Stochastic Shared Embeddings: Data-driven Regularization of Embedding Layers

Liwei Wu
Department of Statistics
University of California, Davis
Davis, CA 95616
liwu@ucdavis.edu

Shuqing Li
Department of Computer Science
University of California, Davis
Davis, CA 95616
qshli@ucdavis.edu

Cho-Jui Hsieh
Department of Computer Science
University of California, Los Angles
Los Angles, CA 90095
chohsieh@cs.ucla.edu

James Sharpnack
Department of Statistics
University of California, Davis
Davis, CA 95616
jsharpna@ucdavis.edu

In deep neural nets, lower level embedding layers account for a large portion of the total number of parameters. Tikhonov regularization, graph-based regularization, and hard parameter sharing are approaches that introduce explicit biases into training in a hope to reduce statistical complexity. Alternatively, we propose stochastic shared embeddings (SSE), a data-driven approach to regularizing embedding layers, which stochastically transitions between embeddings during stochastic gradient descent (SGD). Because SSE integrates seamlessly with existing SGD algorithms, it can be used with only minor modifications when training large scale neural networks. We develop two versions of SSE: SSE-Graph using knowledge graphs of embeddings; SSE-SE using no prior information. We provide theoretical guarantees for our method and show its empirical effectiveness on 6 distinct tasks, from simple neural networks with one hidden layer in recommender systems, to the transformer and BERT in natural languages. We find that when used along with widely-used regularization methods such as weight decay and dropout, our proposed SSE can further reduce overfitting, which often leads to more favorable generalization results.

1 Introduction

Recently, embedding representations have been widely used in almost all AI-related fields, from feature maps [13] in computer vision, to word embeddings [15, 20] in natural language processing, to user/item embeddings [17, 10] in recommender systems. Usually, the embeddings are high-dimensional vectors. Take language models for example, in GPT [22] and Bert-Base model [3], 768-dimensional vectors are used to represent words. Bert-Large model utilizes 1024-dimensional vectors and GPT-2 [23] may have used even higher dimensions in their unreleased large models. In recommender systems, things are slightly different: the dimension of user/item embeddings are usually set to be reasonably small, 50 or 100, but the number of users and items is on a much bigger scale. Contrast this with the fact that the size of word vocabulary that normally ranges from 50,000 to 150,000, the number of users and items can be millions or even billions in large-scale real-world commercial recommender systems [1].

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Given the massive number of parameters in modern neural networks with embedding layers, mitigating over-parameterization can play an important role in preventing over-fitting in deep learning. We propose a regularization method, Stochastic Shared Embeddings (SSE), that uses prior information about similarities between embeddings, such as semantically and grammatically related words in natural languages or real-world users who share social relationships. Critically, SSE progresses by stochastically transitioning between embeddings as opposed to a more brute-force regularization such as graph-based Laplacian regularization and ridge regularization. Thus, SSE integrates seamlessly with existing stochastic optimization methods and the resulting regularization is data-driven.

We will begin the paper with the mathematical formulation of the problem, propose SSE, and provide the motivations behind SSE. We provide a theoretical analysis of SSE that can be compared with excess risk bounds based on empirical Rademacher complexity. We then conducted experiments for a total of 6 tasks from simple neural networks with one hidden layer in recommender systems, to the transformer and BERT in natural languages and find that when used along with widely-used regularization methods such as weight decay and dropout, our proposed methods can further reduce over-fitting, which often leads to more favorable generalization results.

2 Related Work

Regularization techniques are used to control model complexity and avoid over-fitting. ℓ_2 regularization [8] is the most widely used approach and has been used in many matrix factorization models in recommender systems; ℓ_1 regularization [29] is used when a sparse model is preferred. For deep neural networks, it has been shown that ℓ_p regularizations are often too weak, while dropout [7, 27] is more effective in practice. There are many other regularization techniques, including parameter sharing [5], max-norm regularization [26], gradient clipping [19], etc.

Our proposed SSE-graph is very different from graph Laplacian regularization [2], in which the distances of any two embeddings connected over the graph are directly penalized. Hard parameter sharing uses one embedding to replace all distinct embeddings in the same group, which inevitably introduces a significant bias. Soft parameter sharing [18] is similar to the graph Laplacian, penalizing the l_2 distances between any two embeddings. These methods have no dependence on the loss, while the proposed SSE-graph method is data-driven in that the loss influences the effect of regularization. Unlike graph Laplacian regularization, hard and soft parameter sharing, our method is stochastic by nature. This allows our model to enjoy similar advantages as dropout [27].

Interestingly, in the original BERT model's pre-training stage [3], a variant of SSE-SE is already implicitly used for token embeddings but for a different reason. In [3], the authors masked 15% of words and 10% of the time replaced the [mask] token with a random token. In the next section, we discuss how SSE-SE differs from this heuristic. Another closely related technique to ours is the label smoothing [28], which is widely used in the computer vision community. We find that in the classification setting if we apply SSE-SE to one-hot encodings associated with output y_i only, our SSE-SE is closely related to the label smoothing, which can be treated as a special case of our proposed method.

3 Stochastic Shared Embeddings

Throughout this paper, the network input x_i and label y_i will be encoded into indices $j_1^i, \ldots, j_M^i$ which are elements of $\mathcal{I}_1 \times \ldots \mathcal{I}_M$, the index sets of embedding tables. A typical choice is that the indices are the encoding of a dictionary for words in natural language applications, or user and item tables in recommendation systems. Each index, j_l, within the lth table, is associated with an embedding $E_l[j_l]$ which is a trainable vector in $\mathbb{R}^{d_l}$. The embeddings associated with label y_i are usually non-trainable one-hot vectors corresponding to label look-up tables while embeddings associated with input x_i are trainable embedding vectors for embedding look-up tables. In natural language applications, we appropriately modify this framework to accommodate sequences such as sentences.

The loss function can be written as the functions of embeddings:

$$R_n(\Theta) = \sum_i \ell(x_i, y_i|\Theta) = \sum_i \ell(E_1[j_1^i], \ldots, E_M[j_M^i]|\Theta), \tag{1}$$

Algorithm 1 SSE-Graph for Neural Networks with Embeddings

Input: input x_i, label y_i, backpropagate T steps, mini-batch size m, knowledge graphs on embeddings $\{E_1, \ldots, E_M\}$
Define $p_l(.,.|\Phi)$ based on knowledge graphs on embeddings, $l = 1, \ldots, M$
for $t = 1$ **to** T **do**
  Sample one mini-batch $\{x_1, \ldots, x_m\}$
  for $i = 1$ **to** m **do**
    Identify the set of embeddings $\mathcal{S}_i = \{E_1[j_1^i], \ldots, E_M[j_M^i]\}$ for input x_i and label y_i
    for each embedding $E_l[j_l^i] \in \mathcal{S}_i$ **do**
      Replace $E_l[j_l^i]$ with $E_l[k_l]$, where $k_l \sim p_l(j_l^i, .|\Phi)$
    end for
  end for
  Forward and backward pass with the new embeddings
end for
Return embeddings $\{E_1, \ldots, E_M\}$, and neural network parameters Θ

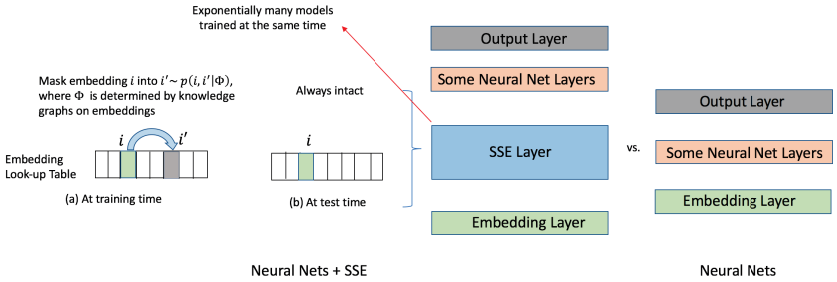

Figure 1: SSE-Graph described in Algorithm 1 and Figure 2 can be viewed as adding exponentially many distinct reordering layers above the embedding layer. A modified backpropagation procedure in Algorithm 1 is used to train exponentially many such neural networks at the same time.

where y_i is the label and Θ encompasses all trainable parameters including the embeddings, $\{E_l[j_l] : j_l \in \mathcal{I}_l\}$. The loss function ℓ is a mapping from embedding spaces to the reals. For text input, each $E_l[j_l^i]$ is a word embedding vector in the input sentence or document. For recommender systems, usually there are two embedding look-up tables: one for users and one for items [6]. So the objective function, such as mean squared loss or some ranking losses, will comprise both user and item embeddings for each input. We can more succinctly write the matrix of all embeddings for the ith sample as $\mathbf{E}[\mathbf{j}^i] = (E_1[j_1^i], \ldots, E_M[j_M^i])$ where $\mathbf{j}^i = (j_1^i, \ldots, j_M^i) \in \mathcal{I}$. By an abuse of notation we write the loss as a function of the embedding matrix, $\ell(\mathbf{E}[\mathbf{j}^i]|\Theta)$.

Suppose that we have access to knowledge graphs [16, 14] over embeddings, and we have a prior belief that two embeddings will share information and replacing one with the other should not incur a significant change in the loss distribution. For example, if two movies are both comedies and they are starred by the same actors, it is very likely that for the same user, replacing one comedy movie with the other comedy movie will result in little change in the loss distribution. In stochastic optimization, we can replace the loss gradient for one movie's embedding with the other similar movie's embedding, and this will not significantly bias the gradient if the prior belief is accurate. On the other hand, if this exchange is stochastic, then it will act to smooth the gradient steps in the long run, thus regularizing the gradient updates.

3.1 General SSE with Knowledge Graphs: SSE-Graph

Instead of optimizing objective function $R_n(\Theta)$ in (1), SSE-Graph described in Algorithm 1, Figure 1, and Figure 2 is approximately optimizing the objective function below:

$$S_n(\Theta) = \sum_i \sum_{\mathbf{k} \in \mathcal{I}} p(\mathbf{j}^i, \mathbf{k}|\Phi)\ell(\mathbf{E}[\mathbf{k}]|\Theta), \tag{2}$$

where $p(\mathbf{j}, \mathbf{k}|\Phi)$ is the transition probability (with parameters Φ) of exchanging the encoding vector $\mathbf{j} \in \mathcal{I}$ with a new encoding vector $\mathbf{k} \in \mathcal{I}$ in the Cartesian product index set of all embedding tables. When there is a single embedding table ($M = 1$) then there are no hard restrictions on the transition

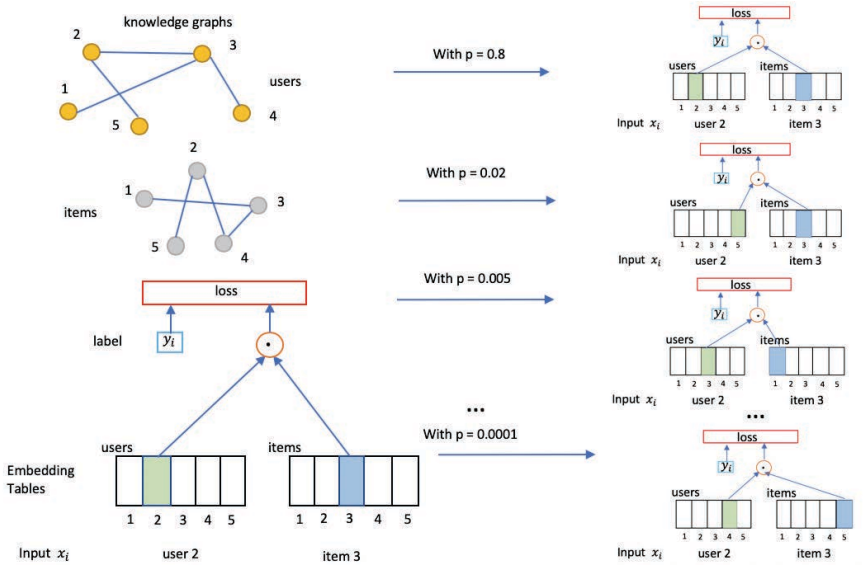

Figure 2: Illustration of how SSE-Graph algorithm in Figure 1 works for a simple neural network.

probabilities, $p(.,.)$, but when there are multiple tables ($M > 1$) then we will enforce that $p(.,.)$ takes a tensor product form (see (4)). When we are assuming that there is only a single embedding table ($M = 1$) we will not bold $j, E[j]$ and suppress their indices.

In the single embedding table case, $M = 1$, there are many ways to define transition probability from j to k. One simple and effective way is to use a random walk (with random restart and self-loops) on a knowledge graph $\mathcal{G}$, i.e. when embedding j is connected with k but not with l, we can set the ratio of $p(j, k|\Phi)$ and $p(j, l|\Phi)$ to be a constant greater than 1. In more formal notation, we have

$$j \sim k, j \not\sim l \longrightarrow p(j, k|\Phi)/p(j, l|\Phi) = \rho, \tag{3}$$

where $\rho > 1$ and is a tuning parameter. It is motivated by the fact that embeddings connected with each other in knowledge graphs should bear more resemblance and thus be more likely replaced by each other. Also, we let $p(j, j|\Phi) = 1 - p_0$, where p_0 is called the *SSE probability* and embedding retainment probability is $1 - p_0$. We treat both p_0 and ρ as tuning hyper-parameters in experiments. With (3) and $\sum_k p(j, k|\Phi) = 1$, we can derive transition probabilities between any two embeddings to fill out the transition probability table.

When there are multiple embedding tables, $M > 1$, then we will force that the transition from $\mathbf{j}$ to $\mathbf{k}$ can be thought of as independent transitions from j_l to k_l within embedding table l (and index set $\mathcal{I}_l$). Each table may have its own knowledge graph, resulting in its own transition probabilities $p_l(.,.)$. The more general form of the SSE-graph objective is given below:

$$S_n(\Theta) = \sum_i \sum_{k_1,\ldots,k_M} p_1(j_1^i, k_1|\Phi) \cdots p_M(j_M^i, k_M|\Phi)\ell(E_1[k_1], \ldots, E_M[k_M]|\Theta), \tag{4}$$

Intuitively, this SSE objective could reduce the variance of the estimator.

Optimizing (4) with SGD or its variants (Adagrad [4], Adam [12]) is simple. We just need to randomly switch each original embedding tensor $\mathbf{E}[\mathbf{j}^i]$ with another embedding tensor $\mathbf{E}[\mathbf{k}]$ randomly sampled according to the transition probability (see Algorithm 1). This is equivalent to have a randomized embedding look-up layer as shown in Figure 1.

We can also accommodate sequences of embeddings, which commonly occur in natural language application, by considering $(j_{l,1}^i, k_{l,1}), \ldots, (j_{l,n_l^i}^i, k_{l,n_l^i})$ instead of (j_l^i, k_l) for l-th embedding table in (4), where $1 \le l \le M$ and n_l^i is the number of embeddings in table l that are associated with (x_i, y_i). When there is more than one embedding look-up table, we sometimes prefer to use different p_0 and ρ for different look-up tables in (3) and the SSE probability constraint. For example, in recommender systems, we would use p_u, ρ_u for user embedding table and p_i, ρ_i for item embedding table.