Northeastern University

Master's Thesis Proposal

CS 7990

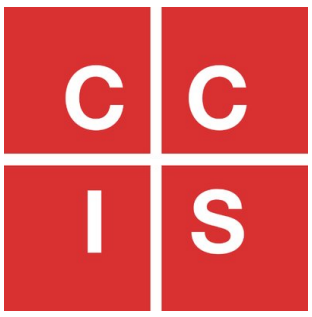
Word-vector Regularization for text classification algorithms

Author:
Ramkishan Panthena

Advisor:
Dr. Virgil Pavlu

Official Reader:
Dr. Byron Wallace

December 13, 2018



1 Introduction

The goal of text classification is to classify text documents into one or more classes according to their content. For this purpose, a document must be transformed into a representation which is suitable for the learning algorithm and the classification task. Representing documents as bag-of-words is a commonly used method in document classification where the frequency of occurrence of each word is used as a feature for training a classifier. However, one should note that when using this representation, some document information is lost as the model disregards grammar and word ordering.

1.1 Problem Statement

Although the bag-of-words model is widely used and performs exceptionally well in most text classification problems, it contains several limitations. As per Zipf's law [1], given a large sample of words used, the frequency of any word is inversely proportional to its rank in the frequency table. So, word number n has a frequency proportional to 1/n. Thus, a large vocabulary can cause extremely sparse representations.

The classification accuracy we observe on the test set largely depends on the quality of training sets we have used to build our models. That is, if the training information is sparse, then we can expect the category model to be a poor representation of a category thereby leading to poor classification accuracy. Words that occur rarely do not give a learning algorithm enough information to determine its influence on classification correctly.

Thus, in such a case any linear classifier like logistic regression, naive bayes, linear SVM would treat each word independently and assign them different coefficients based on the impact of the individual word on the response variable. As the training data for rare words would be sparse, their coefficients would be near 0 implying that the impact of these features is small. This may not be the case as these rare words might be misrepresented due to sparse data. Our model tries to solve this problem by using a word-vector regularizer that assigns similar coefficients to words which are used in a similar context, thereby boosting the effect of similar but rare features on the final prediction.

We use logistic regression for binary text classification on a bag-of-words model. Let $\theta^{(i)}$ be the regression coefficients of word$_i$ which determines the association between each feature value (word occurrence in document) and what the target we are trying to predict.
The cost function for logistic regression would be given as:

$$Cost(h_{(\theta)}(x), y) = \begin{cases} -log(h_{(\theta)}(x)), & \text{if y=1} \\ -log(1 - h_{(\theta)}(x)), & \text{if y=0} \end{cases} \tag{1}$$

In general, the features with larger coefficients are more important because they make a significant contribution in predicting the correct class. However if a word$_k$ is rare, its corresponding regression coefficient ($\theta^{(k)}$) could be very small or very large (minimizing the loss) due to lack of evidence in the training

set; which is not useful for predictions. Our plan here is to constrain such coefficients by the word semantic similarity with other more frequent terms, thus simulating a higher occurence and prohibiting extreme behavior. This also helps with non-frequent synonym words in making their coefficients more uniform.

Example: In order to get a better understanding, consider a classification problem where we are trying to identify whether a document is talking about animals or not. For this example, let's say we only have five features. The bag-of-words representation using term frequencies for 5 different documents would look like:

	dog	football	canine	movies	cat
D1	3	0	0	0	4
D2	0	0	0	6	0
D3	5	0	1	0	6
D4	0	8	0	0	0
D5	0	7	0	4	0

Table 1: Toy dataset

From the above example, we can see that the word 'canine' occurs only once in one of the document. If we train a linear classifier on the above dataset, then 'canine' would have a very low feature importance due to its sparse representation. When we test this classifier on a document where 'canine' is more densely represented as compared to 'dog' or 'cat', then that document would get misclassified as "not talking about animals".

In contrast, since our model would assign similar weights to similar words and considering 'dog' and 'canine' are synonyms, our model would assign a higher feature importance to 'canine', thereby increasing the probability of making a correct prediction on a new document where 'canine' is more densely represented as compared to the train data.

1.2 Related work

A number of approaches have been proposed to increase the classification accuracy on the bag-of-words model.

To aggressively reduce the dimensionality of models, Joachims [2] (1996), Yang and Pedersen[3] (1997) suggested pruning of infrequent words. Mansuy and Hilder [4] (2006) recommended removing of stop words and part-of-speech tags. Porter [5] (1980) proposed removal of suffixes from words. However, Joachims [2] test results revealed that the performance of the system is higher when more words are used as features, with the highest performance achieved using the largest feature set. Any approach that limited the number of words to the most important ones was likely to reduce the classification accuracy as these pruned words lose their ability to contribute to the classification of text. Quinlan [6] (1993) suggested choosing words which have high mutual information with the target concept. However, picking words with high mutual

information had relatively poor performance due to its bias towards favoring rare terms, and its sensitivity to probability estimation errors.

In an attempt to address the issue of related concepts in text classification, many researchers have incorporated features using dictionaries and encyclopedias. Mavroeidis et. al [7] (2005) proposed to extend the traditional bag of words representation by incorporating syntactic and semantic relationships among words using a Word Sense Disambiguation approach. Wang and Domeniconi [8] (2008) explored a similar approach by embedding background knowledge derived from Wikipedia to enrich the representation of documents. Although empirical results have shown improvements in some cases, the applicability of using dictionaries to improve classification accuracy is limited. Ontology is manually built, and the coverage is far too restricted. Recently, Heal et. al [9] (2017) introduced a method for enriching the bag-of-words model by complementing rare term information with related terms from Word Vector models. However, it was revealed that these methods achieved significantly better results only when the training sets were small. There wasn't enough evidence of achieving better results on large datasets.

In addition to incorporating related concepts to improve classification performance, other approaches have also been proposed. One of these approaches considers using part-of-speech tags associated with words contained in a document (Scott and Matwin [10] 1998), (Jensen and Martinez [11] 2000). Since words can have multiple meanings depending upon how and where they are used in a sentence, the part-of-speech may be relevant to text classification. However, a different paper from Mansuy [4] revealed that there was no significant difference between the accuracy of the classifiers whether part-of-speech tags are utilized or not.

To deal with overfitting, different regularization techniques have also been proposed. Regularization adds a penalty on the different parameters of the model to reduce the freedom of the model. Hence, the model will be less likely to fit the noise and improve its generalization abilities. The Lasso regularization acts as a way of feature selection by shrinking some parameters to zero, whereas the Ridge regularization will force the parameters to be relatively small but are not cut to zero.

2 Proposed Method

In this method, we propose to enhance the bag-of-words model for text classification by presenting a novel regularizer which would assign similar coefficients to words used in similar context.

Formally, the problem we are trying to solve can be formulated as follows: Given a set of m documents with n features, the documents would be represented by matrix X $\in$ R^{mxn}. We want to find the coefficients that predict the output Y from the documents X and we want to build a model where the coefficients ($\theta^{(i)}$) are a function of its word$_i$ representation ($v^{(i)}$), i.e.,

$$\theta^{(i)} = f(v^{(i)}) \tag{2}$$

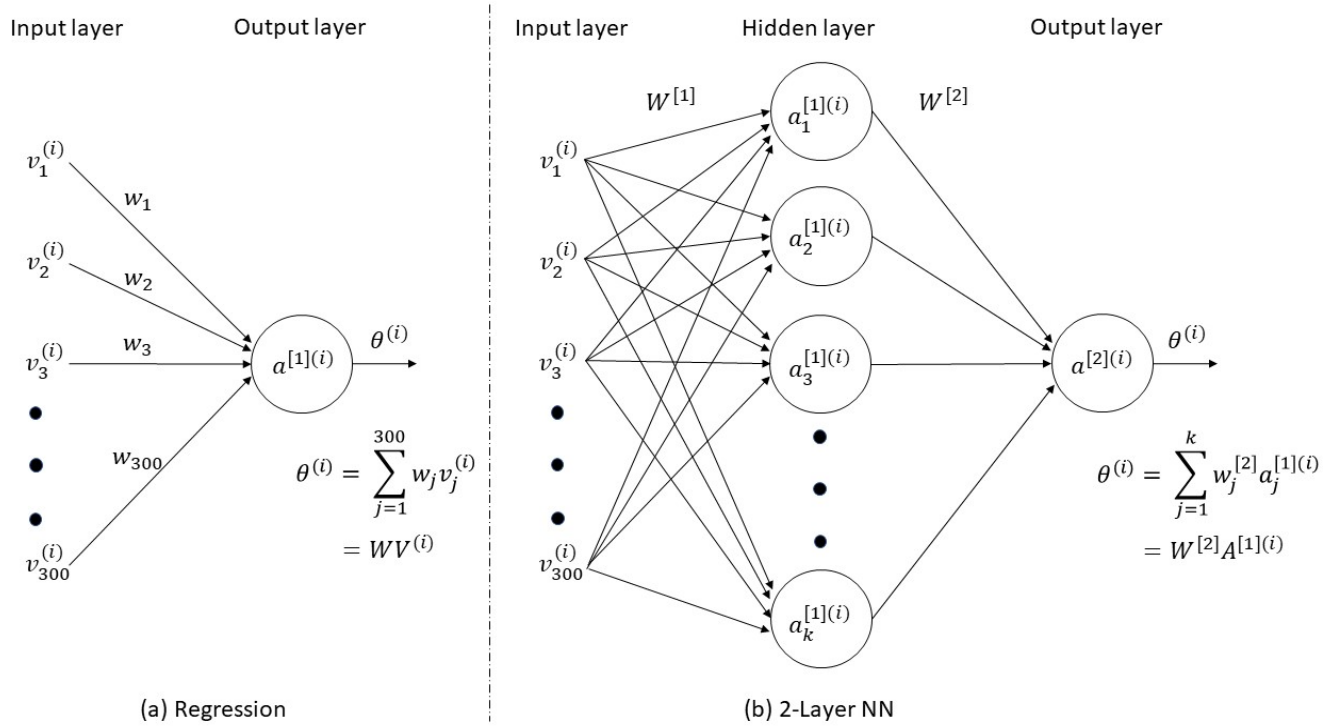


Figure 1: Network diagram to learn $\theta^{(i)}$ from word$_i$ representation f($v^{(i)}$)

This would act as a regularization constraint so that similar word representations would have similar coefficients, which satisfies a continuity condition

$$|f(v^{(1)}) - f(v^{(2)}))| \;\leq\; sim(v^{(1)}, v^{(2)}) \tag{3}$$

where, sim dictates the similarity between words $v^{(1)}$ and $v^{(2)}$ and $f(v^{(1)}), f(v^{(2)})$ are coefficients of their respective words as per (2)

Since words used in similar context will have similar word-vectors and by making the regression co-efficient of a word to be a function of its word-vector representation, we can regularize the regression coefficients of two similar words to have similar values. So, if one word occurs more frequently and has higher regression co-efficient, the other word even if it is very rare would be considered an important feature and would have a higher regression co-efficient. This would improve the classification performance in cases where these rare words occur more frequently in the test set.

We explore learning the function $f(v^{(i)})$ via several algorithms:

- regression
- 2-layer neural network
- boosting trees

Fig. 1 is a network diagram showing how the coefficients (θ_i) would be obtained for a word$_i$ representation using both the above mentioned methods.

Learning f as Regression: We will initially determine if we can learn the coefficients from the word representation of the features using a linear regressor, i.e.,

$$\theta^{(i)} = w_1 v_1^{(i)} + w_2 v_2^{(i)} + ... + w_d v_d^{(i)} = \sum_{d=1}^{D} w_d^{(i)} v_d^{(i)} \tag{4}$$

where, $w = (w_1, w_2, ..., w_d)$ are the parameters of the regression function.
$v = (v_1, v_2, ..., v_d)$ are d-dimensional word-representations of word i

Within the class of linear functions, our task shall be to find the best parameters w that minimize the error function such that,

$$Cost(h_{(\theta|w)}(x), y) = \begin{cases} -log(h_{(\theta|w)}(x)), & \text{if y=1} \\ -log(1 - h_{(\theta|w)}(x)), & \text{if y=0} \end{cases} \tag{5}$$

Learning f as 2-layer neural network: If there is a non-linear relationship between the coefficients and word representations, we can learn them through a multi-layer neural network. We start with a 2-layer neural network having one hidden layer and one output layer. The hidden layer requires the d-dimensional word$_i$ representations as input and outputs an activation value.

Let $a^{[1](i)} = (a_1^{[1](i)}, a_2^{[1](i)}, a_3^{[1](i)}..)$ be the activation of the neurons in the hidden layer, and $a^{[2](i)}$ be the activation of the neurons in the output layer.

where, a = g(z), where g is some activation function (like sigmoid, relu, tanh)
z = Wx + b, where x are the features from the input layer
$a_k^{[l]}$ denotes the activation of the k^{th} unit in layer l
$x^{(i)}$ with parenthesis refers to the ith word
a$^{[1]}$ is the concatenation of all first layer activations

For a 2-layer neural network, we would have the following parameters:

$$z^{[1](i)} = W^{[1]} V^{(i)} + b^{[1]} \tag{6}$$
$$a^{[1](i)} = g(z^{[1](i)}) \tag{7}$$
$$z^{[2](i)} = W^{[2]} a^{[1](i)} + b^{[2]} \tag{8}$$
$$\theta^{(i)} = a^{[2](i)} = g(z^{[2]}) \tag{9}$$

We can then train a neural network model to learn the coefficients ($\theta^{(i)}$) which would help minimize the cost function in (5)

2.1 Obtain word-vectors

Out of the successful deep learning models used for word-embeddings, two of the most popular ones are Word2Vec [12] and Global Vectors [13]. For our research, we will be using Google's Word2Vec which has pre-trained word-vectors with 300 dimensions trained using over 100 billion words.

Also it has been seen that a smaller domain specific Word Vector model is modelled better than a general model trained over a much larger corpus of text. Thus Google's pre-trained Word Vector model which was trained over three million unique words and phrases will be retrained on the training data to generate domain specific Word Vectors.

2.2 Train new model

Once the features have been extracted using the bag-of-words model and their corresponding word-vectors obtained, we can train a neural network model based on the cost function in (5)

Below is the link to its source code on GitHub. It will be made publicly available upon thesis completion:
`https://github.com/RamkishanPanthena/Master-s-Thesis`

3 Experiments

We compare our model to the vanilla logistic regression algorithm by implementing both using the TensorFlow library. Both the models can perform multi-class and multi-label classification. For multi-class, we use the softmax function to assign probabilities to each class which add up to 1. We use the sigmoid function for multi-label classification to train an independent logistic regression model for each class and accept all labels greater than a certain threshold as predictions.

We tune our model using learning rate and regularization parameter as hyper-parameters and train with different number of epochs till the loss converges. For multi-label problems, we also use prediction threshold as one of the hyper-parameters to list labels with probabilities greater than the threshold value.

3.1 Evaluation

To evaluate the performance of both models for multi-class problems, we use accuracy and F1 measures.

For multi-label problems, predictions for an instance is a set of labels, and therefore the prediction can be fully correct, partially-correct or fully-incorrect. Thus, to better evaluate our models we will use the following three measures: *set accuracy*, which is the ratio of perfectly matched instances to the total number of instances; *instance-F1*, which evaluates the performance of partially correct predictions averaged over instances; *label-F1*, which evaluates the performance of partially correct predictions averaged over labels.

For a dataset with ground truth labels $y^{(n)}$ and predictions $\hat{y}^{(n)}$, and n instances where n = 1,2,...,N, these three measures are defined as:

$$set\ accuracy = \frac{1}{N}\sum_{n=1}^{N} I(y^{(n)} = \hat{y}^{(n)}) \tag{10}$$

$$instance\text{-}F1 = \frac{1}{N}\sum_{n=1}^{N} \frac{2\sum_{l=1}^{L} y_l^{(n)}\hat{y}_l^{(n)}}{\sum_{l=1}^{L} y_l^{(n)} + \sum_{l=1}^{L} \hat{y}_l^{(n)}} \qquad label\text{-}F1 = \frac{1}{N}\sum_{n=1}^{N} \frac{2\sum_{n=1}^{N} y_l^{(n)}\hat{y}_l^{(n)}}{\sum_{n=1}^{N} y_l^{(n)} + \sum_{n=1}^{N} \hat{y}_l^{(n)}} \tag{11}$$

where for each instance n, $y_l^{(n)}$ = 1 if label l is a given label in ground truth; $\hat{y}_l^{(n)}$ = 1 if label l is a predicted label.

3.2 Datasets

Below is a table showing how our model performed as compared to logistic regression.

Datasets	Baseline Set Accuracy	Our model Set Accuracy	Baseline Instance-F1	Our model Instance-F1
Slashdot	32.63	34.21	41.18	43.02
Medical	66.06	67.87	74.66	76.91

Table 2: Set-Accuracy and Instance-F1 Results.

We also run a validation experiment with artificially modified dataset 20newsgroup to mimic sparse representation: we found the most similar words in the train set with cosine similarity greater than 0.3 and make the feature values of all but one of the words equal to zero. We then compared the performance of our model with logistic regression before and after zeroing feature values of similar words. The below results show that our model was able to get a good test accuracy after the imposed sparsity.

	Logistic Regression	Our model
Before zeroing features	82.6	84.01
After zeroing features	75.67	83.54

4 Thesis Plan

Below is the action plan for the thesis in the Spring 2019 semester. We plan to defend by May 1st 2019.

No.	Action	% Complete	Time needed
1	Implement and validate f through regression	70%	3 weeks
2	Implement and validate f through 2-layer NN	0%	2 months
3	Run method on 5 datasets	20%	1 month
4	Particular runs - medical data	30%	2 weeks
5	Rare feature analysis - check their importance	0%	1 month

Table 3: Thesis Work Timeline

References

Wentian Li. Random texts exhibit zipf's-law-like word frequency distribution. *IEEE Transactions on information theory*, 38(6):1842–1845, 1992.

Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report, Carnegie-mellon univ pittsburgh pa dept of computer science, 1996.

Yiming Yang and Jan O Pedersen. A comparative study on feature selection in text categorization. In *Icml*, volume 97, pages 412–420, 1997.

Trevor Mansuy and Robert J Hilderman. A characterization of wordnet features in boolean models for text classification. In *Proceedings of the fifth Australasian conference on Data mining and analystics-Volume 61*, pages 103–109. Australian Computer Society, Inc., 2006.

Martin F Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

Dimitrios Mavroeidis, George Tsatsaronis, Michalis Vazirgiannis, Martin Theobald, and Gerhard Weikum. Word sense disambiguation for exploiting hierarchical thesauri in text classification. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 181–192. Springer, 2005.

Pu Wang and Carlotta Domeniconi. Building semantic kernels for text classification using wikipedia. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 713–721. ACM, 2008.

Bradford Heap, Michael Bain, Wayne Wobcke, Alfred Krzywicki, and Susanne Schmeidl. Word vector enrichment of low frequency words in the bag-of-words model for short text multi-class classification problems. *arXiv preprint arXiv:1709.05778*, 2017.

Sam Scott and Stan Matwin. Text classification using wordnet hypernyms. *Usage of WordNet in Natural Language Processing Systems*, 1998.

Lee S Jensen and Tony Martinez. *Improving text classification by using conceptual and contextual features*. PhD thesis, Brigham Young University. Department of Computer Science, 2000.

Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196, 2014.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.