Self-supervised learning: non-generative representation learning

Lesson No. 07

José Dorivaldo Nascimento Souza Júnior - RA170862

1 Introduction

In this document, initially it is briefly introduced what is a **self-supervised method**, and its main characteristics are highlighted. Afterwards, some relevant works in this field are cited. Finally, it is discussed in more detail one of the most relevant self-supervised works, that is the *word2vec*[5].

2 Self-Supervised Learning

2.1 What is Self-Supervised Learning

One of the fundamental definitions in the machine learning area is the difference between *supervised* and *unsupervised learning*. Supervised learning is expected to have a better performance (since there are labels for the data), but since the majority of data available are unlabeled, and usually require an astonishing effort to label, unsupervised learning is also a paramount field.

Given that situation above, *self-supervised learning* is a way to cleverly have the best world of both methods in some tasks. Given an item x from an unlabeled dataset, the general ideia is that x is going to pass through some transformations before being inputted into the model, and then what the model should predict from the transformed input x' is the actual x. Since we know for each x' the corresponding x, this method guides us to a *supervised loss function for a unlabeled dataset*[9].

It is important to not mix the concepts of *self-supervised learning* with *semi-supervised learning*, since the second concept is about having small part of the dataset labelled, and a bigger one unlabelled[2].

Furthermore, it is possible to point that Generative Models might be considered as self-supervised learning. However, the field of methods analyzed herein have a different goal from generative models. Instead of generating new data, it aims to provide a correct output for a task[9].

2.2 Some Relevant Works using Self-Supervised Learning

A wide range of task were performed by different works using self-supervised learning. Some of them are cited in the next sub-sections.

2.2.1 Images

In the context of images, some of the works are:

- **Reconstruct from a corrupted version**[7]: By appliyng a noise in the input image, it is possible to predict a corrupted image.
- **Relative Position**[3]: Given two parts extracted from an image of a dataset, it is possible to predict the position of these views in the original data.
- **Rotation**[4]: Given an initial image, it is possible to predict its real orientation, by rotating the input image.

2.2.2 Videos

In the context of videos, some of the works are:

- **Frame Sequence**[6]: Predict future frames of a video.
- **Video Colorization**[8]: Given a grayscale video, coherently colorize the entire video after manually colorize one frame.

3 Word2Vec

3.1 Predicting Neighbouring Context

One of the possibilities for self-supervised learning is prediction of neighbouring context. Given an item from the dataset, you should *mask part of the input and then predict this masked part in the output*[9]. Using this concept, word2vec[5] used a dataset of documents (texts) to generate meaningful feature vectors for a entire vocabulary. The main idea of this method is explained in the next subsections.

3.2 One-hot Vector

Before diving into word2vec, it is a good idea to glance the one-hot vector method. Given a vocabulary V, one of the most simple ways to represent a word from this vocabulary is by using an vector $v \in \mathbb{R}^{|V|}$, where:

1. There is one, and only one c, where $v_c = 1$
2. For every $i \neq c$, $v_i = 0$
3. Given two different words $v^{(a)}$ and $v^{(b)}$, they do not have the same c.

One of the main problems is that the features vectors learned in this method provides no information about the context[1], since the word vectors are orthogonal from each other in the subspace $\mathbb{R}^{|V|}$.

3.3 Word2Vec - Method Description

Given the one-hot vector problem cited beforehand, Word2Vec[5] aims to generate word feature vectors that represent context among the words in the vocabulary. Given a set of documents, the words are represented by its *context*. In this method, context is defined by the words which surrounds the target word in a document inside a window of size *m*.

For instance, given the statement *My siblings and I were sitting at the dinner table*, if we define a window m of size 3, the context for the word *sitting* (henceforth called *center word*) will be *(and, I, were, at, the, dinner)*. By this definition of context, Word2Vec developed two different models.

3.3.1 Continuous Bag of Words Model

In CBOW model, the goal is to *predict the center word given its context.* In summary, the one-hot vectors of each context word are inputted, aiming to output the center word's one hot vector.

In more detail, given the set of documents, the method iteratively passes through each document, and for each word in this document, it applies the following pipeline.

1. Get the one-hot vectors corresponding to the center word and its context.
2. Input every word from the context (at the same time) in a fully-connected neural network with one hidden layer of a arbitrary size *n*, an input layer of size *2m*|*V*|, and a output layer of size |*V*|.
3. Apply softmax in the output (So given the output $\mathbf{y}$, the value y_i will represent the probability of the word i being the center word).
    - Word i is the word that $c = i$ in its one-hot vector.

4. Use Cross Entropy as loss function.

Concerning the loss function, it is important to note that, given the original definition of Cross Entropy would be:

$$H(\mathbf{x},\mathbf{y}) = -\sum_{j=1}^{|V|} x_j \ln(y_j) \tag{1}$$

Where **x** is the one hot vector for the center word, and **y** is the output after the softmax. However, we know by definition that **y** is equal to 1 at the position c, and equal to 0 elsewhere. Therefore, every item from $\sum$ will be 0, except when $j = c$. As a consequence, the cross entropy function is defined as:

$$H(\mathbf{x},\mathbf{y}) = -x_c \ln(y_c) \tag{2}$$

Since $|V|$ is expected to be a very large number (since is the size of the entire vocabulary), this considerably speeds up the training.

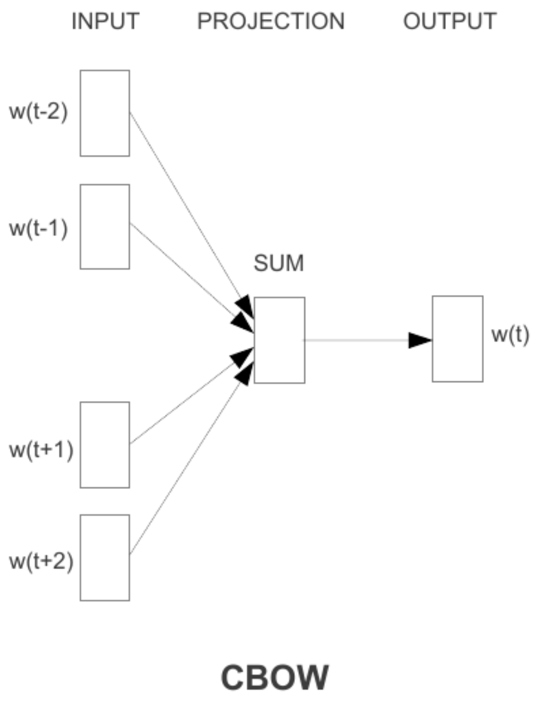


Figure 1. CBOW architecture[5]

3.3.2 Skip-Gram Model

On the other hand, the Skip-Gram model tries to predict the surrounding context given the center word. Basically, it follows the same structure from the CBOW, but it invertes the input and output, and hence the cross-entropy function changes. The pipeline for the Skip-Gram model is:

1. Get the one-hot vectors corresponding to the center word and its context.
2. Input the center word in a fully-connected neural network with one hidden layer of a arbitrary size n, an input layer of size $|V|$, and a output layer of size $2m|V|$ (representing $2m$ word in the context). Since the output represents each word in the context, it will henceforth be referred as $\mathbf{y} \in \mathbb{R}^{2m\times|V|}$.
3. Apply Softmax in each $y^{(i)}$, $1 < i < 2m$.
4. Use Cross-Entropy as loss function.

Related to the cross-entropy function, given the definition from CBOW in equation **2**, we can easily extend it to the Skip-Gram model, given that the difference now is that the method predicts the $2m$ vectors from the context, instead of the only vector from the center. Hence, the Cross-Entropy function is:

$$H(\mathbf{x},\mathbf{y}) = \sum_{j=0}^{2m} H(\mathbf{x}^{(\mathbf{j})},\mathbf{y}^{(\mathbf{j})}) = -\sum_{j=0, j\neq m}^{2m} x_c^{(j)} \ln(y_c^{(j)}) \tag{3}$$

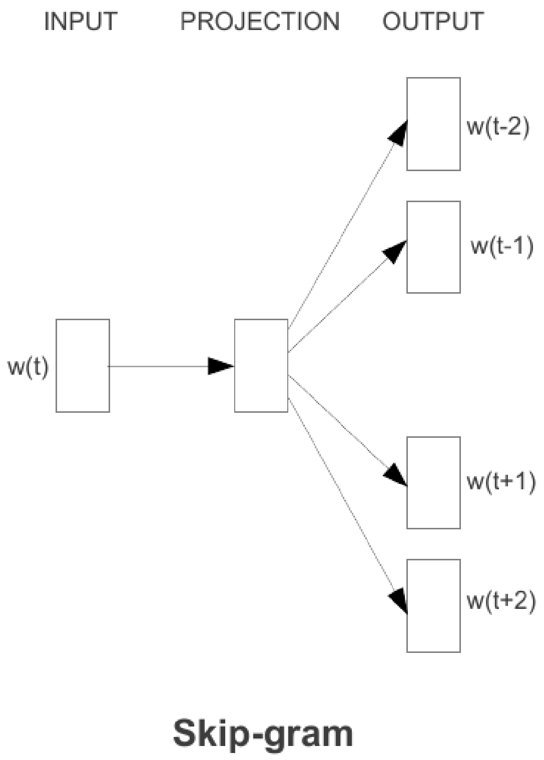


Figure 2. Skip-Gram architecture[5]

3.3.3 Context Word Vectors

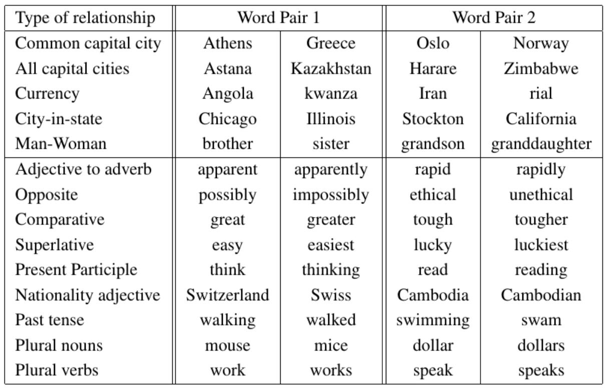
As a result of this procedure, meaningful feature vectors are generated for downstream tasks. A relevant characteristic of these vectors that is noted by the authors is the possibility of answering question by algebric functions. For instance, given $X = vector("biggest") - vector("big") + vector("small")$, the autors reports that the closest feature vector from X is $vector("smallest")$. The autors performed an amount of tests in this sense, achieving good results[5].

References

[1] CS224n: Natural Language Processing with Deep Learning Lecture Notes:Part I Word Vectors I: Introduction, SVD and Word2Vec.

[2] Self-supervised learning gets us closer to autonomous learning. *https://hackernoon.com/self-supervised-learning-gets-us-closer-to-autonomous-learning-be77e6c86b5a*

[3] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visualrepresentation learning by context prediction.CoRR, abs/1505.05192, 2015



Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

Figure 3. Questions answered in the test at the Word2Vec paper by applying algebric functions[5]

[4] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised rep-resentation learning by predicting image rotations.CoRR, abs/1803.07728,2018

[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient esti-mation of word representations in vector space, 2013.

[6] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert.Unsupervisedlearning using sequential verification for action recognition.CoRR,abs/1603.08561, 2016.

[7] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Man-zagol. Extracting and composing robust features with denoising autoen-coders. InProceedings of the 25th International Conference on MachineLearning, ICML '08, page 1096–1103, New York, NY, USA, 2008. Associa-tion for Computing Machinery.

[8] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama,and Kevin Murphy.Tracking emerges by colorizing videos.CoRR,abs/1806.09594, 2018.

[9] LilianWeng.Self-supervised representation learning. *lilianweng.github.io/lil-log*, 2019.