Incremental Skip-gram Model with Negative Sampling

Nobuhiro Kaji and **Hayato Kobayashi**
Yahoo Japan Corporation
{nkaji,hakobaya}@yahoo-corp.jp

Abstract

This paper explores an incremental training strategy for the skip-gram model with negative sampling (SGNS) from both empirical and theoretical perspectives. Existing methods of neural word embeddings, including SGNS, are multi-pass algorithms and thus cannot perform incremental model update. To address this problem, we present a simple incremental extension of SGNS and provide a thorough theoretical analysis to demonstrate its validity. Empirical experiments demonstrated the correctness of the theoretical analysis as well as the practical usefulness of the incremental algorithm.

1 Introduction

Existing methods of neural word embeddings are typically designed to go through the entire training data multiple times. For example, negative sampling (Mikolov et al., 2013b) needs to pre-compute the noise distribution from the entire training data before performing Stochastic Gradient Descent (SGD). It thus needs to go through the training data at least twice. Similarly, hierarchical soft-max (Mikolov et al., 2013b) has to determine the tree structure and GloVe (Pennington et al., 2014) has to count co-occurrence frequencies before performing SGD.

The fact that those existing methods are multi-pass algorithms means that they cannot perform incremental model update when additional training data is provided. Instead, they have to re-train the model on the old and new training data from scratch.

However, the re-training is obviously inefficient since it has to process the entire training data received thus far whenever new training data is provided. This is especially problematic when the amount of the new training data is relatively smaller than the old one. One such situation is that the embedding model is updated on a small amount of training data that includes newly emerged words for instantly adding them to the vocabulary set. Another situation is that the word embeddings are learned from ever-evolving data such as news articles and microblogs (Peng et al., 2017) and the embedding model is periodically updated on newly generated data (*e.g.*, once in a week or month).

This paper investigates an incremental training method of word embeddings with a focus on the skip-gram model with negative sampling (SGNS) (Mikolov et al., 2013b) for its popularity. We present a simple incremental extension of SGNS, referred to as *incremental SGNS*, and provide a thorough theoretical analysis to demonstrate its validity. Our analysis reveals that, under a mild assumption, the optimal solution of incremental SGNS agrees with the original SGNS when the training data size is infinitely large. See Section 4 for the formal and strict statement. Additionally, we present techniques for the efficient implementation of incremental SGNS.

Three experiments were conducted to assess the correctness of the theoretical analysis as well as the practical usefulness of incremental SGNS. The first experiment empirically investigates the validity of the theoretical analysis result. The second experiment compares the word embeddings learned by incremental SGNS and the original SGNS across five benchmark datasets, and demonstrates that those word embeddings are of comparable quality. The last experiment explores the training time of incremental SGNS, demonstrating that it is able to save much training time by avoiding expensive re-training when additional training data is provided.

2 SGNS Overview

As a preliminary, this section provides a brief overview of SGNS.

Given a word sequence, $w_1, w_2, \ldots, w_n$, for training, the skip-gram model seeks to minimize the following objective to learn word embeddings:

$$\mathcal{L}_{\mathrm{SG}} = -\frac{1}{n} \sum_{i=1}^{n} \sum_{\substack{|j| \le c \\ j \ne 0}} \log p(w_{i+j} \mid w_i),$$

where w_i is a target word and w_{i+j} is a context word within a window of size c. $p(w_{i+j} \mid w_i)$ represents the probability that w_{i+j} appears within the neighbor of w_i, and is defined as

$$p(w_{i+j} \mid w_i) = \frac{\exp(\mathbf{t}_{w_i} \cdot \mathbf{c}_{w_{i+j}})}{\sum_{w \in \mathcal{W}} \exp(\mathbf{t}_{w_i} \cdot \mathbf{c}_{w})}, \qquad (1)$$

where $\mathbf{t}_w$ and $\mathbf{c}_w$ are w's embeddings when it behaves as a target and context, respectively. $\mathcal{W}$ represents the vocabulary set.

Since it is too expensive to optimize the above objective, Mikolov et al. (2013b) proposed negative sampling to speed up skip-gram training. This approximates Eq. (1) using sigmoid functions and k randomly-sampled words, called *negative samples*. The resulting objective is given as

$$\mathcal{L}_{\mathrm{SGNS}} = -\frac{1}{n} \sum_{i=1}^{n} \sum_{\substack{|j| \le c \\ j \ne 0}} \psi^{+}_{w_i, w_{i+j}} + k \mathbb{E}_{v \sim q(v)}[\psi^{-}_{w_i, v}],$$

where $\psi^{+}_{w,v} = \log \sigma(\mathbf{t}_w \cdot \mathbf{c}_v)$, $\psi^{-}_{w,v} = \log \sigma(-\mathbf{t}_w \cdot \mathbf{c}_v)$, and $\sigma(x)$ is the sigmoid function. The negative sample v is drawn from a smoothed unigram probability distribution referred to as *noise distribution*: $q(v) \propto f(v)^{\alpha}$, where $f(v)$ represents the frequency of a word v in the training data and α is a smoothing parameter ($0 < \alpha \le 1$).

The objective is optimized by SGD. Given a target-context word pair (w_i and w_{i+j}) and k negative samples ($v_1, v_2, \ldots, v_k$) drawn from the noise distribution, the gradient of $-\psi^{+}_{w_i, w_{i+j}} - k \mathbb{E}_{v \sim q(v)}[\psi^{-}_{w_i, v}] \approx -\psi^{+}_{w_i, w_{i+j}} - \sum_{k'=1}^{k} \psi^{-}_{w_i, v_{k'}}$ is computed. Then, the gradient descent is performed to update $\mathbf{t}_{w_i}$, $\mathbf{c}_{w_{i+j}}$, and $\mathbf{c}_{v_1}, \ldots, \mathbf{c}_{v_k}$.

SGNS training needs to go over the entire training data to pre-compute the noise distribution $q(v)$ before performing SGD. This makes it difficult to perform incremental model update when additional training data is provided.

3 Incremental SGNS

This section explores incremental training of SGNS. The incremental training algorithm (Section 3.1), its efficient implementation (Section 3.2), and the computational complexity (Section 3.3) are discussed in turn.

3.1 Algorithm

Algorithm 1 presents *incremental SGNS*, which goes through the training data in a single-pass to update word embeddings incrementally. Unlike the original SGNS, it does not pre-compute the noise distribution. Instead, it reads the training data word by word[1] to incrementally update the word frequency distribution and the noise distribution while performing SGD. Hereafter, the original SGNS (*c.f.*, Section 2) is referred to as *batch SGNS* to emphasize that the noise distribution is computed in a batch fashion.

The learning rate for SGD is adjusted by using AdaGrad (Duchi et al., 2011). Although the linear decay function has widely been used for training batch SGNS (Mikolov, 2013), adaptive methods such as AdaGrad are more suitable for the incremental training since the amount of training data is unknown in advance or can increase unboundedly.

It is straightforward to extend the incremental SGNS to the mini-batch setting by reading a subset of the training data (or mini-batch), rather than a single word, at a time to update the noise distribution and perform SGD (Algorithm 2). Although this paper primarily focuses on the incremental SGNS, the mini-batch algorithm is also important in practical terms because it is easier to be multi-threaded.

Alternatives to Algorithms 2 might be possible. Other possible approaches include computing the noise distribution separately on each subset of the training data, fixing the noise distribution after computing it from the first (possibly large) subset, and so on. We exclude such alternatives from our investigation because it is considered difficult to provide them with theoretical justification.

3.2 Efficient implementation

Although the incremental SGNS is conceptually simple, implementation issues are involved.

[1]In practice, Algorithm 1 buffers a sequence of words $w_{i-c}, \ldots, w_{i+c}$ (rather than a single word w_i) at each step, as it requires an access to the context words w_{i+j} in line 7. This is not a practical problem because the window size c is usually small and independent from the training data size n.

Algorithm 1 Incremental SGNS

```
1: $f(w) \leftarrow 0$ for all $w \in \mathcal{W}$
2: for $i = 1, \ldots, n$ do
3:     $f(w_i) \leftarrow f(w_i) + 1$
4:     $q(w) \leftarrow \frac{f(w)^\alpha}{\sum_{w' \in \mathcal{W}} f(w')^\alpha}$ for all $w \in \mathcal{W}$
5:     for $j = -c, \ldots, -1, 1, \ldots, c$ do
6:         draw $k$ negative samples from $q(w)$: $v_1, \ldots, v_k$
7:         use SGD to update $\mathbf{t}_{w_i}$, $\mathbf{c}_{w_{i+j}}$, and $\mathbf{c}_{v_1}, \ldots, \mathbf{c}_{v_k}$
8:     end for
9: end for
```

Algorithm 2 Mini-batch SGNS

```
1: for each subset $\mathcal{D}$ of the training data do
2:     update the noise distribution using $\mathcal{D}$
3:     perform SGD over $\mathcal{D}$
4: end for
```

3.2.1 Dynamic vocabulary

One problem that arises when training incremental SGNS is how to maintain the vocabulary set. Since new words emerge endlessly in the training data, the vocabulary set can grow unboundedly and exhaust a memory.

We address this problem by dynamically changing the vocabulary set. The Misra-Gries algorithm (Misra and Gries, 1982) is used to approximately keep track of top-m frequent words during training, and those words are used as the dynamic vocabulary set. This method allows the maximum vocabulary size to be explicitly limited to m, while being able to dynamically change the vocabulary set.

3.2.2 Adaptive unigram table

Another problem is how to generate negative samples efficiently. Since k negative samples per target-context pair have to be generated by the noise distribution, the sampling speed has a significant effect on the overall training efficiency.

Let us first examine how negative samples are generated in batch SGNS. In a popular implementation (Mikolov, 2013), a word array (referred to as a *unigram table*) is constructed such that the number of a word w in it is proportional to $q(w)$. See Table 1 for an example. Using the unigram table, negative samples can be efficiently generated by sampling the table elements uniformly at random. It takes only $O(1)$ time to generate one negative sample.

The above method assumes that the noise distribution is fixed and thus cannot be used directly for the incremental training. One simple solution is to reconstruct the unigram table whenever new training data is provided. However, such a method is not effective for the incremental SGNS, because the unigram table reconstruction requires $\mathcal{O}(|\mathcal{W}|)$ time.[2]

w	a	b	c
$q(w)$	0.5	0.3	0.2

$T = (a, a, a, a, a, b, b, b, c, c)$

Table 1: Example noise distribution $q(w)$ for the vocabulary set $\mathcal{W} = \{a, b, c\}$ (left) and the corresponding unigram table T of size 10 (right).

Algorithm 3 Adaptive unigram table.

```
1: $f(w) \leftarrow 0$ for all $w \in \mathcal{W}$
2: $z \leftarrow 0$
3: for $i = 1, \ldots, n$ do
4:     $f(w_i) \leftarrow f(w_i) + 1$
5:     $F \leftarrow f(w_i)^\alpha - (f(w_i) - 1)^\alpha$
6:     $z \leftarrow z + F$
7:     if $|T| < \tau$ then
8:         add $F$ copies of $w_i$ to $T$
9:     else
10:        for $j = 1, \ldots, \tau$ do
11:            $T[j] \leftarrow w_i$ with probability $\frac{F}{z}$
12:        end for
13:    end if
14: end for
```

We propose a reservoir-based algorithm for efficiently updating the unigram table (Vitter, 1985; Efraimidis, 2015) (Algorithm 3). The algorithm incrementally update the unigram table T while limiting its maximum size to τ. In case $|T| < \tau$, it can be easily confirmed that the number of a word w in T is $f(w)^\alpha (\propto q(w))$. In case $|T| = \tau$, since $z = \sum_{w \in \mathcal{W}} f(w)^\alpha$ is equal to the normalization factor of the noise distribution, it can be proven by induction that, for all j, $T[j]$ is a word w with probability $q(w)$. See (Vitter, 1985; Efraimidis, 2015) for reference.

Note on implementation In line 8, F copies of w_i are added to T. When F is not an integer, the copies are generated so that their expected number becomes F. Specifically, $\lceil F \rceil$ copies are added to T with probability $F - \lfloor F \rfloor$, and $\lfloor F \rfloor$ copies are added otherwise.

The loop from line 10 to 12 becomes expensive if implemented straightforwardly because the maximum table size τ is typically set large (*e.g.*, $\tau = 10^8$ in `word2vec` (Mikolov, 2013)). For acceleration, instead of checking all elements in the unigram table, randomly chosen $\frac{\tau F}{z}$ elements are substituted with w_i. Note that $\frac{\tau F}{z}$ is the expected

[2] This overhead is amortized in mini-batch SGNS if the mini-batch size is sufficiently large. Our discussion here is dedicated to efficiently perform the incremental training irrespective of the mini-batch size.

number of table elements to be substituted in the original algorithm. This approximation achieves great speed-up because we usually have $F \ll z$. In fact, it can be proven that it takes $O(1)$ time when $\alpha = 1.0$. See Appendix[3] A for more discussions.

3.3 Computational complexity

Both incremental and batch SGNS have the same space complexity, which is independent of the training data size n. Both require $\mathcal{O}(|\mathcal{W}|)$ space to store the word embeddings and the word frequency counts, and $\mathcal{O}(|T|)$ space to store the unigram table.

The two algorithms also have the same time complexity. Both require $\mathcal{O}(n)$ training time when the training data size is n. Although incremental SGNS requires extra time for updating the dynamic vocabulary and adaptive unigram table, these costs are practically negligible, as will be demonstrated in Section 5.3.

4 Theoretical Analysis

Although the extension from batch to incremental SGNS is simple and intuitive, it is not readily clear whether incremental SGNS can learn word embeddings as well as the batch counterpart. To answer this question, in this section we examine incremental SGNS from a theoretical point of view.

The analysis begins by examining the difference between the objectives optimized by batch and incremental SGNS (Section 4.1). Then, probabilistic properties of their difference are investigated to demonstrate the relationship between batch and incremental SGNS (Sections 4.2 and 4.3). We shortly touch the mini-batch SGNS at the end of this section (Section 4.4).

4.1 Objective difference

As discussed in Section 2, batch SGNS optimizes the following objective:

$$\mathcal{L}_{\mathrm{B}}(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{\substack{|j|\le c\\ j\ne 0}} \psi^{+}_{w_i,w_{i+j}} + k\mathbb{E}_{v\sim q_n(v)}[\psi^{-}_{w_i,v}],$$

where $\theta = (\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_{|\mathcal{W}|}, \mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{|\mathcal{W}|})$ collectively represents the model parameters[4] (*i.e.*, word embeddings) and $q_n(v)$ represents the noise distribution. Note that the noise distribution is represented in a different notation than Section 2 to make its dependence on the whole training data explicit. The function $q_i(v)$ is defined as $q_i(v) = \frac{f_i(v)^\alpha}{\sum_{v'\in\mathcal{W}} f_i(v')^\alpha}$, where $f_i(v)$ represents the word frequency in the first i words of the training data.

In contrast, incremental SGNS computes the gradient of $-\psi^{+}_{w_i,w_{i+j}} - k\mathbb{E}_{v\sim q_i(v)}[\psi^{-}_{w_i,v}]$ at each step to perform gradient descent. Note that the noise distribution does not depend on n but rather on i. Because it can be seen as a sample approximation of the gradient of

$$\mathcal{L}_{\mathrm{I}}(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{\substack{|j|\le c\\ j\ne 0}} \psi^{+}_{w_i,w_{i+j}} + k\mathbb{E}_{v\sim q_i(v)}[\psi^{-}_{w_i,v}],$$

incremental SGNS can be interpreted as optimizing $\mathcal{L}_{\mathrm{I}}(\theta)$ with SGD.

Since the expectation terms in the objectives can be rewritten as $\mathbb{E}_{v\sim q_i(v)}[\psi^{-}_{w_i,v}] = \sum_{v\in\mathcal{W}} q_i(v)\psi^{-}_{w_i,v}$, the difference between the two objectives can be given as

$$\begin{aligned}\Delta\mathcal{L}(\theta) &= \mathcal{L}_{\mathrm{B}}(\theta) - \mathcal{L}_{\mathrm{I}}(\theta)\\ &= \frac{1}{n}\sum_{i=1}^{n}\sum_{\substack{|j|\le c\\ j\ne 0}} k\sum_{v\in\mathcal{W}}(q_i(v)-q_n(v))\psi^{-}_{w_i,v}\\ &= \frac{2ck}{n}\sum_{i=1}^{n}\sum_{v\in\mathcal{W}}(q_i(v)-q_n(v))\psi^{-}_{w_i,v}\\ &= \frac{2ck}{n}\sum_{w,v\in\mathcal{W}}\sum_{i=1}^{n}\delta_{w_i,w}(q_i(v)-q_n(v))\psi^{-}_{w,v}\end{aligned}$$

where $\delta_{w,v} = \delta(w = v)$ is the delta function.

4.2 Unsmoothed case

Let us begin by examining the objective difference $\Delta\mathcal{L}(\theta)$ in the unsmoothed case, $\alpha = 1.0$.

The technical difficulty in analyzing $\Delta\mathcal{L}(\theta)$ is that it is dependent on the word order in the training data. To address this difficulty, we assume that the words in the training data are generated from some stationary distribution. This assumption allows us to investigate the property of $\Delta\mathcal{L}(\theta)$ from a probabilistic perspective. Regarding the validity of this assumption, we want to note that this assumption is already taken by the original SGNS: the probability that the target and context words co-occur is assumed to be independent of their position in the training data.

[3]The appendices are in the supplementary material.

[4]We treat words as integers and thus $\mathcal{W} = \{1, 2, \ldots |\mathcal{W}|\}$.

We below introduce some definitions and notations as the preparation of the analysis.

Definition 1. Let $\mathrm{X}_{i,w}$ be a random variable that represents $\delta_{w_i,w}$. It takes 1 when the i-th word in the training data is $w \in \mathcal{W}$ and 0 otherwise.

Remind that we assume that the words in the training data are generated from a stationary distribution. This assumption means that the expectation and (co)variance of $\mathrm{X}_{i,w}$ do not depend on the index i. Hereafter, they are respectively denoted as $\mathbb{E}[\mathrm{X}_{i,w}] = \mu_w$ and $\mathbb{V}[\mathrm{X}_{i,w}, \mathrm{X}_{j,v}] = \rho_{w,v}$.

Definition 2. Let $\mathrm{Y}_{i,w}$ be a random variable that represents $q_i(w)$ when $\alpha = 1.0$. It is given as $\mathrm{Y}_{i,w} = \frac{1}{i}\sum_{i'=1}^{i} \mathrm{X}_{i',w}$.

4.2.1 Convergence of the first and second order moments of $\Delta\mathcal{L}(\theta)$

It can be shown that the first order moment of $\Delta\mathcal{L}(\theta)$ has an analytical form.

Theorem 1. *The first order moment of $\Delta\mathcal{L}(\theta)$ is given as*

$$\mathbb{E}[\Delta\mathcal{L}(\theta)] = \frac{2ck(H_n - 1)}{n} \sum_{w,v\in\mathcal{W}} \rho_{w,v}\psi^{-}_{w,v},$$

where H_n is the n-th harmonic number.

Sketch of proof. Notice that $\mathbb{E}[\Delta\mathcal{L}(\theta)]$ can be written as

$$\frac{2ck}{n} \sum_{w,v\in\mathcal{W}} \sum_{i=1}^{n} \big(\mathbb{E}[\mathrm{X}_{i,w}\mathrm{Y}_{i,v}] - \mathbb{E}[\mathrm{X}_{i,w}\mathrm{Y}_{n,v}]\big)\psi^{-}_{w,v}.$$

Because we have, for any i and j such that $i \leq j$,

$$\mathbb{E}[\mathrm{X}_{i,w}\mathrm{Y}_{j,v}] = \sum_{j'=1}^{j} \mathbb{E}[\mathrm{X}_{i,w}\frac{\mathrm{X}_{j',v}}{j}] = \mu_w\mu_v + \frac{\rho_{w,v}}{j},$$

plugging this into $\mathbb{E}[\Delta\mathcal{L}(\theta)]$ proves the theorem. See Appendix B.1 for the complete proof. □

Theorem 1 readily gives the convergence property of the first order moment of $\Delta\mathcal{L}(\theta)$:

Theorem 2. *The first-order moment of $\Delta\mathcal{L}(\theta)$ decreases in the order of $\mathcal{O}(\frac{\log(n)}{n})$:*

$$\mathbb{E}[\Delta\mathcal{L}(\theta)] = \mathcal{O}\left(\frac{\log(n)}{n}\right),$$

and thus converges to zero in the limit of infinity:

$$\lim_{n\to\infty} \mathbb{E}[\Delta\mathcal{L}(\theta)] = 0.$$

Proof. We have $H_n = \mathcal{O}(\log(n))$ from the upper integral bound, and thus Theorem 1 gives the proof. □

A similar result to Theorem 2 can be obtained for the second order moment of $\Delta\mathcal{L}(\theta)$ as well.

Theorem 3. *The second-order moment of $\Delta\mathcal{L}(\theta)$ decreases in the order of $\mathcal{O}(\frac{\log(n)}{n})$:*

$$\mathbb{E}[\Delta\mathcal{L}(\theta)^2] = \mathcal{O}\left(\frac{\log(n)}{n}\right),$$

and thus converges to zero in the limit of infinity:

$$\lim_{n\to\infty} \mathbb{E}[\Delta\mathcal{L}(\theta)^2] = 0.$$

Proof. Omitted. See Appendix B.2. □

4.2.2 Main result

The above theorems reveal the relationship between the optimal solutions of the two objectives, as stated in the next lemma.

Lemma 4. *Let θ^* and $\hat{\theta}$ be the optimal solutions of $\mathcal{L}_B(\theta)$ and $\mathcal{L}_I(\theta)$, respectively: $\theta^* = \arg\min_\theta \mathcal{L}_B(\theta)$ and $\hat{\theta} = \arg\min_\theta \mathcal{L}_I(\theta)$. Then,*

$$\lim_{n\to\infty} \mathbb{E}[\mathcal{L}_B(\hat{\theta}) - \mathcal{L}_B(\theta^*)] = 0, \tag{2}$$

$$\lim_{n\to\infty} \mathbb{V}[\mathcal{L}_B(\hat{\theta}) - \mathcal{L}_B(\theta^*)] = 0. \tag{3}$$

Proof. The proof is made by the squeeze theorem. Let $l = \mathcal{L}_\mathrm{B}(\hat{\theta}) - \mathcal{L}_\mathrm{B}(\theta^*)$. The optimality of θ^* gives $0 \leq l$. Also, the optimality of $\hat{\theta}$ gives

$$\begin{aligned} l &= \mathcal{L}_\mathrm{B}(\hat{\theta}) - \mathcal{L}_\mathrm{I}(\theta^*) + \mathcal{L}_\mathrm{I}(\theta^*) - \mathcal{L}_\mathrm{B}(\theta^*) \\ &\leq \mathcal{L}_\mathrm{B}(\hat{\theta}) - \mathcal{L}_\mathrm{I}(\hat{\theta}) + \mathcal{L}_\mathrm{I}(\theta^*) - \mathcal{L}_\mathrm{B}(\theta^*) \\ &= \Delta\mathcal{L}(\hat{\theta}) - \Delta\mathcal{L}(\theta^*). \end{aligned}$$

We thus have $0 \leq \mathbb{E}[l] \leq \mathbb{E}[\Delta\mathcal{L}(\hat{\theta}) - \Delta\mathcal{L}(\theta^*)]$. Since Theorem 2 implies that the right hand side converges to zero when $n \to \infty$, the squeeze theorem gives Eq. (2). Next, we have

$$\begin{aligned} \mathbb{V}[l] = \mathbb{E}[l^2] - \mathbb{E}[l]^2 &\leq \mathbb{E}[l^2] \\ &\leq \mathbb{E}[(\Delta\mathcal{L}(\hat{\theta}) - \Delta\mathcal{L}(\theta^*))^2] \\ &\leq \mathbb{E}[(\Delta\mathcal{L}(\hat{\theta}) - \Delta\mathcal{L}(\theta^*))^2] \\ &\quad + \mathbb{E}[(\Delta\mathcal{L}(\hat{\theta}) + \Delta\mathcal{L}(\theta^*))^2] \\ &= 2\mathbb{E}[\Delta\mathcal{L}(\hat{\theta})^2] + 2\mathbb{E}[\Delta\mathcal{L}(\theta^*)^2]. \end{aligned} \tag{4}$$

Theorem 3 suggests that Eq. (4) converges to zero when $n \to \infty$. Also, the non-negativity of the variance gives $0 \leq \mathbb{V}[l]$. Therefore, the squeeze theorem gives Eq. (3). □

We are now ready to provide the main result of the analysis. The next theorem shows the convergence of $\mathcal{L}_{\mathrm{B}}(\hat{\theta})$.

Theorem 5. *$\mathcal{L}_B(\hat{\theta})$ converges in probability to $\mathcal{L}_B(\theta^*)$:*

$$\forall\epsilon > 0, \lim_{n\to\infty} \Pr\left[|\mathcal{L}_B(\hat{\theta}) - \mathcal{L}_B(\theta^*)| \geq \epsilon\right] = 0.$$

Sketch of proof. Let again $l = \mathcal{L}_{\mathrm{B}}(\hat{\theta}) - \mathcal{L}_{\mathrm{B}}(\theta^*)$. Chebyshev's inequality gives, for any $\epsilon_1 > 0$,

$$\lim_{n\to\infty} \frac{\mathbb{V}[l]}{\epsilon_1^2} \geq \lim_{n\to\infty} \Pr\left[|l - \mathbb{E}[l]| \geq \epsilon_1\right].$$

Remember that Eq. (2) means that for any $\epsilon_2 > 0$, there exists n' such that if $n' \leq n$ then $|\mathbb{E}[l]| < \epsilon_2$. Therefore, we have

$$\lim_{n\to\infty} \frac{\mathbb{V}[l]}{\epsilon_1^2} \geq \lim_{n\to\infty} \Pr\left[|l| \geq \epsilon_1 + \epsilon_2\right] \geq 0.$$

The arbitrary property of ϵ_1 and ϵ_2 allows $\epsilon_1 + \epsilon_2$ to be rewritten as ϵ. Also, Eq. (3) implies that $\lim_{n\to\infty} \frac{\mathbb{V}[l]}{\epsilon_1^2} = 0$. This completes the proof. See Appendix B.3 for the detailed proof. $\square$

Informally, this theorem can be interpreted as suggesting that the optimal solutions of batch and incremental SGNS agree when n is infinitely large.

4.3 Smoothed case

We next examine the smoothed case ($0 < \alpha < 1$). In this case, the noise distribution can be represented by using the ones in the unsmoothed case:

$$q_i(w) = \frac{f_i(w)^\alpha}{\sum_{w'\in\mathcal{W}} f_i(w')^\alpha} = \frac{\left(\frac{f_i(w)}{F_i}\right)^\alpha}{\sum_{w'\in\mathcal{W}}\left(\frac{f_i(w')}{F_i}\right)^\alpha}$$

where $F_i = \sum_{w'\in\mathcal{W}} f_i(w')$ and $\frac{f_i(w)}{F_i}$ corresponds to the unsmoothed noise distribution.

Definition 3. Let $\mathrm{Z}_{i,w}$ be a random variable that represents $q_i(w)$ in the smoothed case. Then, it can be written by using $\mathrm{Y}_{i,w}$:

$$\mathrm{Z}_{i,w} = g_w(\mathrm{Y}_{i,1}, \mathrm{Y}_{i,2}, \ldots, \mathrm{Y}_{i,|\mathcal{W}|})$$

where $g_w(x_1, x_2, \ldots, x_{|\mathcal{W}|}) = \frac{x_w^\alpha}{\sum_{w'\in\mathcal{W}} x_{w'}^\alpha}$.

Because $\mathrm{Z}_{i,w}$ is no longer a linear combination of $\mathrm{X}_{i,w}$, it becomes difficult to derive similar proofs to the unsmoothed case. To address this difficulty, $\mathrm{Z}_{i,w}$ is approximated by the first-order Taylor expansion around

$$\mathbb{E}[(\mathrm{Y}_{i,1}, \mathrm{Y}_{i,2}, \ldots, \mathrm{Y}_{i,|\mathcal{W}|})] = (\mu_1, \mu_2, \ldots, \mu_{|\mathcal{W}|}).$$

The first-order Taylor approximation gives

$$\mathrm{Z}_{i,w} \approx g_w(\mu) + \sum_{v\in\mathcal{W}} M_{w,v}(\mathrm{Y}_{i,v} - g_v(\mu))$$

where $\mu = (\mu_1, \mu_2, \ldots, \mu_{|\mathcal{W}|})$ and $M_{w,v} = \frac{\partial g_w(x)}{\partial x_v}|_{x=\mu}$. Consequently, it can be shown that the first and second order moments of $\Delta\mathcal{L}(\theta)$ have the order of $\mathcal{O}(\frac{\log(n)}{n})$ in the smoothed case as well. See Appendix C for the details.

4.4 Mini-batch SGNS

The same analysis result can also be obtained for the mini-batch SGNS. We can prove Theorems 2 and 3 in the mini-batch case as well (see Appendix D for the proof). The other part of the analysis remains the same.

5 Experiments

Three experiments were conducted to investigate the correctness of the theoretical analysis (Section 5.1) and the practical usefulness of incremental SGNS (Sections 5.2 and 5.3). Details of the experimental settings that do not fit into the paper are presented in Appendix E.

5.1 Validation of theorems

An empirical experiment was conducted to validate the result of the theoretical analysis. Since it is difficult to assess the main result in Section 4.2.2 directly, the theorems in Sections 4.2.1, from which the main result is readily derived, were investigated. Specifically, the first and second order moments of $\Delta\mathcal{L}(\theta)$ were computed on datasets of increasing sizes to empirically investigate the convergence property.

Datasets of various sizes were constructed from the English Gigaword corpus (Napoles et al., 2012). The datasets made up of n words were constructed by randomly sampling sentences from the Gigaword corpus. The value of n was varied over $\{10^3, 10^4, 10^5, 10^6, 10^7\}$. $10,000$ different datasets were created for each size n to compute the first and second order moments.

Figure 1 (top left) shows log-log plots of the first order moments of $\Delta\mathcal{L}(\theta)$ computed on the different sized datasets when $\alpha = 1.0$. The crosses

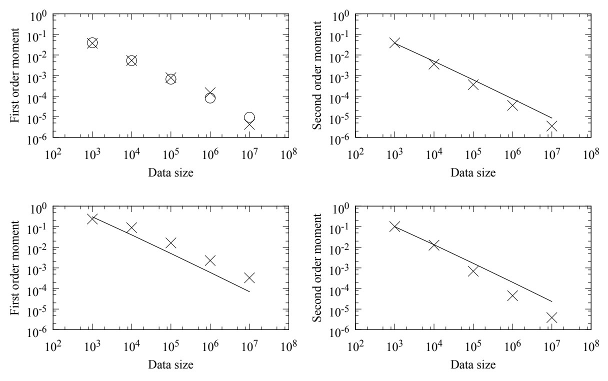

Figure 1: Log-log plots of the first and second order moments of $\Delta\mathcal{L}(\theta)$ on the different sized datasets when $\alpha = 1.0$ (top left and top right) and $\alpha = 0.75$ (bottom left and bottom right).

and circles represent the empirical values and theoretical values obtained by Theorem 1, respectively. Figure 1 (top right) similarly illustrates the second order moments of $\Delta\mathcal{L}(\theta)$. Since Theorem 3 suggests that the second order moment decreases in the order of $\mathcal{O}(\frac{\log(n)}{n})$, the graph $y \propto \frac{\log(x)}{x}$ is also shown. The graph was fitted to the empirical data by minimizing the squared error.

The top left figure demonstrates that the empirical values of the first order moments fit the theoretical result very well, providing a strong empirical evidence for the correctness of Theorem 1. In addition, the two figures show that the first and second order moments decrease almost in the order of $\mathcal{O}(\frac{\log(n)}{n})$, converging to zero as the data size increases. This result validates Theorems 2 and 3.

Figures 1 (bottom left) and (bottom right) show similar results when $\alpha = 0.75$. Since we do not have theoretical estimates of the first order moment when $\alpha \neq 1.0$, the graphs $y \propto \frac{\log(n)}{n}$ are shown in both figures. From these, we can again observe that the first and second order moments decrease almost in the order of $\mathcal{O}(\frac{\log(n)}{n})$. This indicates the validity of the investigation in Section 4.3. The relatively larger deviations from the graphs $y \propto \frac{\log(n)}{n}$, compared with the top right figure, are considered to be attributed to the first-order Taylor approximation.

5.2 Quality of word embeddings

The next experiment investigates the quality of the word embeddings learned by incremental SGNS through comparison with the batch counterparts.

The Gigaword corpus was used for the training. For the comparison, both our own implementation of batch SGNS as well as WORD2VEC (Mikolov et al., 2013c) were used (denoted as **batch** and **w2v**). The training configurations of the three methods were set the same as much as possible, although it is impossible to do so perfectly. For example, incremental SGNS (denoted as **incremental**) utilized the dynamic vocabulary (*c.f.*, Section 3.2.1) and thus we set the maximum vocabulary size m to control the vocabulary size. On the other hand, we set a frequency threshold to determine the vocabulary size of **w2v**. We set $m = 240$k for **incremental**, while setting the frequency threshold to 100 for **w2v**. This yields vocabulary sets of comparable sizes: $220,389$ and $246,134$.

The learned word embeddings were assessed on five benchmark datasets commonly used in the literature (Levy et al., 2015): WordSim353 (Agirre et al., 2009), MEN (Bruni et al., 2013), SimLex-999 (Hill et al., 2015), the MSR analogy dataset (Mikolov et al., 2013c), the Google analogy dataset (Mikolov et al., 2013a). The former three are for a semantic similarity task, and the remaining two are for a word analogy task. As evaluation measures, Spearman's ρ and prediction accuracy were used in the two tasks, respectively.

Figures 2 (a) and (b) represent the results on the similarity datasets and the analogy datasets. We see that the three methods (**incremental**, **batch**, and **w2v**) perform equally well on all of the datasets. This indicates that incremental SGNS can learn as good word embeddings as the batch counterparts, while being able to perform incremental model update. Although **incremental** performs slightly better than the batch methods in some datasets, the difference seems to be a product of chance.

The figures also show the results of incremental SGNS when the maximum vocabulary size m was reduced to 150k and 100k (**incremental-150k** and **incremental-100k**). The resulting vocabulary sizes were $135,447$ and $86,993$, respectively. We see that **incremental-150k** and **incremental-100k** perform comparatively well with **incremental**, although relatively large performance drops are observed in some datasets (MEN and MSR). This demonstrates that the Misra-Gries algorithm can effectively control the vocabulary size.

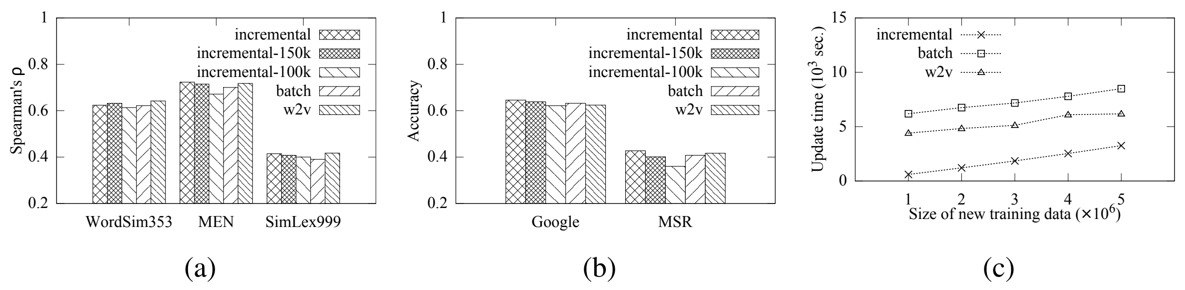

Figure 2: (a): Spearman's ρ on the word similarity datasets. (b): Accuracy on the analogy datasets. (c): Update time when new training data is provided.

5.3 Update time

The last experiment investigates how much time incremental SGNS can save by avoiding re-training when updating the word embeddings.

In this experiment, **incremental** was first trained on the initial training data of size[5] n_1 and then updated on the new training data of size n_2 to measure the update time. For comparison, **batch** and **w2v** were re-trained on the combination of the initial and new training data. We fixed $n_1 = 10^7$ and varied n_2 over $\{1\times10^6, 2\times10^6, \ldots, 5\times10^6\}$. The experiment was conducted on Intel® Xeon® 2GHz CPU. The update time was averaged over five trials.

Figure 2 (c) compares the update time of the three methods across various values of n_2. We see that **incremental** significantly reduces the update time. It achieves 10 and 7.3 times speed-up compared with **batch** and **w2v** (when $n_2 = 10^6$). This represents the advantage of the incremental algorithm, as well as the time efficiency of the dynamic vocabulary and adaptive unigram table. We note that **batch** is slower than **w2v** because it uses AdaGrad, which maintains different learning rates for different dimensions of the parameter, while **w2v** uses the same learning rate for all dimensions.

6 Related Work

Word representations based on distributional semantics have been common (Turney and Pantel, 2010; Baroni and Lenci, 2010). The distributional methods typically begin by constructing a word-context matrix and then applying dimension reduction techniques such as SVD to obtain high-quality word meaning representations. Although some studies investigated incremental updating of the word-context matrix (Yin et al., 2015; Goyal and Daume III, 2011), they did not explore the reduced representations. On the other hand, neural word embeddings have recently gained much popularity as an alternative. However, most previous studies have not explored incremental strategies (Mikolov et al., 2013a,b; Pennington et al., 2014).

Peng et al. (2017) proposed an incremental learning method of hierarchical soft-max. Because hierarchical soft-max and negative sampling have different advantages (Peng et al., 2017), the incremental SGNS and their method are complementary to each other. Also, their updating method needs to scan not only new but also old training data, and thus is not an incremental algorithm in a strict sense. As a consequence, it potentially incurs the same time complexity as the re-training. Another consequence is that their method has to retain the old training data and thus wastes space, while incremental SGNS can discard old training examples after processing them.

Very recently, May et al. (2017) also proposed an incremental algorithm for SGNS. However, their work differs from ours in that their algorithm is not designed to use smoothed noise distribution (*i.e.*, the smoothing parameter α is assumed fixed as $\alpha = 1.0$ in their method), which is a key to learn high-quality word embeddings. Another difference is that they did not provide theoretical justification for their algorithm.

There are publicly available implementations for training SGNS, one of the most popular being WORD2VEC (Mikolov, 2013). However, it does not support an incremental training method. GENSIM (Řehůřek and Sojka, 2010) also offers SGNS training. Although GENSIM allows the incremental updating of SGNS models, it is done in an ad-hoc manner. In GENSIM, the vocabulary set as well as the unigram table are fixed once trained, meaning that new words cannot be added. Also,

[5]The number of sentences here.

they do not provide any theoretical accounts for the validity of their training method. Finally, we want to note that most of the existing implementations can be easily extended to support the incremental (or mini-batch) SGNS by simply keep updating the noise distribution.

7 Conclusion and Future Work

This paper proposed incremental SGNS and provided thorough theoretical analysis to demonstrate its validity. We also conducted experiments to empirically demonstrate its effectiveness. Although the incremental model update is often required in practical machine learning applications, only a little attention has been paid to learning word embeddings incrementally. We consider that incremental SGNS successfully addresses this situation and serves as an useful tool for practitioners.

The success of this work suggests several research directions to be explored in the future. One possibility is to explore extending other embedding methods such as GloVe (Pennington et al., 2014) to incremental algorithms. Such studies would further extend the potential of word embedding methods.

References

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasca, and Aitor Soroa. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of NAACL*, pages 19–27.

Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computatoinal Linguistics*, 36:673–721.

E. Bruni, N. K. Tran, and M. Baroni. 2013. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–49.

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

Pavlos S. Efraimidis. 2015. Weighted random sampling over data streams. ArXiv:1012.0256.

Amit Goyal and Hal Daume III. 2011. Approximate scalable bounded space sketch for large data nlp. In *Proceedings of EMNLP*, pages 250–261.

Felix Hill, Roi Reichart, and Anna Korhonen. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41:665–695.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225.

Chandler May, Kevin Duh, Benjamin Van Durme, and Ashwin Lall. 2017. Streaming word embeddings with the space-saving algorithm. ArXiv:1704.07463.

Tomas Mikolov. 2013. word2vec. https://code.google.com/archive/p/word2vec.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *Workshop at ICLR*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in NIPS*, pages 3111–3119.

Tomas Mikolov, Wen-Tau Yih, and Geoffrey Zweig. 2013c. Linguistic regularities in continuous space word representations. In *Proceedings of NAACL*, pages 746–751.

Jayadev Misra and David Gries. 1982. Finding repeated elements. *Science of Computer Programming*, 2(2):143–152.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated english gigaword ldc2012t21.

Hao Peng, Jianxin Li, Yangqiu Song, and Yaopeng Liu. 2017. Incrementally learning the hierarchical softmax function for neural language models. In *Proceedings of AAAI*, pages 3267–3273.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543.

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50.

Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Jeffrey S. Vitter. 1985. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11:37–57.

Wenpeng Yin, Tobias Schnabel, and Hinrich Schütze. 2015. Online updating of word representations for part-of-speech tagging. In *Proceedings of EMNLP*, pages 1329–1334.