of reference objects. We explore architectural hyper-parameters and compare with the state-of-the-art geometrical method based on the n-simplex projection. Preliminary results show that our model provides a comparable or slightly degraded performance while being more efficient and applicable to generic metric spaces.

Keywords: Distance estimation · Metric spaces · Regression · Deep neural networks · Pivoted embeddings

1 Introduction

Thanks to the impetus given by recent developments in deep learning, machine learning has gained unprecedented popularity and spread into unimaginable number domains of computing. Perhaps one of the most unexpected application domains is that of index structures. In an exploratory research paper, Kraska et al. [8] showed how machine learning models, including deep learning ones, can fully or partially replace existing index structures, such as B-Tree or Bloom filters. This work paved the way for a whole new area of research in index structure, termed *learned index*. The core idea of learned indexes is to obtain a more compact index representation or performance gains by learning from data distribution.

In particular, Kraska et al. show that an index can be seen as a model f that predicts the position y of a record x, i.e. $y = f(x)$. Seen with the eyes of machine learning, this problem is known as a regression problem. Therefore, in

© Springer Nature Switzerland AG 2020
S. Satoh et al. (Eds.): SISAP 2020, LNCS 12440, pp. 361–368, 2020.
https://doi.org/10.1007/978-3-030-60936-8_28

this context a learned index is simply an ML model (such as linear regression or a neural network) that replaces a B-Tree and predicts the position of query key. In this work, we would like to generalize the concept of learned index by providing a broader definition, in which the regression allows us to predict the representation of metric objects in vector form from the knowledge of distances from some anchors. This type of transformation of data is useful in the problems of searching in large-scale metric spaces, where a compromise between accuracy and speed of response to queries is often required. In large-scale scenarios, the amount of distance computations between objects needed for an exact search tends to saturate the available computational budget for obtaining reasonable response times, considering also that in metric spaces, distance functions are often expensive to compute.

The idea of reconstructing the distance between any pair of objects in a metric space by exploiting distances with a group of reference objects was probably first addressed in [9]. The authors proposed an *embedding* into another metric space where it is possible to deduce upper and lower bounds on the actual distance of any pair of objects. Connor et al. [4–6] observed that for a large class of metric spaces, distances to a set of n pivots can be used to project the data objects into a n-dimensional Euclidean space such that in the projected space the Euclidean distance between any two points is an upper or lower bound of the actual distance. They called this approach *n-Simplex projection*, and they proved that it can be used in all the metric spaces meeting the *n-point property* [2]. As also pointed out in [3], many common metric spaces meet the desired property, like Cartesian spaces of any dimension with the Euclidean, cosine or quadratic form distances, probability spaces with the Jenson-Shannon or the Triangular distance, and more generally any Hilbert-embeddable space [2,10]. This approach has recently been used in an inverted index for approximate research on the n-nearest neighbors to obtain an estimate of the real distances of the objects present in the results of a query [12].

We intend to develop a similar approach to the n-simplex projection, however instead of using a handcrafted deterministic algorithm, in this paper we test a machine learning approach based on deep neural networks to probe their capabilities in this context. The rest of the paper is structured as follows. Section 2 describes the general idea of method developed and the model used. Section 3 presents the dataset and experimental evaluation. Section 4 concludes.

2 Method

2.1 Model Definition

Let $\mathcal{X}$ a metric space with distance function $d : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^+$ and $\mathcal{P} = \{ p_i \in \mathcal{X} : i = 1 \dots N \}$ a set of reference points (or pivots) in $\mathcal{X}$. We define the *pivoted embedding* $e(x, \mathcal{P})$ of an object $x \in \mathcal{X}$ w.r.t $\mathcal{P}$ as an N-dimensional real-valued vector where the i-th component is the distance between x and the i-th pivot, i.e.

$$e(x, \mathcal{P}) = [\, d(x, p_1), \dots, d(x, p_i), \dots, d(x, p_N) \,] \in \mathbb{R}^N \,. \tag{1}$$

We are interested in estimating the distance $d(x, y)$ between a pair of objects $x, y \in \mathcal{X}$ given their pivoted embeddings $\mathbf{e}_x = e(x, \mathcal{P}), \mathbf{e}_y = e(y, \mathcal{P})$ with respect to a common set of reference objects $\mathcal{P}$. We formulate this task as a regression problem: we define a parametric model f that outputs the estimated distance given the pivoted embeddings and optimize its parameters on a training set via gradient descent. In addition to $\mathbf{e}_x$ and $\mathbf{e}_y$, we include the distances between pivots $\{\, d(p_i, p_j) : i = 1 \dots N, j = 1 \dots N, i < j \,\}$ in the inputs of our model as a real-valued vector $\mathbf{p} \in \mathbb{R}^{\frac{N(N-1)}{2}}$ is commonly computed once offline and available. Formally,

$$\tilde{d}(x, y) = f(\mathbf{e}_x, \mathbf{e}_y, \mathbf{p}; \theta)\,, \tag{2}$$

where $\tilde{d}(x, y)$ indicates the estimate for $d(x, y)$ of the model f having parameters θ. Following common practice in metric learning, we define

$$f(\mathbf{e}_x, \mathbf{e}_y, \mathbf{p}; \theta) = |\Phi(\mathbf{e}_x, \mathbf{p}; \theta) - \Phi(\mathbf{e}_y, \mathbf{p}; \theta)|_2\,, \tag{3}$$

where $\Phi(\mathbf{e}, \mathbf{p}; \theta)$ is a neural network that takes as input a pivoted embedding $\mathbf{e}$ and the distances between pivots $\mathbf{p}$ and outputs a real-valued vector representation.

As the architecture of $\Phi(\mathbf{e}, \mathbf{p}; \theta)$, we choose a two-branch fully-connected residual network: $\mathbf{e}$ and $\mathbf{p}$ are independently processed by two MLPs with residual connections whose outputs are then merged by concatenation and followed by one or more additional fully-connected layer. Each branch comprises multiple residual blocks [7] having the structure reported in Fig. 1a. We explore and evaluate architectural hyperparameters such as depth and branch merging point in Sect. 3.

2.2 Model Training

We train our model with mini-batch gradient descent. Given a training set $\mathcal{X}_{\text{tr}} \subset \mathcal{X}$, to form a training batch we randomly draw N objects as pivots $\mathcal{P}$ and B pairs of objects $(x_i, y_i), i = 1 \dots B, x_i, y_i \in \mathcal{X}_{\text{tr}}$, and we adopt the original metric distance d to obtain the inputs (the pivoted embedding of the objects $\mathbf{e}_{x_i}, \mathbf{e}_{y_i}$ and distances between pivots $\mathbf{p}$) and the target (exact distances between objects $d(x_i, y_i)$) of our model. We optimize the loss function

$$\mathcal{L} = \frac{1}{B} \sum_{i=1}^{B} \text{SmoothL1}\left(f(\mathbf{e}_{x_i}, \mathbf{e}_{y_i}, \mathbf{p}),\, d(x_i, y_i) \right), \tag{4}$$

with

$$\text{SmoothL1}(a, b) = \begin{cases} \frac{1}{2}(a-b)^2, & \text{if } |a-b| < 1 \\ |a-b| - \frac{1}{2}, & \text{otherwise}\,. \end{cases} \tag{5}$$

We choose the SmoothL1 function to avoid huge gradients in the early phase of training that often lead to numerical instabilities. At each batch, we sample new pivots and couples and repeat the procedure. We periodically evaluate our model on a test set $\mathcal{X}_{\text{ts}}$ by measuring the estimation error, and we adopt early stopping when the test error stops decreasing.

3 Experiments

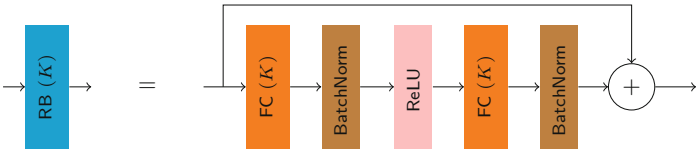
3.1 Dataset

Throughout all experiments, we adopt a subset of the YFCC100M-HNfc6 deep features dataset [1]—a dataset of 100M 4096-dimensional features extracted from YFCC100M [11] images using the HybridCNN [13] deep convolutional pretrained network and selecting the output of the *fc6* layer. In the original space, features are compared with the L_2 distance, i.e. $d = L_2$. We select the first 1M features and divide them in training, validation and test sets with a 750K/150K/100K split. As a metric of performance, we report the mean absolute error and the mean absolute percentage error (MAPE) computed on the test set together with their standard deviations.

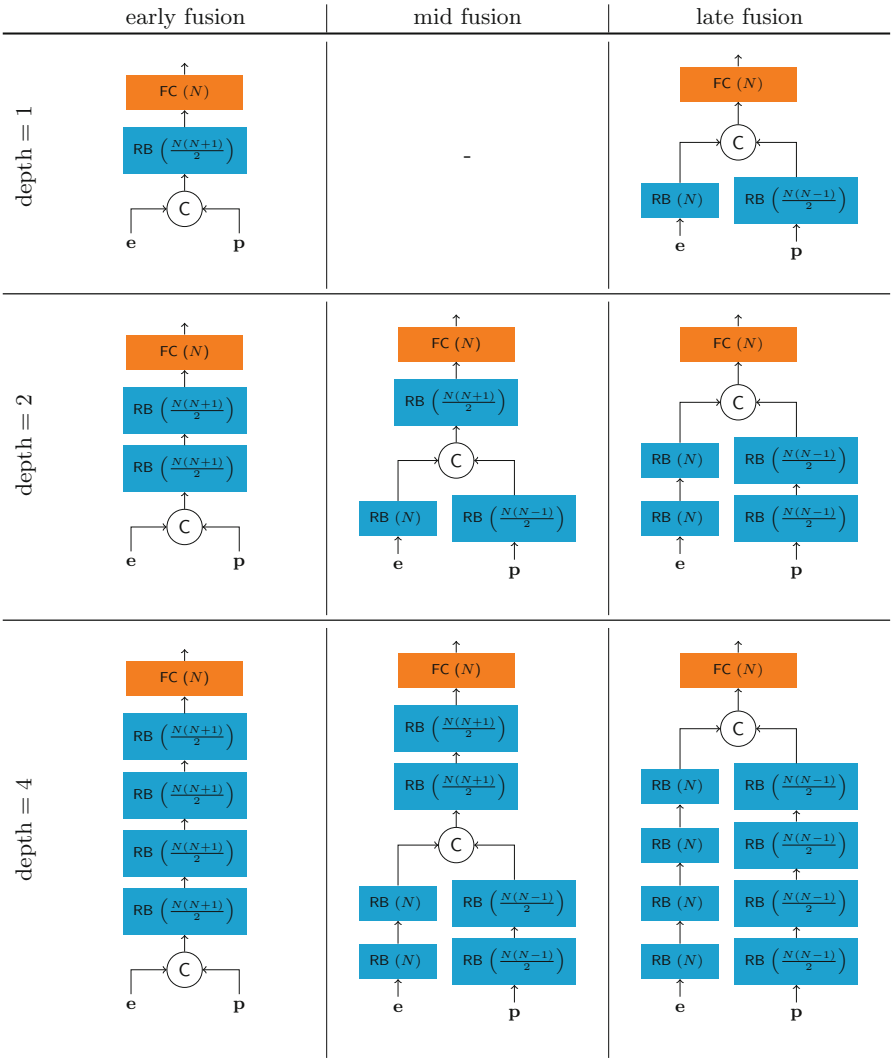
3.2 Choice of the Φ Network

We perform experiments to evaluate two main architectural hyperparameters of Φ—the *depth* of the network and the *fusion strategy* of the two branches. For the former, we test a number of intermediate layers in $\{1, 2, 4\}$, while for the latter, we test concatenation of the two branches at the input level (*early fusion*), at half depth (*mid fusion*), and right before the final layer (*late fusion*). Figure 1 depicts the tested architectures for each parameter combination. We decided not to apply any bottleneck layer to reduce the dimensionality of the input, thus every layer keeps the dimensionality of its input except for the last projection. We are aware this leads to prohibitive memory requirements as N increases, as the dimensionality of $\mathbf{p}$ is $O(N^2)$, but in this preliminary phase, this reduces the architectural search space and enables us to evaluate the model without introducing performance caps. We train all models with SGD with momentum 0.9, learning rate of 0.05 (divided by 10 when the validation loss plateaus), batch size 100 for 10K iterations, validating every 100 iterations. We adopt early stopping by monitoring the MAPE on the validation set and selecting the model reaching the minimum[1]. On our single-GPU configuration, we were able to test all variations with N up to 128, as larger values are prohibitively expensive in terms of GPU memory required for training (+10GB); we left the exploration of larger values with reduced models to future work. Results in terms of MAPE and MAE are reported in Table 1. We notice that on average, an early fusion strategy is able to reach slightly better results with more performance gains with deeper networks. On the other hand, deeper networks with other fusion strategy suffer from numerical instabilities leading to divergence; we left to future work the tuning of optimization hyperparameters that may alleviate this phenomenon. Among shallower (and thus more efficient) ones, the model with late fusion and depth = 1 shows good overall performance on all tested N values with respect to other variants.

[1] The code for reproducing our results is available at https://github.com/fabiocarrara/pivoted-estimation.



(a) Composition of a Residual Block. Output dimensionality is reported in brackets for fully connected layers.



(b) Architectural variations tested. The output dimensionality (K in Fig. (1a)) of each layer is reported in brackets. C indicates concatenation.

Fig. 1. Explored architectures for $\Phi(\mathbf{e}, \mathbf{p})$

Table 1. Performance of architectural variations of our model on YFCC100M-HNfc6 features test subset. **Bold entries** indicate the model reaching the best mean absolute percentage error (MAPE) for each N. Dashes (-) indicate configurations that do not converge.

		N						
arch		2	4	8	16	32	64	128
early	1	$36.7_{\pm 24.3}$	$28.1_{\pm 20.1}$	$23.0_{\pm 16.4}$	$18.4_{\pm 13.1}$	$16.4_{\pm 12.1}$	$16.1_{\pm 12.1}$	$14.8_{\pm 11.3}$
	2	$36.9_{\pm 24.7}$	$27.2_{\pm 20.5}$	$22.8_{\pm 17.1}$	$19.2_{\pm 14.3}$	$18.2_{\pm 14.2}$	$17.9_{\pm 16.7}$	$18.0_{\pm 14.3}$
	4	$36.3_{\pm 25.0}$	$\mathbf{25.0_{\pm 20.0}}$	$\mathbf{18.7_{\pm 15.2}}$	$\mathbf{14.0_{\pm 11.7}}$	$\mathbf{12.7_{\pm 11.5}}$	$\mathbf{12.9_{\pm 11.8}}$	$15.3_{\pm 15.7}$
mid	2	$37.4_{\pm 26.2}$	$39.6_{\pm 28.2}$	$33.8_{\pm 26.1}$	$25.9_{\pm 18.8}$	$32.9_{\pm 30.1}$	$30.0_{\pm 29.2}$	$31.1_{\pm 28.1}$
	4	$39.1_{\pm 26.4}$	-	-	-	-	-	-
late	1	$37.1_{\pm 24.9}$	$27.0_{\pm 19.9}$	$21.3_{\pm 15.8}$	$17.4_{\pm 12.4}$	$15.6_{\pm 11.4}$	$13.6_{\pm 10.3}$	$\mathbf{12.9_{\pm 9.9}}$
	2	$\mathbf{35.6_{\pm 24.7}}$	$25.7_{\pm 21.9}$	$48.8_{\pm 46.1}$	$88.9_{\pm 70.9}$	-	-	-
	4	$56.3_{\pm 36.3}$	-	-	-	-	-	-

(a) Absolute Percentage Error (%, mean and standard deviation)

		N						
arch		2	4	8	16	32	64	128
early	1	$0.46_{\pm 0.30}$	$0.35_{\pm 0.25}$	$0.29_{\pm 0.20}$	$0.23_{\pm 0.16}$	$0.21_{\pm 0.15}$	$0.20_{\pm 0.15}$	$0.19_{\pm 0.14}$
	2	$0.46_{\pm 0.31}$	$0.34_{\pm 0.25}$	$0.29_{\pm 0.21}$	$0.24_{\pm 0.18}$	$0.23_{\pm 0.18}$	$0.23_{\pm 0.21}$	$0.23_{\pm 0.18}$
	4	$0.46_{\pm 0.31}$	$\mathbf{0.32_{\pm 0.25}}$	$\mathbf{0.23_{\pm 0.19}}$	$\mathbf{0.18_{\pm 0.15}}$	$\mathbf{0.16_{\pm 0.14}}$	$\mathbf{0.16_{\pm 0.15}}$	$0.19_{\pm 0.19}$
mid	2	$0.47_{\pm 0.33}$	$0.50_{\pm 0.36}$	$0.43_{\pm 0.33}$	$0.33_{\pm 0.24}$	$0.42_{\pm 0.39}$	$0.38_{\pm 0.37}$	$0.39_{\pm 0.35}$
	4	$0.49_{\pm 0.33}$	-	-	-	-	-	-
late	1	$0.47_{\pm 0.31}$	$0.34_{\pm 0.25}$	$0.27_{\pm 0.20}$	$0.22_{\pm 0.15}$	$0.20_{\pm 0.14}$	$0.17_{\pm 0.13}$	$\mathbf{0.16_{\pm 0.12}}$
	2	$\mathbf{0.45_{\pm 0.31}}$	$0.32_{\pm 0.27}$	$0.62_{\pm 0.57}$	$1.13_{\pm 0.89}$	-	-	-
	4	$0.71_{\pm 0.46}$	-	-	-	-	-	-

(b) Absolute Error (mean and standard deviation)

3.3 Comparison with the State of the Art

We compare in Fig. 2 our models selected from Table 1 (the best for each N) with the *n-Simplex projection* on the same dataset. The n-Simplex projection provides geometrical upper (simplex-U) and lower (simplex-L) bounds for distance estimates in super-metric spaces given **e** and **p**. We also report the estimation obtained by averaging the upper and lower bound (simplex-M) that usually provides a finer estimate. A main drawback of this technique is the iterative $O(N^3)$ simplex building procedure that needs to be executed for each value of **p** we are willing to use. Our approach provides a comparable or slightly degraded performance, but once trained, it can cope with different values of **p** without the need of expensive procedures. Moreover, our approach can be applied to generic metric spaces.

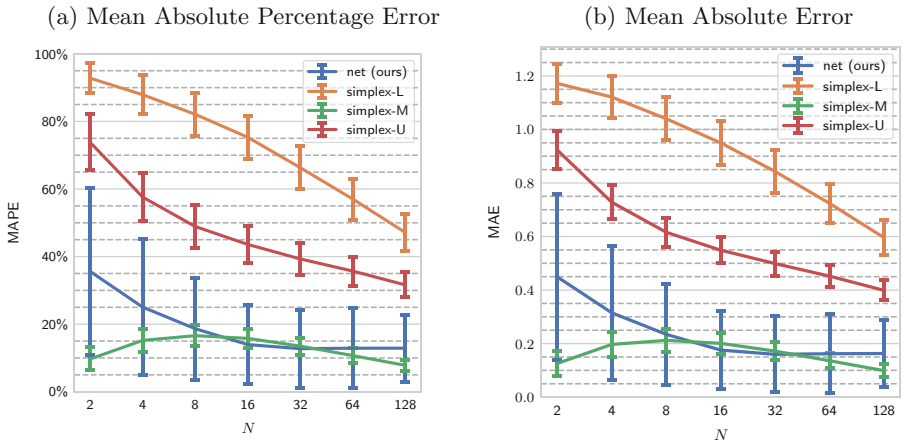


Fig. 2. Comparison with the state-of-the-art n-Simplex estimator: simplex suffixes -U, -L, and -M indicate estimation using respectively the upper bound, lower bound, and their mean.

4 Conclusion

We explored the use of neural regressors for estimating distances from pivoted embedding in generic metric spaces. Preliminary experiments on deep-learned image descriptors suggest that the proposed approach can be used in approximated regimes providing a performance comparable to exact geometrical bounds while being more efficient. Moreover, our formulation is not limited to supermetric spaces and can be applied seamlessly to different set of reference points—properties that pave the way to advanced indexing structures including dynamically chosen reference points sets. Future work comprises the development of more compact architectures for higher dimensionalities and extended experimentation on additional metric and retrieval datasets.

Acknowledgment. This work was partially supported by "Automatic Data and documents Analysis to enhance human-based processes" (ADA, CUP CIPE D55F1 7000290009) and the AI4EU project (funded by the EC, H2020 - Contract n. 825619). We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Tesla K40 GPU used for this research.

References

1. Amato, G., Falchi, F., Gennaro, C., Rabitti, F.: YFCC100M hybridnet fc6 deep features for content-based image retrieval. In: Proceedings of the 2016 ACM Workshop on Multimedia COMMONS, pp. 11–18 (2016)
2. Blumenthal, L.M.: Theory and Applications of Distance Geometry. Clarendon Press, Oxford (1953)

3. Connor, R., Cardillo, F.A., Vadicamo, L., Rabitti, F.: Hilbert exclusion: improved metric search through finite isometric embeddings. ACM Trans. Inf. Syst. **35**(3), 17:1–17:27 (2016). https://doi.org/10.1145/3001583
4. Connor, R., Vadicamo, L., Cardillo, F.A., Rabitti, F.: Supermetric search with the four-point property. In: Amsaleg, L., Houle, M.E., Schubert, E. (eds.) SISAP 2016. LNCS, vol. 9939, pp. 51–64. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46759-7_4
5. Connor, R., Vadicamo, L., Cardillo, F.A., Rabitti, F.: Supermetric search. Inf. Syst. **80**, 108–123 (2019)
6. Connor, R., Vadicamo, L., Rabitti, F.: High-dimensional simplexes for supermetric search. In: Beecks, C., Borutta, F., Kröger, P., Seidl, T. (eds.) Similarity Search and Applications, Proceedings of SISAP 2017, pp. 96–109. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68474-1_7
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
8. Kraska, T., Beutel, A., Chi, E.H., Dean, J., Polyzotis, N.: The Case for Learned Index Structures. Association for Computing Machinery, New York (2018)
9. Micó, M.L., Oncina, J., Vidal, E.: A new version of the nearest-neighbour approximating and eliminating search algorithm (AESA) with linear preprocessing time and memory requirements. Pattern Recogn. Lett. **15**(1), 9–17 (1994). https://doi.org/10.1016/0167-8655(94)90095-7
10. Schoenberg, I.J.: Metric spaces and completely monotone functions. Ann. Math. **39**(4), 811–841 (1938)
11. Thomee, B., et al.: YFCC100M: the new data in multimedia research. Commun. ACM **59**(2), 64–73 (2016)
12. Vadicamo, L., Gennaro, C., Falchi, F., Chávez, E., Connor, R., Amato, G.: Re-ranking via local embeddings: a use case with permutation-based indexing and the nSimplex projection. Inf. Syst. 101506 (2020)
13. Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., Oliva, A.: Learning deep features for scene recognition using places database. In: Advances in Neural Information Processing Systems, pp. 487–495 (2014)