Document Classification on Neural Networks Using Only Positive Examples

Larry M. Manevitz MANEVITZ@CS.HAIFA.AC.IL
Malik Yousef YOUSEF@CS.HAIFA.AC.IL
Department of Computer Science, University of Haifa, Haifa, Israel

Abstract

In this paper, we show how a simple feed-forward neural network can be trained to filter documents when only positive information is available, and that this method seems to be superior to more standard methods, such as tf-idf retrieval based on an "average vector". A novel experimental finding that retrieval is enhanced substantially in this context by carrying out a certain kind of uniform transformation ("Hadamard") of the information prior to the training of the network.

1. Introduction

The goal of this research is to develop a filter that can examine a corpus of documents and choose those of interest.

This requires a method of defining what it means to be "of interest" and a method of matching the documents to this definition. It is natural and convenient to assume that the definition of interest be learned (see also [4] and [6]) by observing examples, and, in this context, it is pertinent to assume only positive examples. That is, one can have a sample set of examples of documents which are "interesting" and from this set develop a filter which can be applied to select other such "interesting" documents. The reason for using only positive examples is that one can (i) obtain such examples simply by observation; i.e. for many applications an "active" teacher will not be necessary (ii) in many contexts, it is easier to find "typical" examples rather than typical "non-examples". See [2] for other papers on the use of positive examples only.

2. The Neural Networks Classifier

The basic design of the filter under discussion here is a feed-forward neural network. In order to incorporate the restriction of positive examples only, we used the design of a feed-forward network with a "bottleneck". we choose a three level network with m inputs, m outputs and k neurons on the hidden level, where $k < m$. Then the network is trained, under standard back-propagation to learn the identity function on the sample examples.

The idea is that while the bottleneck prevents learning the full identity function on m-space; the identity on the small set of examples is in fact learnable. Then the set of vectors for which the network acts as the identity function is a sort of sub-space which is similar to the trained set. (This avoids the "saturation" problem of learning from only positive examples.) Thus the filter is defined by applying the network to a given vector; if the result is the identity, then the vector is "interesting".

In most our experiments we used 20 real valued inputs and output and 6 hidden level neurons. All neurons were standard sigmoids. Training proceeded according to standard back-propagation with learning parameter .75 and momentum coefficient .08 until the mean-square error fell below a pre-determined level.

For acceptance threshold determination, a sophisticated method was used , based on a combination of variance and calculating the optimal F_1 measure. (See below and [7] for a definition of F_1 .) During training, we checked, at different levels of error, the F_1 values of the test set. We stop the training at the point which F_1 started a steep decline. Then we did a secondary analysis to determine an optimal real multiple of the standard deviation of the average error to serve as the threshold.

We also examined our neural network filter with different sizes of input and output, (corresponding to different numbers of features from the document, see below) in order to investigate how this influenced the performence of the classification task. (See table 2.)

2.1 Text Representation and Feature Selection

To implement the bottleneck filter, we ran experiments with several different representations. In addition, as a comparison, we ran a different algorithm, that simply takes the average vector of the sample examples as a prototype vector and defines the tolerance as the largest angle between this prototype and all sample examples. The filter then accepts all documents whose vector representation is within this tolerance. (Note that this is similiar to the Rocchio algorithm [1] although we are using only positive examples to determine the angle.)

One can use , instead of the word frequency, the *tf-idf* (term-frequency-inverse-document- frequency) representation [5] which is given by the following formula (where $f(word)$ meand the frequency of the word in the document and $N(word)$ means the number of documents the word appears in):

$$tfidf(word) = f(word) \cdot [log\frac{n}{N(word)} + 1].$$

To explain our heuristic mix of neural network encoding and heuristic choice of representation, we will need a few definitions:

Let C, the *"corpus"* be the set of documents to be classified. Let T be a subset of C the class of "interesting" documents. Let E be a subset of T, the positive examples. The problem is to define a function (or "filter"), using only information from E that distinguishes T from $\overline{T}$, the complement of T.

We proceed as follows: Let D be the *dictionary* of all words in $\bigcup E$; with each word is associated its frequency in the list. Heuristically, we eliminate words whose document frequency is less than 3; and use standard algorithms to (i) eliminate connecting words and (ii) strip grammatical endings from common words [3].

From this dictionary, we then chose the m words that appear in the most documents of E. We call these "key-words"; however they are chosen automatically. This choice of m is was influenced by the comparisons with different choices of m (see table 2). For later reference (see "Hadamard product" below) we define v_E as the m-dimensional vector consisting of the frequencies of appearance of each of the keywords throughout the dictionary. Then for each document $\mathbf{e} \in E$ we associate a vector of dimension m, which we will continue to designate as $\mathbf{e}$. Here e_i is the frequency of the i^{th} chosen keyword in the document $\mathbf{e}$.

2.2 Hadamard Product

We discovered experimentally that the following additional transformation, H_E, of vectors substantially enhances performance. Here

$$H_E(e_i) = e_i \cdot v_{Ei},$$

i.e. take the component-wise product with the frequency vector of the dictionary.

It seems reasonable to look for a Bayesean explanation of this phenomenom. Let g represent a given document, w_i the i^{th} word in the dictionary, and E a set of training examples. Assume that e_i represents $P(w_i|g)$ and v_{E_i} represents $P(w_i|E)$; i.e. the probabilities of a chosen word being w_i given that you are either in the document being tested or in the class of interesting examples. One can then argue (under certain independent assumptions) using Bayes rule that the product vector is representing the adjusted probability of a word being chosen given that it is both in the document g and an interesting example.

3. Data Set, Experiments and Results

3.1 Reuters-21578

To test the above ideas, we applied these filters to the standard *Reuters* data-base , a preclassified collection of short articles. This is one of the standard test-beds used to test information retrieval algorithms. So C is the overall collection of articles, and there are a variety of subsets T.

For each choice of subject T, we used 25% of the positive data to train; and then ran the filters on all of C.

We treated each of the 10 categories as a binary classification task and evaluated the classifiers for each category separately. For reporting the results we used the F_1 measure, the recall and the precision values.

For text categorization, the effectiveness meaure of recall and precision are defined as follows. recall is the number of items of category identified devide by the number of category members in test set. precision is the number of items of category identified devide by total items assigned to category.

Van Rijsbergen [7] defined the F_1-meaure as a combination of recall (R) and precision (P) with an equal weight in the following form: F_1(R,P)= $\frac{2RP}{R+P}$

In Table 1, we summarize the results for using Neural Networks classifier with Hadamard and frequency document representation. The Prototype algorithm is presented as baseline algorithms.The results show that the Hadamard representation is superior, and the Neural Networks is more superior than the Prototype algorithm.

Table 1. Neural Networks (NN) Comparison of Hadamard and Frequency representation. Prototype algorithm is presented as the baseline algorithm using *tf-idf* representation.

	NN(Hadamard)			NN(Frequency)			Prototype		
	F_1	R	P	F_1	R	P	F_1	R	P
Earn	0.781	0.800	0.763	0.418	0.805	0.282	0.637	0.569	0.724
Acq	0.534	0.598	0.483	0.347	0.363	0.332	0.468	0.492	0.446
Money	0.542	0.641	0.470	0.475	0.420	0.546	0.484	0.500	0.470
Grain	0.415	0.394	0.439	0.379	0.355	0.408	0.402	0.320	0.542
Crude	0.537	0.505	0.573	0.476	0.410	0.566	0.398	0.322	0.520
Trade	0.573	0.600	0.547	0.536	0.513	0.561	0.557	0.503	0.623
Int	0.496	0.416	0.616	0.478	0.405	0.583	0.454	0.440	0.468
Ship	0.393	0.328	0.492	0.388	0.400	0.376	0.370	0.358	0.382
Wheat	0.507	0.446	0.588	0.414	0.430	0.400	0.262	0.263	0.260
Corn	0.310	0.451	0.236	0.315	0.434	0.247	0.230	0.423	0.158
Average	0.508	0.517	0.520	0.422	0.453	0.430	0.426	0.419	0.459

In Table 2, using only the Hadamard representation, we investigated the affect of increasing the dimension of the features. (That is, allowing a larger number of key-words, while keeping the size of the hidden level the same.) We see some improvement but not dramatic.

Table 2. Comparison of different sizes of Networks Using Hadamard Representation

NN size	20	40	60	100	200
	F_1	F_1	F_1	F_1	F_1
Grain	0.415	0.504	0.515	0.518	0.539
Crude	0.537	0.591	0.565	0.588	0.583
Trade	0.573	0.605	0.616	0.602	0.603
Interest	0.496	0.504	0.497	0.528	0.510

4. Summary

The basic result can be summarized as follows: (i) Using the autoencoder neural network works. (ii) It is sensitive to the choice of representation. a. Representing documents by tf-idf in this context fails. b. Representing documents by frequency works reasonably. c. Modifying the frequency representation by a "Hadamard" operation results in substantially improved results.

5. Acknowledgements

This work was partially supported by *HIACS*, the Haifa Interdisciplinary Center for Advanced Computer Science. This work is part of a doctoral thesis of the second author.

References

[1] T. Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1996.

[2] S. Muggleton. Learning from positive data. *Machine Learning*, 1999.

[3] M. Porter. An algorithm for suffix stripping. *program*, (14):130–137, 1980.

[4] C. Quek. Classification of world wide web documents. Master's thesis, School of Computer Science Camegie Mellon University, 1997.

[5] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, Berlin, 1983.

[6] B.D. Sheth. A learning approach to personalized information filtering. Master's thesis, Massachusetts Institute of Technology, 1994.

[7] C.J. van Rijsbergen. *Information Retrieval.* Butterworths, London, second edition, 1979.