Factor semantics for document retrieval: Project report

Rafael A. Calvo
Language Technology Institute, Carnegie Mellon University,
5000 Forbes Avenue, Pittsburgh PA 15213, USA
rafa@ifir.ifir.edu.ar

December 14, 1998

Abstract

Principal Component Analysis (PCA) is a useful technique for reducing the dimensionality of datasets, a paramount need in high dimensional term spaces. We study three neural networks with Hebbian-like learning that approximately produce the principal components of a document database. The explained variance of the solutions shows how much information the reduced space retains. The retrieval performance on the reduced space is compared with other methods: Smart, LSI and LLSF.

1 Information Retrieval

Several neural network models have recently been applied to dimensionality reduction tasks.

Information Retrieval (IR) is about finding documents relevant to an user's query, from within a set of documents.

Many IR models define a vector space where each document is represented. Standard IR techniques represent a document as a vector where each dimension corresponds to a distinct term. The value of this term is usually the occurrence of this term in the document, or function of it. This space is high dimensional (10,000-100,000) and with high correlations between terms. This fact has been useful since some of the correlations between terms are obvious and this redundancy can be indirectly reduced by using a thesaurus, term weighting, [9], and stemming [8].

Recently a number of researchers have developed other feature extraction techniques as a way of reducing the term space, Latent Semantic Indexing (LSI) [2], information gain [5, 13], mutual information and χ^2 statistics, and principal component analysis [13, 11]. Yang [13] has compared some of these methods for feature extraction as a preprocessing for classification on the Reuters and OHSUMED data bases. All these studies show that retrieval and classification performance can be improved using feature extraction techniques. Our aim is to study how neural networks with Hebbian learning can be used as linear feature extraction methods, that are fast, reliable and easy to interpret. The philosophy of these methods is equivalent to LSI, but the are considerable differences in the implementation.

Section 1.1 introduces standard term weighting and section 1.2 the Hebbian learning models. 2 describes the data set used and section 3 the compared methods Smart, Latent Semantic Indexing (LSI) and Linear Least Square Fit (LLSF). Section 4 describes the results and section 5 concludes. Section 6 talks about the implementation, and where to find the software.

1.1 The standard model

Document indexing and retrieval techniques have been around for many years and a set of established and well tested set of methods exists.

Many of these techniques are used in commercial retrieval systems. Many have also been extensively tested through the TREC project, where evaluations of different IR systems have been rigorously compared on very large data sets.

From the original documents the indexed terms are usually nouns and verbs. A stop word list is usually used to eliminate prepositions and conjunctions and a stemming algorithm is used to leave only stems (or roots) rather than full words (eg. singular/plural forms).

Once each document is represented in this term space, traditional IR seeks exact matches between words in queries and words in documents. However retrieval performance can be improved by weighting each term. There are three elements used in weighting documents:

When searching on data base O, for documents matching query q, the similarity of each document to the query is measured, using a dot product or cosine.

$$similarity(q, d_i) = cos(q, d_i)$$

This method makes the underlying assumption that each word subspace is orthogonal to any other word space. This assumption is obviously wrong, and produces two kinds of errors: synonymy and polysemy [2]. Synonymy because there are many ways to refer to the same object, in fact some experiments show that in only 10-20% of the cases two people use the same word to describe an object. Polysemy refers to the fact that most words have more than one meaning (homography).

1.2 Feature extraction techniques

Since the beginning of this century, several researchers [4, 7], have developed dimensionality reduction techniques; principal component analysis (PCA) is one of them (see [1] for a review). In mathematical terms, n correlated random variables are transformed into a set of $d \leqslant n$ uncorrelated variables. These uncorrelated variables are linear combinations of the original variables and can be used to express the data in a reduced form. Data modelling and pattern recognition are better able to work on this reduced form, it's also more efficient for storage and transmission.

Although only d of the n dimensions are used, we see that the complexity of matrix methods grow at least quadratically with the dimension of the data. This can render the matrix methods impractical when n is large, or when time is at a premium.

For this reason neural network models that perform PCA have recently been developed [3], most of them using Hebbian learning rules. In general these neural networks will contain processing units with forward connections given by matrix $\mathbf{W}$, where the column vectors $\mathbf{w}_i$ are the connections between the inputs $\mathbf{x}_k$ and the output y_i. The i^{th} weight, $\mathbf{w}_i$ is used to approximate the i^{th} eigenvector, $\boldsymbol{\alpha}_i$. The output units have a value given by:

$$y_i = \mathbf{w}_i^T \mathbf{x}_k$$

In some of Hebbian architectures there are also lateral connections between outputs, such that these outputs effect each other (see [3] for a full description).

In the k^{th} iteration, Hebbian learning rules update the weights using:

$$\mathbf{w}_i^{k+1} = \mathbf{w}_i^k + \Phi(y_i, \mathbf{x}_k) \qquad (1)$$

where Φ is some function of the inputs and outputs.

We compare, in this article, several adaptive PCA techniques. The Simple PCA (SPCA) [6] method that produces approximate solutions without the need for calculating a variance-covariance matrix and then diagonalizing it (as do most traditional methods) and does not depend on learning parameters (as other neural networks do). Most importantly, for high dimensional datasets, SPCA is faster than other existing techniques and is easy to implement as a batch or an adaptive technique.

Sanger's rule is a popular Hebbian technique for finding a principal component, and is representative of many of the approaches which have been studied. Its form for Φ is given by:

$$\Phi(y_i, \mathbf{x}_k) = \beta y_i (\mathbf{x}_k - \sum_{k=1}^{i} y_k W_{kj}) \qquad (2)$$

where β_k is a learning rate, $y_i \mathbf{x}_k$ is known as the Hebbian term, and $y_i y_i \mathbf{w}_i^k$ is an anti-Hebbian term. The Hebbian term is responsible for driving the weight $\mathbf{w}_i^k$ towards the eigenvector $\boldsymbol{\alpha}_i$, and the anti-Hebbian term is responsible for keeping the weights bounded. All Hebbian learning rules have the Hebbian term; they vary in the learning parameter, the anti-Hebbian term and the lateral connections matrix. It can be shown that under some reasonable conditions, that $\mathbf{w}_i^k \to \boldsymbol{\alpha}_i$ as $k \to \infty$ (see [3]).

The complexity of Hebbian rules is $O(dmn)$ for the determination of d principal components of m n-dimensional data. The constants depend on the time required until convergence, which in turn is dependant largely on the learning rate β_k.

SPCA and $SPCA_2$ are Hebbian-like methods and were described in [6]. They are data oriented methods, this means they do not start with the computation of the covariance matrix, to then diagonalize it. These methods do not require the tuning of learning parameters (a problem with Hebbian learning) and convergence is obtained with very few iterations.

$SPCA_2$ is equivalent to Oja's one unit learning rule with deflation.

SPCA does not necessarily add a Hebbian term (linear function of $y_i \mathbf{x}_k$), instead, SPCA considers other forms of Φ. In particular, we will show here the results for two different functions Φ_1 and Φ_2. As SPCA is intended to be a fast approximator for the principal components, we consider the threshold function:

$$\Phi_1(y_i, \mathbf{x}_k) = \begin{cases} \mathbf{x}_k & \text{if } y_i \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

We also consider a second function, Φ_2 which gives the Hebbian term.

$$\Phi_2(y_i, \mathbf{x}_k) = y_i \mathbf{x}_k$$

Each principal component direction found is normalized, in order to avoid divergence, so we do not use the anti-Hebbian term. Either of these algorithms may be implemented in an iterative or a batch mode. Initially $\mathbf{w}^0$ can be set to any vector. In a batch mode, the sum of $\Phi(y_i, \mathbf{x}_j)$ over all samples $\mathbf{x}_j$, is calculated where y_i is calculated relative to the old estimate $\mathbf{w}^k$. (For Φ_1 this is equivalent to summing those vectors positively correlated with $\mathbf{w}^k$.) This sum is then normalized to give the new estimate (equation 3).

$$\mathbf{a}^{k+1} = \frac{\sum_j \Phi(y_i, \mathbf{x}_j)}{\left\| \sum_j \Phi(y_i, \mathbf{x}_j) \right\|}\,; \qquad y_i = \left(\mathbf{a}^k\right)^T \mathbf{x}_j \tag{3}$$

In an iterative mode, updating is performed according to equation 1, and the estimate is normalized once every pass. Unfortunately with Φ_2, the terms $\mathbf{a}^k$ and y can diverge during the pass, if unconstrained. Hence Φ_2 is modified for the iterative rule to include a normalizing constant:

$$\Phi_2^{\text{iter.}}(y_i, \mathbf{x}_k) = \frac{1}{||\mathbf{a}^k||} y_i \mathbf{x}_k$$

So far the method of finding one principal component according to SPCA has been described. In order to find the next principal component, the effect of the first principal component is removed from the dataset, so as to avoid being found again. The removal of a component from the dataset, is called "deflation". This is performed by subtracting the component of each datum which is parallel to the principal component from that datum, leaving only an orthogonal component. That is, datum $\mathbf{x}_i$, is deflated by the unit principal component direction $\mathbf{w}$, according to:

$$\mathbf{x}_i' = \mathbf{x}_i - (\mathbf{a}^T \mathbf{x}_i)\mathbf{a}$$

The batch SPCA_1 algorithm is geometrically described in Figures 1 to 3.

The algorithmic complexity of the SPCA algorithms is $O(dmn)$ which is equal to the complexity of Hebbian learning algorithms, however each iteration of SPCA is faster and we experimentally show that the time taken until convergence is faster.

2 Data Set

MEDLARS is a monolingual corpus, with 1033 documents, 30 queries and human relevance judgements. The queries length varies between 2 to 62 words, with an average of 20. The number of relevant documents per query varies between 9 to 39 with an average of 23.

3 Methods

3.1 Smart

Smart is the oldest and most popular research tool in Information retrieval. It was designed and made publicly

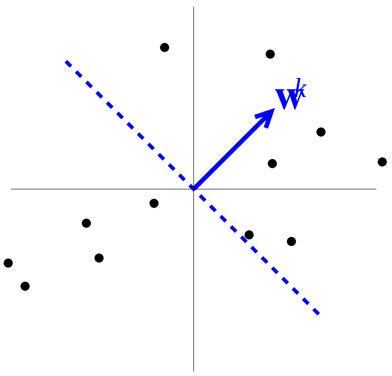

Figure 1: Data points (dots) with an approximate PC direction (an arrow), and its orthogonal plane (a dashed line).

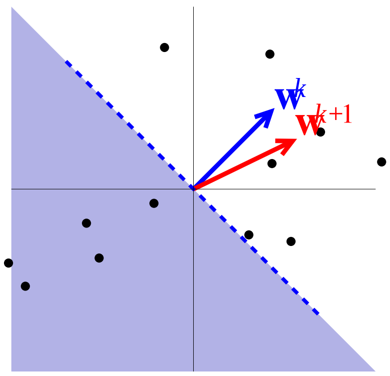

Figure 2: A better approximation to the PC direction is given by the normalized sum of the points "above" the plane.

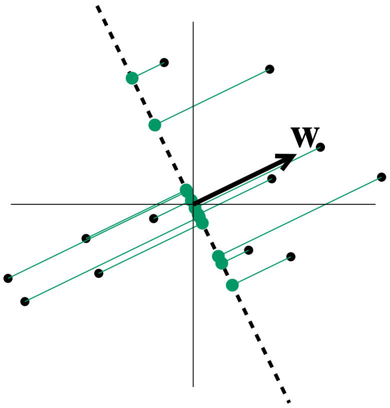

Figure 3: The data is deflated onto the plane orthogonal to the PC found. SPCA is then repeated on the deflated data.

available at Cornell University.

Smart [**?**] was runed using different weighting schemes.

3.2 Latent Semantic Indexing (LSI)

Latent Semantic Indexing (LSI) is a mathematical/statistical technique for extracting and representing the similarity of meaning of words and passages by analysis of large bodies of text. It uses singular value decomposition, a general form of factor analysis, to condense a very large matrix of document-terms data into a much smaller, but still large-typically 100-500 dimensional-representation (Deerwester, Dumais, Furnas, Landauer Harshman, 1990).

3.3 Linear Least square Fit (LLSF)

This method uses Linear Least square Fit approach to find a mapping function between the user queries and the documents. [12] gives a complete description.

4 Results

4.1 Explained variance

Traditionally, the first step in PCA is to estimate the covariance (sometimes correlation) matrix. The matrix is then diagonalized using some numeric algebra method like SVD. This diagonalisation is only a transformation that rotates the data, and as such it preserves the trace. The trace is equal to the total variance of the data and the explained variance of factor i is $\frac{\sigma_i^2}{\sigma^2}$, where σ^2 is the original total variance. The accumulated explained variance is shown in figures 7 and **??**.

The estimation of the eigenvectors and principal components -the projection of the data points in these directions- takes 3hrs for the 1033 factors.

4.2 Factor semantics

PCA has several advantages over other multidimensional techniques. It allows the representation of words and documents in the same space. And linear PCA solutions can be interpreted easily. Nonlinear PCA instead, is usually very hard [10]. Table 4.3 shows the terms with highest score on the first 3 factrors.

4.3 Retrieval Performance

Reducing the dimension of a data set doesn't usually solve any problem, in general its aim is to simplify it. After reducing the dimension we want to see how well other methods perform solving a retrieval task, in the reduced and the original data sets.

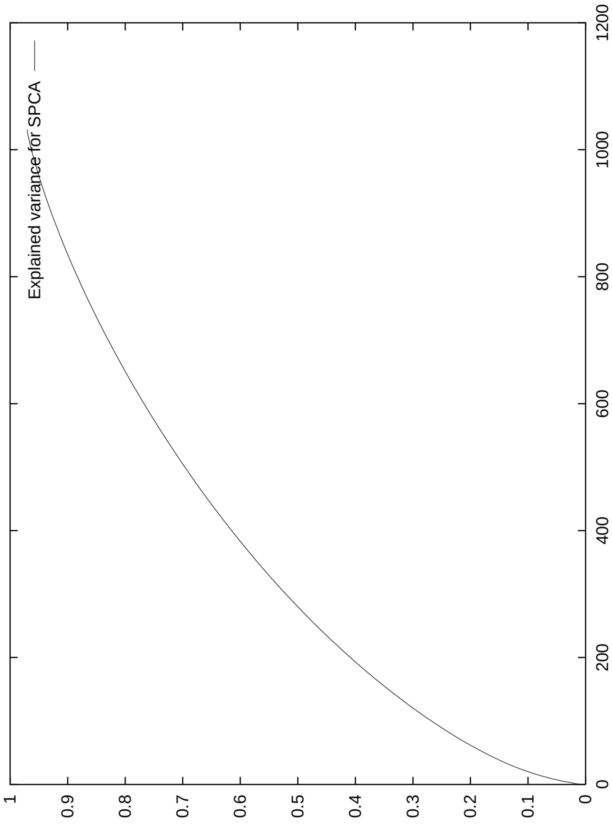

Figure 4: Explained variance and its error vs. factors for the SPCA method.

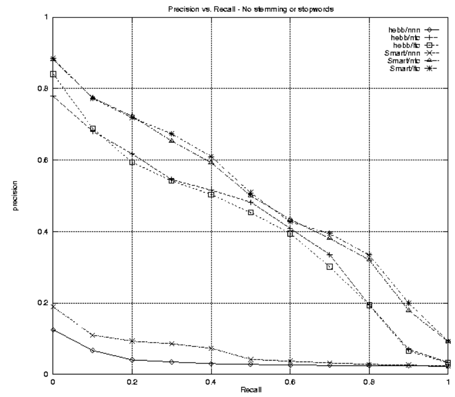

Figure 5: Precision/Recall for SPCA and Smart with different weighting schemes.

We compared the precision/recall curves and the 11-pt average precision for each method. SPCA is bellow any of the other methods, and possible reasons are discussed later.

5 Conclusions

1. Performance of Hebbian SPCA is not good enough. (Convergence?)

2. There is a big difference with LSI - LLSF. And between weighting schemes.

3. The interpretation of the factors is not clear enough for domain specific data

To Do

1. Understand the difference in the results with LSI

2. More efficient implementation. Try other algorithms

3. Try bigger data base (not domain specific?)

6 Implementation

Makefile.hebb runs everything.

smart2hebb, gets the document vectors from smart.

hebb was written in C++, at a time I was experimenting with literate programming, so it looks pretty much like a latex document with C++ code in the middle. I left all the C++ code in case you don't have noweb (the literate programming interpreter I used), you can run make on it, but do not clean (or you will need to get the code out of the original noweb files). It contains several Hebbian learning method: SPCA, $SPCA_2$, Oja's rule, Sanger's rule, and APEX. Also,

	11-pt precition
LSI-10	0.4787
LSI-50	0.7174
LSI-100	0.6803
LSI-200	0.6325
LSI-300	0.6084
LSI-600	0.5632
SPCA (300)	0.4374
LLSF (300)	0.6181
Smart	0.5391 height

Table 1:

Factor 1	Factor 2	Factor 3
cellś	ventriculo	cancer
ratten	acid	patients
acid	hypotherm	lung
dna	pressures	disease
len	oxygen	alveoli
antigens	blood	breast
tissue	aortico	fatty
proteinic	nickelous	tumor
cultures	plasma	pregnancy height

Table 2:

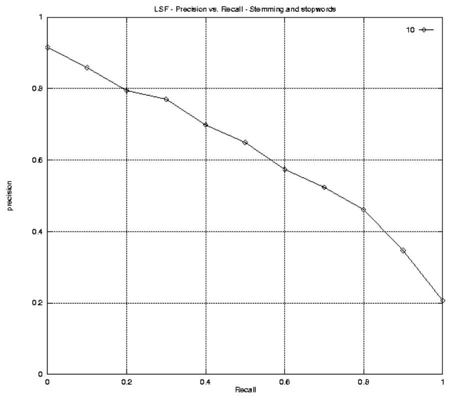

Figure 6: Precision/Recall for LLSF.

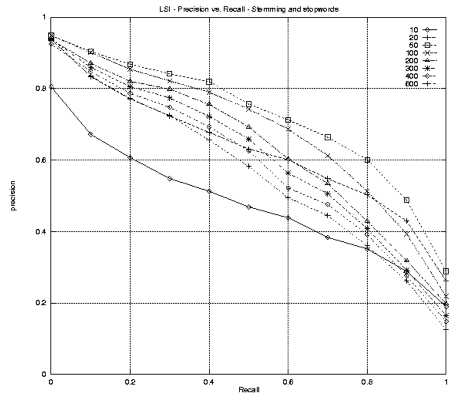

Figure 7: Precision/Recall for LSI

for smallish data abses you have Power method that performs SVD directly on the Covariance matrix.

After training the network a matrix containing the eigenvectors (or weights in NN terms), is saved, as well as the mean instance, the eigenvalues,...

These things are used by a set of perl scripts that can look at the factor semntics (interpret.pl) or run the queries on the reduced factor space.

References

[1] A. Basilevsky. *Statistical factor analysis and related methods: theory and applications*. Wiley, New York, 1994.

[2] S. Deerwester, S. Dumais, G. Furnas, TK Landauer, and R. Harshman. Indexing by latent semantic indexing. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[3] K. Diamantaras and Kung. *Principal component neural networks : theory and applications*. Wiley, New York, 1996.

[4] H. Hotelling. Analysis of a complex of statistical variables into principal components. *J. Educ. Psych.*, 24:417–441, 498–520, 1933.

[5] D.D. Lewis and M. Ringuette. Comparison of two learning algorithms for text categorization. In *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval (SDAIR'94)*, 1994.

[6] M. Partridge and R. A. Calvo. Fast dimensionality reduction and simple pca. *Intelligent Data Analysis*, 2(3), 1998 (to appear).

[7] K. Pearson. On lines and planes of closest fit to a system of points in space. *Philosophical Magazine*, 2:79–156, 1901.

[8] MF Porter. An algorithm for suffix striping. *Program*, 14:130–137, 1980.

[9] SE Robertson and K Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27:129–146, 1976.

[10] B. Schlkopf, A. Smola, and K Mlller. Nonlinear component analysis. *Neural Computation*, as a Kernel Eigenvalue Problem1996.

[11] H. Schultze, D.A. Hull, and Pedersen J.O. A comparison of classifiers and document repre-

sentations for the routing problem. In *Int. ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 22–34, 1995.

[12] Y. Yang and C.G. Chute. An example-based mapping method for text categorization and retrieval. *ACM Transaction on Information Systems (TOIS)*, pages 253–277, 1994.

[13] Y. Yang and J.P. Pedersen. Feature selection in statistical learning of text categorization. In *The Fourteenth International Conference on Machine Learning*, pages 412–420, 1997.