Journal of the American Statistical Association

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/uasa20

Embedding Learning

Ben Dai, Xiaotong Shen & Junhui Wang

To cite this article: Ben Dai, Xiaotong Shen & Junhui Wang (2022) Embedding Learning, Journal of the American Statistical Association, 117:537, 307-319, DOI: 10.1080/01621459.2020.1775614

To link to this article: https://doi.org/10.1080/01621459.2020.1775614

JOURNAL OF THE AMERICAN STATISTICAL ASSOCIATION
2022, VOL. 117, NO. 537, 307–319: Theory and Methods
https://doi.org/10.1080/01621459.2020.1775614

Embedding Learning

Ben Dai[a], Xiaotong Shen[a], and Junhui Wang[b]

[a]School of Statistics, University of Minnesota, Minneapolis, MN; [b]School of Data Science, City University of Hong Kong, Kowloon, Hong Kong

ABSTRACT

Numerical embedding has become one standard technique for processing and analyzing unstructured data that cannot be expressed in a predefined fashion. It stores the main characteristics of data by mapping it onto a numerical vector. An embedding is often unsupervised and constructed by transfer learning from large-scale unannotated data. Given an embedding, a downstream learning method, referred to as a two-stage method, is applicable to unstructured data. In this article, we introduce a novel framework of embedding learning to deliver a higher learning accuracy than the two-stage method while identifying an optimal learning-adaptive embedding. In particular, we propose a concept of *U*-minimal sufficient learning-adaptive embeddings, based on which we seek an optimal one to maximize the learning accuracy subject to an embedding constraint. Moreover, when specializing the general framework to classification, we derive a graph embedding classifier based on a hyperlink tensor representing multiple hypergraphs, directed or undirected, characterizing multi-way relations of unstructured data. Numerically, we design algorithms based on blockwise coordinate descent and projected gradient descent to implement linear and feed-forward neural network classifiers, respectively. Theoretically, we establish a learning theory to quantify the generalization error of the proposed method. Moreover, we show, in linear regression, that the one-hot encoder is more preferable among two-stage methods, yet its dimension restriction hinders its predictive performance. For a graph embedding classifier, the generalization error matches up to the standard fast rate or the parametric rate for linear or nonlinear classification. Finally, we demonstrate the utility of the classifiers on two benchmarks in grammatical classification and sentiment analysis. Supplementary materials for this article are available online.

ARTICLE HISTORY
Received December 2018
Accepted May 2020

KEYWORDS
Automatic feature generation; Deep learning; Natural language processing; Representational learning; Sentimental analysis; Text mining

1. Introduction

Numerical embedding has become an essential part of modern data analysis, particularly for unstructured data that cannot be organized in predefined structures, including texts, graphs, videos, objects, and chemical compounds. It maps unstructured data onto lower-dimensional numerical vectors while retaining key characteristics of the original unstructured data. Then the numerical vectors can be used by any downstream methods that are designed for numerical data to facilitate unstructured data analysis, which would be otherwise rather difficult. This strategy has been widely adopted, such as strings of plain text in sentiment analysis, proteins in biological state prediction, and search tokens and users' watch history in *YouTube* video recommendation.

Numerical embedding learns numerical features from large external unannotated data such as relational and knowledge graphs based on certain principles. For instance, *Word2Vec* (Pratt 1993; Mikolov et al. 2013; Mikolov, Yih, and Zweig 2013) constructs embedding vectors through a skip-gram model by maximizing the conditional likelihood of a word given other surrounding words based on a large external unannotated text corpus. Local linear embedding (LLE; Roweis and Saul 2000) and Laplacian eigenmaps (LAE; Belkin and Niyogi 2002) construct neighborhood to preserve embeddings for each node in a graph, which are used to capture pairwise similarities between unstructured data, such as protein–protein interaction networks (Grover and Leskovec 2016), and word-word co-occurrence networks (Pennington, Socher, and Manning 2014).

After numerical embedding, any downstream method is directly applicable to analyze unstructured data, which we refer to as a two-stage method, and has become a standard tool of machine learning. In sentiment analysis, logistic regression is built on paragraph embedding with *Doc2Vec* (Le and Mikolov 2014), and support vector machine is conducted on word embedding with *Word2Vec* (Cortes and Vapnik 1995; Socher et al. 2013). To predict the biological functionality of each protein, multi-label classification is developed based on a graph embedding from the protein–protein interaction network (Grover and Leskovec 2016). In *YouTube* video recommendation, a user's video watch history, and search tokens are embedded as numerical features to fit a deep neural network (Covington, Adams, and Sargin 2016). By comparison, traditional classification and regression hinge on features derived from limited domain knowledge or manually constructed features. For instance, in sentiment analysis, Wang et al. (2016) used bag-of-words for ordinal classification, Genkin, Lewis, and Madigan (2007), Taddy (2013), and Miratrix et al. (2011) relied on word counts or frequencies of a document for the logistic or multinomial regression. Empirical evidence of Mikolov et al. (2013) and Mikolov, Yih, and Zweig (2013) suggests that numerical embedding such as *Word2Vec* and

CONTACT Ben Dai bdai@umn.edu School of Statistics, University of Minnesota, Minneapolis, MN 55455.
Supplementary materials for this article are available online. Please go to www.tandfonline.com/r/JASA.

Doc2Vec is more effective for some learning tasks than the traditional approach based on manually constructed features.

Despite the prominent role in the analysis of unstructured data, a two-stage method has three major issues emerge. First, it uses a generic embedding that is often unrelated to the objective of learning in that it is unsupervised and constructed from unannotated data. Second, little is known about theoretical aspects of any two-stage method, particularly with respect to how the embedding impacts the prediction accuracy, although its superior empirical performance has been widely acknowledged. Third, it is unclear how it can effectively integrate external unannotated data in prediction. As a result, a common practice is to use a large yet redundant embedding for a specific learning task to capture all the information on original data, thus impeding the learning performance. For instance, to keep essential characteristics of text from external relational data, numerical embedding, such as *Word2Vec* or *Doc2Vec*, requires a sufficiently large vector, say 300 to more than 1000.

The main contribution of this article is the development of a novel framework of embedding learning to enhance supervised learning through the construction of a learning-specific embedding based on annotated and unannotated data. Moreover, we provide a theoretical justification for the proposed method and illustrate its advantage over a two-stage method. As shown in Lemma 5, in linear regression, the two-stage methods may fail to provide adaptiveness. Moreover, it indicates that the one-hot encoder (Weinberger et al. 2009) is more preferable among two-stage methods, its dimension restriction hinders its predictive performance as compared to the embedding learner. Finally, we develop a large-scale computational tool for the proposed methods.

Regarding the methodological development, we impose an embedding constraint to maximize the prediction accuracy of an embedding learner while identifying a U-minimal sufficient learning-adaptive embedding. In a sense, we integrate annotated unstructured data with unannotated data via the method of regularization. Then we apply the embedding learning framework to graph embedding classification, based on a hyperlink tensor reflecting multi-way relations of unstructured data (Berge 1973). For linear graph embedding classification, we develop blockwise coordinate descent algorithms with exact and inexact updates. For the corresponding nonlinear neural network classification, we develop a projected gradient descent algorithm based on neural network back-propagation in *TensorFlow*.[1]

Theoretically, we develop the generalization error bounds to shed light on why and when the proposed method is expected to deliver higher accuracy than the two-stage method, and the embedding learning without constraint. The generalization error of the proposed method is determined by the complexity of the embedding space induced by an U-embedding loss. Particularly, for graph embedding classification, the error rate depends on the relational structure of an U-embedding loss, characterized by the eigenvalues and eigenvectors of the corresponding graph matrix, which addresses the question of how an U-embedding loss contributes to learning. Moreover, under some conditions, the proposed linear and nonlinear classifiers can achieve a fast rate of linear classification and a standard rate of smooth classification, as if an optimal embedding were known. Additionally, as indicated in Table 1, the proposed method generally achieves faster convergence rates than its competitors, except in a situation in which the U-embedding loss is not informative.

Table 1. Comparison of the learning regret error rate (upper bound) for three methods, where K^*, $|\mathcal{S}|$, p^*, and α are the effective dimension, the size of unstructured data, the optimal size of embedding in Corollary 1, and $a \lesssim b$ indicates a/b does not converge to ∞, and e_{EL}, e_{OH}, and e_{ELN} denote the learning regrets of the embedding learner, one-hot encoder, and the embedding learner without constraint, respectively.

f^* is linear	$K^* \le \lvert\mathcal{S}\rvert/p^* - 1$	$K^* > \lvert\mathcal{S}\rvert/p^* - 1$
	$e_{\mathrm{EL}} \lesssim e_{\mathrm{OH}} \lesssim e_{\mathrm{ELN}}$	$e_{\mathrm{OH}} \lesssim e_{\mathrm{EL}} \lesssim e_{\mathrm{ELN}}$
$f^* \in \mathcal{W}^{\alpha}_{\infty}$	$K^* \le \left(\lvert\mathcal{S}\rvert - \left(n/\lvert\mathcal{S}\rvert\right)^{\frac{p^*}{\alpha(2-\kappa)}}\right)/p^*$	$K^* > \left(\lvert\mathcal{S}\rvert - \left(n/\lvert\mathcal{S}\rvert\right)^{\frac{p^*}{\alpha(2-\kappa)}}\right)/p^*$
	$e_{\mathrm{EL}} \lesssim e_{\mathrm{OH}} \lesssim e_{\mathrm{ELN}}$	$e_{\mathrm{OH}} \lesssim e_{\mathrm{EL}} \lesssim e_{\mathrm{ELN}}$

NOTE: All logarithm terms are omitted for simplicity.

Finally, we apply the embedding learning framework to two domains of application: grammatical classification in Wikipedia network and sentiment analysis in movie reviews, where the former explores grammatical category based on relations between interacting words, while the latter learns the sentiment level of a document. In particular, we investigate grammatical classification supported by LLE (Roweis and Saul 2000) and Laplacian eigenmaps (Belkin and Niyogi 2002) and movie reviews built on *Doc2Vec*, *Word2Vec*, and *Word2Vec-GoogleNews*.[2] In these examples, the proposed method consistently yields higher prediction accuracy.

This article is organized as follows. Section 2 introduces the framework of embedding learning, which incorporates embedding into learning using the concept of U-minimal sufficient embedding. Section 3 develops a learning theory to quantify the benefits of the proposed method over its competitors. Section 4 applies the embedding learning framework to graph embedding classification. Section 5 develops scalable computation schemes and implements linear classification and neural network classification. Section 6 discusses the connection of the proposed method with other methods. Section 7 investigates the numerical performance of the proposed method and compares with its competitors in real applications. The technical proofs are contained in the supplementary materials.

2. Embedding Learning

Consider a learning task of predicting the output Y, possibly a vector, by unstructured predictors $s \in \mathcal{S} = \{s_1, \ldots, s_{|\mathcal{S}|}\}$ such as words, texts, or documents, where $|\mathcal{S}|$ denotes the number of unstructured elements in $\mathcal{S}$. In sentiment analysis, words of a document, represented by s in a dictionary $\mathcal{S}$, are used to identify the level of sentiment Y of an entire document. In protein analysis, $s \in \mathcal{S}$ denotes proteins to be used to predict its functionality categorized by Y.

To reduce the dimension of $\mathcal{S}$, an embedding $\mathcal{X} = \{\mathcal{X}(s)\}_{s\in\mathcal{S}}$ is constructed, where $\mathcal{X}(s)$ maps from $s \in \mathcal{S}$ onto a p-dimensional vector $\mathcal{X}(s) \in \mathbb{R}^p$, where p, a tuning parameter,

[1] https://www.tensorflow.org/

[2] https://code.google.com/archive/p/word2vec/

is usually no greater than $|\mathcal{S}|$. Typically, an embedding such as *Word2Vec* (Mikolov et al. 2013) is designed to characterize unstructured data so that it can be analyzed by a learning method. Given $\mathcal{X}(s)$, a learning function or vector $f(\mathcal{X}(s))$ is estimated to predict the outcome of Y.

Next we introduce the notation of sufficiency of $\mathcal{X}(s)$ to describe the situation of no information loss when using $\mathcal{X}(s)$ for learning.

Definition 1 (Sufficient learning-adaptive embedding). An embedding $\mathcal{X} = \{\mathcal{X}(s)\}_{s\in\mathcal{S}}$ is learning-adaptive sufficient for predicting the outcome of Y if the conditional distribution of Y given an embedding $\mathcal{X}(s) = x$ does not depend on $S = s$, that is, $\mathbb{P}\big(Y = y|S = s, \mathcal{X}(s) = x\big) = \mathbb{P}\big(Y = y\big|\mathcal{X}(s) = x\big)$.

This definition is analogous to that of the sufficient statistic for numerical data (Casella and Berger 2002). As suggested by Lemma 1, learning from a sufficient embedding is equivalent to learning from original unstructured data.

Let $L(\hat{y}, y)$ be any learning loss; for example, $L(\hat{y}, y) = (\hat{y}-y)^2$ is the L_2-loss for regression, and $L(\hat{y}, y) = I(\hat{y}y \le 0)$ is the zero-one loss for classification.

Lemma 1. Let $\mathcal{X}$ be a sufficient learning-adaptive embedding. Then

$$\min_f \mathbb{E}\big(L\big(f(\mathcal{X}(S)), Y\big)\big) = \min_g \mathbb{E}\big(L\big(g(S), Y\big)\big). \tag{1}$$

In (1), minimizing the right-hand side is generally infeasible due to unstructured data, whereas the left-hand side is manageable given a sufficient embedding. Moreover, $\mathcal{X}(S)$ retains only the information relevant to prediction, allowing further dimension-reduction of an embedding. Note that any one-to-one mapping of a sufficient embedding is also sufficient. Thus, we seek one U-minimal sufficient embedding $\mathcal{X}$ over sufficient embeddings to minimize $U(\mathcal{X})$, what we refer to as an U-embedding loss, which measures the amount of information concerning s captured by $\mathcal{X}(s)$.

2.1. Embedding Loss

An U-embedding loss is often defined with a hyperlink tensor $W = (w^{(d)}_{u_1\dots u_k})$ describing the multi-way relations between elements of $\mathcal{S}$, where $w^{(d)}_{u_1\dots u_k}$ represents the k-order interactions among $s_{u_1}, \dots, s_{u_k}$ from the dth data source. W can be obtained from large-scale unannotated external data, or a pre-estimated scientific knowledge graph (He et al. 2017), such as protein–protein interactions (Stark et al. 2010), WordNet (Fellbaum 2010), and text-to-text frequencies. More specifically, given that each element of $\mathcal{S}$ is embedded as a vector in $\mathbb{R}^p$, the U-embedding loss is defined as $U(\mathcal{X}) : \mathbb{R}^p \times \cdots \times \mathbb{R}^p \to \mathbb{R}$, where $\mathbb{R}^p \times \cdots \times \mathbb{R}^p$ is the $|\mathcal{S}|$-way Cartesian power of $\mathbb{R}^p$.

Different U-embedding losses use disparate principles based on various aspects of original data. We now give some examples of $U(\mathcal{X})$ as follows:

$$\text{Locally linear} \sum_{d=1}^{D}\sum_{u_1} \|\mathcal{X}(s_{u_1}) - \sum_{u_2} w^{(d)}_{u_2u_1}\mathcal{X}(s_{u_2})\|_2^2, \tag{2}$$

$$\text{Laplacian eigenmaps} \sum_{d=1}^{D}\sum_{u_1,u_2} w^{(d)}_{u_1u_2}\|\mathcal{X}(s_{u_1}) - \mathcal{X}(s_{u_2})\|_2^2, \tag{3}$$

$$\text{Ordinal embedding} \sum_{d=1}^{D}\sum_{u_1,u_2,u_3} w^{(d)}_{u_1u_2u_3} I\big(\|\mathcal{X}(s_{u_1}) - \mathcal{X}(s_{u_3})\|_2^2 \le \|\mathcal{X}(s_{u_2}) - \mathcal{X}(s_{u_3})\|_2^2\big), \tag{4}$$

$$\text{Word2Vec-multi-word} - \sum_{d=1}^{D}\sum_{u_1,\dots,u_k} w^{(d)}_{u_1\dots u_k} \log\Big(\frac{\exp\big(\frac{1}{k-1}(\sum_{l=1}^{k-1}\mathcal{X}(s_{u_l}))^\top\mathcal{X}(s_{u_k})\big)}{\sum_{v=1}^{|\mathcal{S}|}\exp(\frac{1}{k-1}(\sum_{l=1}^{k-1}\mathcal{X}(s_{u_l}))^\top\mathcal{X}(s_v))}\Big), \tag{5}$$

where $\|\cdot\|_2$ is the L_2-norm. In particular, (2) and (3) construct embeddings based on neighborhood preserving principle in a pairwise relational hyper-graph (Roweis and Saul 2000; Belkin and Niyogi 2002). A typical example is the WordNet dataset, two different types ($D = 2$) of similarities are presented for each pair of words, including semantic and syntactic relations (Fellbaum 2010). Then, (2) and (3) provide close numerical vectors in the Euclidean space to present words with similar semantics and syntactics. Moreover, (4) focuses on ordinal data and penalizes the misordering embeddings, both structured or unstructured, where the weight $w^{(d)}_{u_1u_2u_3}$ measures the probability of misordering embeddings in terms of the L_2-Euclidean distance for an ordered triple u_1, u_2, u_3 (Bădoiu et al. 2008). Multiple-words *Word2Vec* embedding (5) uses the information on the interaction between $(s_{u_1}, \dots, s_{u_{k-1}})$ and s_{u_k}, where $w^{(d)}_{u_1,\dots,u_k}$ presents the frequency of word s_{u_k} given previous words $(s_{u_1}, \dots, s_{u_{k-1}})$, estimated from the dth unannotated text corpus. The loss function in (5) is actually the multinomial likelihood or the Kullback–Leibler loss on the probability of $s_{u_k}|s_{u_1}, \dots, s_{u_{k-1}}$ induced by an inner product of embeddings (Mikolov et al. 2013). Consequently, similar words can be close in terms of the inner product of the estimated embeddings.

2.2. Transfer Embedding Loss

Many U-embedding losses do not have analytic expressions, particularly *Node2Vec* (Grover and Leskovec 2016), which explores the neighborhood structure based on random walks. On the other hand, some embeddings are pre-trained by transfer learning from large unannotated data, specifically, *Word2Vec-GoogleNews*, and Global Vectors for Word Representation *GloVe* (Pennington, Socher, and Manning 2014). One way to incorporate these embeddings is constructing a transfer embedding loss to identify a sufficient learning-adaptive embedding in a neighborhood of a pre-trained embedding. In particular, a transfer embedding loss $U(\mathcal{X})$ can be $\sum_{u=1}^{|\mathcal{S}|} d(\mathcal{X}(s_u), \widetilde{\mathcal{X}}(s_u))$, where $\widetilde{\mathcal{X}}$ denotes a pre-trained embedding and $d(\mathcal{X}(s_u), \widetilde{\mathcal{X}}(s_u))$ is a certain metric measuring the discrepancy between $\mathcal{X}(s_u)$ and $\widetilde{\mathcal{X}}(s_u)$, for instance, $d(\mathcal{X}(s_u), \widetilde{\mathcal{X}}(s_u)) = \|\mathcal{X}(s_u) - \widetilde{\mathcal{X}}(s_u)\|_2^2$. As demonstrated in the numerical result of Section 6.2, a transfer embedding loss yields a good prediction accuracy in sentiment analysis.

2.3. Proposed Framework

This subsection proposes a framework of embedding learning to identify a sufficient learning-adaptive embedding while integrating training data with large external unannotated data through regularization.

For a given U-embedding loss, it is natural to ask if there is an embedding $\mathcal{X}$ that is sufficient for predicting Y while being the most efficient in terms of the U-embedding loss. This leads to the definition of U-minimal sufficiency.

Definition 2 (U-minimal sufficient embedding). A sufficient learning-adaptive embedding $\mathcal{X}$ is U-minimal if it minimizes an U-embedding loss $U(\mathcal{X})$ over all sufficient embeddings, or,

$$\mathcal{X}^* \in \underset{\text{sufficient embedding } \mathcal{X}}{\operatorname{argmin}} \quad U(\mathcal{X}).$$

Definition 2 says that $\mathcal{X}^*$ is a most efficient embedding that minimizes the U-embedding loss $U(\mathcal{X})$ among all sufficient embeddings. Denote $C^* = U(\mathcal{X}^*)$, then equivalently, by Lemma 1, $\mathcal{X}^*$ minimizes the corresponding learning loss subject to the embedding constraint, that is, $\mathcal{X}^* \in \operatorname{argmin}_{\mathcal{X}:U(\mathcal{X})\le C^*} \min_f \mathbb{E}\big(L\big(f(\mathcal{X}(S)), Y\big)\big)$. Since the value of C^* is unknown, we replace it by a tuning parameter C to be estimated by cross-validation as described in Section 6, which is a manner similar to Tibshirani (1996), Mazumder, Hastie, and Tibshirani (2010), and Hastie, Friedman, and Tibshirani (2009). Then an embedding learner seeks $(f, \mathcal{X})$ such that

$$\min_{f,\mathcal{X}} \mathbb{E}\Big(L\big(f(\mathcal{X}(S)), Y\big)\Big) \quad \text{subj to} \quad U\big(\mathcal{X}\big) \le C. \tag{6}$$

Lemma 2. Let $f^* = \operatorname{argmin}_f \mathbb{E}\big(L\big(f(\mathcal{X}^*(S)), Y\big)\big)$, then $\big(f^*, \mathcal{X}^*\big)$ is a minimizer of (6) for $C \ge C^*$, and $\mathbb{E}\big(L\big(f^*(\mathcal{X}^*(S)), Y\big)\big) = \min_g \mathbb{E}\big(L\big(g(S), Y\big)\big)$.

Lemma 2 says that (6) is able to recover a U-minimal sufficient embedding $\mathcal{X}^*$ defined in Definition 2.

In light of the foregoing discussion, we propose an empirical embedding learning cost function for (6) to integrate training data $\{s^i, y^i\}_{i=1}^n$ with large external data summarized by $U(\mathcal{X})$:

$$\min_{f\in\mathcal{F},\mathcal{X}} n^{-1}\sum_{i=1}^{n} V\big(f(\mathcal{X}(s^i)), y^i\big) + \lambda J(f) \quad \text{subj to} \quad U\big(\mathcal{X}\big) \le C, \tag{7}$$

where V is a surrogate loss of L for the computation purpose, for example, in classification, $V\big(f(\mathcal{X}(s)), y\big) = \big(1 - yf(\mathcal{X}(s))\big)_+$ (Cortes and Vapnik 1995), $V(f(\mathcal{X}(s)), y) = \log\big(yf(\mathcal{X}(s))/(1 - yf(\mathcal{X}(s)))\big)$ (Zhu and Hastie 2002), and $V(f(\mathcal{X}(s)), y) = \min(1, (1 - yf(\mathcal{X}(s)))_+)$ (Shen et al. 2003) with z_+ denoting the nonnegative part of z, are the hinge, logistic and ψ-loss; in regression, $V(f(\mathcal{X}(s)), y) = (y - f(\mathcal{X}(s)))^2$ is the L_2-loss. Here $J(f)$ is a nonnegative regularizer for f such as the inverse of the separation margin (Cortes and Vapnik 1995), $\lambda \ge 0$ is a tuning parameter controlling the trade-off between learning and regularization of f, and $\mathcal{F}$ is a class of candidate functions.

3. Theory

This section develops an asymptotic theory to quantify the learning accuracy of an embedding learner, as measured by learning regret $e(\theta) = \mathbb{E}\big(L\big(f(\mathcal{X}(S)), Y\big) - L\big(f^*(\mathcal{X}^*(S)), Y\big)\big) \ge 0$, where $\theta = (f, \mathcal{X})$. In addition, $\widehat{\theta} = (\widehat{f}, \widehat{\mathcal{X}})$ denotes a minimizer of (7), and $\theta^* = (f^*, \mathcal{X}^*)$ denotes the optimal learning function as well as the optimal embedding.

Let $e_V(\theta) = \mathbb{E}\big(V\big(f(\mathcal{X}(S)), Y\big) - V\big(f^*(\mathcal{X}^*(S)), Y\big)\big)$ be the learning regret under the surrogate loss V. $\bar{\theta} = (\bar{f}, \bar{\mathcal{X}})$ be any approximation of θ^*, where $\bar{f} \in \mathcal{F}$, $\bar{\mathcal{X}} \in \mathcal{D}_C = \big\{\mathcal{X} : U(\mathcal{X}) \le C,\ \sup_{s\in\mathcal{S}} \|\mathcal{X}(s)\|_\infty \le B\big\}$, and B is a constant such that $B \ge \sup_{s\in\mathcal{S}} \|\mathcal{X}^*(s)\|_\infty$, as f^* may not belong to the class of candidate functions $\mathcal{F}$ in (7). Then the corresponding approximation error is defined as $\upsilon_n^2 = e_V(\bar{\theta})$. One typical example is, $\bar{\mathcal{X}} = \mathcal{X}^*$ when $C \ge C^*$, and thus $\bar{f} = \operatorname{argmin}_{f\in\mathcal{F}} \mathbb{E}\big(V\big(f(\mathcal{X}(S)), Y\big)$. Note that the approximation error bound for υ_n^2 has been studied in Niyogi and Girosi (1996) and Yarotsky (2017). It is sensible to assume that $\sup_{s\in\mathcal{S}} \|\widehat{\mathcal{X}}(s)\|_\infty \le B$ as an embedding is typically finite in practice. The following technical conditions are assumed.

Assumption A1 (Conversion). There exist constants $\mu > 0$ and $c_1 > 0$ such that for any small $\epsilon > 0$,

$$\sup_{\{f\in\mathcal{F},\mathcal{X}\in\mathcal{D}_C: e_V(\theta)\le\epsilon\}} e(\theta) \le c_1\epsilon^\mu.$$

Assumption A2 (Variance). There exist constants $1 \ge \kappa \ge 0$ and $c_2 > 0$ such that for any $\epsilon > 0$,

$$\sup_{\{f\in\mathcal{F},\mathcal{X}\in\mathcal{D}_C: e_V(\theta)\le\epsilon\}} r_V(\theta) \le c_2\epsilon^\kappa,$$

where $r_V(\theta) = \operatorname{var}\big(V\big(f(\mathcal{X}(S)), Y\big) - V\big(f^*(\mathcal{X}^*(S)), Y\big)\big)$.

Assumption A3 (Complexity). For some constants $c_j > 0; j = 3, 4, 5$, there exists $\epsilon_n > 0$ such that

$$\sup_{r\ge 2} \phi(\epsilon_n, r) \le c_3 n^{1/2}, \quad \phi(\epsilon_n, r) = \int_{c_5\zeta}^{c_4^{1/2}\zeta^{\kappa/2}} H^{1/2}\big(u, \mathcal{V}(r)\big)du/\zeta, \tag{8}$$

where $\zeta = \min(\epsilon_n^2 + \lambda\bar{J}(r/2 - 1), 1)$, $H(u, \mathcal{V}(r))$ is the L_2-bracketing metric entropy H_2 or the L_∞-bracketing metric entropy H_∞, and $\mathcal{V}(r) = \big\{V\big(f(\mathcal{X}(s)), y\big) - V\big(\bar{f}(\bar{\mathcal{X}}(s)), y\big) : f \in \mathcal{F}, J(f) \le \bar{J}r, U(\mathcal{X}) \le C\big\}$, where $\bar{J} = \max(J(\bar{f}), 1)$.

Assumption A1 quantifies the relation between $e(\theta)$ and $e_V(\theta)$ to establish convergence of the learning regret of a global minimizer $\widehat{\theta}$ of (7). Assumption A2 characterizes the degree of smoothness in terms of the mean and variance of V to quantify the behavior of the resultant empirical processes. Both the assumptions are widely used in the literature and can be directly verified with many popular choices of V, including the ψ-loss with $\mu = 1$ and $\kappa = 1$ (Shen et al. 2003) and the hinge loss with $\mu = 1/2$ and $\kappa = 1$ (Shen and Wang 2007). Assumption A3 measures the complexity of $(f, \mathcal{X})$ with $f \in \mathcal{F}$ and $\mathcal{X} \in \{\mathcal{X} : U(\mathcal{X}) \le C\}$. Consequently, the U-embedding loss $U(\mathcal{X})$ incorporates the information of unannotated data to reduce the complexity of the embedding space by tuning C.

Theorem 1. Under Assumptions A1–A3, if $C \geq U(\mathcal{X}^*)$ in (7), and $\widehat{\theta} = (\widehat{f}, \widehat{\mathcal{X}})$ is a global minimizer of (7), then there exist nonnegative constants c_6–c_8 such that

$$\mathbb{P}\big(e(\widehat{\theta}) \geq c_6 \delta_n^{2\mu}\big) \leq c_7 \exp\big(-c_8 n(\lambda \bar{J})^{2-\kappa}\big), \tag{9}$$

provided that $\lambda^{-1} \geq 2\delta_n^{-2}\bar{J}$ and $\delta_n^2 = \min(\epsilon_n^2 + \upsilon_n^2, 1)$. Therefore, $e(\widehat{\theta}) = O_P(\delta_n^{2\mu})$ when $n(\lambda\bar{J})^{2-\kappa}$ is bounded away from zero.

Worthy of note is that the learning regret $e(\widehat{\theta})$ is governed by the approximation error υ_n^2 and the estimation error determined by ϵ_n^2 from the metric entropy equation in (8). Clearly, the learning regret is governed by $\mathcal{F}$, V, U, and C. In particular, the surrogate loss V determines the degrees of smoothness μ and κ in Assumptions A1 and A2, for example, $\mu = \kappa = 1$ in regression (Shen and Wang 2007); $\mu = 1$ and $0 \leq \kappa \leq 1$ when $L(\hat{y}, y) = I(\hat{y}y \leq 0)$ and $V(\hat{y}, y) = \min((1 - y\hat{y})_+, 1)$ in classification and $\kappa = 1$ under the low-noise assumption (Shen and Wong 1994; Shen et al. 2003). The estimation error ϵ_n^2 increases but the approximation error υ_n^2 decreases when $\mathcal{F}$ becomes larger. As illustrated by Corollary 2, the best learning regret in the nonlinear deep network case can be obtained by striking an optimal trade-off between ϵ_n^2 and υ_n^2 in (12). Finally, the U-embedding loss and the tuning parameter C restrict the embedding space $\mathcal{D}_C$, which in turn impacts ϵ_n^2. This aspect is illustrated by Corollaries 1 and 2.

4. Graph Embedding Classification

This section applies the embedding learning framework (6) to binary classification with unstructured training data $\{s^i, y^i\}_{i=1}^n$, where $y^i \in \{-1, 1\}$ is a label. The unstructured data is characterized by a graph $\mathcal{G} = (\mathcal{S}, \boldsymbol{W})$, describing the pairwise (two-way) relations among the unstructured data elements, where $\boldsymbol{W} = (\boldsymbol{W}^{(d)})_{d=1}^D$, and $\boldsymbol{W}^{(d)} = (w_{uv}^{(d)}) \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{S}|}$ is a hyperlink tensor obtained from external data or prior knowledge, whose the (u, v)th element $w_{uv}^{(d)}$ is a d-source weight assigned to undirected/directed hyperlink from s_u to s_v.

Next, we analyze an U-embedding loss in a form of $U(\mathcal{X}) = \frac{1}{2}\sum_{d=1}^D \mathrm{Tr}(\boldsymbol{X}\boldsymbol{Q}^{(d)}\boldsymbol{X}^T)$, where $\boldsymbol{X} = \big(\mathcal{X}(s_1), \ldots, \mathcal{X}(s_{|\mathcal{S}|})\big) \in \mathbb{R}^{p\times|\mathcal{S}|}$ is an embedding matrix of $\mathcal{X}$, and $\mathrm{Tr}(\cdot)$ is the trace of a matrix. This includes, but is not limited to LLE in (2) with $\boldsymbol{Q}^{(d)} = (\boldsymbol{I}_{|\mathcal{S}|} - \boldsymbol{W}^{(d)})^\top(\boldsymbol{I}_{|\mathcal{S}|} - \boldsymbol{W}^{(d)})$ and Laplacian eigenmaps when $\boldsymbol{Q}^{(d)} = \boldsymbol{T}^{(d)} - \boldsymbol{W}^{(d)}$ is the graph Laplacian and $\boldsymbol{T}^{(d)}$ is a diagonal matrix with $T_{ii}^{(d)} = \sum_j W_{ij}^{(d)}$. Let $\boldsymbol{Q} = \sum_{d=1}^D \boldsymbol{Q}^{(d)}$, and its eigendecomposition is $\boldsymbol{Q} = \boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{P}^T$, where $\boldsymbol{\Lambda} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_{|\mathcal{S}|})$ is a diagonal matrix with eigenvalues of $\boldsymbol{Q}$ ($\sigma_1 \leq \sigma_2 \leq \cdots \leq \sigma_{|\mathcal{S}|}$), and $\boldsymbol{P}$ is an orthogonal matrix consisting of the corresponding eigenvector vectors.

By (7), the cost function of an embedding learner for graph embedding classification is

$$\min_{f\in\mathcal{F},\mathcal{X}} n^{-1}\sum_{i=1}^n V\Big(f\big(\mathcal{X}(s^i)\big), y^i\Big) + \lambda J(f) \text{ subj to } \frac{1}{2}\sum_{d=1}^D \mathrm{Tr}(\boldsymbol{X}\boldsymbol{Q}^{(d)}\boldsymbol{X}^T) \leq C. \tag{10}$$

Solving (10) yields an estimated $(\widehat{f}, \widehat{\mathcal{X}})$. Next we make a technical assumption for (10).

Assumption B. Assume that the surrogate loss $V(f, y)$ is a margin loss in that $V(f, y) = V(z)$ with $z = yf$, where $V(z)$ is Lipschitz-continuous in z.

Assumption B is a continuity condition, which is met by most surrogate losses, such as the hinge loss $V(\hat{y}, y) = (1 - y\hat{y})_+$ for support vector machines (SVM) (Cortes and Vapnik 1995) and ψ-loss $V(\hat{y}, y) = \min(1, (1 - y\hat{y})_+)$ for ψ-learning (Shen et al. 2003). Subsequently, the classification regret of an embedding learner is given in Corollaries 1 and 2.

In linear classification, let $(f^*, \mathcal{X}^*)$ be a minimizer of (6) with $L(\hat{y}, y) = I(y\hat{y} \leq 0)$. Suppose $\bar{f}(x) = f^*(x) = x^\top\beta^* \in \mathcal{F} = \{f(x) = \beta^\top x : \beta \in \mathbb{R}^{p^*}\}$, and $\bar{\mathcal{X}} = \mathcal{X}^*$, where p^* is the dimension of $\mathcal{X}^*(s)$. Moreover, let $J(f) = \|\beta\|_2^2$ in (10) and $\bar{J} = \max(\|\beta^*\|_2^2, 1)$. In this case, the approximation error υ_n^2 is zero, when $C \geq C^*$ and $p \geq p^*$.

Corollary 1 (Linear classification). Under Assumptions A1–A2 and B, for a global minimizer $\widehat{\theta}$ of (10), $e(\widehat{\theta}) = O_p(\delta_n^{2\mu})$ with

$$\delta_n^2 = \Big(\frac{(K^*+1)p}{n}\log\Big(\frac{n\sqrt{p\bar{J}}(1 + \|P\|_1\|P\|_\infty)}{(K^*+1)p}\Big)\Big)^{\frac{1}{2-\kappa}}, \tag{11}$$

provided that $K^* = \Big|\big\{k : \sigma_k < 4p\bar{J}C\|P\|_\infty^2\big(\frac{n}{(1+k)p}\big)^{\frac{2}{2-\kappa}}\big\}\Big|$ is the effective dimension of the underlying problem, $C \geq U(\mathcal{X}^*)$, $p \geq p^*$, $\|P\|_\infty$ and $\|P\|_1$ are the infinity-norm and 1-norm of matrix P, and $O_p(\cdot)$ denotes stochastic ordering (Vapnik 1998).

In nonlinear classification, we consider a multi-layer neural network with a rectified linear unit (ReLU) activation function (Glorot, Bordes, and Bengio 2011). In this situation, we approximate the optimal learning function and embedding $\theta^* = (f^*(\mathcal{X}^*), \mathcal{X}^*)$ by $\bar{\theta} = (\bar{f}, \bar{\mathcal{X}})$, where $\bar{\mathcal{X}} = \mathcal{X}^* \in \mathcal{D}_C$ with $C \geq C^*$, $\bar{f} = \mathrm{argmin}_{f\in\mathcal{F}}\|f - f^*\|_\infty$, $\|\cdot\|_\infty$ is the supremum norm, and $\mathcal{F} = \big\{f_M(x) = A_M\mathbf{f}_{M-1} + b_M$ with $(\mathbf{f}_m = \sigma(A_m\mathbf{f}_{m-1} + b_m))_{m=1}^{M-1}, \mathbf{f}_0 = x : A_m \in \mathbb{R}^{h_{m-1}\times h_m}, b_m \in \mathbb{R}^{h_m}, m = 1, \ldots, M\big\}$ is the space of neural network functions, M is the number of layers, h_m is the number of nodes in the mth layer with $h_0 = p$ and $h_M = 1$, $\sigma(z_m) = (\sigma(z_{m,1}), \ldots, \sigma(z_{m,h_m}))^\top$, and $\sigma(z)$ is a ReLU activation function.

Assume that the optimal learning function f^* belongs to a Sobolev space $W_\infty^\alpha[-B, B]^{p^*}$ (Yarotsky 2017), where B is previously defined with $\sup_{s\in\mathcal{S}}\|\widehat{\mathcal{X}}(s)\|_\infty \leq B$. Let $J(f) = \sum_{m=1}^M \big(\|A_m\|_{1,1} + \|b_m\|_1\big)$ in (10), and $\bar{J}_M = \max(J(\bar{f}), 1)$. Denote by $\boldsymbol{\Theta}_N = \{(A_m, b_m)\}_{m=1}^M$ the collection of all parameters of the neural network and $|\boldsymbol{\Theta}_N| = \sum_{m=1}^M (h_{m-1}h_m + h_m)$ is the total number of parameters of the neural network.

Corollary 2 (Nonlinear classification via deep ReLU networks). Under Assumptions A1–A2 and B, for a global minimizer $\widehat{\theta}$ of (10), $e(\widehat{\theta}) = O_p(\delta_n^{2\mu})$ with $\delta_n^2 = \epsilon_n^2 + \upsilon_n^2$ and

$$\epsilon_n^2 = \Big(\frac{|\boldsymbol{\Theta}_N| + K^*p}{n}\log\Big(\frac{n(M + \|P\|_1\|P\|_\infty)(2\bar{J}_M)^{2M}}{M(|\boldsymbol{\Theta}_N| + K^*p)}\Big)\Big)^{\frac{1}{2-\kappa}},$$
$$\upsilon_n^2 = \max\Big\{\frac{1}{e^M}, |\boldsymbol{\Theta}_N|^{-\frac{\alpha}{p}}\log\Big(\frac{n}{M(|\boldsymbol{\Theta}_N| + K^*p)}\Big)\Big\}, \tag{12}$$

provided that $K^* = \big|\{k : \sigma_k < 2C\|P\|_\infty^2(2\bar{J}_M/M)^{2M}\big(\frac{n}{M(|\Theta_N|+kp)}\big)^{\frac{2}{2-\kappa}}\}\big|$, $C \geq C^*$, and $p \geq p^*$.

In (11) and (12), the optimal C with the constraint $C \geq U(\mathcal{X}^*)$ is $C^* = U(\mathcal{X}^*)$, since a large value of C increases the complexity of the embedding space $\mathcal{D}_C$ as well as the effective dimension K^* in the rates, while the optimal value of the embedding dimension p is $p = p^*$. In practice, C^* and p^* can be estimated by cross-validation as discussed in Section 6. For deep neural network, the depth M and the total number of parameters $|\Theta_N|$ are optimized to minimize δ_n^2, which balances the trade-off between υ_n^2 and ϵ_n^2, depending on p^*, C^*, and the eigenvalues and eigenvectors of Q.

Corollaries 1 and 2 suggest that the rates in (11) and (12) are proportional to the value of K^*, which characterizes the impact of the U-embedding loss on the performance of embedding learning. Note that K^* usually decreases as $(\sigma_k)_{k=1}^{|\mathcal{S}|}$ increases and as $\|P\|_\infty$ or $U(\mathcal{X}^*)$ decreases, leading to faster rates in (11) and (12). Interestingly, when K^* is a constant order, the rate becomes a standard one as if the optimal numerical embedding were known in advance. For example, in the linear case, if there exists a constant $\bar{K}$ independent of n such that $\sigma_{\bar{K}}/\|P\|_\infty = O(C^*p^*(n/p^*)^{\frac{2}{2-\kappa}}$, the rate becomes $O_p\big(\big(\frac{p^*}{n}\log(\frac{n}{p^*})\big)^{\frac{\mu}{2-\kappa}}\big)$, which is the squared parametric rate when $\mu = 1$ and $\kappa = 1$ as in ψ-learning (Shen et al. 2003) and is the parametric rate when $\mu = 1/2$ and $\kappa = 1$ as in SVM (Shen and Wang 2007). For deep neural network classification, the rate is $O_p((n^{\zeta-1}\log^2 n)^{\frac{\mu}{2-\kappa}})$, which agrees with the generalization error rate of a deep network classifier (Schmidt-Hieber 2017; Xu and Wang 2018), where $\zeta = \frac{p^*}{p^*+\alpha(2-\kappa)} \in [0,1)$, when $\sigma_{\bar{K}}/\|P\|_\infty = O(C^*(2\bar{J}_M/M)^M(\frac{n}{M(|\Theta_N|+p^*)})^{\frac{2}{2-\kappa}})$ with $M \sim \log(n)$ and $|\Theta_N| \sim n^\zeta$. When $\mu = \kappa = 1$, the rate reduces to $n^{-\frac{\alpha}{\alpha+p^*}}\log^2 n$. While we conjecture that this rate is nearly optimal, we are not aware of any lower bound for (12), except that Tsybakov (2004) gives a lower bound of order $n^{-\frac{\alpha}{\alpha+p^*-1}}$ for estimation of classification sets of smooth epigraphs with a smaller metric entropy.

5. Scalable Computation for Graph Embedding

This section develops a scalable computing scheme to solve (10) for linear embedding classification and nonlinear embedding classification.

5.1. Linear Classification

In linear embedding classification, we consider (10) with $f(\mathcal{X}(s)) = \beta^\top\mathcal{X}(s)$ by a blockwise coordinate descent (BCD; Chen et al. 2012). The scheme relaxes bi-convex minimization (10) into a sequence of convex subproblems, where each subproblem can be efficiently solved. Specifically, (10) can be written as

$$\min_{\beta,X} n^{-1}\sum_{i=1}^n V\big(y^j\beta^\top\mathcal{X}(s^j)\big) + \lambda\|\beta\|_2^2, \text{ subj to } \frac{1}{2}\mathrm{Tr}(XQX^\top) \leq C. \tag{13}$$

Next, we consider learning and embedding blocks β and X separately, and use $V(z) = (1-z)_+$ for illustration. Then (13) is solved by alternating two convex subproblems.

5.1.1. Learning Block

Given X, β is updated by solving a linear SVM by Liblinear (Fan et al. 2008):

$$\min_\xi n^{-1}\sum_{i=1}^n \xi_i + \lambda_1\|\beta\|_2^2, \text{ subj to } 1 - y^j\beta^\top(X\mathbf{1}_{s^j}) \leq \xi_i, \quad \xi_i \geq 0. \tag{14}$$

5.1.2. Embedding Block

Given β, X is updated by solving

$$\min_{\mathrm{vec}(X)} n^{-1}\sum_{i=1}^n \big(1 - y^j(\mathbf{1}_{s^j}\otimes\beta)^\top\mathrm{vec}(X)\big)_+, \text{ subj to } \frac{1}{2}\mathrm{vec}(X)^\top\big(Q\otimes I_p\big)\mathrm{vec}(X) \leq C, \tag{15}$$

where $\otimes$ is the Kronecker product and $\mathrm{vec}(X)$ is the column-vectorization of X. Note that (15) is convex in $\mathrm{vec}(X)$. Then we work with an equivalent regularization form of (15) with regularization parameter λ_2 corresponding to C:

$$\min_{\mathrm{vec}(X),\xi} n^{-1}\sum_{i=1}^n \xi_i + \frac{\lambda_2}{2}\mathrm{vec}(X)^\top\big(Q\otimes I_p\big)\mathrm{vec}(X), \text{ subj to } 1 - y^j(\mathbf{1}_{s^j}\otimes\beta)^\top\mathrm{vec}(X) \leq \xi_i, \quad \xi_i \geq 0; i = 1,\ldots,n, \tag{16}$$

which involves a $|\mathcal{S}| \times p$ dimensional vector $\mathrm{vec}(X)$. By equivalence tuning C is equivalent to that of λ_2 (Rockafellar and Wets 2011). Therefore, we directly tune λ_2 without converting back to C in computation. Finally, to alleviate the potential issue of memory, we consider the dual form of (16)

$$\min_\alpha \frac{\|\beta\|_2^2}{2\lambda_2}\sum_{i=1}^n\sum_{j=1}^n \alpha_i\alpha_j y^i y^j\big(\mathbf{1}_{s^i}^\top Q^{-1}\mathbf{1}_{s^j}\big) - \sum_{i=1}^n\alpha_i \text{ subj to } 0 \leq \alpha_i \leq \frac{1}{n}, \tag{17}$$

which is a box-constrained quadratic problem and solved by a standard coordinate descent algorithm (Fan et al. 2008). Then the solution of $\mathrm{vec}(X)$ in (17) has the form $\mathrm{vec}(X) = \frac{1}{\lambda_2}\sum_{i=1}^n\alpha_i y^i\big(Q^{-1}\otimes\beta\big)$, where $\{\alpha_i\}_{i=1}^n$ is a solution of (17).

The foregoing computational scheme is summarized in Algorithm 1, where the final solution is denoted as $(\widehat{\beta},\widehat{X})$. Then, each unstructured data s is classified by $\mathrm{sign}(\widehat{\beta}^\top\widehat{\mathcal{X}}(s))$.

Algorithm 1 (BCD with exact update).

Step 1 (Initialization). Initialize $\mathrm{vec}(X)$ and specify a tolerance level.

Step 2 (Learning-block). Update β by solving (14) with fixed $\mathrm{vec}(X)$.

Step 3 (Embedding-block). Update $\mathrm{vec}(X)$ by solving $(\alpha_i)_{i=1}^n$ in (17)with fixed β.

Step 4 (Termination). Iterate Steps 2 and 3 until the decrement of the cost function is less than the tolerance level.

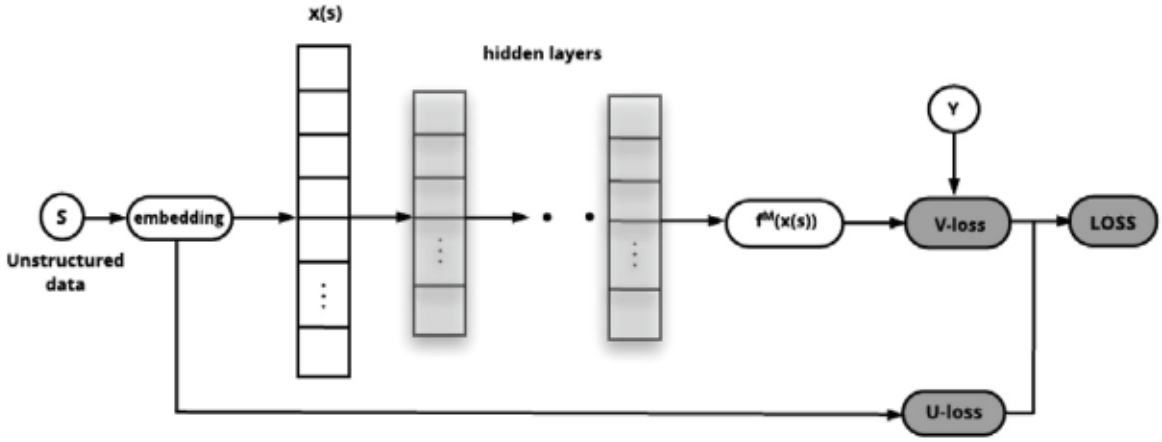

Figure 1. Architecture of nonlinear deep neural network embedding classification.

Note that (14) and (17) are strictly convex minimization, although (13) is nonconvex. We define a point $(\beta^0, \mathcal{X}^0)$ to be a stationary point of a cost function $\tilde{l}_V$ in (13) if

$$\beta^0 = \underset{\beta}{\operatorname{argmin}}\ \tilde{l}_V(\beta, X^0), \quad \text{and} \quad \mathcal{X}^0 = \underset{\mathcal{X}:U(\mathcal{X})\le C}{\operatorname{argmin}}\ \tilde{l}_V(\beta^0, X).$$

The numerical convergence properties of Algorithm 1 can be established as follows.

Lemma 3 (Convergence of Algorithm 1). A solution $(\widehat{\beta}, \widehat{X})$ of Algorithm 1 is a stationary point of (13). If l_V is regular at $(\widehat{\beta}, \widehat{X})$, then the solution is a local minimizer.

In the literature of scalable computing, it has become prevailing to consider blockwise coordinate gradient descent method (BCGD), which only updates parameters along their gradients for one step. Specifically, the exact update in Steps 2 and 3 of Algorithm 1 can be replaced by an inexact update for scalability, which is summarized in Algorithm 1a.

Algorithm 1a (BCD with inexact update).

Step 1 (Initialization). Initialize X and β, and specify a tolerance level.

Step 2 (Learning-block). Update β along its gradient,

$$\beta^{(k+1)} = \beta^{(k)} - n^{-1}\gamma^{(k)} \sum_{i=1}^{n} y^i V' \left(y^i (\mathcal{X}^{(k)}(s^i))^\top \beta^{(k)}\right) \mathcal{X}(s^i) - 2\gamma^{(k)}\lambda_1 \beta^{(k)},$$

where $\gamma^{(k)}$ is a step size of the algorithm.

Step 3 (Embedding-block). Update X along its gradient,

$$X^{(k+1)} = X^{(k)} - n^{-1}\gamma^{(k)} \sum_{i=1}^{n} y^i V' \left(y^i (\mathcal{X}^{(k)}(s^i))^\top \beta^{(k+1)}\right) \beta \mathbf{1}_{s^i}^\top - \gamma^{(k)}\lambda_2 X^{(k)} Q. \quad (18)$$

Step 4 (Termination). Iterate Steps 2 and 3 until the decrement of the cost function is less than the tolerance level.

The key difference between Algorithms 1 and 1a lies in the update of β and X along their gradients for one step based on the idea of BCGD. Yet, convergence properties of BCGD remain largely unknown for nonconvex minimization, although it delivers superior performance in some empirical studies (Udell et al. 2016). Moreover, the update of β and X can be approximately solved by, for example, the blockwise coordinate proximal gradient method (Udell et al. 2016). Finally, classical gradient descent method is also applicable to update β and X simultaneously, which leads to a local minimum (Lee et al. 2016).

5.2. Deep Neural Network Classification

This subsection implements a deep neural network classifier for solving (10) when $f(x)$ is a neural net. Let $\mathbf{f}_m(\mathcal{X}(s^i)) = \sigma(A_m \mathbf{f}_{m-1}(\mathcal{X}(s^i)) + b_m) \in \mathbb{R}^{h_m}$, $\mathbf{f}_0(\mathcal{X}(s^i)) = \mathcal{X}(s^i) \in \mathbb{R}^p$, $A_m \in \mathbb{R}^{h_{m-1}\times h_m}$, $b_m \in \mathbb{R}^{h_m}$; $m = 1, \ldots, M$, M is the number of layers, h_m is the number of nodes in the mth layer with $h_0 = p$ and $h_M = 1$, and $\sigma(z)$ is a ReLU. Other choices of $\sigma(z)$ are possible, including the sigmoid and tanh functions. The neural network architecture is displayed in Figure 1. From (10), we solve

$$\min_{\Theta_N, X} n^{-1} \sum_{i=1}^{n} V\left(y^i \mathbf{f}_M(\mathcal{X}(s^i))\right) + \lambda \sum_{m=1}^{M} \left(\|A_m\|_{1,1} + \|b_m\|_1\right) \text{ subj to } \frac{1}{2}\operatorname{Tr}(XQX^\top) \le C. \quad (19)$$

For (19), we employ the method of projected gradient descent, and integrate it into the classical neural network back-propagation algorithm (Linnainmaa 1976) for classification. Specifically, we employ a projected gradient decent method together with an analytic update of gradients, where the gradients for learning and embedding parameters are given subsequently.

5.2.1. Gradients for Learning

Let $V^i = V(y^i \mathbf{f}_M^i)$, $V_n = \sum_{i=1}^n V^i$, $z_m^i = A_m \mathbf{f}_{m-1}^i + b_m$, and $\mathbf{f}_m^i = \mathbf{f}_m(\mathcal{X}(s^i))$. The back-propagation of the gradient of Θ_N is computed as

$$\frac{\partial V_n}{\partial b_m} = \frac{1}{n}\sum_{i=1}^{n} \frac{\partial V^i}{\partial b_m} = \frac{1}{n}\sum_{i=1}^{n} \delta_m^i,$$

$$\frac{\partial V_n}{\partial A_m} = \frac{1}{n}\sum_{i=1}^{n} \frac{\partial V^i}{\partial A_m} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{f}_{m-1}^i (\delta_m^i)^\top, \quad (20)$$

where $\delta_M^i = (y^i \nabla_{\mathbf{f}_M} V^i) \circ \sigma'(z_M^i)$, $\delta_m^i = \left(A_{m+1}^\top \delta_{m+1}^i\right) \circ \sigma'(z_m^i)$; $m = M-1, \ldots, 1$, and $\circ$ is the Hadamard product for matrices.

5.2.2. Gradients for Embedding

The gradient of V_n with respect to X is,

$$\frac{\partial V_n}{\partial X} = \frac{1}{n}\sum_{i=1}^n \frac{\partial V^i}{\partial X} = \frac{1}{n}\sum_{i=1}^n \frac{\partial V^i}{\partial z_1^i}\frac{\partial z_1^i}{\partial X} = A_1^\top\Big(\frac{1}{n}\sum_{i:s^i=s_1}\delta_1^i, \ldots, \frac{1}{n}\sum_{i:s^i=s_{|\mathcal{S}|}}\delta_1^i\Big) = A_1^\top \Delta_n^1. \quad (21)$$

The detail of projected gradient descent is summarized as follows.

Algorithm 2 (BGD of deep neural network classification):

Step 1 (Initialization): Initialize $X^{(0)}$ and specify the learning loss $V(\cdot)$, an U-embedding loss $U(\cdot)$ and a number of training iterations.

Step 2 (Gradient update): Update the neural network parameters Θ_N and embedding X by (23) and (22) via the back-propagation as follows.

Feed-forward: Compute $\mathbf{f}_0^i = \mathcal{X}(s^i)$, $z_m^i = A_m \mathbf{f}_{m-1}^i + b_m$, and $\mathbf{f}_m^i = \sigma(z_m^i)$; $i = 1, \ldots, n$, and $m = 1, \ldots, M$.

Back-propagation: Compute $\delta_M^i = (y^i \nabla_{\mathbf{f}_M^i} V^i) \circ \sigma'(z_M^i)$, $\delta_m^i = \left(A_{m+1}^\top \delta_{m+1}^i\right) \circ \sigma'(z_m^i)$; $i = 1, \ldots, n$, $m = M-1, \ldots, 1$.

Projected gradient descent: Based on (21), X in k-iteration is updated as

$$\begin{aligned} Z^{(k+1)} &= \mathcal{P}_{\mathcal{D}_C}(X^{(k)} - A_1^\top \Delta_n^1) \\ &= \operatorname*{argmin}_{\{Z:\mathrm{Tr}(ZQZ^\top)\le 2C\}} \|Z - X^{(k)} - A_1^\top \Delta_n^1\|_F^2, \\ X^{(k+1)} &= X^{(k)} + \gamma^{(k)}(Z^{(k+1)} - X^{(k)}), \end{aligned} \quad (22)$$

where $\gamma^{(k)}$ is a step size determined by Armijo's line search as shown in the Appendix, $\mathcal{P}_{\mathcal{D}_C}$ is the projection operator to $\mathcal{D}_C$, $Z^{(k)}$ is the projection for $X^{(k)} - A_1^\top \Delta^1$ into the convex set $\mathcal{D}_C$. Then we consider an equivalent regularization form of (22) with a regularization parameter λ_2 corresponding to C, where analytic expression can be computed. Then X and Θ_N in k-iteration are updated as

$$\begin{aligned} Z^{(k+1)} &= \operatorname*{argmin}_Z \Big(\|Z - X^{(k)} - A_1^\top \Delta_n^1\|_F^2 + \frac{\lambda_2}{2}\mathrm{Tr}(ZQZ^\top)\Big) \\ &= (X^{(k)} - A_1^\top \Delta_n^1)(I + \lambda_2 Q)^{-1}, \\ b_m^{(k+1)} &= b_m^{(k)} - \frac{\gamma^{(k)}}{n}\sum_{i=1}^n \delta_m^i, \quad A_m^{(k+1)} \\ &= A_m^{(k)} - \frac{\gamma^{(k)}}{n}\sum_{i=1}^n \mathbf{f}_i^{m-1}(\delta_m^i)^\top, m = M, M-1, \ldots, 1, \\ X^{(k+1)} &= X^{(k)} + \gamma^{(k)}(Z^{(k+1)} - X^{(k)}), \end{aligned} \quad (23)$$

where $(I_{|\mathcal{S}|} + \lambda_2 Q)^{-1}$ is pre-computed before the iterations.

Step 3 (Termination): Iterate Step 2 for the training iterations.

Note that the updates in (23) can be computed by a stochastic gradient scheme (Gelfand and Mitter 1991).

Lemma 4 (Convergence of Algorithm 2). A solution $(\widehat{\Theta}_N, \widehat{X})$ of Algorithm 2 is a local minimizer of (13).

6. Connections With Other Methods

An embedding learner in (7) has its distinct characteristics, although it may appear remotely related to supervised dimension reduction (Rish et al. 2008; Guo 2009), and multi-task learning in reinforcement learning (de Bruin et al. 2018). It differs from the aforementioned methods in that it is designed for unstructured data and leverages external unannotated data to reduce the estimation error through the embedding constraint, which collaboratively links the learning loss $L(f(\mathcal{X}(s)), y)$ to $U(\mathcal{X})$.

Next, we examine two methods, a two-stage method and the proposed method without the embedding loss constraint, particularly for unstructured data.

6.1. Two-Stage Method

A two-stage method uses an embedding $\widetilde{\mathcal{X}}$:

$$\min_{f\in\mathcal{F}} n^{-1}\sum_{i=1}^n \Big(V\big(f(\widetilde{\mathcal{X}}(s^i)), y^i\big)\Big) + \lambda J(f), \quad \widetilde{\mathcal{X}} = \operatorname*{argmin}_{\mathcal{X}} U(\mathcal{X}), \quad (24)$$

possibly with some additional standardization constraints to eliminate certain trivial embeddings, for example, $\frac{1}{|\mathcal{S}|}\sum_{u=1}^{|\mathcal{S}|} \mathcal{X}_j^2(s_u) = 1$ and $\sum_{u=1}^{|\mathcal{S}|} \mathcal{X}_j(s_u) = 0$; $j = 1, \ldots, p$, in the case of LLE. In short, (24) first produces $\widetilde{\mathcal{X}}$ from $U(\mathcal{X})$ at the first stage, and then proceeds with learning f given $\widetilde{\mathcal{X}}$ in the second stage. Critically, $\widetilde{\mathcal{X}}$ may not be sufficient or unrelated to the outcome of learning, and hence that the learning accuracy of (24) may be governed primarily by the approximation error of using $\widetilde{\mathcal{X}}$.

6.1.1. One-Hot Encoder

One-hot encoder (Weinberger et al. 2009) is a two-stage method, which maps s to $\mathcal{X}(s) = \big(I(s = s_1), \ldots, I(s = s_{|\mathcal{S}|})\big)^\top$ and then solves

$$\min_{\beta\in\mathbb{R}^{|\mathcal{S}|}} n^{-1}\sum_{i=1}^n \Big(V\big(\beta^\top \mathcal{X}(s^i), y^i\big)\Big) + \lambda J(\beta), \quad \mathcal{X}(s^i) = \big(I(s^i = s_1), \ldots, I(s^i = s_{|\mathcal{S}|})\big)^\top, \quad (25)$$

which treats each individual $s \in \mathcal{S}$ separately and ignores interactions among $s_1, \ldots, s_{|\mathcal{S}|}$ completely. As shown in Lemma 5, the one-hot encoder is more preferable among all two-stage methods in terms of prediction.

To understand the impact of the constraint $U(\mathcal{X}) \le C$ in (7), we consider (7) without the embedding constraint.

6.2. Embedding Learner Without Constraint

It solves the following problem:

$$\min_{f\in\mathcal{F},\mathcal{X}} n^{-1}\sum_{i=1}^n V\big(f(\mathcal{X}(s^i)), y^i\big) + \lambda J(f), \quad (26)$$

which can be thought of as adding an embedding layer of a neural network on the top of one-hot encoder.

Lemma 5 gives the adaptiveness of a two-stage method based on $\widetilde{\mathcal{X}}$.

Lemma 5. Consider linear regression in (24), where $V(f(\mathcal{X}(s)), y) = (f(\mathcal{X}(s)) - y)^2$ and $f(x) = \beta^\top x$. Let P be the distribution of (S, Y), $g = (g(s_1), \ldots, g(s_{|\mathcal{S}|}))^\top$ and $g(s) = \mathbb{E}(Y|S = s)$ be the optimal prediction function. Then, for any $P \in \mathcal{P} = \{P : \|g\|_2 \le 1\}$,

$$\inf_{\beta \in \mathbb{R}^p} \mathbb{E}\big(g(S) - \beta^\top \widetilde{\mathcal{X}}(S)\big)^2 = 0, \tag{27}$$

if and only if the rank of the embedding matrix $\widetilde{X}$ is $|\mathcal{S}|$, where $\widetilde{X} = (\widetilde{\mathcal{X}}(s_1), \ldots, \widetilde{\mathcal{X}}(s_{|\mathcal{S}|}))$ is a $p \times |\mathcal{S}|$ matrix.

As suggested by Lemma 5, for an embedding $\widetilde{\mathcal{X}}$, (27) is a minimal requirement. If it is violated, then $\widetilde{\mathcal{X}}$ incurs a bias, that is, there exists a (S, Y) cannot be approximated by a linear function of $\widetilde{\mathcal{X}}$, and the consistency may not hold. On this ground, the approximation error is zero implies that the dimension of $\widetilde{\mathcal{X}}$ is no less than $|\mathcal{S}|$, or $p \ge |\mathcal{S}|$, which yields a slow rate of convergence in the L_2-loss due to the high dimensionality. On the other hand, if $p < |\mathcal{S}|$, the approximation error of the embedding in the first stage is always bounded away from zero regardless of the sample size, leading to inconsistent estimation. In other words, $p = |\mathcal{S}|$ yields the best convergence rate for a two-stage method, one particular example is the one-hot encoder, which has a rate $O_P\big(\big(\frac{|\mathcal{S}|}{n}\big)^{\frac{\mu}{2-\kappa}}\big)$ for linear regression or classification.

As illustrated in Table 1, the embedding learner and one-hot encoder are more preferred than the embedding learner without the embedding loss constraint, implying that the embedding constraint $U(\mathcal{X}) \le C$ is indeed critical for the embedding learner because it helps reduce the estimation complexity. Additionally, the embedding learner is more preferable than the one-hot encoder, when the effective dimension K^* defined in (1) is not too large. Furthermore, the dimension of U-minimal sufficient embedding $\mathcal{X}^*$ is usually smaller than $|\mathcal{S}|$, yielding a more compact embedding than the one-hot encoder.

7. Numerical Examples

This section examines the numerical performance of an embedding learner in two applications, grammatical classification in Wikipedia network[3] (Grover and Leskovec 2016) and sentiment analysis of movie reviews[4] (Pang and Lee 2005). In particular, we compare an embedding learner in (10) against a two-stage method in (24) based on four different embedding methods, LLE, Laplacian eigenmaps (LAE), *Doc2Vec*, and *Word2Vec*, in linear and nonlinear classification. Moreover, tuning parameters (λ_1, λ_2) and p in Algorithms 1 and 5.2.2 can be optimized by cross-validation on a validation set, as in Tibshirani (1996), Mazumder, Hastie, and Tibshirani (2010), and Hastie, Friedman, and Tibshirani (2009). This is achieved by minimizing the misclassification error on a validation set with respect to the tuning parameters.

7.1. Grammatical Classification in Wikipedia Network

A co-occurrence network is commonly used in representational learning to provide a graphic visualization of potential relations between interacting units such as organizations, concepts, organisms like bacteria, and other entities. One of its applications is to predict the outcome of one unit by utilizing its co-occurrence relations between other units characterized by the network, for example, automatic grammatical classification of new words or phrases through word relations. In this situation, co-occurrence means an above-chance frequency of occurrence of two words or phrases, indicating semantic proximity or idiomatic expressions. The data consist of a graph with 4777 nodes and 184,812 edges, where each node corresponds to one word or phrase and an edge between two nodes represents their co-occurrence relations. The label Y is defined by if a word is tagged as a specific grammatical category by a benchmark tagger, Stanford POS-Tagger (Toutanova et al. 2003).

In this application, we examine the performance for embedding methods, based on linear SVM classification with the hinge loss $V(f, y) = V(fy) = (1 - yf)_+$, $f(x) = \beta^\top x$, and the U-embedding loss as LLE or LAE. For two-stage methods, LLE produces numerical embeddings as the bottom $p + 1$ eigenvectors of $Q = (I_{|\mathcal{S}|} - W)^\top (I_{|\mathcal{S}|} - W)$ except the eigenvectors corresponding to the smallest eigenvalue. LAE provides numerical embeddings as the partial eigenvalue decomposition on graph Laplacian of W. Here $W = (w_{uv}) \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$ is given and w_{uv} is the frequency of occurrence of s_u and s_v.

For implementation, we code Algorithms 1 and 1a for an embedding learner (10) and the two-stage method (24) in Python and utilize a Python package for machine learning *scikit-learn*.[5]

To evaluate the performance of each method, we randomly split the dataset into three sets of training, validation, and testing with respect to the nodes for a partition rate: 50%, 20%, and 30%. Then we minimize the misclassification error on the validation set, followed by an evaluation of a classifier on the test set. For an embedding learner, we employ a grid search to estimate two tuning parameters λ_1 and λ_2 in (16) of Algorithms 1 and 1a, where 60 grid points of $\lambda_j; j = 1, 2$ are chosen as $\{10^{(v-31)/10}; v = 1, 3, \ldots, 61\}$. For the two-stage method, λ is tuned through a grid search. Finally, the average test errors of the two methods based on 50 random replications are reported in Table 2 for $p = 5, 10, 50, 100, 200$.

As suggested by Table 2, blockwise coordinate descent with an inexact update (iBCD) in Algorithm 1a and gradient descent (GD) have similar performance, which slightly outperform the blockwise coordinate descent with an exact update (BCD) in Algorithm 1. Moreover, the embedding learner outperforms the two-stage method in all cases. For a fixed p, the largest amount of improvement on the two-stage method is $(0.456 - 0.195)/0.456 = 57.2\%$ when $p = 10$ for LLE, and $(0.462 - 0.193)/0.462 = 58.2\%$ for LAE. For the best performance across $p = 5, 10, 50, 100, 200$, the amount of improvement becomes $(0.434 - 0.195)/0.434 = 55.1\%$ for LLE, and $(0.378 - 0.193)/0.378 = 48.9\%$ for LAE. The best performance of the proposed method is achieved at $p = 10$, which is either better or

[3] https://snap.stanford.edu/node2vec/#datasets

[4] https://www.cs.cornell.edu/people/pabo/movie-review-data/

[5] https://scikit-learn.org/stable/

Table 2. Test errors (standard errors in parentheses) of the embedding learner in (10) and the two-stage method (24) for SVM classification of the Wikipedia network data based on 50 random partitions, denoted by EL and two-stage.

$p=5$	$p=10$		$p=50$	$p=100$	$p=200$
EL (BCD): SVM+LLE	0.198(0.002)	**0.195(0.002)**	0.198(0.002)	0.201(0.002)	0.202(0.002)
EL (iBCD): SVM+LLE	0.197(0.002)	**0.195(0.002)**	0.197(0.002)	0.197(0.002)	0.195(0.002)
EL (GD): SVM+LLE	0.197(0.002)	**0.195(0.002)**	0.197(0.002)	0.197(0.002)	0.195(0.002)
Two-stage: SVM+LLE	0.453(0.003)	0.456(0.003)	0.442(0.002)	**0.434(0.003)**	0.435(0.002)
EL (BCD): SVM+LAE	0.198(0.001)	0.199(0.001)	0.197(0.002)	**0.196(0.001)**	0.196(0.002)
EL (iBCD): SVM+LAE	0.197(0.001)	**0.193(0.002)**	0.196(0.002)	0.197(0.002)	0.197(0.002)
EL (GD): SVM+LAE	0.197(0.002)	0.197(0.002)	0.196(0.002)	**0.195(0.002)**	0.195(0.002)
Two-stage: SVM+LAE	0.381(0.002)	**0.378(0.013)**	0.465(0.004)	0.462(0.004)	0.445(0.002)

NOTE: Here locally liner embedding loss (LLE) and Laplacian eigenmaps (LAE) are employed, and EL is implemented by gradient descent (GD), and blockwise coordinate descent with an exact update (BCD) in Algorithm 1 and with inexact update (iBCD) in Algorithm 1a. The best performer is marked in bold.

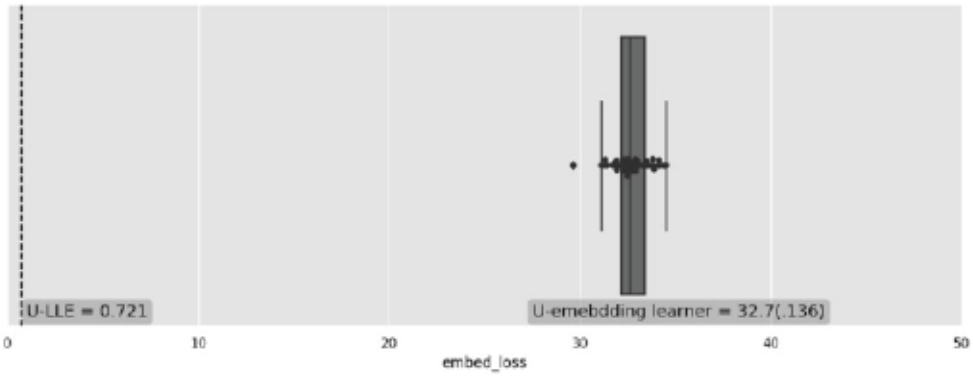

Figure 2. Boxplot of the U-embedding loss (LLE) values for an embedding estimated by an embedding learner based on 50 random partitions (right) in grammatical classification as well as those values for the two-stage method (LLE) as a reference.

comparable to that at $p = 50, 100, 200$. The amount of improvement of the best performance of the embedding learner over that the two-stage method that is tuned for $p = 5, 10, 50, 100, 200$ is $(0.456 - 0.195)/0.378 = 57.2\%$ and $(0.378 - 0.199)/0.378 = 47.4\%$ for both LLE and LAE eigenmaps, respectively.

Moreover, as illustrated in Figure 2, although the U-embedding loss (LLE) value for a two-stage method appears much smaller than that for the embedding learner (proposed), the two-stage method performs much worse in the learning accuracy, as reported in Table 2. In other words, a minimizer of an U-embedding loss does not necessarily render a good embedding for a specific task of learning. By comparison, the embedding learner not only produces a low-dimensional and task-adaptive embedding but also delivers a higher prediction accuracy than its two-stage competitor.

7.2. Sentiment Analysis in Movie Reviews

Sentiment analysis identifies sentiment from textual data toward a specific event of interest. Now we examine the movie review data, which is available at polarity dataset v2.0.[6] This data consist of 1000 positive and 1000 negative processed movie text reviews, serving as a positive or negative label for sentiment, where each text review maybe comprised of hundreds of words and phrases. For this data, we compare the neural network embedding classifier with its two-stage counterpart, based on U-embedding losses, including LLE, LAE, and a transfer embedding based on *GoogleNews*.[7] Moreover, we also scrutinize two-stage methods based on six embeddings LLE, LAE, *Bag-of-words*, *Doc2Vec*, *Word2Vec*, and embeddings built on *GoogleNews*.

To construct embeddings, we preprocess text reviews by removing stop words, and use *Gensim*[8] to construct embeddings $\widetilde{\mathcal{X}}$ for *Doc2Vec*, *Word2Vec*, and use pre-trained embeddings *GoogleNews* based on a Google news database of about 100 billion words. For *Word2Vec* and *GoogleNews*, text review embeddings are generated by averaging their associated numerical vectors of sentiment words identified in Hu and Liu (2004). For *Bag-of-words*, embeddings are generated by the number of occurrences of each word in the given text review against a domain dictionary, and it becomes an embedding learner with constraint when $M \geq 1$.

For this data, the weight matrix W is implicit, and hence that we obtain a rough estimate based on Figure 3, that is we set $w_{uv} = 1$ if the number of positive words exceeds that of negative words by 12 or the number of negative words exceeds that of positive words by 4 in a text, where positive words and negative words are identified by Hu and Liu (2004), and assigning 0, otherwise. Next, LLE and LAE are produced based on the weight W, and the transfer embedding loss $U(\mathcal{X}) = \sum_{u=1}^{|\mathcal{S}|} \|\mathcal{X}(s_u) - \widetilde{\mathcal{X}}(s_u)\|_2^2$ is used to incorporate the information about pre-trained embedding, where $\widetilde{\mathcal{X}}$ is obtained from *GoogleNews* when $p = 300$.

[6] *http://www.cs.cornell.edu/people/pabo/movie-review-data/poldata.README.2.0.txt*

[7] *https://code.google.com/archive/p/word2vec/*

[8] *https://radimrehurek.com/gensim/*

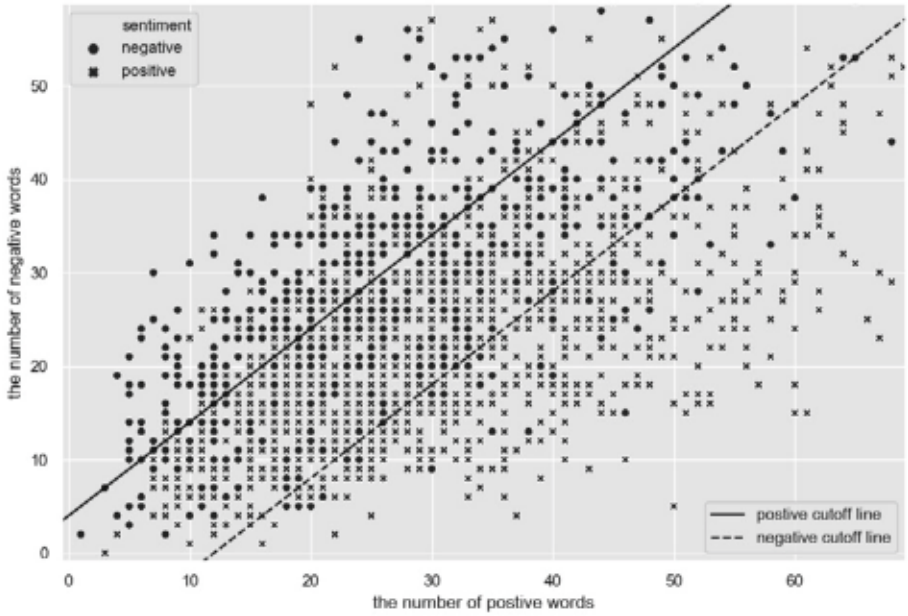

Figure 3. Numbers of positive and negative words in labeled text reviews. Reviews tend to be negative when the number of negative words exceeds that of positive words by 4, whereas it is the opposite when the number of positive words exceeds that of negative words by 12.

In this application, we construct a feed-forward neural network of a fixed number of nodes per layer $h_0 = p, h_1 = \cdots = h_{M-1} = 128$, $h_M = 1$ and four different depths of hidden layer $M = 0, 1, 2, 4$, where $M = 0$ indicates the network of no hidden layers as a linear SVM. In what is to follow, we consider four different embedding dimensions $p = 20, 50, 100, 300$ given that the dimension of the pre-trained embedding generated from GoogleNew is fixed at 300.

For implementation,[9] we code Algorithm 2 for the embedding learner (10) and the two-stage method (24) as well as *Bag-of-words* in *TensorFlow*,[10] with 500 number of steps and a learning rate decay as Adam. In this case, the learning loss is the hinge loss, ReLU and linear the activation function for each hidden layer and the output layer.

For evaluation, we randomly split the dataset into three sets of training, validation, and testing for a partition ratio: 50%, 20%, and 30%, Then we minimize the misclassification error on the validation set, followed by an evaluation of a classifier on the test set. For the embedding learner, we fix $\lambda_1 = 0$ with no regularization and employ a grid search for the validation set to estimate the optimal λ_2 for Algorithm 2, where the grid is set as $\{0.001, 0.01, 0.1, 1, 5, 10, 15\}$. The average misclassification errors of all methods are reported in Table 3.

As indicated in Table 3, the embedding learner based on LLE performs among all the methods for $p = 20, 50, 100, 300$. In this case, the amounts of improvement on its competitors two-stage methods based on LLE, LAE, transfer embedding, *Doc2Vec*, *Word2Vec*, and *Bag-of-words* are $(0.407 - 0.188)/0.407 = 54\%$, $(0.329 - 0.188)/0.329 = 42.9\%$, $(0.246 - 0.188)/0.246 = 24\%$, $(0.290 - 0.188)/0.290 = 35\%$, $(0.257 - 0.188)/0.257 = 27\%$, and $(0.352 - 0.188)/0.352 = 47\%$. Moreover, the embedding learner outperforms its two-stage counterpart for all embedding sizes $p = 20, 50, 100, 300$.

It is worth mentioning that the two-stage method based on the *Word2Vec-GoogleNews* transfer embedding delivers higher performance than its counterpart *Word2Vec* without using it across all the cases. This suggests that unannotated data plays a significant role. Overall, the embedding learner not only yields a higher prediction accuracy for unstructured predictors but also produces more compact and task-specific embeddings leading to parsimonious function representations.

8. Conclusion and Summary

This article introduces an embedding learning framework for unstructured data, which constructs a learning-specific embedding from large external unannotated data. The proposed method uses a concept of U-minimal sufficient learning-adaptive embeddings and links a specific learning task to a class of sufficient embeddings through a constraint defined by an U-embedding loss. When applying the proposed embedding learning framework to classification, we derive a graph embedding classifier for unstructured data analysis. Numerically, we design algorithms based on blockwise coordinate descent and projected gradient descent for various implementations including feed-forward neural networks. Theoretically, we establish a learning theory to demonstrate that the proposed embedding learner is advantageous to its two-stage counterpart in terms of the learning accuracy. Moreover, we also show that in linear regression the one-hot encoder is more preferable among various embeddings in a two-stage method, yet its dimension restriction hinders its prediction performance. Although the advantage of the proposed embedding leaner over its two-stage counterpart has been demonstrated empirically and theoretically, further investigation may be necessary. In particular, the issue of how to choose an optimal U-embedding loss for a specific learning task deserves attention.

[9] A simple demonstration for embedding learning is provided in Github (*https://github.com/statmlben/embedding-learning*).

[10] *https://www.tensorflow.org/*

Table 3. Test errors (standard errors in parentheses) of the embedding learner in (10) and the corresponding two-stage methods in (24) for deep SVM neural network classification with a ReLU for the sentence polarity movie-review data, based on 50 random partitions.

Method	Embed-dim	$M=0$	$M=1$	$M=2$	$M=4$
EL: SVM+LLE	$p=20$	0.335(0.003)	0.306(0.005)	0.293(0.006)	0.341(0.008)
	$p=50$	0.317(0.003)	0.270(0.004)	0.276(0.008)	0.319(0.006)
	$p=100$	0.324(0.004)	0.266(0.005)	0.249(0.006)	0.331(0.004)
	$p=300$	0.209(0.003)	**0.188(0.003)**	0.215(0.004)	0.218(0.005)
EL: SVM+GoogleNews	$p=300$	0.217(0.002)	**0.216(0.002)**	0.217(0.005)	0.226(0.005)
Two-stage: SVM+LLE	$p=20$	0.505(0.003)	0.494(0.004)	0.439(0.018)	0.471(0.015)
	$p=50$	0.499(0.003)	0.477(0.004)	0.425(0.017)	0.451(0.017)
	$p=100$	0.500(0.003)	0.466(0.008)	0.427(0.016)	0.428(0.019)
	$p=300$	0.490(0.005)	0.450(0.012)	**0.407(0.018)**	0.416(0.017)
Two-stage: SVM+LAE	$p=20$	0.503(0.001)	0.503(0.001)	0.449(0.005)	0.419(0.002)
	$p=50$	0.502(0.001)	0.501(0.001)	0.376(0.001)	0.367(0.001)
	$p=100$	0.504(0.001)	0.485(0.004)	0.353(0.001)	0.344(0.001)
	$p=300$	0.504(0.001)	0.400(0.005)	**0.329(0.001)**	0.332(0.001)
Two-stage: SVM+GoogleNews (Socher et al. 2013)	$p=300$	0.265(0.013)	**0.246(0.002)**	0.271(0.002)	0.295(0.003)
Two-stage: SVM+Doc2Vec (Le and Mikolov 2014)	$p=20$	0.434(0.014)	0.356(0.005)	0.354(0.004)	0.365(0.004)
	$p=50$	0.412(0.017)	0.309(0.003)	0.331(0.003)	0.348(0.005)
	$p=100$	0.332(0.014)	**0.290(0.004)**	0.329(0.004)	0.353(0.004)
	$p=300$	0.310(0.009)	0.294(0.004)	0.343(0.005)	0.348(0.004)
Two-stage: SVM+Word2Vec (Socher et al. 2013)	$p=20$	0.491(0.010)	0.338(0.004)	0.355(0.005)	0.367(0.005)
	$p=50$	0.473(0.009)	0.304(0.005)	0.331(0.005)	0.338(0.005)
	$p=100$	0.420(0.008)	0.293(0.004)	0.316(0.004)	0.340(0.006)
	$p=300$	0.353(0.024)	**0.257(0.005)**	0.310(0.005)	0.344(0.004)
SVM+Bag-of-words (Weinberger et al. 2009) SVM+ELN ($M \geq 1$)	$p=\lvert\mathcal{S}\rvert=3799$	**0.352(0.004)**	0.370(0.003)	0.386(0.003)	0.416(0.003)

NOTE: Here "EL," "ELN," and "two-stage" denote the embedding learner, that without constraint, and the two-stage method. Six embedding losses are considered, including LLE, LAE, the transfer embedding utilizing *GoogleNews*, *Doc2Vec* (Le and Mikolov 2014), *Word2Vec* (Mikolov et al. 2013), and *Bag-of-words* (Weinberger et al. 2009). For the neural network, the number of nodes per layer is 128 and the depth of the neural network is $M = 0, 1, 2, 4$. Note that the embedding value for *GoogleNews* is only available. Moreover, the results for embedding learner with transfer embedding loss on *Doc2Vec* (Le and Mikolov 2014) and *Word2Vec* (Mikolov et al. 2013) are worse than that on *GoogleNews*, and thus is omitted due to a space constraint. The best performance in each case is marked in bold.

Supplementary Materials

The supplementary materials provide proofs of theorems and Python codes used in real data application.

Acknowledgments

The authors thank the editors, the associate editor, and three anonymous referees for helpful comments and suggestions.

Funding

Research supported in part by NSF grants DMS-1712564, DMS-1721216, DMS-1952539, NIH funding: 1R01GM126002, R01HL105397, and HK RGC grants GRF-11303918, GRF-11300919.

References

Bădoiu, M., Demaine, E. D., Hajiaghayi, M., Sidiropoulos, A., and Zadimoghaddam, M. (2008), "Ordinal Embedding: Approximation Algorithms and Dimensionality Reduction," in *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, eds. A. Goel, K. Jansen, J. D. P. Rolim, and R. Rubinfeld, Berlin, Heidelberg: Springer, pp. 21–34. [309]

Belkin, M., and Niyogi, P. (2002), "Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering," in *Advances in Neural Information Processing Systems*, pp. 585–591. [307,308,309]

Berge, C. (1973), *Graphs and Hypergraphs*, Rome: Food and Agriculture Organization of the United States. [308]

Casella, G., and Berger, R. L. (2002), *Statistical Inference*, Pacific Grove, CA: Duxbury. [309]

Chen, B., He, S., Li, Z., and Zhang, S. (2012), "Maximum Block Improvement and Polynomial Optimization," *SIAM Journal on Optimization*, 22, 87–107. [312]

Cortes, C., and Vapnik, V. (1995), "Support-Vector Networks," *Machine Learning*, 20, 273–297. [307,310,311]

Covington, P., Adams, J., and Sargin, E. (2016), "Deep Neural Networks for YouTube Recommendations," in *Proceedings of the 10th ACM Conference on Recommender Systems*, ACM, pp. 191–198. [307]

de Bruin, T., Kober, J., Tuyls, K., and Babuška, R. (2018), "Integrating State Representation Learning Into Deep Reinforcement Learning," *IEEE Robotics and Automation Letters*, 3, 1394–1401. [314]

Fan, R. E., Chang, K. W., Hsieh, C. J., Wang, X. R., and Lin, C. J. (2008), "LIBLINEAR: A Library for Large Linear Classification," *Journal of Machine Learning Research*, 9, 1871–1874. [312]

Fellbaum, C. (2010), "WordNet," in *Theory and Applications of Ontology: Computer Applications*, eds. R. Poli, M. Healy, and A. Kameas, Dordrecht: Springer, pp. 231–243. [309]

Gelfand, S. B., and Mitter, S. K. (1991), "Recursive Stochastic Algorithms for Global Optimization in $\mathbb{R}^d$," *SIAM Journal on Control and Optimization*, 29, 999–1018. [314]

Genkin, A., Lewis, D. D., and Madigan, D. (2007), "Large-Scale Bayesian Logistic Regression for Text Categorization," *Technometrics*, 49, 291–304. [307]

Glorot, X., Bordes, A., and Bengio, Y. (2011), "Deep Sparse Rectifier Neural Networks," in *International Conference on Artificial Intelligence and Statistics*, pp. 315–323. [311]

Grover, A., and Leskovec, J. (2016), "node2vec: Scalable Feature Learning for Networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, pp. 855–864. [307,309,315]

Guo, Y. (2009), "Supervised Exponential Family Principal Component Analysis via Convex Optimization," in *Advances in Neural Information Processing Systems*, pp. 569–576. [314]

Hastie, T., Friedman, J., and Tibshirani, R. (2009), *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, New York: Springer. [310,315]

He, H., Balakrishnan, A., Eric, M., and Liang, P. (2017), "Learning Symmetric Collaborative Dialogue Agents With Dynamic Knowledge Graph Embeddings," arXiv no. 1704.07130. [309]

Hu, M., and Liu, B. (2004), "Mining and Summarizing Customer Reviews," in *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, pp. 168–177. [316]

Le, Q., and Mikolov, T. (2014), "Distributed Representations of Sentences and Documents," in *International Conference on Machine Learning*, pp. 1188–1196. [307,318]

Lee, J. D., Simchowitz, M., Jordan, M. I., and Recht, B. (2016), "Gradient Descent Only Converges to Minimizers," in *Conference on Learning Theory*, pp. 1246–1257. [313]

Linnainmaa, S. (1976), "Taylor Expansion of the Accumulated Rounding Error," *BIT Numerical Mathematics*, 16, 146–160. [313]

Mazumder, R., Hastie, T., and Tibshirani, R. (2010), "Spectral Regularization Algorithms for Learning Large Incomplete Matrices," *Journal of Machine Learning Research*, 11, 2287–2322. [310,315]

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013), "Distributed Representations of Words and Phrases and Their Compositionality," in *Advances in Neural Information Processing Systems*, pp. 3111–3119. [307,309,318]

Mikolov, T., Yih, W.-T., and Zweig, G. (2013), "Linguistic Regularities in Continuous Space Word Representations," in *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 746–751. [307]

Miratrix, L., Jia, J., Gawalt, B., Yu, B., and Ghaoui, L. E. (2011), "Summarizing Large-Scale, Multiple-Document News Data: Sparse Methods and Human Validation." [307]

Niyogi, P., and Girosi, F. (1996), "On the Relationship Between Generalization Error, Hypothesis Complexity, and Sample Complexity for Radial Basis Functions," *Neural Computation*, 8, 819–842. [310]

Pang, B., and Lee, L. (2005), "Seeing Stars: Exploiting Class Relationships for Sentiment Categorization With Respect to Rating Scales," in *Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics, pp. 115–124. [315]

Pennington, J., Socher, R., and Manning, C. (2014), "Glove: Global Vectors for Word Representation," in *Proceeding of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1532–1543. [307,309]

Pratt, L. Y. (1993), "Discriminability-Based Transfer Between Neural Networks," in *Advances in Neural Information Processing Systems*, pp. 204–211. [307]

Rish, I., Grabarnik, G., Cecchi, G. A., Pereira, F., and Gordon, G. J. (2008), "Closed-Form Supervised Dimensionality Reduction With Generalized Linear Models," in *ICML* (Vol. 8), pp. 832–839. [314]

Rockafellar, R., and Wets, R. (2011), *Variational Analysis* (Vol. 317), Berlin, Heidelberg: Springer. [312]

Roweis, S. T., and Saul, L. K. (2000), "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science*, 290, 2323–2326. [307,308,309]

Schmidt-Hieber, J. (2017), "Nonparametric Regression Using Deep Neural Networks With ReLU Activation Function," arXiv no. 1708.06633. [312]

Shen, X., Tseng, G. C., Zhang, X., and Wong, W. H. (2003), "On ψ-Learning," *Journal of the American Statistical Association*, 98, 724–734. [310,311,312]

Shen, X., and Wang, L. (2007), "Generalization Error for Multi-Class Margin Classification," *Electronic Journal of Statistics*, 1, 307–330. [310,311,312]

Shen, X., and Wong, W. H. (1994), "Convergence Rate of Sieve Estimates," *The Annals of Statistics*, 22, 580–615. [311]

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013), "Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank," in *Proceedings of 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642. [307,318]

Stark, C., Breitkreutz, B.-J., Chatr-Aryamontri, A., Boucher, L., Oughtred, R., Livstone, M. S., Nixon, J., Van Auken, K., Wang, X., Shi, X., and Reguly, T. (2010), "The BioGRID Interaction Database: 2011 Update," *Nucleic Acids Research*, 39, D698–D704. [309]

Taddy, M. (2013), "Multinomial Inverse Regression for Text Analysis," *Journal of the American Statistical Association*, 108, 755–770. [307]

Tibshirani, R. (1996), "Regression Shrinkage and Selection via the Lasso," *Journal of the Royal Statistical Society*, Series B, 58, 267–288. [310,315]

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003), "Feature-Rich Part-of-Speech Tagging With a Cyclic Dependency Network," in *Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, Association for Computational Linguistics, pp. 173–180. [315]

Tsybakov, A. B. (2004), "Optimal Aggregation of Classifiers in Statistical Learning," *The Annals of Statistics*, 32, 135–166. [312]

Udell, M., Horn, C., Zadeh, R., and Boyd, S. (2016), "Generalized Low Rank Models," *Foundations and Trends® in Machine Learning*, 9, 1–118. [313]

Vapnik, V. (1998), *Statistical Learning Theory* (Vol. 3), New York: Wiley. [311]

Wang, J., Shen, X., Sun, Y., and Qu, A. (2016), "Classification With Unstructured Predictors and an Application to Sentiment Analysis," *Journal of the American Statistical Association*, 111, 1242–1253. [307]

Weinberger, K., Dasgupta, A., Attenberg, J., Langford, J., and Smola, A. (2009), "Feature Hashing for Large Scale Multitask Learning," arXiv no. 0902.2206. [308,314,318]

Xu, Y., and Wang, X. (2018), "Understanding Weight Normalized Deep Neural Networks With Rectified Linear Units," in *Advances in Neural Information Processing Systems*, pp. 130–139. [312]

Yarotsky, D. (2017), "Error Bounds for Approximations With Deep ReLU Networks," *Neural Networks*, 94, 103–114. [310,311]

Zhu, J., and Hastie, T. (2002), "Kernel Logistic Regression and the Import Vector Machine," in *Advances in Neural Information Processing Systems*, pp. 1081–1088. [310]