Learning Feature-based Semantics with Autoencoder

Wonhong Lee WONHONG@STANFORD.EDU
Minjong Chung MJIPEO@STANFORD.EDU

Abstract

It is essential to reduce the dimensionality of features, not only for computational efficiency, but also for extracting the most meaningful pattern of features. This is particularly important in certain area such as computer vision or natural language processing, in which researchers usually have difficulty to manually specify feature. In this project, we applied the autoencoder model to represent the semantics of phrases and evaluated it by computing the correlation between the similarity of autoencoded feature and the rating manually tagged by human.

1. Introduction

Even though a lot of supervised learning algorithm such SVM, linear regression works well with a set of appropriate features, it is tedious to manually specify the feature of the input data in some domains like computer vision or natural language processing, and sometimes even quite difficult to define the features representing the input data in an intuitive way. Therefore, it is getting more and more important to reduce the dimensionality of the data and to extract only a set of meaningful features from them, which eventually leads to computational efficiency as well as the learning accuracy.

Natural language processing is one of the most promising area for which the autoencoder approach can be applied. Especially, compared to their syntactic information, the semantics of word, phrase, or sentence is quite difficult to be modeled accurately and evaluated systematically. Furthermore, supervised approach to learn semantic of word or phrase has certain limitation to retrieve the labeled accurate training data. Turian emphasized the efficiency of the semi-supervised approach for word representation.

In this project, we utilized autoencoder to automatically learn useful features from the input data, and use the compressed representation to find similar phrases given a phrase, which is usually hard to manually specify the features.

First of all, we made a simple assumption about the autoencoder model to reduce the number of parameter, which leads to fast convergence compared to the old version, and derived new update rule for this model. We applied this model to visualize the trained hidden unit to figure out what kind of interesting patterns our model tried to extract from input features.

To apply our model to natural language processing, we used Jeff Mitchell and Mirella Lapata's phrase pairs, with manual rating by humans, to train our autoencoder model, and to compute the Pearson correlation between the manual rating and the autoencoded (compressed) similarity to evaluate the model.

Furthermore, we compared the performance of our modified autoencoder model to other method to compute the similarity of phrases.

2. Background

2.1. Autoencoder

Autoencoder networks are feed forward neural networks that can have more than one hidden layer. These networks attempt to reconstruct the input data at the output layer. The targets at the output layer are the same as the input data, thus the size of the output layer is also the same as the size of the input layer. Also it assume that the values of input data are equal to the those of output data. In other word, it tries to learn a function h(W,b) $\approx$ x; it is trying to find the similarity between input features and output features. By adjusting the number of nodes in the hidden layer, we can discover very interesting feature between data sets. From the interesting structure of autoencoder, we also expect to find semantic correlations between various sentences.

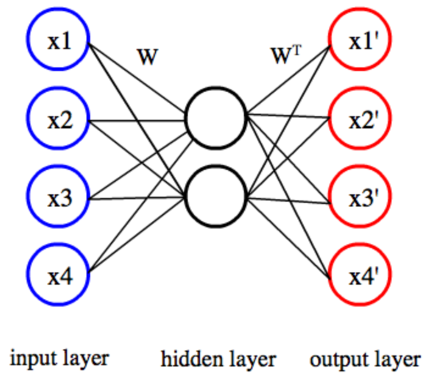

Figure 1. Autoencoder model

Autoencoder is unsupervised neural networks model that are trained using a gradient descent method, such as back propagation. Since the size of the hidden layer in an autoencoder is smaller than the size of the input data, the dimensionality of input data is reduced to a smaller-dimensional code space at the hidden layer. The outputs from the hidden layer are then reconstructed into the original data at the output layer. Like PCA, the autoencoders can give mappings in both directions between the data and the code space.

2.2. Word representation

Semi-supervised approaches such as Ando and Zhang (2005), Suzuki and Isozaki (2008) achieve remarkabe results in terms of accuracy. However, those approaches are limited in choosing a training model because of their distinct characteristics. Therefore, it is very difficult to utilize an exsiting supervised Natural Language Processing system in the semi-supervised methods. Consequently, it is getting more preferable to use unsupervised technics to induce word features.

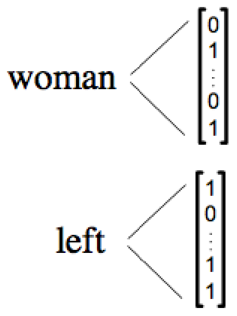

Figure 2. Word representation

A word can be represented as a mathematical object such as a vector associated with the word. Each dimension's value corresponds to a feature and might even have a semantic or grammatical interpretation, so we call it a word feature. However, in real world, the labeled training data to represent words is rare so that it results in poor estimation. Because of this limitation, NLP researchers tried to investigate suitable unsupervised methods to induce word features. One of powerful approach is to use clustering. This technique was used by a variety of researchers. In this project, we will choose data sets in those categories.

3. Model

3.1. Autoencoder

3.1.1. Assumption

This model starts from the assumption:

$$W_2 = {W_1}^T$$

In other words, we assume that the weight matrix of the second layer is the transpose of the weight matrix of the first layer, which effectively reduce the number of the parameters of the weight matrix into half. With this assumption, the output vector of the autoencoder can be written by three parameters, W, b_1, b_2:

$$Y = W^T f(WX + b_1) + b_2$$

The parameter b_1, b_2 represent the bias in the first and second layer, respectively, and $f(x)$ is the activation function. In this project, we use $tanh(x)$ as an activation function.

Finally, the cost function we should minimize is given by:

$$J(W, b_1, b_2) = \frac{1}{2}(Y - X)^T(Y - X)$$

3.1.2. Derivation

The update rule of the weight matrix W for the stochastic gradient descent algorithm is given by:

$$W := W - \alpha \frac{dJ}{dW}$$

Since $X \in R^n$ and $Y \in R^n$, the cost function can be written by the summation of the value for each dimension.

$$\begin{aligned} J(W, b_1, b_2) &= \frac{1}{2}(Y - X)^T(Y - X) \\ &= \frac{1}{2}\sum_{i=1}^{n}(Y_i - X_i)^2 \end{aligned}$$

Now, we can apply the chain rule to compute the derivative of the cost function with respect to weight parameter. (Note that $W \in R^{m\times n}$, $b_1 \in R^m$, and $b_2 \in R^n$)

$$\begin{aligned}\frac{dJ}{dY_i} &= \frac{d}{dY_i}\frac{1}{2}\sum_{j=1}^{n}(Y_j - X_j)^2\\ &= Y_i - X_i\end{aligned}$$

$$\begin{aligned}\frac{dJ}{dW_{pq}} &= \sum_{i=1}^{n}\frac{dJ}{dY_i}\frac{dY_i}{dW_{pq}}\\ &= \sum_{i=1}^{n}(Y_i - X_i)\frac{dY_i}{dW_{pq}}\end{aligned}$$

The value of the i-th dimension of Y can be written by:

$$\begin{aligned}Y_i &= \sum_{k=1}^{m} W_{ki} f(\sum_{l=1}^{n} W_{kl}X_l + b_{1k}) + b_{2i}\\ \frac{dY_i}{dW_{pq}} &= \sum_{i=1}^{n}(Y_i - X_i)W_{pi}f'(\sum_{j=1}^{n} W_{pj}X_j + b_{1p})X_q\\ &+ \quad (Y_q - X_q)f(\sum_{i=1}^{n} W_{pj}X_j + b_{1p})\end{aligned}$$

To sum up, the update rule for each parameter is given by: (derivation for b_1 and b_2 is similar)

$$\begin{aligned}\frac{dJ}{dW} &= W(Y - X)1^T \bullet f'(WX + b_1)X^T\\ &+ \quad f(WX + b_1)(Y - X)^T\\ \frac{dJ}{db_1} &= W(Y - X) \bullet f'(WX + b_1)\\ \frac{dJ}{db_2} &= Y - X\end{aligned}$$

(Note that $\bullet$ is used to denote the element-wise product operator, often called Hadamard product)

3.2. Phrase representation

We want to let our modified autoencoder model to learn the semantic of phrase consisting of two words, expecting each hidden unit to try to catch certain meaningful pattern of the composition of words. Jeff Mitchell and Mirella Lapata (2008) introduced vector-based models of semantic composition of two words and showed that the model works well on a phrase similarity task. In this project, we introduced a different form of composition of the vector representation of words: concatenation.

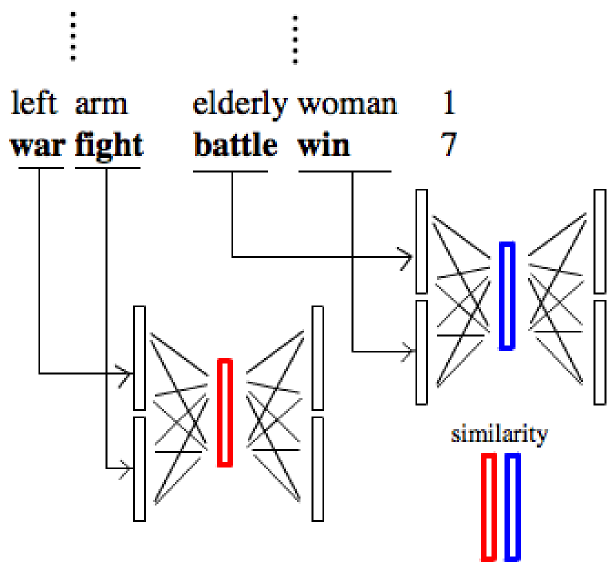

Figure 3. Phrase representation

Since the autoencoder operates in unsupervised way, we can take advantage of unbounded amounts of training data without any human effort manually labeling the data. This approach is especially effective with language model since it is quite difficult to define for which each dimension of phrase feature vector should represent.

After training the model by optimizing the weight matrix as well as two bias vectors with the stochastic gradient descent method, we can represent each phrase by computing $WX + b_1$, and compare the similarity between those vectors which supposedly represent the semantic of the phrases. The similarity measure we used is Euclidean distance and cosine similarity, which defined as:

$$\begin{aligned}EUC(u,v) &= ||u - v||_2\\ COS(u,v) &= \frac{u \cdot v}{||u|| \cdot ||v||}\end{aligned}$$

4. Experiment

4.1. Visualization

To verify our model assumption, we wanted to visualize what the model learned with autoencoder. The model were trained by randomly picking one of the 10 images, then randomly sampling an 8x8 image patch from the selected 512x512 images, and set the size of the hidden units to 30 and the input units of 64. After training the model, each cell in the below images represents the input image that maximizes the activation of the corresponding hidden unit.

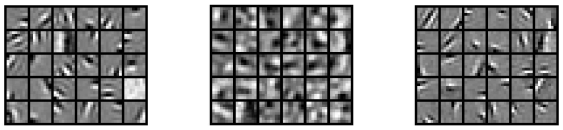

Figure 4. Visualization of trained hidden units

It is easy to find out that each hidden unit is trying to capture a different pattern of input feature, which leads to effective dimensional reduction.

4.2. Phrase similarity

The preprocessed datasets by Collobert and Weston's neural language model (2008) was used as input features of this natural language processing experiment. From the data sets, each word can be represented as n-dimensional vecotor. (n=25,50,100,200) All the datasets are provided in either scaled or unscaled version.

As mentioned before, the stochastic gradient descent method is used as an optimization algorithm to minimize the cost function $J(W, b_1, b_2)$. The learning rate parameter is tried for 0.0002, 0.002, 0.02, 0.2. Furthermore, in order to avoid the case the weight matrix grows too big, we introduced so-called weight decay term at the end of the cost function. The decay parameter is tried for 0.0001, 0.001, 0.01, 0.1. The combination of the parameters of the best performance is 0.02 for learning rate and 0.001 for decay parameter. The number of iterations is also tried differently, but it tends to converge before 2 million iterations.

The bias parameters are initialized to zero vector and weight matrix to randomly computed of the range $[0, \frac{1}{\sqrt{q}}]$, when q is the dimension of the input feature. We tried different initialization approaches, but there wasn't notable difference among them.

The Pearson correlation is used to verify the accuracy, which is computed between the similarities of autoencoded phrases and the level of similarity manually rated by human. For the Euclidean distance measure, there should be a negative correlation, and for the cosine similarity, it should be as close as possible to 1.

For comparison, we also implemented several different models, including sparse autoencoder. In the table below, "AUTO" represents for the similarity of autoencoded phrases, "RAW" for the uncompressed concatenated phrase vectors, "PAIRAVG" for the pairwise average similarity between corresponding words of phrases, and "SPARSEAUTO" for the sparse autoencoded phrases, respectively.

Measure	Correlation
EUC, AUTO	-0.0842
COS, AUTO	0.0502
EUC, RAW	0.0341
COS, RAW	-0.0201
EUC, PAIRAVG	0.0341
COS, PAIRAVG	0.0063
EUC, SPARSEAUTO	0.0742
COS, SPARSEAUTO	0.0020

Table 1. Correlation between similarities computed by different model

Note that the "RAW" and "PAIRAVG" models do not utilize the training data to learn parameters, just computing the similarity given words and their representation.

5. Discussion

Even though the Pearson correlation value of our Autoencoder model is not significant compared to other measures, it consistently showed a negative correlation, in case of the Euclidean distance measure, with various combination of parameters such as learning rate or weight decay. For cosine similarity measure, it still shows good performance with relatively higher positive correlation compared to other ones.

In the other hands, the raw representation of phrases, which is just concatenation of two words, continuously showed inconsistent correlation with different parameter combination, nearly random pattern. It is also true for the pairwise average similarity between corresponding words. In this result, we can conclude that these two model to represent phrase consisting of two words failed to effectively capture the essential parts of input features, while the autoencoder model showed consistent correlation pattern.

Although it shows consistent correlation pattern, the absolute value is not significant enough to conclude that there is strong correlation between them. There are several factors which possibly have an effect on the experiment result.

First of all, the training data size is not large enough to cover large dimension of the input feature, such as 100 or 200 dimensional representation of Collobert and Weston's embeddings. Since we can take advantage of the power of unsupervised learning by utilizing large unlabeled, unprocessed dataset, this is one of the future work which should be followed by this research.

Furthermore, we can still utilize a variety of different feature vectors to represent each word, not only the Collobert and Weston's neural language model. Since the autoencoder tries to capture interesting pattern of concatenated words, we believe that uncompressed raw vector representation should be applied for this model, in which each dimension of feature has certain meaning such as POS tag, frequency, and other syntactic or semantic information.

Finally, a variety of composition method to represent phrases given words can be tried as input feature of this model. These include Jeff Mitchell and Mirella Lapata's additive, multiplicative, etc.

6. Acknowledgement

Thanks to Richard Socher for useful discussion.

References

Ronan Collobert, Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. *ICML '08 Proceedings of the 25th international conference on Machine learning*

Jeff Mitchell, Mirella Lapata. 2008. Vector-based Models of Semantic Composition. *In Proceedings of ACL-08: HLT*

Joseph Turian, Lev Ratinov, Yoshua Bengio. 2010 Word representations: A simple and general method for semi-supervised learning. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*

R. Ando, T. Zhang. 2005. A high- performance semi-supervised learning method for text chunking. *ACL*

J. Suzuki, H. Isozaki. 2008. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. *ACL-08: HLT (pp. 665673).*

Richard Socher. 2010. Deep Chinese Character *Department of Computer Science, Stanford University*

Andrew Ng. 2010. Sparse Autoencoder *Department of Computer Science, Stanford University*