Word2Vec and Sentiment Analysis

王嬰(Student ID:18210980068)

Abstract—In this project, we were asked to use word2vec models for sentiment analysis. Firstly, train word2vec to learn high-quality distributed vector representations with SST dataset and the cost at convergence is 11.19 at dim=20, C=9. Furthermore, Use the average of all the word vectors in each sentence as its feature, train a softmax regression classifier with SGD. Finally, the accuracy are 29.96%, 30.43%, 28.14% in train, dev and test set.

I. The skip-gram model theory

The training objective of the skip-gram model is to find word representations that are useful for predicting the surrounding words in a sentence or a document. More formally, given a sequence of training words $w_1, w_2, w_3, \dots, w_T$, the objective of the skip-gram model is to maximize the average log probability

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\neq 0} logp(w_{t+j}|w_t)$$

Where c is the size of the training context (which can be a function of the center word w_t). Larger c results in more training examples and thus can lead to a higher accuracy, at the expense of the training time. The basic skip-gram formulation defines $\mathrm{p}(w_{t+j}|w_t)$ using the softmax function:

$$\mathrm{p}(w_O|w_I) = \frac{\exp({v'_{w_O}}^T v_{w_I})}{\sum_{w=1}^{W}\exp({v'_w}^T v_{w_I})}$$

where v_w ad v'_w are the 'input' and 'output' vector representation of w, and W is the number of words in the vocabulary. This formulation is impractical because the cost of computing $\nabla\mathrm{logp}(w_O|w_I)$ is proportional to W, which is often large($10^5 - 10^7\ terms$).

II. Experiment

A : Word2Vec

Implement the word2vec models and train my own word vectors with stochastic gradient descent (SGD) and finally visualize word vectors.

(1) Write a helper function to **normalize rows** of a matrix.

For each rows of the matrix, calculate the L2 norm for every sample. Each units divide by the L2 norm.Use package sklearn, I can directly normalize each rows of a matrix by Sklearn.preprocessing.normalize(x, norm='l2')

(2) Fill in the implementation for the **softmax** cost and gradient functions.

Softmax-CE loss function:

$$J_{softmax-CE}(\mathrm{o}, v_c, \mathrm{U}) = CE(y, \hat{y}) = -\log \mathrm{p}(\mathrm{o}|\mathrm{c})$$

the gradients with respect to v_c:

$$\frac{\partial J_{softmax-CE}}{\partial v_c} = -u_o + \sum_{t=1}^{V} p(t|c)u_t$$

the gradients with respect to u_w:

$$\frac{\partial J_{softmax-CE}}{\partial u_w} = \begin{cases} \frac{\exp(u_w^T v_c)\, v_c}{\sum_{t=1}^{V} \exp(u_t^T v_c)} & (w \neq o) \\ -v_c + \frac{\exp(u_o^T v_c)\, v_c}{\sum_{t=1}^{V} \exp(u_t^T v_c)} & (w = o) \end{cases}$$

(3) Fill in the implementation for the **negative sampling** cost and gradient functions.
Negative sampling loss function:

$$J_{neg-sample}(\mathrm{o}, v_c, \mathrm{U}) = -\log\left(\sigma(u_o^T v_c)\right) - \sum_{k=1}^{K} \log(\sigma(-u_k^T v_c))$$

the gradients with respect to v_c:

$$\frac{\partial J_{neg-sample}}{\partial v_c} = (\sigma(u_o^T v_c) - 1)u_o + \sum_{k=1}^{K} (1 - \sigma(-u_k^T v_c))u_k$$

the gradients with respect to u_w:

$$\frac{\partial J_{neg-sample}}{\partial u_k} = \begin{cases} \left(1 - \sigma(-u_k^T v_c)\right) v_c & (k \neq o) \\ (\sigma(u_o^T v_c) - 1)v_c + \left(1 - \sigma(-u_k^T v_c)\right) v_c & (k = o) \end{cases}$$

(4) Fill in the implementation of the cost and gradient functions for **the skipgram model**.
Skip gram cost:

$$J_{skip-gram}(word_{c-m\ldots c+m}) = \sum_{-m \leq j \leq m, j \neq 0} F(w_{c+j}, v_c)$$

The gradients with respect to v_c:

$$\frac{\partial J_{skip-gram}}{\partial v_c} = \sum\nolimits_{-m \leq j \leq m, j \neq 0} \frac{\partial F(w_{c+j}, v_c)}{\partial v_c}$$

The gradients with respect to u_w:

$$\frac{\partial J_{neg-sampleskip-gram}}{\partial u_{w_c+j}} = \sum\nolimits_{-m \leq j \leq m, j \neq 0} \frac{\partial F(w_{c+j}, v_c)}{\partial u_{w_c+j}}$$

B: Sentiment Analysis
For each sentence in the Stanford Sentiment Treebank dataset, I am going to use the average of all the word vectors in that sentence as its feature, train a softmax regression classifier, and try to predict the sentiment level of the said sentence.

III. Result
(1) Fixed dimVectors, compare different Context size(C).

Table 1 The accuracy in dimVectors = 20

C	Best regularization	Train accuracy (%)	Dev accuracy (%)	Test accuracy (%)
5	1.0E-06	29.763577	28.156222	27.737557
7	1.0E-05	29.646536	29.246140	28.054299
9	1.0E-08	**29.962547**	**30.426885**	**28.144796**

Table2 The accuracy in dimVectors = 10

C	Best regularization	Train accuracy (%)	Dev accuracy (%)	Test accuracy (%)
5	1.0E-06	29.143258	28.792007	26.923077

7	1.0E-07	**29.459270**	29.064487	**28.099548**
9	1.0E-06	29.435861	**30.245232**	27.511312

Table3 The accuracy in dimVectors = 5

C	Best regularization	Train accuracy (%)	Dev accuracy (%)	Test accuracy (%)
5	1.0E-06	**28.148408**	**28.610354**	25.384615
7	1.0E-04	27.773876	26.975477	24.705882
9	1.0E-06	27.832397	27.247956	**25.791855**

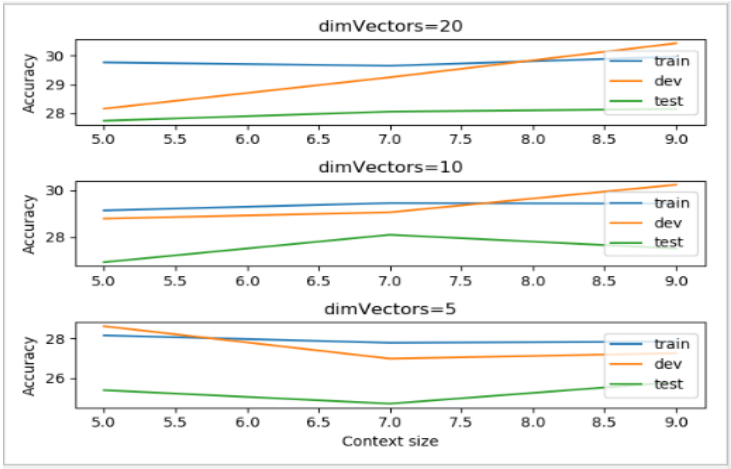

Fig1. Compare different Context size with Fixed dimVectors

For dimVectors=20, with the context size increases, the accuracy increases. However, for dimVectors=5, model which trained with the smallest context size 5 reach at the highest accuracy in the train and dev set. Therefore, it is not definitely right that large context size lead to high accuracy. Maybe there is a overfitting problem when too large context size with fixed dimVectors.

(2) Fixed Context size, compare different dimVectors.

Table4 The accuracy in Context size = 9

dimVedctors	Train accuracy (%)	Dev accuracy (%)	Test accuracy (%)
5	27.832397	27.247956	25.791855
10	29.435861	30.245232	27.511312
20	**29.962547**	**30.426885**	**28.144796**

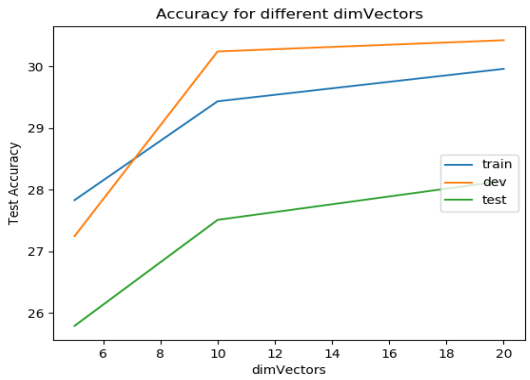

Fig2. different dimVectors with Fixed Context size =9

The accuracy increases with high dimVecotors when fixed the context size. It can reach at 28.14% on dev set with dimVectors=20, Context_size=9.

(3) Fixed dimVectors and Context size compare different regularization

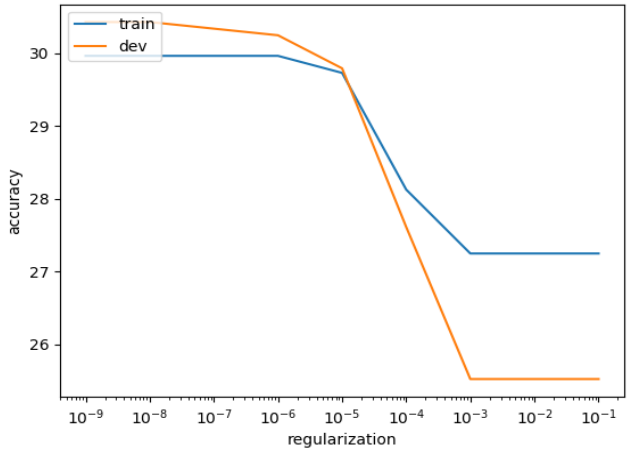

Fig3. Fixed dimVectors and Context size compare different regularization

Regularization is to deal with overfitting problem. We can see higher regularization may lead to lower accuracy in train and dev set from fig3.

(4) Visualization for word vectors

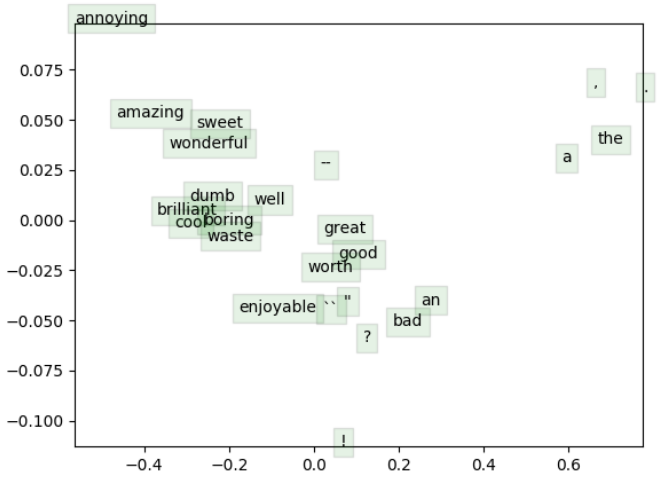

Fig4. Visualization for word vectors in dimVectors=20 Context size = 9

Words such as 'wonderful', 'sweet', 'amazing' represents positive attitude were clustered according to their word representations. Moreover, the trained skip-gram model can tell apart the positive words 'great', 'good', 'worth' and the negative words 'boring', 'waste'.

In conclusion, the highest accuracy are 29.96%, 30.43%, 28.14% in train, dev and test set.

VI.Reference

1. Numpy broadcasting documentation
2. Distributed Representations of Words and Phrases and their Compositionality.