15 Construction of Random Projections of Word Histograms by Pointers

Teuvo Kohonen

In the basic *vector space model* [1], documents were represented statistically as real vectors in which each component corresponds to the frequency of occurrence of a particular word in the document: the model or document vector can be viewed as a weighted word histogram. The main problem of the vector space model is the large vocabulary in any sizable collection of free-text documents, which means a vast dimensionality of the model vectors.

It was shown previously that the dimensionality of the document vectors can be reduced radically by a random projection method [2,3] without essentially losing the power of discrimination between the documents.

Before description of the new encoding of the documents, some experimental results are presented that motivate its idea. Table 6 compares a few projection methods in which the model vectors, except in the first case, were always 315-dimensional. As the material in this experiment we used 18,540 English documents from 20 Usenet newsgroups of Internet. When the text was preprocessed as explained in Sec. 14, the remaining vocabulary consisted of 5,789 words or word forms. When the *document map* as discussed more closely in Sec. (*ibid*) was formed, each document was mapped onto one of its grid points. These points were then classified according to the majority of newsgroup names in them. All documents that represented a minority group at any grid point were counted as classification errors.

The classification accuracy of 68.0 per cent reported on the first row of Table 6 refers to a classification that was carried out with the vector-space model with full 5789-dimensional histograms as document vectors. In practice, this kind of classification would be orders of magnitude too slow.

Random projection of the original document vectors onto a 315-dimensional space yielded, within the statistical accuracy of computation, the same figures as the basic vector space method. This is reported on the second row. The figures are averages from seven statistically independent tests, like in the rest of the cases.

Consider now that we want to simplify the projection matrix in order to speedup computations. We do this by thresholding the matrix elements, or using sparse matrices. Such experiments are reported next. The following rows have the following meaning: Third row, the originally random matrix elements were thresholded to $+1$ or -1; fourth row, exactly 5 randomly distributed ones were generated in each column, and the other elements were zeroes; fifth row, the number of ones was 3; and sixth row, the number of ones was 2, respectively.

The sparse projection matrices have now turned out sufficiently good in producing reasonable classification accuracies, and we shall next concentrate on fast computation of the matrix-vector products. Consider first the following trivial-looking piece of pseudocode, where we form the product $\mathbf{x} = \mathbf{R}\mathbf{n}$:

Table 6: Classification accuracies of documents, in per cent, with different projection matrices $\mathbf{R}$. The figures are averages from seven test runs with different random elements of $\mathbf{R}$.

	Accuracy	Standard deviation due to different randomization of $\mathbf{R}$
Vector space model	68.0	–
Normally distributed $\mathbf{R}$	68.0	0.2
Thresholding to $+1$ or -1	67.9	0.2
5 ones in each column	67.8	0.3
3 ones in each column	67.4	0.2
2 ones in each column	67.3	0.2

```
for i:=1 step 1 until m do x(i):=0 ;
for all (i,j) such that R(i,j)=1 begin
      x(i):=x(i)+n(j) ;
end
```

This scheme is supposed to give us the idea that if we reserve a memory array for $\mathbf{x}$ that acts like an accumulator, another array for $\mathbf{n}$, and *permanent address pointers* from all the locations of the $\mathbf{n}$ array to all such locations of the $\mathbf{x}$ array for which the matrix element $R(i,j)$ of $\mathbf{R}$ is equal to one, we can form the product very fast by following the pointers.
In the method that is actually used we do not project ready histograms, but the pointers are already used with each word in the text in the construction of the low-dimensional document vectors. When scanning the text, the hash address for each word is formed, and if the word resides in the hash table, those elements of the $\mathbf{x}$ array that are found by the (say, three) address pointers stored at the due hash table location are incremented by the weight value of that word. The weighted, randomly projected word histogram obtained in the above way may be normalized optionally. The computing time needed to form the histograms in the above way is about 20 per cent of that of the usual matrix-product method. We have now some indication for the same speedup holding for larger maps, too.

References

[1] Salton G, McGill MJ. *Introduction to modern information retrieval.* McGraw-Hill, New York, 1983

[2] Kaski S. Data exploration using self-organizing maps. Acta Polytechnica Scandinavica, Mathematics, Computing and Management in Engineering Series No 82, 1997. Dr Tech Thesis, Helsinki University of Technology, Finland

[3] Kaski S. Dimensionality reduction by random mapping. In: Proc of IJCNN'98, Int Joint Conf on Neural Networks. IEEE Press, Piscataway, NJ, 1998, pp 413-418