Provable word embedding for the skip-gram word2vec model

Anonymous

Abstract

To be written

(In magenta color is our comments to drive the writing; in plain black color will be our text)

Introduction

What is the problem we focus on? Word-embedding, why it is important, where it is used, some references from public science & media showing its significance.

What is the current state of the art & what are the shortcomings? Here, we should describe how people solve this problem, what are the tools they use (SGD, Riemannian, etc.) and what are the problems with these methods. We shouldn't spend that much space on related work as there will be a Related Work section next.

What is our perspective? Computational. We need to stress out that we do not focus on finding a better linguistic metric, but given word2vec model interpretation as matrix factorization, we identify the problems of using classical methods that involve huge matrix manipulations, and we propose alternatives. At the same time, we are interested in proposing theory that justifies partially what we observe in practice. E.g., here we should say that current approaches are expensive computationally, as well as non-convex with no theory.

Word embedding represents one of the most successful applications of unsupervised learning. It has shown geralization power in varities of NLP tasks, part-of-speech tagging (Abka 2016), document level metric learning(Kusner et al. 2015), machine translation (Mikolov et al. 2013)(Zou et al. 2013) etc.

In this paper, we train Skip-Gram model with negative sampling (SGND) under the framework of Bi-factorized gradient descent (BFGD). We show that with BFGD, which does not require singular value decomposition at each iteration, we can obtain similar liguistic performance compared to Splitting Projection Algorithm. And BFGD is able to train the model efficiently as the size of a corpus goes up, since it does not require singular value decomposition at each iteration. As we will see, with update rule in word2vec a special version of stochastic gradient descent under BFGD framework, this discussion opens a critical topic on how to train SGNS both efficiently and effectively — how to design a good stochastic/batched version of BFGD and train Skip-Gram model without loss of performance. What are our contributions? We need to write them down in bullets; this will be written after we have all the rest.

Background & Related Work

Set up the problem: notation + mathematical description What are the works before us: a more detailed description. What did they contribute, how did they evolve the field?

What questions are still open?

The Skip-Gram model is introduced in (Mikolov et al. 2013), which assumes a corpus of words w_1, w_2 ... w_n and correponding contexts. For every individual word w_i, a context c is defined as a word within a $L-$sided window surrounding it, i.e. $w_{i-L}, \cdots, w_{i-1}, w_{i+1}, \cdots, w_L$. Following notations in (Levy and Goldberg 2014), we denote $\#(w, c)$ as number of word-context pairs and $\#(w)$ are abbreviation for

$$\#(w) = \sum_{c'} \#(w, c'), \quad \#(c) = \sum_{w'} \#(w', c) \tag{1}$$

It is also convenient to define the set of observed word-context pairs as D, and $|D| = \sum_{w,c} \#(w, c)$

The negative sampling takes a word-context pair and samples k negative pairs (w, c_N) aligning with it, where k is a hyperparameter and equals 5 in experiments. In every negative pairs, the context is generated from the context distribution from the corpus, which is:

$$c_N \sim P_D(c) = \frac{\#(c)}{|D|}^1 \tag{2}$$

Therefore for every word-context pair in vocabulary, the SGNS objective is:

$$l_{SGNS}(w, c) = \#(w, c) \log[\sigma(w^T c)] + k \cdot \mathbb{E}_{c_N \sim P_D} \log[\sigma(-w^T c_N)] \tag{3}$$

[1] In word2vec implementation, $c_N \sim P_D(c) = \frac{\#(c)^{3/4}}{|D|}$, but in mathematics view SGNS is still doing factorization

When the online learning goes through all word-context pairs in corpus, SGNS model learns distributed word representation by maximizing the following objective function:

$$\max_{w,c} \sum_{w,c} \#(w,c) \log \sigma(w^T c) + k \cdot \mathbb{E}_{c_N \sim P_D}[\log \sigma(-w^T c_N)] \tag{4}$$

Matrix Factorization

Project-Splitting algorithm on SGNS

On (Fonarev et al. 2017) illustrates a general two-step scheme for training SGNS word embedding model and suggested a search of a solution in the low-rank form via Riemannian optimization framework.

Linguistic scores

Our approach

Description of the algorithm, details and discussion on initialization + step size, maybe already here have some plots to show how these behave (without giving away comparison results, just showing what is their trend in

In this paper, we follow literature dicussions characterizing SGNS as a matrix factorization problem (Levy and Goldberg 2014)(Levy, Goldberg, and Dagan 2015). The expectation term $\mathbb{E}_{c_N \sim P_D}[\log \sigma(-w^T c_N)]$ can be explicitly expressed as:

$$\mathbb{E}_{c_N \sim P_D}[\log \sigma(-w^T c_N)] = \sum_{c_N} \frac{\#(c_N)}{|D|} \log \sigma(-w^T c_N) \tag{5}$$

And one can show that SGNS objective 4 is factorizing the following matrix (Levy and Goldberg 2014):

$$X = W^T C = PMI(w_i, c_j) - \log k \tag{6}$$

where columns in W and C are w_i and c_i respectively. And in our setting, the vocabulary size is V , and the hidden layer size is d, so both W and C are $d \times V$.

Different from projector-splitting scheme(Fonarev et al. 2017) which shows advantages in optimizing a low-rank X on SGNS model, we observed that the matrix form SGNS objective is both smooth and convex in X. And with the explicit expression of SGNS objective $L_{SGNS}(X)$:

$$L_{SGNS}(X) = \sum_{w,c} \{\#(w,c) \log \sigma(w^T c) + k \cdot \sum_{c_N} \frac{\#(c_N)}{|D|} [\log \sigma(-w^T c_N)\} \tag{7}$$

$L_{SGNS}(X)$ is $L-$smooth with lipschitz constant

$$L = \frac{1}{4} \| \{\#(w,c) + k \frac{\#(w)\#(c)}{|D|}\}_{w,c} \|_F$$

Then we can borrow ideas from Bi-factorized gradient descent (Park et al. 2016)

Here, we should have a figure with the algorithm's steps etc.

Experimental results

We will move a bit unconventionally and show first some experimental results: this is what we are currently working on

Algorithm 1 BFGD on Skip-Gram Model with Negative Sampling

procedure BFGD(W_0, C_0, η, K) ▷ W_0, C_0 are initial encoding and decoding matrices, η is the step size and K is the total number of iterations
  $W \leftarrow W_0$
  $C \leftarrow C_0$
  for $i \leftarrow 1, \cdots, K$ **do**
    Calculate gradient of loss $\nabla L(W^T C)$
    $W \leftarrow W + \eta \cdot C \nabla L(W^T C)$
    $C \leftarrow C + \eta \cdot W \nabla L(W^T C)^T$
  end for
  return W, C
end procedure

dim		sem	syn	wordsim	men	simlex	murk
100	SGD						
	PS						
	BFGD						
300	SGD						
	PS						
	BFGD						
300	SGD						
	PS						
	BFGD						

Table 1: Comparison of different methods on liguistic scores, on different dimensions

Experimental Settings In experiments, we trained skip-gram negative sampling with Bi-Factorized gradient descent on two corpora: "enwik9" corpus (Mahoney 2011) and New York Times corpus (NYT) (Sandhaus 2008). The "enwik9" contains the first billions bytes of the Wikipedia dump on Mar. 3, 2006. We ignore words that appear less than 100 times in this dump and train a model with vocabulary size 37360. The New York Times Annotated Corpus contains over a million articles tagged with metadata. These articles are published between 1987 and 2007. We pick articles from 2000 to 2005 for training. We preprocessed the data with Stanford CoreNLP tookenizer (Manning et al. 2014). It has 13,517,603 sentences and we use a dictionary of the 40,000 most frequent words from this subcorpus.

In order to reduce training noise from frequent words: we do subsampling and ignore a word w in a sentence with a probability

$$P(f(w)) = 1 - (\sqrt{\frac{f(w)}{t}} + 1) \cdot \frac{t}{f(w)} \tag{8}$$

where where t, $f(w)$ are subsampling threshold and the frequency of the word respectively. We have to point out equation 8 is used in word2vec[2] and is an adapted version of subsampling in (Mikolov et al. 2013).

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

dataset		sem	syn	wordsim	men	simlex	mum
NYT	SGD						
	PS						
	BFGD						
enwik	SGD						
	PS						
	BFGD						

Table 2: Comparison of different methods on liguistic scores, on different dataset, we kept dimension $d = 300$

Theoretical guarantees

This is where we will try to focus on after we have fixed the experiments.

- What are the properties of the objective?
- What is known out there w.r.t. theory? What can we reuse for our algorithm?
- Initialization: what can we say about it? Any theory?
- Are there local minima for the non-convex problem we have?
- what about the stochastic version? SVRG version? Are there prior results on this?
- Can we use momentum/acceleration? Can we gain theoretically?

Conclusions

In discussion, we should claim that our purpose is to design a distributed version of the non-convex algorithm that can scale up and out.

References

Abka, A. F. 2016. Evaluating the use of word embeddings for part-of-speech tagging in bahasa indonesia. In *Computer, Control, Informatics and its Applications (IC3INA), 2016 International Conference on*, 209–214. IEEE.

Fonarev, A.; Grinchuk, O.; Gusev, G.; Serdyukov, P.; and Oseledets, I. 2017. Riemannian optimization for skip-gram negative sampling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 2028–2036.

Kusner, M.; Sun, Y.; Kolkin, N.; and Weinberger, K. 2015. From word embeddings to document distances. In *International Conference on Machine Learning*, 957–966.

Levy, O., and Goldberg, Y. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, 2177–2185.

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

Mahoney, M. 2011. Large text compression benchmark. *URL: http://www. mattmahoney. net/text/text. html*.

Manning, C.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S.; and McClosky, D. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 55–60.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In Burges, C. J. C.; Bottou, L.; Welling, M.; Ghahramani, Z.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 26*. Curran Associates, Inc. 3111–3119.

Park, D.; Kyrillidis, A.; Caramanis, C.; and Sanghavi, S. 2016. Finding low-rank solutions to matrix problems, efficiently and provably. *arXiv preprint arXiv:1606.03168*.

Sandhaus, E. 2008. The new york times annotated corpus, linguistic data consortium. *Philadelphia* 6(12):e26–752.

Zou, W. Y.; Socher, R.; Cer, D.; and Manning, C. D. 2013. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1393–1398.