Unified Interpretation of Softmax Cross-Entropy and Negative Sampling: With Case Study for Knowledge Graph Embedding

Hidetaka Kamigaito
Tokyo Institute of Technology
kamigaito@lr.pi.titech.ac.jp

Katsuhiko Hayashi
Gunma University, RIKEN AIP
khayashi0201@gmail.com

Abstract

In knowledge graph embedding, the theoretical relationship between the softmax cross-entropy and negative sampling loss functions has not been investigated. This makes it difficult to fairly compare the results of the two different loss functions. We attempted to solve this problem by using the Bregman divergence to provide a unified interpretation of the softmax cross-entropy and negative sampling loss functions. Under this interpretation, we can derive theoretical findings for fair comparison. Experimental results on the FB15k-237 and WN18RR datasets show that the theoretical findings are valid in practical settings.

1 Introduction

Negative Sampling (NS) (Mikolov et al., 2013) is an approximation of softmax cross-entropy (SCE). Due to its efficiency in computation cost, NS is now a fundamental loss function for various Natural Language Processing (NLP) tasks such as used in word embedding (Mikolov et al., 2013), language modeling (Melamud et al., 2017), contextualized embedding (Clark et al., 2020b,a), and knowledge graph embedding (KGE) (Trouillon et al., 2016). Specifically, recent KGE models commonly use NS for training. Considering the current usages of NS, we investigated the characteristics of NS by mainly focusing on KGE from theoretical and empirical aspects.

First, we introduce the task description of KGE. A knowledge graph is a graph that describes the relationships between entities. It is an indispensable resource for knowledge-intensive NLP applications such as dialogue (Moon et al., 2019) and question-answering (Lukovnikov et al., 2017) systems. However, to create a knowledge graph, it is necessary to consider a large number of entity combinations and their relationships, making it difficult to construct a complete graph manually. Therefore, the prediction of links between entities is an important task.

Currently, missing relational links between entities are predicted using a scoring method based on KGE (Bordes et al., 2011). With this method, a score for each link is computed on vector space representations of embedded entities and relations. We can train these representations through various loss functions. The SCE (Kadlec et al., 2017) and NS (Trouillon et al., 2016) loss functions are commonly used for this purpose.

Several studies (Ruffinelli et al., 2020; Ali et al., 2020) have shown that link-prediction performance can be significantly improved by choosing the appropriate combination of loss functions and scoring methods. However, the relationship between the SCE and NS loss functions has not been investigated in KGE. Without a basis for understanding the relationships among different loss functions, it is difficult to make a fair comparison between the SCE and NS results.

We attempted to solve this problem by using the Bregman divergence (Bregman, 1967) to provide a unified interpretation of the SCE and NS loss functions. Under this interpretation, we can understand the relationships between SCE and NS in terms of the model's predicted distribution at the optimal solution, which we called the **objective distribution**. By deriving the objective distribution for a loss function, we can analyze different loss functions, the objective distributions of which are identical under certain conditions, from a unified viewpoint.

We summarize our theoretical findings not restricted to KGE as follows:

- The objective distribution of NS with uniform noise (NS w/ Uni) is equivalent to that of SCE.
- The objective distribution of self-adversarial negative sampling (SANS) (Sun et al., 2019)

is quite similar to SCE with label smoothing (SCE w/ LS) (Szegedy et al., 2016).

- NS with frequency-based noise (NS w/ Freq) in word2vec[1] has a smoothing effect on the objective distribution.
- SCE has a property wherein it more strongly fits a model to the training data than NS.

To check the validity of the theoretical findings in practical settings, we conducted experiments on the FB15k-237 (Toutanova and Chen, 2015) and WN18RR (Dettmers et al., 2018) datasets. The experimental results indicate that

- The relationship between SCE and SCE w/ LS is also similar to that between NS and SANS in practical settings.
- NS is prone to underfitting because it weakly fits a model to the training data compared with SCE.
- SCE causes underfitting of KGE models when their score function has a bound.
- Both SANS and SCE w/ LS perform well as pre-training methods.

The structure of this paper is as follows: Sec. 2 introduces SCE and Bregman divergence; Sec. 3 induces the objective distributions for NS; Sec. 4 analyzes the relationships between SCE and NS loss functions; Sec. 5 summarizes and discusses our theoretical findings; Sec. 6 discusses empirically investigating the validity of the theoretical findings in practical settings; Sec. 7 explains the differences between this paper and related work; and Sec. 8 summarizes our contributions. Our code will be available at https://github.com/kamigaito/acl2021kge

2 Softmax Cross Entropy and Bregman Divergence

2.1 SCE in KGE

We denote a link representing a relationship r_k between entities e_i and e_j in a knowledge graph as (e_i, r_k, e_j). In predicting the links from given queries $(e_i, r_k, ?)$ and $(?, r_k, e_j)$, the model must predict entities corresponding to each ? in the queries. We denote such a query as x and the entity to be predicted as y. By using the softmax function, the probability $p_\theta(y|x)$ that y is predicted from x with the model parameter θ given a score function $f_\theta(x,y)$ is expressed as follows:

$$p_\theta(y|x) = \frac{\exp(f_\theta(x,y))}{\sum_{y' \in Y} \exp(f_\theta(x,y'))}, \quad (1)$$

where Y is the set of all predictable entities. We further denote the pair of an input x and its label y as (x,y). Let $D = \{(x_1,y_1), \cdots, (x_{|D|}, y_{|D|})\}$ be observed data that obey a distribution $p_d(x,y)$.

2.2 Bregman Divergence

Next, we introduce the Bregman divergence. Let $\Psi(z)$ be a differentiable function; the Bregman divergence between two distributions f and g is defined as follows:

$$d_{\Psi(z)}(f,g) = \Psi(f) - \Psi(g) - \nabla\Psi(g)^T(f-g). \quad (2)$$

We can express various divergences by changing $\Psi(z)$. To take into account the divergence on the entire observed data, we consider the expectation of $d_\Psi(f,g)$: $B_{\Psi(z)}(f,g) = \sum_{x,y} d_{\Psi(z)}(f(y|x), g(y|x)) p_d(x,y)$. To investigate the relationship between a loss function and learned distribution of a model at an optimal solution of the loss function, we need to focus on the minimization of $B_{\Psi(z)}$. Gutmann and Hirayama (2011) showed that $B_{\Psi(z)}(f,g) = 0$ means that f equals g almost everywhere when $\Psi(z)$ is a differentiable strictly convex function in its domain. Note that all $\Psi(z)$ in this paper satisfy this condition. Accordingly, by fixing f, minimization of $B_{\Psi(z)}(f,g)$ with respect to g is equivalent to minimization of

$$B_{\Psi(z)}(f,g) = \sum_{x,y} \left[-\Psi(g) + \nabla\Psi(g)^T g - \nabla\Psi(g)^T f \right] p_d(x,y) \quad (3)$$

We use $B_\Psi(f,g)$ to reveal a learned distribution of a model at optimal solutions for the SCE and NS loss functions.

2.3 Derivation of SCE

For the latter explanations, we first derive the SCE loss function from Eq. (3). We denote a probability for a label y as $p(y)$, vector for all y as $\mathbf{y}$, vector of probabilities for $\mathbf{y}$ as $p(\mathbf{y})$, and dimension size of $\mathbf{z}$ as $len(\mathbf{z})$. In Eq. (3), by setting f as $p_d(\mathbf{y}|x)$ and g as $p_\theta(\mathbf{y}|x)$ with $\Psi(\mathbf{z}) = \sum_{i=1}^{len(\mathbf{z})} z_i \log z_i$ (Banerjee

[1] The word2vec uses unigram distribution as the frequency-based noise.

et al., 2005), we can derive the SCE loss function as follows:

$$B_{\Psi(\mathbf{z})}(p_d(\mathbf{y}|x), p_\theta(\mathbf{y}|x))$$
$$= -\sum_{x,y}\left[\sum_{i=1}^{|Y|} p_d(y_i|x)\log p_\theta(y_i|x)\right] p_d(x,y) \quad (4)$$
$$= -\frac{1}{|D|}\sum_{(x,y)\in D}\log p_\theta(y|x). \quad (5)$$

This derivation indicates that $p_\theta(y|x)$ converges to the observed distribution $p_d(y|x)$ through minimizing $B_{\Psi(\mathbf{z})}(p_d(\mathbf{y}|x), p_\theta(\mathbf{y}|x))$ in the SCE loss function. We call the distribution of $p_\theta(\mathbf{y}|x)$ when $B_{\Psi(\mathbf{z})}$ equals zero an **objective distribution**.

3 Objective Distribution for Negative Sampling Loss

We begin by providing a definition of NS and its relationship to the Bregman divergence, following the induction of noise contrastive estimation (NCE) from the Bregman divergence that was established by Gutmann and Hirayama (2011). We denote $p_n(y|x)$ to be a known non-zero noise distribution for y of a given x. Given ν noise samples from $p_n(y|x)$ for each $(x,y) \in D$, NS estimates the model parameter θ for a distribution $G(y|x;\theta) = \exp(-f_\theta(x,y))$.

By assigning to each (x,y) a binary class label C: $C = 1$ if (x,y) is drawn from observed data D following a distribution $p_d(x,y)$ and $C = 0$ if (x,y) is drawn from a noise distribution $p_n(y|x)$, we can model the posterior probabilities for the classes as follows:

$$p(C=1,y|x;\theta) = \frac{1}{1+\exp(-f_\theta(x,y))} = \frac{1}{1+G(y|x;\theta)},$$
$$p(C=0,y|x;\theta) = 1-p(C=1,y|x;\theta) = \frac{G(y|x;\theta)}{1+G(y|x;\theta)}.$$

The objective function $\ell^{NS}(\theta)$ of NS is defined as follows:

$$\ell^{NS}(\theta) = -\frac{1}{|D|}\sum_{(x,y)\in D}\Big[\log(P(C=1,y|x;\theta)) + \sum_{i=1,y_i\sim p_n}^{\nu}\log(P(C=0,y_i|x;\theta))\Big]. \quad (6)$$

By using the Bregman divergence, we can induce the following propositions for $\ell^{NS}(\theta)$.

Proposition 1. *$\ell^{NS}(\theta)$ can be induced from Eq. (3) by setting $\Psi(z)$ as:*

$$\Psi(z) = z\log(z) - (1+z)\log(1+z). \quad (7)$$

Proposition 2. *When $\ell^{NS}(\theta)$ equals 0, the following equation is satisfied:*

$$G(y|x;\theta) = \frac{\nu p_n(y|x)}{p_d(y|x)}. \quad (8)$$

Proposition 3. *The objective distribution of $P_\theta(y|x)$ for $\ell^{NS}(\theta)$ is*

$$\frac{p_d(y|x)}{p_n(y|x)\sum_{y_i\in Y}\frac{p_d(y_i|x)}{p_n(y_i|x)}}. \quad (9)$$

Proof. We give the proof of Props. 1, 2, and 3 in Appendix A of the supplemental material. □

We can also investigate the validity of Props. 1, 2, and 3 by comparing them with the previously reported result. For this purpose, we prove the following proposition:

Proposition 4. *When Eq. (8) satisfies $\nu = 1$ and $p_n(y|x) = p_d(y)$, $f_\theta(x,y)$ equals point-wise mutual information (PMI).*

Proof. This is described in Appendix B of the supplemental material. □

This observation is consistent with that by Levy and Goldberg (2014). The differences between their representation and ours are as follows. (1) Our noise distribution is general in the sense that its definition is not restricted to a unigram distribution; (2) we mainly discuss $p_\theta(y|x)$ not $f_\theta(x,y)$; and (3) we can compare NS- and SCE-based loss functions through the Bregman divergence.

3.1 Various Noise Distributions

Different from the objective distribution of SCE, Eq. (9) is affected by the type of noise distribution $p_n(y|x)$. To investigate the actual objective distribution for $\ell^{NS}(\theta)$, we need to consider separate cases for each type of noise distribution. In this subsection, we further analyze Eq. (9) for each separate case.

3.1.1 NS with Uniform Noise

First, we investigated the case of a uniform distribution because it is one of the most common noise distributions for $\ell^{NS}(\theta)$ in the KGE task. From Eq. (9), we can induce the following property.

Proposition 5. *When $p_n(y|x)$ is a uniform distribution, Eq. (9) equals $p_d(y|x)$.*

Proof. This is described in Appendix C of the supplemental material. □

Dyer (2014) indicated that NS is equal to NCE when $\nu = |Y|$ and $P_n(y|x)$ is uniform. However, as we showed, in terms of the objective distribution, the value of ν is not related to the objective distribution because Eq. (9) is independent of ν.

3.1.2 NS with Frequency-based Noise

In the original setting of NS (Mikolov et al., 2013), the authors chose as $p_n(y|x)$ a unigram distribution of y, which is independent of x. Such a frequency-based distribution is calculated in terms of frequencies on a corpus and independent of the model parameter θ. Since in this case, different from the case of a uniform distribution, $p_n(y|x)$ remains on the right side of Eq. (9), $p_\theta(y|x)$ decreases when $p_n(y|x)$ increases. Thus, we can interpret frequency-based noise as a type of smoothing for $p_d(y|x)$. The smoothing of NS w/ Freq decreases the importance of high-frequency labels in the training data for learning more general vector representations, which can be used for various tasks as pre-trained vectors. Since we can expect pre-trained vectors to work as a prior (Erhan et al., 2010) that prevents models from overfitting, we tried to use NS w/ Freq for pre-training KGE models in our experiments.

3.1.3 Self-Adversarial NS

Sun et al. (2019) recently proposed SANS, which uses $p_\theta(y|x)$ for generating negative samples. By replacing $p_n(y|x)$ with $p_\theta(y|x)$, the objective distribution when using SANS is as follows:

$$p_\theta(y|x) = \frac{p_d(y|x)}{p_{\hat{\theta}}(y|x) \sum_{y_i \in Y} \frac{p_d(y_i|x)}{p_{\hat{\theta}}(y_i|x)}}, \tag{10}$$

where $\hat{\theta}$ is a parameter set updated in the previous iteration. Because both the left and right sides of Eq. (10) include $p_\theta(y|x)$, we cannot obtain an analytical solution of $p_\theta(y|x)$ from this equation. However, we can consider special cases of $p_\theta(y|x)$ to gain an understanding of Eq. (10). At the beginning of the training, $p_\theta(y|x)$ follows a discrete uniform distribution $u\{1,|Y|\}$ because θ is randomly initialized. In this situation, when we set $p_{\hat{\theta}}(y|x)$ in Eq. (10) to a discrete uniform distribution $u\{1,|Y|\}$, $p_\theta(y|x)$ becomes

$$p_\theta(y|x) = p_d(y|x). \tag{11}$$

Next, when we set $p_{\hat{\theta}}(y|x)$ in Eq. (10) as $p_d(y|x)$, $p_\theta(y|x)$ becomes

$$p_\theta(y|x) = u\{1,|Y|\}. \tag{12}$$

In actual mini-batch training, θ is iteratively updated for every batch of data. Because $p_\theta(y|x)$ converges to $u\{1,|Y|\}$ when $p_{\hat{\theta}}(y|x)$ is close to $p_d(y|x)$ and $p_\theta(y|x)$ converges to $p_d(y|x)$ when $p_{\hat{\theta}}(y|x)$ is close to $u\{1,|Y|\}$, we can approximately regard the objective distribution of SANS as a mixture of p_d and $u\{1,|Y|\}$. Thus, we can represent the objective distribution of $p_\theta(y|x)$ as

$$p_\theta(y|x) \approx (1-\lambda)p_d(y|x) + \lambda u\{1,|Y|\} \tag{13}$$

where λ is a hyper-parameter to determine whether $p_\theta(y|x)$ is close to $p_d(y|x)$ or $u\{1,|Y|\}$. Assuming that $p_\theta(y|x)$ starts from $u\{1,|Y|\}$, λ should start from 0 and gradually increase through training. Note that λ corresponds to a temperature α for $p_{\hat{\theta}}(y|x)$ in SANS, defined as

$$p_{\hat{\theta}}(y|x) = \frac{\exp(\alpha f_\theta(x,y))}{\sum_{y' \in Y} \exp(\alpha f_\theta(x,y'))}, \tag{14}$$

where α also adjusts $p_{\hat{\theta}}(y|x)$ to be close to $p_d(y|x)$ or $u\{1,|Y|\}$.

4 Theoretical Relationships among Loss Functions

4.1 Corresponding SCE form to NS with Frequency-based Noise

We induce a corresponding cross entropy loss from the objective distribution for NS with frequency-based noise. We set $T_{x,y} = p_n(y|x) \sum_{y_i \in Y} \frac{p_d(y_i|x)}{p_n(y_i|x)}$, $q(y|x) = T_{x,y}^{-1} p_d(y|x)$, and $\Psi(\mathbf{z}) = \sum_{i=1}^{len(\mathbf{z})} z_i \log z_i$. Under these conditions, following induction from Eq. (4) to Eq. (5), we can reformulate $B_{\Psi(\mathbf{z})}(q(\mathbf{y}|x), p(\mathbf{y}|x))$ as follows:

$$\begin{aligned} & B_{\Psi(\mathbf{z})}(q(\mathbf{y}|x), p_\theta(\mathbf{y}|x)) \\ &= -\sum_{x,y} \left[\sum_{i=1}^{|Y|} T_{x,y}^{-1} p_d(y_i|x) \log p_\theta(y_i|x) \right] p_d(x,y) \\ &= -\frac{1}{|D|} \sum_{(x,y) \in D} T_{x,y}^{-1} \log p_\theta(y|x). \end{aligned} \tag{15}$$

Loss	Objective Distribution	$\Psi(z)$ or $\Psi(\mathbf{z})$	Remarks
NS w/ Uni	$p_d(y\|x)$	$\Psi(z) = z\log(z) - (1+z)\log(1+z)$	
NS w/ Freq	$T_{x,y}^{-1} p_d(y\|x)$	$\Psi(z) = z\log(z) - (1+z)\log(1+z)$	$T_{x,y} = p_n(y\|x) \sum_{y_i \in Y} \frac{p_d(y_i\|x)}{p_n(y_i\|x)}$
SANS	$(1-\lambda)p_d(y\|x) + \lambda u\{1, \|Y\|\}$	$\Psi(z) = z\log(z) - (1+z)\log(1+z)$	Approximately derived. λ increases from zero in training.
SCE	$p_d(y\|x)$	$\Psi(\mathbf{z}) = \sum_{i=1}^{len(\mathbf{z})} z_i \log z_i$	
SCE w/ BC	$T_{x,y}^{-1} p_d(y\|x)$	$\Psi(\mathbf{z}) = \sum_{i=1}^{len(\mathbf{z})} z_i \log z_i$	$T_{x,y} = p_n(y\|x) \sum_{y_i \in Y} \frac{p_d(y_i\|x)}{p_n(y_i\|x)}$
SCE w/ LS	$(1-\lambda)p_d(y\|x) + \lambda u\{1, \|Y\|\}$	$\Psi(\mathbf{z}) = \sum_{i=1}^{len(\mathbf{z})} z_i \log z_i$	λ is fixed.

Table 1: Summary of our theoretical findings. w/ Uni denotes with uniform noise, w/ Freq denotes with frequency-based noise, w/ BC denotes with backward correction, and w/ LS denotes with label smoothing.

Except that $T_{x,y}$ is conditioned by x and not normalized for y, we can interpret this loss function as SCE with backward correction (SCE w/ BC) (Patrini et al., 2017). Taking into account that backward correction can be a smoothing method for predicting labels (Lukasik et al., 2020), this relationship supports the theoretical finding that NS can adopt a smoothing to the objective distribution.

Because the frequency-based noise is used in word2vec as unigram noise, we specifically consider the case in which $p_n(y|x)$ is set to unigram noise. In this case, we can set $p_n(y|x) = p_d(y)$. Since relation tuples do not appear twice in a knowledge graph, we can assume that $p_d(x,y)$ is uniform. Accordingly, we can change $T_{x,y}^{-1}$ to $\frac{1}{p_d(y)\sum_{y_i \in Y} \frac{p_d(y_i|x)}{p_d(y_i)}} = \frac{1}{p_d(y)\sum_{y_i \in Y} \frac{p_d(y_i,x)}{p_d(y_i)p_d(x)}} = \frac{p_d(x)}{p_d(y)C}$, where C is a constant value, and we can reformulate Eq. (15) as follows:

$$
\begin{aligned}
&-\frac{1}{|D|}\sum_{(x,y)\in D}\frac{p_d(x)}{p_d(y)C}\log p_\theta(y|x) \\
\propto &-\frac{1}{|D|}\sum_{(x,y)\in D}\frac{\#x}{\#y}\log p_\theta(y|x), \qquad (16)
\end{aligned}
$$

where $\#x$ and $\#y$ respectively represent frequencies for x and y in the training data. We use Eq. (16) to pre-train models for SCE-based loss functions.

4.2 Corresponding SCE form to SANS

We induce a corresponding cross entropy loss from the objective distribution for SANS by setting $q(y|x) = (1-\lambda)p_d(y|x) + \lambda u\{1,|Y|\}$ and $\Psi(\mathbf{z}) = \sum_{i=1}^{len(\mathbf{z})} z_i \log z_i$. Under these conditions, on the basis of induction from Eq. (4) to Eq. (5), we can reformulate $B_{\Psi(\mathbf{z})}(q(\mathbf{y}|x), p_\theta(\mathbf{y}|x))$ as follows:

$$
\begin{aligned}
&B_{\Psi(\mathbf{z})}(q(\mathbf{y}|x), p_\theta(\mathbf{y}|x)) \\
=&-\sum_{x,y}\Bigg[\sum_{i=1}^{|Y|}(1-\lambda)p_d(y_i|x)\log p_\theta(y_i|x) \\
&\qquad +\sum_{i=1}^{|Y|}\lambda u\{1,|Y|\}\log p_\theta(y_i|x)\Bigg]p_d(x,y) \\
=&-\frac{1}{|D|}\sum_{(x,y)\in D}\Bigg[(1-\lambda)\log p_\theta(y|x) \\
&\qquad +\sum_{i=1}^{|Y|}\frac{\lambda}{|Y|}\log p_\theta(y_i|x)\Bigg]. \qquad (17)
\end{aligned}
$$

The equation in the brackets of Eq. (17) is the cross entropy loss that has a corresponding objective distribution to that of SANS. This loss function is similar in form to SCE with label smoothing (SCE w/ LS) (Szegedy et al., 2016). This relationship also accords with the theoretical finding that NS can adopt a smoothing to the objective distribution.

5 Understanding Loss Functions for Fair Comparisons

We summarize the theoretical findings from Sections 2, 3, and 4 in Table 1. To compare the results from the theoretical findings, we need to understand the differences in their objective distributions and divergences.

5.1 Objective Distributions

The objective distributions for NS w/ Uni and SCE are equivalent. We can also see that the objective distribution for SANS is quite similar to that for SCE w/ LS. These theoretical findings will be important for making a fair comparison between scoring methods trained with the NS and SCE loss functions. When a dataset contains low-frequency

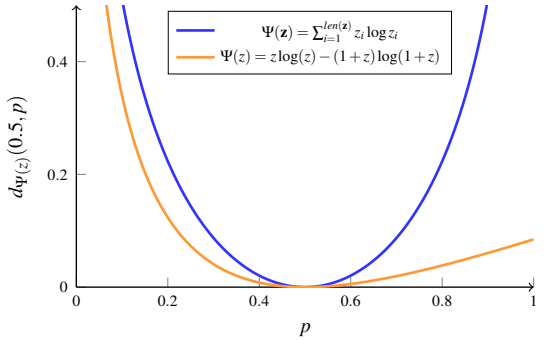

Figure 1: Divergence between 0.5 and p in $d_{\Psi(z)}$ for each $\Psi(z)$.

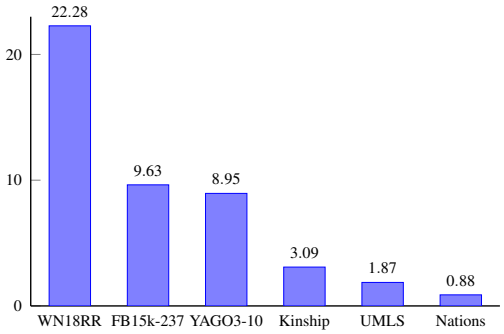

Figure 2: KL divergence of $p_d(y|x)$ between training and test relations for each dataset

entities, SANS and SCE w/ LS can improve the link-prediction performance through their smoothing effect, even if there is no performance improvement from the scoring method itself. For comparing the SCE and NS loss functions fairly, therefore, it is necessary to use the vanilla SCE against NS w/ Uni and use SCE w/ LS against SANS.

However, we still have room to discuss the relationship between SANS and SCE w/ LS because λ in SANS increases from zero during training, whereas λ in SCE w/ LS is fixed. To introduce the behavior of λ in SANS to SCE w/ LS, we tried a simple approach in our experiments that trains KGE models via SCE w/ LS using pre-trained embeddings from SCE as initial parameters. Though this approach is not exactly equivalent to SANS, we expected it to work similarly to increasing λ from zero in training.

We also discuss the relationship between NS w/ Freq and SCE w/ BC. While NS w/ Freq is often used for learning word embeddings, neither NS w/ Freq nor SCE w/ BC has been explored in KGE. We investigated whether these loss functions are effective in pre-training KGE models[2]. Because SANS and SCE w/ LS are similar methods to NS w/ Freq and SCE w/ BC in terms of smoothing, in our experiments, we also compared NS w/ Freq with SANS and SCE w/ BC with SCE w/ LS as pre-training methods.

5.2 Divergences

Comparing $\Psi(z)$ for NS and SCE losses is as important as focusing on their objective distributions. The $\Psi(z)$ determines the distance between model-predicted and data distributions in the loss. It has an important role in determining the behavior of the model. Figure 1 shows the distance in Eq. (3) between the probability p and probability 0.5 for each Ψ in Table 1[3]. As we can see from the example, $d_{\Psi(z)}(0.5, p)$ of the SCE loss has a larger distance than that of the NS loss. In fact, Painsky and Wornell (2020) proved that the upper bound of the Bregman divergence for binary labels when $\Psi(\mathbf{z}) = \sum_{i=1}^{len(\mathbf{z})} z_i \log z_i$. This means that the SCE loss imposes a larger penalty on the same predicted value than the NS loss when the value of the learning target is the same between the two losses[4].

However, this does not guarantee that the distance of SCE is always larger than NS. This is because the values of the learning target between the two losses are not always the same. To take into account the generally satisfied property, we also focus on the convexity of the functions. In each training instance, the first-order and second-order derivatives of these loss functions indicate that SCE is convex, but NS is not in their domains[5]. Since this property is independent of the objective distribution, we can consider SCE fits the model more strongly to the training data in general. Because of these features, SCE can be prone to overfitting.

Whether the overfitting is a problem depends on how large the difference between training and test data is. To measure the difference between training and test data in a KG dataset, we calculated the Kullback-Leibler (KL) divergence for $p(\mathbf{y}|x)$ between the training and test data of commonly used KG datasets. To compute $p(y|x)$, we first calculated

[2] As a preliminary experiment, we also trained KGE models via NS w/ Freq and SCE w/ BC. However, these methods did not improve the link-prediction performance because frequency-based noise changes the data distribution drastically.

[3] In this setting, we can expand $\Psi(\mathbf{z}) = \sum_{i=1}^{len(\mathbf{z})} z_i \log z_i$ to $\Psi(z) = z\log z + (1-z)\log(1-z)$.

[4] See Appendix. E for the further details.

[5] Goldberg and Levy (2014) discuss the convexity of the inner product in NS. Different from theirs, our discussion is about the convexity of the loss functions itself.

Method	Loss	FB15k-237				WN18RR			
		MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
TuckER	NS	0.257	0.151	0.297	0.472	0.431	0.407	0.440	0.473
	SANS	0.330	0.238	0.365	0.512	0.445	0.421	0.455	0.489
	SCE	0.338	0.246	0.372	0.521	0.453	0.424	0.465	0.507
	SCE w/ LS	0.343	0.251	0.378	0.529	0.472	0.441	0.483	0.528
RESCAL	NS	0.337	0.247	0.368	0.516	0.385	0.354	0.405	0.437
	SANS	0.339	0.249	0.372	0.520	0.389	0.363	0.404	0.434
	SCE	0.352	0.260	0.387	0.537	0.451	0.417	0.470	0.512
	SCE w/ LS	0.363	0.269	0.400	0.548	0.469	0.435	0.485	0.529
ComplEx	NS	0.296	0.211	0.324	0.468	0.394	0.373	0.403	0.432
	SANS	0.300	0.214	0.328	0.472	0.432	0.407	0.442	0.480
	SCE	0.300	0.218	0.326	0.466	0.463	0.434	0.473	0.521
	SCE w/ LS	0.318	0.231	0.348	0.493	0.477	0.441	0.491	0.546
DistMult	NS	0.304	0.219	0.336	0.470	0.389	0.374	0.394	0.416
	SANS	0.320	0.234	0.352	0.489	0.410	0.386	0.419	0.452
	SCE	0.342	0.252	0.374	0.521	0.438	0.407	0.447	0.497
	SCE w/ LS	0.344	0.254	0.377	0.526	0.448	0.410	0.460	0.527
TransE	NS	0.284	0.182	0.319	0.498	0.218	0.011	0.390	0.510
	SANS	0.328	0.230	0.365	0.525	0.219	0.016	0.394	0.514
	SCE	0.324	0.232	0.359	0.508	0.229	0.054	0.366	0.523
	SCE w/ LS	0.323	0.231	0.359	0.508	0.229	0.054	0.369	0.522
RotatE	NS	0.301	0.203	0.333	0.505	0.469	0.429	0.484	0.547
	SANS	0.333	0.238	0.371	0.523	0.472	0.431	0.487	0.550
	SCE	0.315	0.228	0.347	0.486	0.452	0.423	0.463	0.507
	SCE w/ LS	0.315	0.228	0.346	0.489	0.447	0.417	0.461	0.502

Table 2: Results for each method in FB15k-237 and WN18RR datasets. Notations are same as those in Table 1.

$p(e_i|r_k,e_j) = p(e_i|r_k) + p(e_i|e_j)$ on the basis of frequencies in the data then calculated $p(e_j|r_k,e_i)$ in the same manner. We treated both $p(e_i|r_k,e_j)$ and $p(e_j|r_k,e_i)$ as $p(y|x)$. We denote $p(\mathbf{y}|x)$ in the training data as P and in the test data as Q. With these notations, we calculated $D_{KL}(P||Q)$ as the KL divergence for $p(\mathbf{y}|x)$ between the test and training data. Figure 2 shows the results. There is a large difference in the KL divergence between FB15k-237 and WN18RR. We investigated how this difference affects the SCE and NS loss functions for learning KGE models.

In a practical setting, the loss function's divergence is not the only factor to affect the fit to the training data. Model selection also affects the fitting. However, understanding a model's behavior is difficult due to the complicated relationship between model parameters. For this reason, we experimentally investigated which combinations of models and loss functions are suitable for link prediction.

6 Experiments and Discussion

We conducted experiments to investigate the validity of what we explained in Section 5 through a comparison of the NS and SCE losses.

6.1 Experimental Settings

We evaluated the following models on the FB15k-237 and WN18RR datasets in terms of the Mean Reciprocal Rank (MRR), Hits@1, Hits@3, and Hits@10 metrics: TuckER (Balazevic et al., 2019); RESCAL (Bordes et al., 2011); ComplEx (Trouillon et al., 2016); DistMult (Yang et al., 2015); TransE (Bordes et al., 2013); RotatE (Sun et al., 2019). We used LibKGE (Broscheit et al., 2020)[6] as the implementation. For each model to be able to handle queries in both directions, we also trained a model for the reverse direction that shares the entity embeddings with the model for the forward direction.

To determine the hyperparameters of these models, for RESCAL, ComplEx, DistMult, and TransE with SCE and SCE w/ LS, we used the settings that achieved the highest performance in a previous study (Ruffinelli et al., 2020) for each loss function as well as the settings from the original papers for TuckER and RotatE. In TransE with NS and SANS, we used the settings used by Sun

[6] https://github.com/uma-pi1/kge

et al. (2019). When applying SANS, we set α to an initial value of 1.0 for LibKGE for all models except TransE and RotatE, and for TransE and RotatE, where we followed the settings of the original paper since SANS was used in it. When applying SCE w/ LS, we set λ to the initial value of LibKGE, 0.3, except on TransE and RotatE. In the original setting of RotatE, because the values of SANS for TransE and RotatE were tuned, we also selected λ from $\{0.3, 0.1, 0.01\}$ using the development data in TransE and RotatE for fair comparison. Appendix D in the supplemental material details the experimental settings.

6.2 Characteristics of Loss functions

Table 2 shows the results for each loss and model combination. In the following subsections, we discuss investigating whether our findings work in a practical setting on the basis of the results.

6.2.1 Objective Distributions

In terms of the objective distribution, when SCE w/ LS improves performance, SANS also improves performance in many cases. Moreover, it accords with our finding that SCE w/ LS and SANS have similar effects. For TransE and RotatE, the relationship does not hold, but as we will see later, this is probably because TransE with SCE and RotatE with SCE did not fit the training data. If the SCE does not fit the training data, the effect of SCE w/ LS is suppressed as it has the same effect as smoothing.

6.2.2 Divergences

Next, let us focus on the distance of the loss functions. A comparison of the results of WN18RR and FB15k-237 shows no performance degradation of SCE compared with NS. This indicates that the difference between the training and test data in WN18RR is not so large to cause overfitting problems for SCE.

In terms of the combination of models and loss functions, the results of NS are worse than those of SCE in TuckER, RESCAL, ComplEx, and DistMult. Because the four models have no constraint to prevent fitting to the training data, we consider that the lower scores are caused by underfitting. This conjecture is on the basis that the NS loss weakly fits model-predicted distributions to training-data distributions compared with the SCE loss in terms of divergence and convexity.

In contrast, the performance gap between NS and SCE is smaller in TransE and RotatE. This is because the score functions of TransE and RotatE have bounds and cannot express minus values. Since SCE has a normalization term, it is difficult to represent values close to 1 when the score function cannot represent negative values. This feature prevents TransE and RotatE from completely fitting to the training data. Therefore, we can assume that NS can be a useful loss function when the score function is bounded.

FB15k-237					
Method	**Pre-train**	**MRR**	**Hits@1**	**Hits@3**	**Hits@10**
RESCAL	-	0.363	0.269	0.400	0.548
+SCE w/ LS	SCE	0.363	0.268	0.400	0.552
	SCE w/ BC	0.361	0.266	0.398	0.547
	SCE w/ LS	0.364	0.269	0.402	0.550
RESCAL	-	0.339	0.249	0.372	0.520
+SANS	NS	0.342	0.251	0.376	0.524
	NS w/ Freq	0.343	0.251	0.378	0.524
	SANS	0.345	0.254	0.380	0.525
WN18RR					
Method	**Pre-train**	**MRR**	**Hits@1**	**Hits@3**	**Hits@10**
ComplEx	-	0.477	0.441	0.491	0.546
+SCE w/ LS	SCE	0.477	0.439	0.493	0.550
	SCE w/ BC	0.469	0.433	0.486	0.533
	SCE w/ LS	0.481	0.444	0.496	0.553
RotatE	-	0.472	0.431	0.487	0.550
+SANS	NS	0.470	0.433	0.483	0.548
	NS w/ Freq	0.470	0.428	0.484	0.553
	SANS	0.471	0.429	0.488	0.552

Table 3: Results of pre-training methods. + denotes combination of model and loss function. Other notations are same as those in Table 1.

6.3 Effectiveness of Pre-training Methods

We also explored pre-training for learning KGE models. We selected the methods in Table 2 that achieved the best MRR for each NS-based loss and each SCE-based loss in each dataset. In accordance with the success of word2vec, we chose unigram noise for both NS w/ Freq and SCE w/ BC.

Table 3 shows the results. Contrary to our expectations, SCE w/ BC does not work well as a pre-training method. Because the unigram noise for SCE w/ BC can drastically change the original data distribution, SCE w/ BC is thought to be effective when the difference between training and test data is large. However, since the difference is not so large in the KG datasets, as discussed in the previous subsection, we believe that the unigram noise may be considered unsuitable for these datasets.

Compared with SCE w/ BC, both SCE w/ LS and SANS are effective for pre-training. This is because the hyperparameters of SCE w/ LS and

SANS are adjusted for KG datasets.

When using vanilla SCE as a pre-training method, there is little improvement in prediction performance, compared with other methods. This result suggests that increasing λ in training is not as important for improving task performance.

For RotatE, there is no improvement in pre-training. Because RotatE has strict constraints on its relation representation, we believe it may degrade the effectiveness of pre-training.

7 Related Work

Mikolov et al. (2013) proposed the NS loss function as an approximation of the SCE loss function to reduce computational cost and handle a large vocabulary for learning word embeddings. NS is now used in various NLP tasks, which must handle a large amount of vocabulary or labels. Melamud et al. (2017) used the NS loss function for training a language model. Trouillon et al. (2016) introduced the NS loss function to KGE. In contextualized pre-trained embeddings, Clark et al. (2020a) indicated that ELECTRA (Clark et al., 2020b), a variant of BERT (Devlin et al., 2019), follows the same manner of the NS loss function.

NS is frequently used to train KGE models. KGE is a task to complement a knowledge graph that describes relationships between entities. Knowledge graphs are used in various important downstream tasks because of its convenience in incorporating external knowledge, such as in a language model (Logan et al., 2019), dialogue (Moon et al., 2019), question-answering (Lukovnikov et al., 2017), natural language inference (K M et al., 2018), and named entity recognition (He et al., 2020). Thus, current KGE is important in NLP.

Due to the importance of KGE, various scoring methods including RESCAL (Bordes et al., 2011), TransE (Bordes et al., 2013), DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016), TuckER (Balazevic et al., 2019), and RotatE (Sun et al., 2019) used in our experiment, have been proposed. However, the relationship between these score functions and loss functions is not clear. Several studies (Ruffinelli et al., 2020; Ali et al., 2020) have investigated the best combinations of scoring method, loss function, and their hyperparameters in KG datasets. These studies differ from ours in that they focused on empirically searching for good combinations rather than theoretical investigations.

As a theoretical study, Levy and Goldberg (2014) showed that NS is equivalent to factorizing a matrix for PMI when a unigram distribution is selected as a noise distribution. Dyer (2014) investigated the difference between NCE (Gutmann and Hyvärinen, 2010) and NS. Gutmann and Hirayama (2011) revealed that NCE is derivable from Bregman divergence. Our derivation for NS is inspired by their work. Meister et al. (2020) proposed a framework to jointly interpret label smoothing and confidence penalty (Pereyra et al., 2017) through investigating their divergence. Yang et al. (2020) theoretically induced that a noise distribution that is close to the true distribution behind the training data is suitable for training KGE models in NS. They also proposed a variant of SANS in the basis of their investigation.

Different from these studies, we investigated the distributions at optimal solutions of SCE and NS loss functions while considering several types of noise distribution in NS.

8 Conclusion

We revealed the relationships between SCE and NS loss functions in KGE. Through theoretical analysis, we showed that SCE and NS w/ Uni are equivalent in objective distribution, which is the predicted distribution of a model at an optimal solution, and that SCE w/ LS and SANS have similar objective distributions. We also showed that SCE more strongly fits a model to the training data than NS due to the divergence and convexity of SCE.

The experimental results indicate that the differences in the divergence of the two losses were not large enough to affect dataset differences. The results also indicate that SCE works well with highly flexible scoring methods, which do not have any bound of the scores, while NS works well with RotatE, which cannot express minus values due to its bounded scoring. Moreover, they indicate that SCE and SANS work better in pre-training than NS w/ Uni, commonly used for learning word embeddings.

For future work, we will investigate the properties of loss functions in out-of-domain data.

Acknowledgements

This work was partially supported by JSPS Kakenhi Grant nos. 19K20339, 21H03491, and 21K17801.

References

Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Sahand Sharifzadeh, Volker Tresp, and Jens Lehmann. 2020. Pykeen 1.0: A python library for training and evaluating knowledge graph emebddings. *arXiv preprint arXiv:2007.14175*.

Ivana Balazevic, Carl Allen, and Timothy Hospedales. 2019. TuckER: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5185–5194, Hong Kong, China. Association for Computational Linguistics.

Arindam Banerjee, Srujana Merugu, Inderjit S. Dhillon, and Joydeep Ghosh. 2005. Clustering with bregman divergences. *Journal of Machine Learning Research*, 6(58):1705–1749.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, volume 26, pages 2787–2795. Curran Associates, Inc.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI'11, page 301–306. AAAI Press.

L.M. Bregman. 1967. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200 – 217.

Samuel Broscheit, Daniel Ruffinelli, Adrian Kochsiek, Patrick Betz, and Rainer Gemulla. 2020. LibKGE - A knowledge graph embedding library for reproducible research. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 165–174.

Kevin Clark, Minh-Thang Luong, Quoc Le, and Christopher D. Manning. 2020a. Pre-training transformers as energy-based cloze models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 285–294, Online. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020b. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, pages 1811–1818.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Chris Dyer. 2014. Notes on noise contrastive estimation and negative sampling. *arXiv preprint arXiv:1410.8251*.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11:625–660.

Yoav Goldberg and Omer Levy. 2014. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *CoRR*, abs/1402.3722.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304.

Michael U. Gutmann and Jun-ichiro Hirayama. 2011. Bregman divergence as general framework to estimate unnormalized statistical models. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, UAI'11, page 283–290, Arlington, Virginia, USA. AUAI Press.

Qizhen He, Liang Wu, Yida Yin, and Heming Cai. 2020. Knowledge-graph augmented word representations for named entity recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7919–7926.

Annervaz K M, Somnath Basu Roy Chowdhury, and Ambedkar Dukkipati. 2018. Learning beyond datasets: Knowledge graph augmented neural networks for natural language processing. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 313–322, New Orleans, Louisiana. Association for Computational Linguistics.

Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. 2017. Knowledge base completion: Baselines strike back. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 69–74, Vancouver, Canada. Association for Computational Linguistics.

Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2177–2185, Cambridge, MA, USA. MIT Press.

Robert Logan, Nelson F. Liu, Matthew E. Peters, Matt Gardner, and Sameer Singh. 2019. Barack's wife hillary: Using knowledge graphs for fact-aware language modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5962–5971, Florence, Italy. Association for Computational Linguistics.

Michal Lukasik, Srinadh Bhojanapalli, Aditya Menon, and Sanjiv Kumar. 2020. Does label smoothing mitigate label noise? In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 6448–6458. PMLR.

Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer. 2017. Neural network-based question answering over knowledge graphs on word and character level. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 1211–1220, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Clara Meister, Elizabeth Salesky, and Ryan Cotterell. 2020. Generalized entropy regularization or: There's nothing special about label smoothing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6870–6886, Online. Association for Computational Linguistics.

Oren Melamud, Ido Dagan, and Jacob Goldberger. 2017. A simple language model based on PMI matrix approximations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1860–1865, Copenhagen, Denmark. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Seungwhan Moon, Pararth Shah, Anuj Kumar, and Rajen Subba. 2019. OpenDialKG: Explainable conversational reasoning with attention-based walks over knowledge graphs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 845–854, Florence, Italy. Association for Computational Linguistics.

A. Painsky and G. W. Wornell. 2020. Bregman divergence bounds and universality properties of the logarithmic loss. *IEEE Transactions on Information Theory*, 66(3):1658–1673.

Giorgio Patrini, Alessandro Rozza, Aditya Krishna Menon, Richard Nock, and Lizhen Qu. 2017. Making deep neural networks robust to label noise: A loss correction approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. 2017. Regularizing neural networks by penalizing confident output distributions. *CoRR*, abs/1701.06548.

Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. 2020. You can teach an old dog new tricks! on training knowledge graph embeddings. In *International Conference on Learning Representations*.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

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

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, Beijing, China. Association for Computational Linguistics.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *ICML*, pages 2071–2080.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Zhen Yang, Ming Ding, Chang Zhou, Hongxia Yang, Jingren Zhou, and Jie Tang. 2020. Understanding negative sampling in graph representation learning. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '20, page 1666–1676, New York, NY, USA. Association for Computing Machinery.

A Proof of Proposition 1, 2, and 3

We can reformulate ℓ_{NS} as follows:

$$\begin{aligned}
\ell^{NS}(\theta) &= -\frac{1}{|D|}\sum_{(x,y)\in D}\left(\log(P(C=1,y|x;\theta)) + \sum_{i=1,y_i\sim p_n}^{\nu}\log(P(C=0,y_i|x;\theta))\right)\\
&= -\frac{1}{|D|}\sum_{(x,y)\in D}\log(P(C=1,y|x;\theta)) - \frac{1}{|D|}\sum_{(x,y)\in D}\sum_{i=1,y_i\sim p_n}^{\nu}\log(P(C=0,y_i|x;\theta))\\
&= -\frac{1}{|D|}\sum_{(x,y)\in D}\log(\frac{1}{1+G(y|x;\theta)}) - \frac{1}{|D|}\sum_{(x,y)\in D}\sum_{i=1,y_i\sim p_n}^{\nu}\log(\frac{G(y_i|x;\theta)}{1+G(y_i|x;\theta)})\\
&= \frac{1}{|D|}\sum_{(x,y)\in D}\log(1+G(y|x;\theta)) + \frac{\nu}{\nu|D|}\sum_{(x,y)\in D}\sum_{i=1,y_i\sim p_n}^{\nu}\log(1+\frac{1}{G(y_i|x;\theta)})\\
&= \sum_{x,y}p_d(y|x)\log(1+G(y|x;\theta))p_d(x) + \sum_{x,y}\nu p_n(y|x)\log(1+\frac{1}{G(y|x;\theta)})p_d(x)
\end{aligned} \tag{18}$$

Letting $u=(x,y)$, $f(u)=\frac{\nu p_n(y|x)}{p_d(y|x)}$, $g(u)=G(y|x;\theta)$, and $p_d(x)=\frac{1}{p_d(y|x)}p_d(x,y)$, we can reformulate Eq. (18) as:

$$\begin{aligned}
\ell^{NS}(\theta) &= \left(\sum_{x,y}p_d(y|x)\log(1+g(u))\frac{1}{p_d(y|x)}p_d(x,y) + \sum_{x,y}\nu p_n(y|x)\log(1+\frac{1}{g(u)})\frac{1}{p_d(y|x)}p_d(x,y)\right)\\
&= \sum_{x,y}\left[\log(1+g(u)) + \log(1+\frac{1}{g(u)})f(u)\right]p_d(x,y)\\
&= \sum_{x,y}\Big[\log(1+g(u)) - \log(g(u))f(u) + \log(1+g(u))f(u)\Big]p_d(x,y)\\
&= \sum_{x,y}\Big[-g(u)\log(1+g(u)) + (1+g(u))\log(1+g(u))\\
&\quad + \log(g(u))g(u) + \log(1+g(u))g(u) - \log(g(u))f(u) + \log(1+g(u))f(u)\Big]p_d(x,y)
\end{aligned} \tag{19}$$

With $\Psi(g(u)) = g(u)\log(g(u)) - (1+g(u))\log(1+g(u))$ and $\nabla\Psi(g(u)) = \log(g(u)) - \log(1+g(u))$, we can reformulate Eq. (19) as:

$$\begin{aligned}
\ell^{NS}(\theta) &= \sum_{x,y}\Big[-\Psi(g(u)) + \nabla\Psi(g(u))g(u) - \nabla\Psi(g(u))f(u)\Big]p_d(x,y)\\
&= B_{\Psi(z)}(f(u),g(u)).
\end{aligned} \tag{20}$$

From Eq. (20), when $\ell^{NS}(\theta)$ is minimized, $g(u)=f(u)$ is satisfied. In this condition, $G(y|x;\theta)$ becomes $\frac{\nu p_n(y|x)}{p_d(y|x)}$, and $\exp(f_\theta(x,y))$ becomes $\frac{p_d(y|x)}{\nu p_n(y|x)}$ as follows:

$$g(u)=f(u) \Leftrightarrow G(y|x;\theta)=\frac{\nu p_n(y|x)}{p_d(y|x)} \Leftrightarrow \exp(f_\theta(x,y)) = \frac{p_d(y|x)}{\nu p_n(y|x)}. \tag{21}$$

Based on the Eq. (1) and Eq. (21), the objective distribution for $p_\theta(y|x)$ is as follows:

$$p_\theta(y|x) = \frac{p_d(y|x)}{p_n(y|x)\sum_{y_i\in Y}\frac{p_d(y_i|x)}{p_n(y_i|x)}}. \tag{22}$$

B Proof of Proposition 4

PMI is induced by multiplying $p_d(x)$ to the right-hand side of Eq. (8) and then computing logarithm for both sides as follows:

$$G(y|x;\theta)=\frac{p_n(y|x)}{p_d(y|x)} \Leftrightarrow \exp(f_\theta(x,y)) = \frac{p_d(y|x)}{p_n(y|x)} = \frac{p_d(y|x)}{p_d(y)} = \frac{p_d(x,y)}{p_d(x)p_d(y)} \Leftrightarrow f_\theta(x,y) = \log\frac{p_d(x,y)}{p_d(y)p_d(y)} \tag{23}$$

C Proof of Proposition 5

When $p_n(y|x)$ is a uniform distribution, $p_n(y|x) \sum_{y_i \in Y} \frac{p_d(y_i|x)}{p_n(y_i|x)} = \sum_{y_i \in Y} p_d(y_i|x) = 1$, and thus, Eq. (9) becomes $p_d(y|x)$.

D Experimental Details

Dataset: We use FB15k-237 (Toutanova and Chen, 2015)[7] and WN18RR (Dettmers et al., 2018)[8] datasets in the experiments. We followed the standard split in the original papers for each dataset. Table 4 lists the statistics for each dataset.

Dataset	Entities	Relations	Tuples		
			Train	Valid	Test
WN18RR	40,943	11	86,835	3,034	3,134
FB15k-237	14,541	237	272,115	17,535	20,466

Table 4: The numbers of each instance for each dataset.

Metric: We evaluated the link prediction performance of models with MRR, Hits@1, Hits@3, and Hits@10 by ranking test triples against all other triples not appeared in the training, valid, and test datasets. We used LibKGE for calculating these metric scores.
Model: We compared the following models: TuckER (Balazevic et al., 2019); RESCAL (Bordes et al., 2011); ComplEx (Trouillon et al., 2016); DistMult (Yang et al., 2015); TransE (Bordes et al., 2013); RotatE (Sun et al., 2019). For each model, we also trained a model for the reverse direction that shares the entity embeddings with the model for the forward direction. Thus, the dimension size of subject and object embeddings are the same in all models.
Implementation: We used LibKGE (Broscheit et al., 2020)[9] as the implementation. We used its 1vsAll setting for SCE-based loss functions and negative sampling setting for NS-based loss functions. We modified LibKGE to be able to use label smoothing on the 1vsAll setting. We also incorporated NS w/ Freq and SCE w/ BC into the implementation.
Hyper-parameter: Table 5 and 6 show the hyper-parameter settings of each method for each dataset. In RESCAL, ComplEx, and DistMult we used the settings that achieved the highest performance for each loss function in the previous study (Ruffinelli et al., 2020)[10]. In TuckER and RotatE, we follow the settings from the original paper. When applying SANS, we set α to an initial value of 1.0 for LibKGE for all models except TransE and RotatE, and for TransE and RotatE, where we followed the settings of the original paper of SANS since SANS was used in it. When applying SCE w/ LS, we set λ to the initial value of LibKGE, 0.3, except on TransE and RotatE. In the original setting of TransE and RotatE, because the value of SANS was tuned for comparison, for fairness, we selected λ from $\{0.3, 0.1, 0.01\}$ by using the development data through a single run for each value. We set the maximum epoch to 800. We calculated MRR every five epochs on the developed data, and the training was terminated when the highest value was not updated ten times. We chose the best model by using the MRR score on the development data. These hyperparameters were also used in the pre-training step.
Validation Score Table 7, 8, and 9 show the best MRR scores of each loss for each model on the validation dataset.
Device: In all models, we used a single NVIDIA RTX2080Ti for training. Except for RotetE with SCE-based loss functions, all models finished the training in one day. The RotetE with SCE-based loss function finished the training in at most one week.

[7] https://www.microsoft.com/en-us/download/confirmation.aspx?id=52312
[8] https://github.com/TimDettmers/ConvE
[9] https://github.com/uma-pi1/kge
[10] https://github.com/uma-pi1/kge-iclr20

FB15k-237																	
Model		Batch	Dim	Initialize	Regularize			Dropout		Optimizer				Sample		λ	α
					Type	Entity	Relation	Entity	Rel.	Type	LR	Decay	P.	sub.	obj.		
TuckER	SCE	128	200	xn: 1.0	-	-	-	0.3	0.4	Adam	0.0005	-	-	All	All	-	-
	SCE w/ LS	128	200	xn: 1.0	-	-	-	0.3	0.4	Adam	0.0005	-	-	All	All	0.3	-
	NS	128	200	xn: 1.0	-	-	-	0.3	0.4	Adam	0.0005	-	-	All	All	-	-
	SANS	128	200	xn: 1.0	-	-	-	0.3	0.4	Adam	0.0005	-	-	All	All	-	1.0
Rescal	SCE	512	128	n: 0.123	-	-	-	0.427	0.159	Adam	7.39E-5	0.95	1	All	All	-	-
	SCE w/ LS	512	128	n: 0.123	-	-	-	0.427	0.159	Adam	7.39E-5	0.95	1	All	All	0.3	-
	NS	256	128	xn: 1.0	lp: 3	1.22E-12	4.80E-14	0.347	-	Adagrad	0.0170	0.95	5	22	155	-	-
	SANS	256	128	xn: 1.0	lp: 3	1.22E-12	4.80E-14	0.347	-	Adagrad	0.0170	0.95	5	22	155	-	1.0
ComlEx	SCE	512	128	u: 0.311	-	-	-	0.0476	0.443	Adagrad	0.503	0.95	7	All	All	-	-
	SCE w/ LS	512	128	u: 0.311	-	-	-	0.0476	0.443	Adagrad	0.503	0.95	7	All	All	0.3	-
	NS	512	256	n: 4.81E-5	lp: 2	6.34E-9	9.08E-18	0.182	0.0437	Adagrad	0.241	0.95	4	1	48	-	-
	SANS	512	256	n: 4.81E-5	lp: 2	6.34E-9	9.08E-18	0.182	0.0437	Adagrad	0.241	0.95	4	1	48	-	1.0
DistMult	SCE	512	128	n: 0.806	-	-	-	0.370	0.280	Adam	0.00063	0.95	1	All	All	-	-
	SCE	512	128	n: 0.806	-	-	-	0.370	0.280	Adam	0.00063	0.95	1	All	All	0.3	-
	NS	1024	256	u: 0.848	lp: 3	1.55E-10	3.93E-15	0.455	0.360	Adagrad	0.141	0.95	9	557	367	-	-
	SANS	1024	256	u: 0.848	lp: 3	1.55E-10	3.93E-15	0.455	0.360	Adagrad	0.141	0.95	9	557	367	-	1.0
TransE	SCE	128	128	u: 1.0E-5	-	-	-	-	-	Adam	0.0003	0.95	5	All	All	-	-
	SCE w/ LS	128	128	u: 1.0E-5	-	-	-	-	-	Adam	0.0003	0.95	5	All	All	0.01	-
	NS	1024	1000	xu: 1.0	-	-	-	-	-	Adam	0.00005	0.95	5	256	256	-	-
	SANS	1024	1000	xu: 1.0	-	-	-	-	-	Adam	0.00005	0.95	5	256	256	-	1.0
Rotate	SCE	1024	1000	xu: 1.0	-	-	-	-	-	Adam	0.00005	0.95	5	All	All	-	-
	SCE w/ LS	1024	1000	xu: 1.0	-	-	-	-	-	Adam	0.00005	0.95	5	All	All	0.01	-
	NS	1024	1000	xu: 1.0	-	-	-	-	-	Adam	0.00005	0.95	5	256	256	-	-
	SANS	1024	1000	xu: 1.0	-	-	-	-	-	Adam	0.00005	0.95	5	256	256	-	1.0

Table 5: The hyper-parameters for each model in FB15k-237. Rel. denotes relation, P. denotes patience, sub. denotes subjective, obj. denotes objective, xn denotes xavier normal, n denotes normal, xu denotes xavier uniform, and u denotes uniform.

WN18RR																	
Model		Batch	Dim	Initialize	Regularize			Dropout		Optimizer				Sample		λ	α
					Type	Entity	Relation	Entity	Rel.	Type	LR	Decay	P.	sub.	obj.		
TuckER	SCE	128	200	xn: 1.0	-	-	-	0.2	0.2	Adam	0.0005	-	-	All	All	-	-
	SCE w/ LS	128	200	xn: 1.0	-	-	-	0.2	0.2	Adam	0.0005	-	-	All	All	0.3	-
	NS	128	200	xn: 1.0	-	-	-	0.2	0.2	Adam	0.0005	-	-	All	All	-	-
	SANS	128	200	xn: 1.0	-	-	-	0.2	0.2	Adam	0.0005	-	-	All	All	-	1.0
Rescal	SCE	512	256	xn: 1.0	-	-	-	-	-	Adam	0.00246	0.95	9	All	All	-	-
	SCE w/ LS	512	256	xn: 1.0	-	-	-	-	-	Adam	0.00246	0.95	9	All	All	0.3	-
	NS	512	128	n: 1.64E-4	-	-	-	-	-	Adam	0.00152	0.95	1	6	8	-	-
	SANS	512	128	n: 1.64E-4	-	-	-	-	-	Adam	0.00152	0.95	1	6	8	-	1.0
ComlEx	SCE	512	128	u: 0.281	lp: 2	4.52E-6	4.19E-10	0.359	0.311	Adagrad	0.526	0.95	5	All	All	-	-
	SCE w/ LS	512	128	u: 0.281	lp: 2	4.52E-6	4.19E-10	0.359	0.311	Adagrad	0.526	0.95	5	All	All	0.3	-
	NS	1024	128	xn: 1.0	-	-	-	0.0466	0.0826	Adam	3.32E-5	0.95	7	6	6	-	-
	SANS	1024	128	xn: 1.0	-	-	-	0.0466	0.0826	Adam	3.32E-5	0.95	7	6	6	-	1.0
DistMult	SCE	512	128	u: 0.311	lp: 2	1.44E-18	1.44E-18	0.0476	0.443	Adagrad	0.503	0.95	7	All	All	-	-
	SCE w/ LS	512	128	u: 0.311	lp: 2	1.44E-18	1.44E-18	0.0476	0.443	Adagrad	0.503	0.95	7	All	All	0.3	-
	NS	1024	128	xn: 1.0	-	-	-	0.0466	0.0826	Adam	3.32E-5	0.95	7	6	6	-	-
	SANS	1024	128	xn: 1.0	-	-	-	0.0466	0.0826	Adam	3.32E-5	0.95	7	6	6	-	1.0
TransE	SCE	128	512	xn: 1.0	lp: 2	2.13E-7	8.99E-13	0.252	-	Adagrad	0.253	0.95	5	All	All	-	-
	SCE w/ LS	128	512	xn: 1.0	lp: 2	2.13E-7	8.99E-13	0.252	-	Adagrad	0.253	0.95	5	All	All	0.01	-
	NS	512	500	xu: 1.0	-	-	-	-	-	Adam	0.00005	0.95	5	1024	1024	-	-
	SANS	512	500	xu: 1.0	-	-	-	-	-	Adam	0.00005	0.95	5	1024	1024	-	0.5
Rotate	SCE	512	500	xu: 1.0	-	-	-	-	-	Adam	0.00005	0.95	5	All	All	-	-
	SCE w/ LS	512	500	xu: 1.0	-	-	-	-	-	Adam	0.00005	0.95	5	All	All	0.01	-
	NS	512	500	xu: 1.0	-	-	-	-	-	Adam	0.00005	0.95	5	1024	1024	-	-
	SANS	512	500	xu: 1.0	-	-	-	-	-	Adam	0.00005	0.95	5	1024	1024	-	0.5

Table 6: The hyper-parameters for each model in WN18RR. The notations are the same as Table 5.

Model	Loss	FB15k-237	WN18RR
TuckER	SCE	0.345	0.451
	SCE w/ LS	0.350	0.470
	NS	0.261	0.433
	SANS	0.337	0.441
RESCAL	SCE	0.359	0.461
	SCE w/ LS	0.369	0.474
	NS	0.344	0.389
	SANS	0.344	0.390
ComplEx	SCE	0.304	0.468
	SCE w/ LS	0.324	0.478
	NS	0.302	0.399
	SANS	0.308	0.433
DistMult	SCE	0.350	0.441
	SCE w/ LS	0.351	0.451
	NS	0.308	0.391
	SANS	0.326	0.412
TransE	SCE	0.328	0.227
	SCE w/ LS	0.322	0.220
	NS	0.289	0.216
	SANS	0.333	0.218
RotatE	SCE	0.320	0.452
	SCE w/ LS	0.320	0.449
	NS	0.306	0.472
	SANS	0.340	0.475

Table 7: The best MRR scores on validation data.

Dataset	Mehotd	MRR
FB15k-237	RESCAL+SCE w/BC	0.149
	RESCAL+NS w/ Freq	0.171
WN18RR	ComplEx+SCE w/ BC	0.361
	RotatE+NS w/ Freq	0.469

Table 8: The best MRR scores of pre-trained models on validation data.

FB15k-237		
Method	Pretrain	MRR
RESCAL+SCE w / LS	SCE	0.369
	SCE w/ BC	0.369
	SCE w/ LS	0.371
RESCAL+SANS	NS	0.349
	NS w/ Freq	0.348
	SANS	0.350
WN18RR		
Method	Pretrain	MRR
ComplEx+SCE w/ LS	SCE	0.483
	SCE w/ BC	0.469
	SCE w/ LS	0.481
RotatE+SANS	NS	0.472
	NS w/ Freq	0.474
	SANS	0.475

Table 9: The best MRR scores of models initialized with pre-trained embeddings on validation data.

E Note: the divergence between the NS and SCE loss functions

Painsky and Wornell (2020) proved that the upper bound of the Bregman divergence for binary labels when $\Psi(\mathbf{z}) = \sum_{i=1}^{len(\mathbf{z})} z_i \log z_i$. However, to compare the SCE and NS loss functions in general, we need to consider the divergence of multi labels in SCE. When $\Psi(\mathbf{z}) = \sum_{i=1}^{len(\mathbf{z})} z_i \log z_i$, we can derive the following inequality by using the log sum inequality:

$$
\begin{aligned}
& d_{\Psi(\mathbf{z})}(p_d(\mathbf{y}|x), p_\theta(\mathbf{y}|x)) \\
=& \sum_{i=1}^{|Y|} p_d(y_i|x) \log \frac{p_d(y_i|x)}{p_\theta(y_i|x)} \\
=& p_d(y_j|x) \log \frac{p_d(y_j|x)}{p_\theta(y_j|x)} + \sum_{i \neq j}^{|Y|} p_d(y_i|x) \log \frac{p_d(y_i|x)}{p_\theta(y_i|x)} \\
\geqq& p_d(y_j|x) \log \frac{p_d(y_j|x)}{p_\theta(y_j|x)} + (\sum_{i \neq j}^{|Y|} p_d(y_i|x)) \log \frac{(\sum_{i \neq j}^{|Y|} p_d(y_i|x))}{(\sum_{i \neq j}^{|Y|} p_\theta(y_i|x))} \\
=& p_d(y_j|x) \log \frac{p_d(y_j|x)}{p_\theta(y_j|x)} + (1 - p_d(y_j|x)) \log \frac{(1 - p_d(y_j|x))}{(1 - p_\theta(y_j|x))}.
\end{aligned} \tag{24}
$$

Eq. (24) shows that the divergence of multi labels is larger than that of binary labels in SCE. As we explained, $d_{\Psi(\mathbf{z})}(f,g)$ of SCE is larger than $d_{\Psi(z)}(f,g)$ of NS in binary labels. Therefore, the SCE loss imposes a larger penalty on the same predicted value than the NS loss when the value of the learning target is the same between the two losses.