Pattern Recognition Letters 94 (2017) 1–6

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Scalable out-of-sample extension of graph embeddings using deep neural networks

Aren Jansen, Gregory Sell*, Vince Lyzinski

Human Language Technology Center of Excellence and the Center for Language and Speech Processing, Johns Hopkins University, Baltimore, MD, 21211 USA

ARTICLE INFO

Article history:
Received 16 June 2016
Available online 26 April 2017

Keywords:
Deep neural networks
Out-of-sample extension
Graph embedding
Manifold

ABSTRACT

Several popular graph embedding techniques for representation learning and dimensionality reduction rely on performing computationally expensive eigendecompositions to derive a nonlinear transformation of the input data space. The resulting eigenvectors encode the embedding coordinates for the training samples only, and so the embedding of novel data samples requires further costly computation. In this paper, we present a method for the out-of-sample extension of graph embeddings using deep neural networks (DNNs) to parametrically approximate these nonlinear maps. Compared with traditional nonparametric out-of-sample extension methods, we demonstrate that the DNNs can generalize with equal or better fidelity and require orders of magnitude less computation at test time. Moreover, we find that unsupervised pretraining of the DNNs improves optimization for larger network sizes, thus removing sensitivity to model selection.

1. Introduction

Manifold learning is a popular data analysis framework that attempts to recover compact low-dimensional embeddings of high-dimensional datasets. Several manifold learning algorithms—including ISOMAP [30], locally linear embedding [24,25], diffusion maps [7], and Laplacian eigenmaps [1]—derive coordinate representations that encode the local neighborhood structure of an unlabeled data sample. In these approaches, data is first mapped into a graph, with each data point acting as a node and edge weights between the nodes determined from some affinity function. This entire graph is then embedded into a Euclidean space, resulting in a new representation of the original data. For the remainder of this paper, we will refer to this process as *graph embedding*. These techniques have found considerable success in a wide array of application domains, including computer vision [8,13,22], speech processing [16–18,21,26,31], and natural language processing [27,28]. In [32], it was shown that these algorithms are all members of a more general graph embedding framework, in which the transformations are derived via a generalized eigendecomposition of the graph Laplacian matrix operator for algorithm-specific graph construction methodologies.

In their basic form, these graph embedding techniques only provide transformations of the training samples used to construct the graph. Thus, even if a large training set is used, computing the output of the estimated map for a novel test sample is not possible. To address this shortcoming, a nonparametric out-of-sample extension technique based on Nyström sampling was developed that leverages the input and target representation pairs for each training sample to approximate what the map would have generated for an arbitrary test point [5,20]. While generally effective, the Nyström extension is a kernel-based method with time complexity that scales linearly with the number of training samples. This increase in computational cost is especially problematic because manifold methods are most effective when provided the benefit of large training sets for representation learning. It would be highly beneficial to remove this trade-off between representation quality and extension feasibility with a more efficiently scaleable method for out-of-sample extension.

Neural networks have long been known to be a powerful learning framework for classification and regression, capable of distilling large training sets into efficiently evaluated parametric models, and thus are a natural choice for modeling manifold embeddings. In their seminal paper, Hornik *et al.* [14] proved that feedforward neural networks can approximate a virtually arbitrary deterministic map between high-dimensional spaces, indicating that they would also be ideally suited for our out-of-sample extension problem. However, there are two caveats for the use of neural networks as universal approximators: (i) there must be sufficient hidden units (i.e. sufficient model parameters), which in turn require additional data samples for training without overfitting; and (ii) the non-convexity of the objective function grows with the num-

* Corresponding author.
E-mail address: gsell@jhu.edu (G. Sell).

http://dx.doi.org/10.1016/j.patrec.2017.04.016
0167-8655/

ber of model parameters, making the search for reliable global solutions increasingly difficult.

With these considerations in mind, we explore the application of recent advances in deep neural network (DNN) training methodology to the out-of-sample extension problem. First, by stabilizing the Lanczos eigendecomposition algorithm, we are able to produce exact graph embeddings for training sets with millions of data samples. This permits an extensive study with deeper architectures than have been previously considered for the task. Second, motivated by success in the supervised classification setting [3,4], we consider unsupervised DNN pretraining procedures to improve optimization as our larger training samples support commensurate increases in model complexity.

In the work that follows, we compare the performance of our parametric DNN approach against a Nyström sampling baseline, both in terms of approximation fidelity and test runtime. We find DNNs to match or outperform the approximation fidelity of the Nyström method for all training sample sizes. Furthermore, since the DNN approach is parametric, its test-time complexity for fixed network size is constant in the training sample size, producing orders-of-magnitude speedup over Nyström sampling for larger training sizes.

The remainder of this paper is organized as follows. We begin with an overview of prior work in out-of-sample extension for graph embeddings. We then describe the strategy for stabilizing eigendecompositions for large training sets, followed by a description of the process for training our DNN out-of-sample extension to approximate the embedding for unseen data. Finally, we analyze the reconstruction accuracy and computation speed of both the Nyström baseline and the DNN approach.

2. Prior work

The most popular methods for extending graph embeddings to unseen data have been based on Nyström sampling [5,20], and thus they will serve as the baseline in our experiments. This is a nonparametric, kernel-based technique that approximates the embedding of each test sample by computing a weighted interpolation of the embeddings for training samples that were nearby in the original input space. Formally (see [5] for details), let $X = \{x_1, \ldots, x_n\}$ be the set of training samples on which we will learn the graph embedding, where each $x_i \in \mathbb{R}^d$. These data points in X are be used to build a graph whose edge weights are derived from some affinity function, and that graph is then embedded via eigendecomposition of the graph Laplacian. Formally, let $\mathcal{L}$ be the symmetric, normalized graph Laplacian operator defined for the set X such that $\mathcal{L} = I - D^{-1/2}AD^{-1/2}$, where $A_{i,j} = K(x_i, x_j)$ for some positive semidefinite kernel function K that must be specialized for each specific graph embedding algorithm; and D is the diagonal matrix defined via $D_{i,i} = \sum_j A_{i,j}$. Let the spectral decomposition of the normalized Laplacian be denoted as $\mathcal{L} = U\Sigma U^T$, where the diagonal entries of $\Sigma = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ are non-increasing. The d'-dimensional embedding of X is then provided by the first d' columns of U, which we shall denote as $U^{(d')}$. Stated simply, the embedding of x_i is given by the i-th row of $U^{(d')}$.

To embed an out-of-sample data point $x \in \mathbb{R}^d$ via the Nyström extension, the p-th dimension of the extension $y_p(x)$ is given by

$$y_p(x) = \frac{1}{\lambda_p} \sum_{i=1}^{n} U_{i,p}^{(d')} K(x, x_i). \tag{1}$$

We see that the complexity of this extension is linear in the size of the training set. In practice, approximate nearest neighbor techniques can be used to speed this up with minimal loss in fidelity (the implementation we benchmark uses k-d tree for this purpose), but the algorithmic complexity still increases with training size. Finally, note that a nearly equivalent formulation based on reproducing Kernel Hilbert space theory was presented in [2], where the kernelization was introduced into the objective function before the eigendecomposition is performed. This formulation has the same scalability limitations as the Nyström extension. These computational difficulties motivate our exploration of DNNs to model embeddings for out-of-sample extension.

Traditional (shallow) neural networks have also been considered for out-of-sample extension in the past in two limited studies involving small datasets and model architectures [6,10]. The idea was introduced in [10], but the study failed to include a meaningful quantitative evaluation to measure the effectiveness of the learned mapping as compared to more typical out-of-sample extensions like the Nyström extension. The experiments in [6], which predated the advent of recent deep learning training methodologies, found neural networks to be one of the worst performing methods. However, with a similar motive of computational efficiency, [12] explored the use of DNNs for approximating expensive sparse coding transformations and produced more compelling results.

3. A scalable out-of-sample extension

A truly scalable out-of-sample extension must simultaneously consume a large amount of training data for detailed modeling and provide a test-time complexity that does not strongly depend on that training set size. The nonparametric nature of the Nyström method leads to a linear dependence on the training set size (logarithmic if kernel approximations are implemented) and thus can get bogged down as we feed more data to the graph embedding training. We begin this section with a simple trick for the eigendecomposition of large graph Laplacians, which permits larger training sets and motivates the need for more computationally efficient extension methods. This is followed by a presentation of the deep neural network architecture we propose to efficiently extend the embedding to arbitrary test points.

3.1. Stabilizing the eigendecomposition

In [9], it is suggested that the stability of the Lanczos eigendecomposition algorithm can be greatly increased (and memory requirements consequently reduced) by reformulating the eigenproblem to recover the largest eigenvalues. We can exploit this by observing that if v is an eigenvector of $\mathcal{L}$ with eigenvalue λ, then v is also an eigenvector of $\widetilde{\mathcal{L}} = I - \mathcal{L}$ with eigenvalue $1 - \lambda$ (which is guaranteed to be less than or equal to 1). Thus, with this small redefinition of the eigenproblem, we can recover the same eigenvectors by considering the largest eigenvalue criterion. Note that when using the ARPACK implementation, a similar effect can also be accomplished by searching for the smallest *algebraic* eigenvalues of $\mathcal{L}$ directly.

While this trick is by no means a fundamental theoretical innovation on our part, its effects have proven dramatic. Our past efforts to solve for the smallest magnitude eigenvalues of the graph Laplacian exceeded our hardware memory limits when our graphs reached the order of 100,000 nodes and 1 million edges. Employing this simple trick, we have now succeeded in processing graphs with order 100 million nodes and order 10 billion edges on conventional hardware, stably solving for the top 100 eigenvectors in a few days using 32 cores and 0.5 TB of RAM. This problem size even exceeds what was reported using approximate singular value decomposition solvers in the past [29]. For the 1.5 million node graphs we consider in our experiments described below, this method was more than adequate for our (offline) embedding training needs.