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.

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. $w_{iL}, \cdots, w_{i1}, w_{i+1}, \cdots, w_L$. 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.

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:

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

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

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

Summing up over all word-context pairs SGNS model learns distributed word representation by maximizing the following objective function:

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

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). in our setting, the vocabulary size is V , and the hidden layer size is d.

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

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

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.

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.