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`

Abstract

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

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

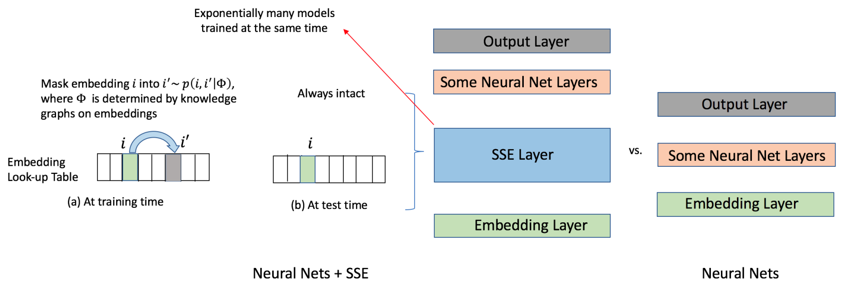


Figure 1: SSE-Graph described in Algorithm 1 and Figure 2 can be viewed as adding exponentially many distinct reordering layers above the embedding layer. A modified backpropagation procedure in Algorithm 1 is used to train exponentially many such neural networks at the same time.

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 $\mathbf{E}[\mathbf{j}^i] = (E_1[j_1^i], \dots, E_M[j_M^i])$ where $\mathbf{j}^i = (j_1^i, \dots, j_M^i) \in \mathcal{I}$. 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:

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

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

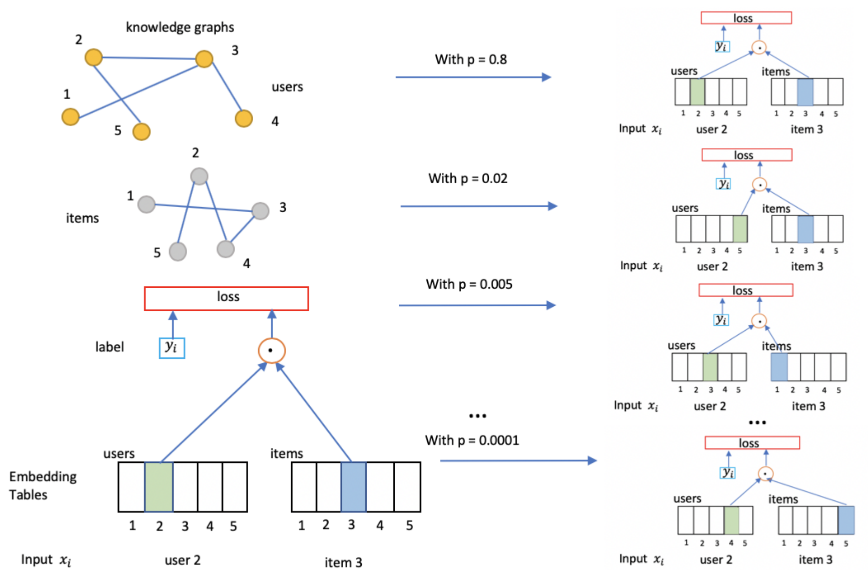


Figure 2: Illustration of how SSE-Graph algorithm in Figure 1 works for a simple neural network.

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 **j** to **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.

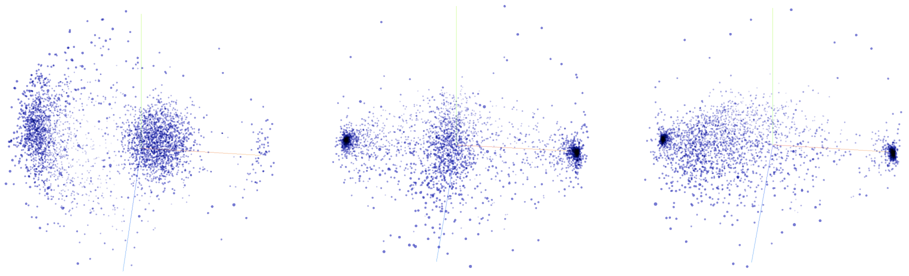


Figure 3: Projecting 50-dimensional embeddings obtained by training a simple neural network without SSE (Left), and with SSE-Graph (Center) , SSE-SE (Right) into 3D space using PCA.

We find that SSE with knowledge graphs, i.e., SSE-Graph, can force similar embeddings to cluster when compared to the original neural network without SSE-Graph. In Figure 3, one can easily see that more embeddings tend to cluster into 2 singularities after applying SSE-Graph when embeddings are projected into 3D spaces using PCA. Interestingly, a similar phenomenon occurs when assuming the knowledge graph is a complete graph, which we would introduce as SSE-SE below.

3.2 Simplified SSE with Complete Graph: SSE-SE

One clear limitation of applying the SSE-Graph is that not every dataset comes with good-quality knowledge graphs on embeddings. For those cases, we could assume there is a complete graph over all embeddings so there is a small transition probability between every pair of different embeddings:

$$p(j,k|\Phi) = \frac{p_0}{N-1}, \quad \forall 1 \le k \neq j \le N, \tag{5}$$

where N is the size of the embedding table. The SGD procedure in Algorithm 1 can still be applied and we call this algorithm SSE-SE (Stochastic Shared Embeddings - Simple and Easy). It is worth noting that SSE-Graph and SSE-SE are applied to embeddings associated with not only input x_i but also those with output y_i. Unless there are considerably many more embeddings than data points and model is significantly overfitting, normally $p_0 = 0.01$ gives reasonably good results.

Interestingly, we found that the SSE-SE framework is related to several techniques used in practice. For example, BERT pre-training unintentionally applied a method similar to SSE-SE to input x_i by replacing the masked word with a random word. This would implicitly introduce an SSE layer for input x_i in Figure 1, because now embeddings associated with input x_i be stochastically mapped according to (5). The main difference between this and SSE-SE is that it merely augments the input once, while SSE introduces randomization at every iteration, and we can also accommodate label embeddings. In experimental Section 4.4, we will show that SSE-SE would improve original BERT pre-training procedure as well as fine-tuning procedure.

3.3 Theoretical Guarantees

We explain why SSE can reduce the variance of estimators and thus leads to better generalization performance. For simplicity, we consider the SSE-graph objective (2) where there is no transition associated with the label y_i, and only the embeddings associated with the input x_i undergo a transition. When this is the case, we can think of the loss as a function of the x_i embedding and the label, $\ell(\mathbf{E}[\mathbf{j}^i], y_i; \Theta)$. We take this approach because it is more straightforward to compare our resulting theory to existing excess risk bounds.

The SSE objective in the case of only input transitions can be written as,

$$S_n(\Theta) = \sum_i \sum_{\mathbf{k}} p(\mathbf{j}^i, \mathbf{k}) \cdot \ell(\mathbf{E}[\mathbf{k}], y_i|\Theta), \tag{6}$$

and there may be some constraint on Θ. Let $\hat{\Theta}$ denote the minimizer of S_n subject to this constraint. We will show in the subsequent theory that minimizing S_n will get us close to a minimizer of $S(\Theta) = \mathbb{E}S_n(\Theta)$, and that under some conditions this will get us close to the Bayes risk. We will use the standard definitions of empirical and true risk, $R_n(\Theta) = \sum_i \ell(x_i, y_i|\Theta)$ and $R(\Theta) = \mathbb{E}R_n(\Theta)$.

Our results depend on the following decomposition of the risk. By optimality of $\hat{\Theta}$,

$$R(\hat{\Theta}) = S_n(\hat{\Theta}) + [R(\hat{\Theta}) - S(\hat{\Theta})] + [S(\hat{\Theta}) - S_n(\hat{\Theta})] \le S_n(\Theta^*) + B(\hat{\Theta}) + \mathcal{E}(\hat{\Theta}) \tag{7}$$

Table 1: Compare SSE-Graph and SSE-SE against ALS-MF with Graph Laplacian Regularization. The p_u and p_i are the SSE probabilities for user and item embedding tables respectively, as in (5). Definitions of ρ_u and ρ_i can be found in (3). Movielens10m does not have user graphs.

	Movielens1m					Movielens10m				
Model	RMSE	ρ_u	ρ_i	p_u	p_i	RMSE	ρ_u	ρ_i	p_u	p_i
SGD-MF	1.0984	-	-	-	-	1.9490	-	-	-	-
Graph Laplacian + ALS-MF	1.0464	-	-	-	-	1.9755	-	-	-	-
SSE-Graph + SGD-MF	**1.0145**	500	200	0.005	0.005	**1.9019**	1	500	0.01	0.01
SSE-SE + SGD-MF	1.0150	1	1	0.005	0.005	1.9085	1	1	0.01	0.01

where $B(\Theta) = |R(\Theta) - S(\Theta)|$, and $E(\Theta) = |S(\Theta) - S_n(\Theta)|$. We can think of $B(\Theta)$ as representing the bias due to SSE, and $E(\Theta)$ as an SSE form of excess risk. Then by another application of similar bounds,

$$R(\hat{\Theta}) \leq R(\Theta^*) + B(\hat{\Theta}) + B(\Theta^*) + E(\hat{\Theta}) + E(\Theta^*). \tag{8}$$

The high level idea behind the following results is that when the SSE protocol reflects the underlying distribution of the data, then the bias term $B(\Theta)$ is small, and if the SSE transitions are well mixing then the SSE excess risk $E(\Theta)$ will be of smaller order than the standard Rademacher complexity. This will result in a small excess risk.

Theorem 1. *Consider SSE-graph with only input transitions. Let $L(\mathbf{E}[\mathbf{j}^i]) = \mathbb{E}_{Y|X=x^i}\ell(\mathbf{E}[\mathbf{j}^i], Y|\Theta)$ be the expected loss conditional on input x^i and $e(\mathbf{E}[\mathbf{j}^i], y|\Theta) = \ell(\mathbf{E}[\mathbf{j}^i], y|\Theta) - L(\mathbf{E}[\mathbf{j}^i]|\Theta)$ be the residual loss. Define the conditional and residual SSE empirical Rademacher complexities to be*

$$\rho_{L,n} = \mathbb{E}_\sigma \sup_\Theta \left| \sum_i \sigma_i \sum_{\mathbf{k}} p(\mathbf{j}^i, \mathbf{k}) \cdot L(\mathbf{E}[\mathbf{k}]|\Theta) \right|, \tag{9}$$

$$\rho_{e,n} = \mathbb{E}_\sigma \sup_\Theta \left| \sum_i \sigma_i \sum_{\mathbf{k}} p(\mathbf{j}^i, \mathbf{k}) \cdot e(\mathbf{E}[\mathbf{k}], y_i; \Theta) \right|, \tag{10}$$

$$\tag{11}$$

respectively where σ is a Rademacher ± 1 random vectors in $\mathbb{R}^n$. Then we can decompose the SSE empirical risk into

$$\mathbb{E} \sup_\Theta |S_n(\Theta) - S(\Theta)| \leq 2\mathbb{E}[\rho_{L,n} + \rho_{e,n}]. \tag{12}$$

Remark 1. *The transition probabilities in (9), (10) act to smooth the empirical Rademacher complexity. To see this, notice that we can write the inner term of (9) as $(P\sigma)^\top L$, where we have vectorized $\sigma_i, L(x_i; \Theta)$ and formed the transition matrix P. Transition matrices are contractive and will induce dependencies between the Rademacher random variables, thereby stochastically reducing the supremum. In the case of no label noise, namely that $Y|X$ is a point mass, $e(x, y; \Theta) = 0$, and $\rho_{e,n} = 0$. The use of L as opposed to the losses, ℓ, will also make $\rho_{L,n}$ of smaller order than the standard empirical Rademacher complexity. We demonstrate this with a partial simulation of $\rho_{L,n}$ on the Movielens1m dataset in Figure 5 of the Appendix.*

Theorem 2. *Let the SSE-bias be defined as*

$$\mathcal{B} = \sup_\Theta \left| \mathbb{E}\left[\sum_i \sum_{\mathbf{k}} p(\mathbf{j}^i, \mathbf{k}) \cdot \big(\ell(\mathbf{E}[\mathbf{k}], y_i|\Theta) - \ell(\mathbf{E}[\mathbf{j}^i], y_i|\Theta) \big) \right] \right|.$$

Suppose that $0 \leq \ell(., .; \Theta) \leq b$ for some $b > 0$, then

$$\mathbb{P}\left\{ R(\hat{\Theta}) > R(\Theta^*) + 2\mathcal{B} + 4\mathbb{E}[\rho_{L,n} + \rho_{e,n}] + \sqrt{n}u \right\} \leq e^{-\frac{u^2}{2b^2}}.$$

Remark 2. *The price for 'smoothing' the Rademacher complexity in Theorem 1 is that SSE may introduce a bias. This will be particularly prominent when the SSE transitions have little to do with the underlying distribution of Y, X. On the other extreme, suppose that $p(\mathbf{j}, \mathbf{k})$ is non-zero over a neighborhood $\mathcal{N}_\mathbf{j}$ of $\mathbf{j}$, and that for data x', y' with encoding $\mathbf{k} \in \mathcal{N}_\mathbf{j}$, x', y' is identically distributed with x_i, y_i, then $\mathcal{B} = 0$. In all likelihood, the SSE transition probabilities will not be supported over neighborhoods of iid random pairs, but with a well chosen SSE protocol the neighborhoods contain approximately iid pairs and $\mathcal{B}$ is small.*

Table 2: SSE-SE outperforms Dropout for Neural Networks with One Hidden Layer such as Matrix Factorization Algorithm regardless of dimensionality we use. p_s is the SSE probability for both user and item embedding tables and p_d is the dropout probability.

	Douban			Movielens10m			Netflix		
Model	RMSE	p_d	p_s	RMSE	p_d	p_s	RMSE	p_d	p_s
MF	0.7339	-	-	0.8851	-	-	0.8941	-	-
Dropout + MF	0.7296	0.1	-	0.8813	0.1	-	0.8897	0.1	-
SSE-SE + MF	0.7201	-	0.008	0.8715	-	0.008	0.8842	-	0.008
SSE-SE + Dropout + MF	**0.7185**	0.1	0.005	**0.8678**	0.1	0.005	**0.8790**	0.1	0.005

Table 3: SSE-SE outperforms dropout for Neural Networks with One Hidden Layer such as Bayesian Personalized Ranking Algorithm regardless of dimensionality we use. We report the metric precision for top k recommendations as $P@k$.

	Movielens1m			Yahoo Music			Foursquare		
Model	$P@1$	$P@5$	$P@10$	$P@1$	$P@5$	$P@10$	$P@1$	$P@5$	$P@10$
SQL-Rank (2018)	**0.7369**	0.6717	0.6183	**0.4551**	**0.3614**	**0.3069**	0.0583	0.0194	**0.0170**
BPR	0.6977	0.6568	0.6257	0.3971	0.3295	0.2806	0.0437	0.0189	0.0143
Dropout + BPR	0.7031	0.6548	0.6273	0.4080	0.3315	0.2847	0.0437	0.0184	0.0146
SSE-SE + BPR	0.7254	**0.6813**	**0.6469**	0.4297	0.3498	0.3005	**0.0609**	**0.0262**	0.0155

4 Experiments

We have conducted extensive experiments on 6 tasks, including 3 recommendation tasks (explicit feedback, implicit feedback and sequential recommendation) and 3 NLP tasks (neural machine translation, BERT pre-training, and BERT fine-tuning for sentiment classification) and found that our proposed SSE can effectively improve generalization performances on a wide variety of tasks. Note that the details about datasets and parameter settings can be found in the appendix.

4.1 Neural Networks with One Hidden Layer (Matrix Factorization and BPR)

Matrix Factorization Algorithm (MF) [17] and Bayesian Personalized Ranking Algorithm (BPR) [25] can be viewed as neural networks with one hidden layer (latent features) and are quite popular in recommendation tasks. MF uses the squared loss designed for explicit feedback data while BPR uses the pairwise ranking loss designed for implicit feedback data.

First, we conduct experiments on two explicit feedback datasets: Movielens1m and Movielens10m. For these datasets, we can construct graphs based on actors/actresses starring the movies. We compare SSE-graph and the popular Graph Laplacian Regularization (GLR) method [24] in Table 1. The results show that SSE-graph consistently outperforms GLR. This indicates that our SSE-Graph has greater potentials over graph Laplacian regularization as we do not explicitly penalize the distances across embeddings, but rather we implicitly penalize the effects of similar embeddings on the loss. Furthermore, we show that even without existing knowledge graphs of embeddings, our SSE-SE performs only slightly worse than SSE-Graph but still much better than GLR and MF.

In general, SSE-SE is a good alternative when graph information is not available. We then show that our proposed SSE-SE can be used together with standard regularization techniques such as dropout and weight decay to improve recommendation results regardless of the loss functions and dimensionality of embeddings. This is evident in Table 2 and Table 3. With the help of SSE-SE, BPR can perform better than the state-of-art listwise approach SQL-Rank [32] in most cases. We include the optimal SSE parameters in the table for references and leave out other experiment details to the appendix. In the rest of the paper, we would mostly focus on SSE-SE as we do not have high-quality graphs of embeddings on most datasets.

4.2 Transformer Encoder Model for Sequential Recommendation

SASRec [11] is the state-of-the-arts algorithm for sequential recommendation task. It applies the transformer model [30], where a sequence of items purchased by a user can be viewed as a sentence in transformer, and next item prediction is equivalent to next word prediction in the language model. In Table 4, we perform SSE-SE on input embeddings ($p_x = 0.1, p_y = 0$), output embeddings ($p_x = 0.1, p_y = 0$) and both embeddings ($p_x = p_y = 0.1$), and observe that all of them significantly improve over state-of-the-art SASRec ($p_x = p_y = 0$). The regularization effects of SSE-SE is even more

Table 4: SSE-SE has two tuning parameters: probability p_x to replace embeddings associated with input x_i and probability p_y to replace embeddings associated with output y_i. We use the dropout probability of 0.1, weight decay of $1e^{-5}$, and learning rate of $1e^{-3}$ for all experiments.

	Movielens1m		Dimension	# of Blocks	SSE-SE Parameters	
Model	NDCG@10	Hit Ratio@10	d	b	p_x	p_y
SASRec	0.5941	0.8182	100	2	-	-
SASRec	0.5996	0.8272	100	6	-	-
SSE-SE + SASRec	0.6092	0.8250	100	2	0.1	0
SSE-SE + SASRec	0.6085	0.8293	100	2	0	0.1
SSE-SE + SASRec	0.6200	0.8315	100	2	0.1	0.1
SSE-SE + SASRec	**0.6265**	**0.8364**	100	6	0.1	0.1

Table 5: Our proposed SSE-SE helps the Transformer achieve better BLEU scores on English-to-German in 10 out of 11 newstest data between 2008 and 2018.

	Test BLEU										
Model	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018
Transformer	21.0	20.7	22.7	20.6	20.6	**25.3**	26.2	28.4	32.1	27.2	38.8
SSE-SE + Transformer	**21.4**	**21.1**	**23.0**	**21.0**	**20.8**	25.2	**27.2**	**29.2**	**33.1**	**27.9**	**39.9**

obvious when we increase the number of self-attention blocks from 2 to 6, as this will lead to a more sophisticated model with many more parameters. This leads to the model overfitting terribly even with dropout and weight decay. We can see in Table 4 that when both methods use dropout and weight decay, SSE-SE + SASRec is doing much better than SASRec without SSE-SE.

4.3 Neural Machine Translation

We use the transformer model [30] as the backbone for our experiments. The baseline model is the standard 6-layer transformer architecture and we apply SSE-SE to both encoder, and decoder by replacing corresponding vocabularies' embeddings in the source and target sentences. We trained on the standard WMT 2014 English to German dataset which consists of roughly 4.5 million parallel sentence pairs and tested on WMT 2008 to 2018 news-test sets. We use the OpenNMT implementation in our experiments. We use the same dropout rate of 0.1 and label smoothing value of 0.1 for the baseline model and our SSE-enhanced model. The only difference between the two models is whether or not we use our proposed SSE-SE with $p_0 = 0.01$ in (5) for both encoder and decoder embedding layers. We evaluate both models' performances on the test datasets using BLEU scores [21].

We summarize our results in Table 5 and find that SSE-SE helps improving accuracy and BLEU scores on both dev and test sets in 10 out of 11 years from 2008 to 2018. In particular, on the last 5 years' test sets from 2014 to 2018, the transformer model with SSE-SE improves BLEU scores by 0.92 on average when compared to the baseline model without SSE-SE.

4.4 BERT for Sentiment Classification

BERT's model architecture [3] is a multi-layer bidirectional Transformer encoder based on the Transformer model in neural machine translation. Despite SSE-SE can be used for both pre-training and fine-tuning stages of BERT, we want to mainly focus on pre-training as fine-tuning bears more similarity to the previous section. We use SSE probability of 0.015 for embeddings (one-hot encodings) associated with labels and SSE probability of 0.015 for embeddings (word-piece embeddings) associated with inputs. One thing worth noting is that even in the original BERT model's pre-training stage, SSE-SE is already implicitly used for token embeddings. In original BERT model, the authors masked 15% of words for a maximum of 80 words in sequences of maximum length of 512 and 10% of the time replaced the [mask] token with a random token. That is roughly equivalent to SSE probability of 0.015 for replacing input word-piece embeddings.

We continue to pre-train Google pre-trained BERT model on our crawled IMDB movie reviews with and without SSE-SE and compare downstream tasks performances. In Table 6, we find that SSE-SE pre-trained BERT base model helps us achieve the state-of-the-art results for the IMDB sentiment classification task, which is better than the previous best in [9]. We report test set accuracy of 0.9542 after fine-tuning for one epoch only. For the similar SST-2 sentiment classification task in Table 7, we also find that SSE-SE can improve BERT pre-trains better. Our SSE-SE pre-trained model achieves 94.3% accuracy on SST-2 test set after 3 epochs of fine-tuning while the standard pre-trained BERT

Table 6: Our proposed SSE-SE applied in the pre-training stage on our crawled IMDB data improves the generalization ability of pre-trained IMDB model and helps the BERT-Base model outperform current SOTA results on the IMDB Sentiment Task after fine-tuning.

	IMDB Test Set		
Model	AUC	Accuracy	F1 Score
ULMFiT [9]	-	0.9540	-
Google Pre-trained Model + Fine-tuning	0.9415	0.9415	0.9419
Pre-training + Fine-tuning	0.9518	0.9518	0.9523
(SSE-SE + Pre-training) + Fine-tuning	**0.9542**	**0.9542**	**0.9545**

Table 7: SSE-SE pre-trained BERT-Base models on IMDB datasets turn out working better on the new unseen SST-2 Task as well.

	SST-2 Dev Set			SST-2 Test Set
Model	AUC	Accuracy	F1 Score	Accuracy (%)
Google Pre-trained + Fine-tuning	0.9230	0.9232	0.9253	93.6
Pre-training + Fine-tuning	0.9265	0.9266	0.9281	93.8
(SSE-SE + Pre-training) + Fine-tuning	0.9276	0.9278	0.9295	94.3
(SSE-SE + Pre-training) + (SSE-SE + Fine-tuning)	**0.9323**	**0.9323**	**0.9336**	**94.5**

model only reports 93.8 after fine-tuning. Furthermore, we show that SSE-SE with SSE probability 0.01 can also improve dev and test accuracy in the fine-tuning stage. If we are using SSE-SE for both pre-training and fine-tuning stage of the BERT base model, we can achieve 94.5% accuracy on the SST-2 test set, approaching the 94.9% accuracy by the BERT large model. We are optimistic that our SSE-SE can be applied to BERT large model as well in the future.

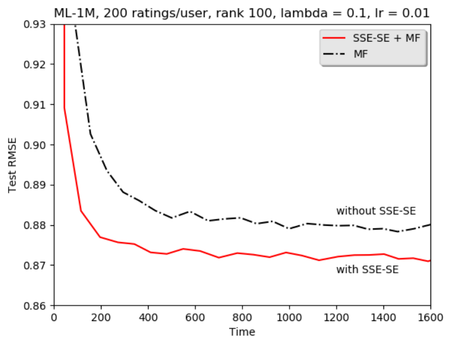
4.5 Speed and Convergence Comparisons.

In Figure 4, it is clear to see that our one-hidden-layer neural networks with SSE-SE are achieving much better generalization results than their respective standalone versions. One can also easily spot that SSE-version algorithms converge at much faster speeds with the same learning rate.

5 Conclusion

We have proposed Stochastic Shared Embeddings, which is a data-driven approach to regularization, that stands in contrast to brute force regularization such as Laplacian and ridge regularization. Our theory is a first step towards explaining the regularization effect of SSE, particularly, by 'smoothing' the Rademacher complexity. The extensive experimentation demonstrates that SSE can be fruitfully integrated into existing deep learning applications.

Acknowledgement. Hsieh acknowledges the support of NSF IIS-1719097, Intel faculty award, Google Cloud and Nvidia.



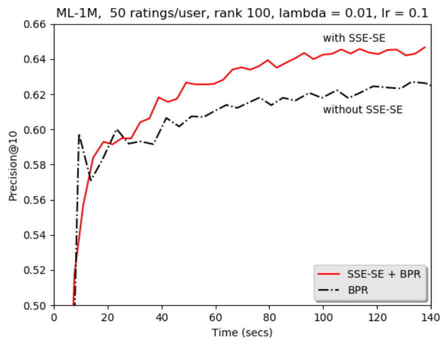


Figure 4: Compare Training Speed of Simple Neural Networks with One Hidden Layer, i.e. MF and BPR, with and without SSE-SE.

References

[1] James Bennett, Stan Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, NY, USA., 2007.

[2] Deng Cai, Xiaofei He, Jiawei Han, and Thomas S Huang. Graph regularized nonnegative matrix factorization for data representation. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1548–1560, 2011.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[5] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[6] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182. International World Wide Web Conferences Steering Committee, 2017.

[7] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[8] Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

[9] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[10] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, volume 8, pages 263–272. Citeseer, 2008.

[11] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. *arXiv preprint arXiv:1808.09781*, 2018.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[14] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195, 2015.

[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[16] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11): 39–41, 1995.

[17] Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2008.

[18] Steven J Nowlan and Geoffrey E Hinton. Simplifying neural networks by soft weight-sharing. *Neural computation*, 4(4):473–493, 1992.

[19] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

[20] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[21] Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels, October 2018. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/W18-6319`.

[22] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*, 2018.

[23] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *URL https://openai. com/blog/better-language-models*, 2019.

[24] Nikhil Rao, Hsiang-Fu Yu, Pradeep K Ravikumar, and Inderjit S Dhillon. Collaborative filtering with graph information: Consistency and scalable methods. In *Advances in neural information processing systems*, pages 2107–2115, 2015.

[25] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.

[26] Nathan Srebro, Jason Rennie, and Tommi S Jaakkola. Maximum-margin matrix factorization. In *Advances in neural information processing systems*, pages 1329–1336, 2005.

[27] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[28] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[29] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[31] Liwei Wu, Cho-Jui Hsieh, and James Sharpnack. Large-scale collaborative ranking in near-linear time. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 515–524. ACM, 2017.

[32] Liwei Wu, Cho-Jui Hsieh, and James Sharpnack. Sql-rank: A listwise approach to collaborative ranking. In *Proceedings of Machine Learning Research (35th International Conference on Machine Learning)*, volume 80, 2018.