Extracting Distributed Representations of Concepts and Relations from Positive and Negative Propositions

Alberto Paccanaro, Geoffrey E. Hinton
Gatsby Computational Neuroscience Unit
University College London, 17 Queen Square, London WC1N 3AR, UK
{alberto,hinton}@gatsby.ucl.ac.uk

Abstract

Linear Relational Embedding (LRE) was introduced (Paccanaro and Hinton, 1999) as a means of extracting a distributed representation of concepts from relational data. The original formulation cannot use negative information and cannot properly handle data in which there are multiple correct answers. In this paper we propose an extended formulation of LRE that solves both these problems. We present results in two simple domains, which show that learning leads to good generalization.

1 Introduction

The problem that we are trying to solve, can be expressed as follows:

Let our universe of discourse consist of concepts and binary relations, and our data consists of triplets of the form `(concept1, relation, concept2)`. *We would like to learn distributed representations of the concepts and the relations that allow good generalization.*

We say that the system has good generalization if, after being shown only some of the possible triplets, it is able to complete any triplet, i.e. answer correctly the third element of any triplet, given the first two. In order to achieve this, such a system must come up with a sensible distributed representation of the objects and the relations, a representation which captures some of their semantic features.

There have been several attempts to tackle this problem. Multidimensional Scaling (Kruskal, 1964; Young and Hamer, 1987) attempts to find a representation of the concepts as vectors in multi-dimensional space, such that similar concepts are mapped onto vectors which are close to each other. The biggest limitation of this technique is that it only deals with the relation of *similarity*, and therefore it cannot take into account other relations which could be present in the data.

Latent Semantic Analysis (LSA) (Deerwester et al., 1990; Landauer and Dumais, 1997; Landauer et al., 1998) assumes that the similarity among two concepts is reflected in the way in which they co-occur across contexts. Again, this technique is limited by the fact that it only deals with the relation of *similarity*, now expressed in the form of co-occurrence.

Another attempt was by Hinton (1986) , where he showed how the distributed patterns of activities in a multi-layer perceptron network trained using the backpropagation algorithm (Rumelhart et al., 1986), could make explicit the semantic features of concepts and relations present in the data. Unfortunately, the system had problems in generalizing when many triplets were missing from the training set. Moreover there was no guarantee that the semantic features extracted for a given concept would be the same independent of the position of the concept within the triplet.

The original version of LRE (Paccanaro and Hinton, 1999) represents concepts using n-dimensional vectors, relations as $(n \times n)$ matrices, and the operation of applying `relation` to `concept1` in order to obtain `concept2` as a matrix-vector multiplication. The goal of learning is to find suitable vectors and matrices such that for each triplet, `concept2` is the vector closest to the result obtained by multiplying together `relation` and `concept1`. This is achieved by maximizing an appropriate discriminative goodness function (see eq. 1). Learning is fast and the system usually reaches a configuration which gives good generalization. There are some shortcomings however, and to explain them better we shall introduce two toy problems with which we are going to play later.

The number problem: the concepts are integer numbers and the relations are operations among numbers. In the examples presented here we shall use integers in the set $N = [1 \ldots 10]$, while the set of operations is

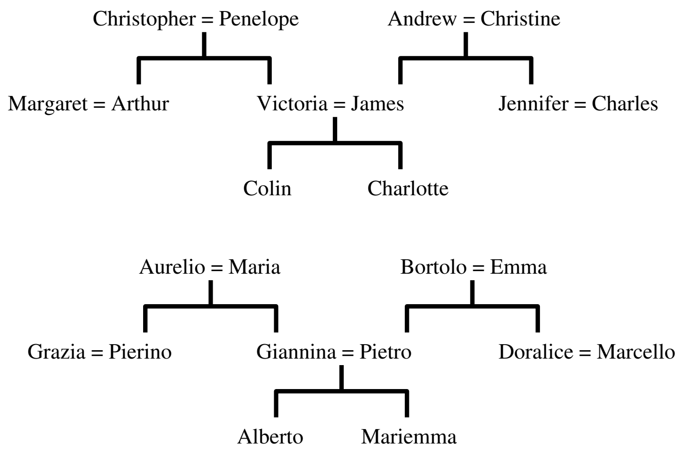

Figure 1: Two isomorphic family trees. The symbol "=" means "married to"

$O = \{+1, -1, +2, -2, +3, -3, +4, -4, +0\}$. Our data is constituted of all the possible triplets $(\mathtt{num}_1, \mathtt{op}, \mathtt{num}_2)$ where $\mathtt{num}_1, \mathtt{num}_2 \in N$, $\mathtt{op} \in O$, and $\mathtt{num}_2$ is the result of applying operation op to number $\mathtt{num}_1$, such as $\{(1, +1, 2), (4, +3, 7), \ldots\}$.

The family tree problem: this problem is the one used by Hinton (1986). Figure 1 shows two family trees. All the information in these trees can be represented in simple propositions of the form `(person1, relation, person2)`. Using the relations `father, mother, husband, wife, son, daughter, uncle, aunt, brother, sister, nephew, niece` there are 112 of such triplets in the two trees.

Now let us analyze two shortcomings in the original LRE formulation. First, it cannot deal with incomplete data, since it has no way to indicate that *it does not know* how to complete a triplet. For example in the number problem, when asked to complete the triplet (9, +3, ?) it will always return as its answer the vector which happens to be the closest to the result of multiplying the matrix representing the relation +3 and the vector representing number 9. In this case the appropriate answer would instead be something like *"I do not know"* or *"I do not know the name of that concept"* since number 12 does not belong to the universe of discourse. The same thing happens if we ask *"who is the father of Christopher"* in the family tree problem: the system always comes up with an answer from among its familiar concepts.

The second shortcoming in the original formulation of LRE appears in those cases in which a triplet has more than one correct completion (i.e. when the data contains triplets which differ only in the third element). In these cases, the system cannot clearly indicate how many correct answers there are and which ones they are. We do find that the answers are correctly ordered. For example, the two vectors closest to the result of multiplying matrix `aunt` and vector `Colin` are `Margaret` and `Jennifer`, but the system has no way to indicate that they are *both* correct, and that they are the *only* correct answers.

In the next section we describe a new version of LRE that is able to solve both these problems.

2 An improved version of Linear Relational Embedding

Let us assume that our data contains N distinct concepts and M binary relations. As before, we decide to represent each concept with an n-dimensional vector, and each relation with an $(n \times n)$ matrix. We call $\mathcal{V} = \{\vec{v}_1, ..., \vec{v}_N\}$ the set of vectors, and $\mathcal{R} = \{R_1, ..., R_M\}$ the set of matrices. The operation that relates a pair $\{\vec{v}_i, R_j\}$ to a vector $\vec{v}_k$ is the matrix-vector multiplication, $R_j \cdot \vec{v}_i$.

Let $\mathcal{P} = \{\vec{v}_i^p, R_j^p, \vec{v}_k^p\}_{p=1}^P$ indicate the set of all the triplets, where $\vec{v}_i, \vec{v}_k \in \mathcal{V}$ and $R_j \in \mathcal{R}$. Let $\mathcal{N} = \{\vec{v}_s^q, R_t^q, \perp\}_{q=1}^Q$ indicate the set of all the triplets, where $\vec{v}_s \in \mathcal{V}$, $R_t \in \mathcal{R}$ and $\perp$ indicates the fact that the result of applying relation R_t to $\vec{v}_s$ does not belong to $\mathcal{V}$. In other words, $\mathcal{P}$ represents *positive* information, which relates concepts in the dataset, while $\mathcal{N}$ represents *negative* information, which says that the correct completion of a triplet `(concept1, relation, ?)` is *not* a concept among the ones available in the dataset.

In the original version of LRE, the goal of learning was to find suitable vectors and matrices such that for each triplet $\{\vec{v}_i^p, R_j^p, \vec{v}_k^p\} \in \mathcal{P}$, $\vec{v}_k^p$ is the vector closest to $R_j^p \cdot \vec{v}_i^p$, while $\mathcal{N}$ was not considered. This version of LRE employed N distinct Gaussians, each centered on a vector representing a concept, and with standard deviation equal to $1/\sqrt{2}$; for each triplet in $\mathcal{P}$ the probability that the vector $R_j^p \cdot \vec{v}_i^p$ was generated from the Gaussian centered on $\vec{v}_k^p$ was maximized, while the probability that it came from any other Gaussian was minimized. This was achieved by maximizing a discriminative goodness function:

$$G = \sum_{p=1}^{P} \log \frac{e^{-|R_j^p \cdot \vec{v}_i^p - \vec{v}_k^p|^2}}{\sum_{\vec{v}_h \in \mathcal{V}} e^{-|R_j^p \cdot \vec{v}_i^p - \vec{v}_h|^2}} - \frac{1}{2}\lambda_v \sum_{j=1}^{N}\sum_{i=1}^{n} ({v_i}^{(j)})^2 - \frac{1}{2}\lambda_m \sum_{k=1}^{M}\sum_{i=1}^{n}\sum_{j=1}^{n} ({m_{i,j}}^{(k)})^2 \tag{1}$$

where ${v_i}^{(j)}$ indicates the component i of vector j, ${m_{i,j}}^{(k)}$ indicates the component (i,j) of matrix k, and λ_v, λ_m are (weight decay) parameters (see Paccanaro and Hinton (1999) for details).

In this paper we introduce a version of LRE in which for each relation R we have a mixture of N Gaussians and an un-normalized Uniform distribution. Each of the Gaussians is centered on a vector representing a concept, and has temperature T_R (i.e. standard deviation equal to $\sqrt{T_R/2}$). The Uniform distribution has value $\frac{1}{2\pi\sqrt{T_R/2}} \exp(-r_R^2/T_R)$. Now we can use both positive and negative information to learn a distributed representation of the data. For each triplet in $\mathcal{P}$ we maximize the probability that the vector $R_j^p \cdot \vec{v}_i^p$ is generated from the Gaussian centered on $\vec{v}_k^p$ while minimizing the probability that it comes from any other Gaussian or the Uniform distribution. At the same time, for each triplet in $\mathcal{N}$ we maximize the probability that the vector $R_j^q \cdot \vec{v}_i^q$ is generated from the Uniform distribution while minimizing the probability that it comes from any of the Gaussians.

The discriminative goodness function becomes:

$$G = \sum_{q=1}^{Q} \log \frac{\exp(-\frac{r_R^2}{T_R})}{\exp(-\frac{r_R^2}{T_R}) + \sum_{\vec{v}_h \in \mathcal{V}} \exp(-\frac{|R_j^q \cdot \vec{v}_i^q - \vec{v}_h|^2}{T_R})} + \sum_{p=1}^{P} \frac{1}{C_p} \cdot \log \frac{\exp(-\frac{|R_j^p \cdot \vec{v}_i^p - \vec{v}_k^p|^2}{T_R})}{\exp(-\frac{r_R^2}{T_R}) + \sum_{\vec{v}_h \in \mathcal{V}} \exp(-\frac{|R_j^p \cdot \vec{v}_i^p - \vec{v}_h|^2}{T_R})} \tag{2}$$

where C_p is the number of triplets in $\mathcal{P}$ having the first two elements equal to the ones of p, but differing in the third one. To understand why we need to introduce this factor, let us consider a set of C triplets, each having the same first two elements, $\vec{a}$ and R, but differing in the third element, which we shall call $\vec{b}_i$ with $i = 1 \dots C$. We would like our system to assign equal probability to each of the correct answers, and therefore the discrete probability distribution that we want to approximate can be written as:

$$\mathcal{P}_{\vec{x}} = \frac{1}{C} \sum_{i=1}^{C} \delta(\vec{b}_i - \vec{x}) \tag{3}$$

where δ is the discrete delta function and $\vec{x}$ ranges over the vectors in $\mathcal{V}$. Our system implements the discrete distribution

$$\mathcal{Q}_{\vec{x}} = \frac{1}{Z} \exp(-\frac{|R \cdot \vec{a} - \vec{x}|^2}{T_R}) \tag{4}$$

where

$$Z = \exp(-\frac{r_R^2}{T_R}) + \sum_{\vec{v}_h \in \mathcal{V}} \exp(-\frac{|R \cdot \vec{a} - \vec{v}_h|^2}{T_R}) \tag{5}$$

is the normalization factor. The Kullback-Leibler divergence between $\mathcal{P}_{\vec{x}}$ and $\mathcal{Q}_{\vec{x}}$, can be written as:

$$KL(\mathcal{P}\|\mathcal{Q}) = \sum_{\vec{x}} \mathcal{P}_{\vec{x}} \cdot \log \frac{\mathcal{P}_{\vec{x}}}{\mathcal{Q}_{\vec{x}}} = \sum_{\vec{x}} \frac{1}{C} \sum_{i=1}^{C} \delta(\vec{b}_i - \vec{x}) \cdot \log \frac{\frac{1}{C}\sum_{i=1}^{C}\delta(\vec{b}_i - \vec{x})}{\frac{1}{Z}\exp(-\frac{|R\cdot\vec{a}-\vec{x}|^2}{T_R})} \tag{6}$$

Thus, minimizing $KL(\mathcal{P}\|\mathcal{Q})$ amounts to minimizing:

$$-\sum_{\vec{u}} \frac{1}{C} \cdot \log(\frac{1}{Z} \exp(-\frac{|R \cdot \vec{a} - \vec{u}|^2}{T_R})) \tag{7}$$

for every $\vec{u}$ that is a solution to the triplet, which is exactly what we do when we maximize eq.2.

In this version of LRE, the Uniform distribution is taking care of the "don't know" symbol $\perp$ and is competing with all the other Gaussians, each of which is representing a vector constituting an answer in our dataset. The Uniform distribution has the value that a Gaussian with temperature T_R would have at a distance r_R from the mean. This is equivalent to saying that, given the point $R_j^p \cdot \vec{v}_i^p$, *all* and *only* the vectors representing correct answers must be found within an n-dimensional sphere of radius r_R centered at that point. In this way the system can handle both incomplete information and multiple correct answers: if no vector is inside the sphere we interpret this as the system answering "I don't know", while if a triplet can be completed in more than one way, all correct completions should be inside the sphere.

During our experiments, we maximized G using gradient ascent. Since temperatures must be constrained to be positive, they were learned using unconstrained optimization in the $\log$ domain. Learning is fast, though it is more prone to get stuck in local optima than the original version of LRE, so the parameters of the learning algorithm need to be tuned more carefully; on the other hand the problem that we are trying to solve is more complex. One effective method of performing the optimization is conjugate gradient. We have also developed an alternative optimization method which is faster and less likely to get trapped in local optima when the task is difficult. The objective function is modified to include a temperature that is annealed during the search. This method uses a line search in the direction of steepest ascent of the modified objective function.

3 Results

This section reports some of the results we obtained applying the new version of LRE to the *Number Problem* and *Family Tree Problem* presented earlier. After training the system, we presented it with each possible question and checked if it would give us the correct answers i.e. we checked that for any pair $(\vec{v}_i, R_j)$ we would obtain all and only the correct completions or the "don't know" answer. We looked at how well the system learned the whole set of data and how well it generalized to unseen triplets, after being trained on a subset of them.

Number Problem - LRE is able to solve the problem using two dimensions. To understand the solution it finds, let us look at figure 2a, which shows the vectors after learning together with the answers given by the system to each question $(\vec{v}_i, R_j)$. The vectors are represented by points on the plane and we plotted a solid line connecting each vector representing a number to the one representing the next number. The dots in the figure represent the answers to questions in $\mathcal{P}$, while the crosses represent the result of multiplying $R_j \cdot \vec{v}_i$ if $(\vec{v}_i, R_j, \perp) \in \mathcal{N}$. The matrices representing the operations turn out to be (almost) orthogonal, with all the rows having (roughly) the same norm. Therefore each can be decomposed into a constant factor which multiplies an orthonormal matrix. Thus we understand how this solution works: applying a certain operation to a number amounts to rotating and scaling its vector, so that if falls close enough to the vector which is the correct solution.

This method is able to generalize well. We trained the system with just 70 triplets chosen randomly from the 90 in the training set, and yet achieved perfect results during testing on all the 90 cases.

Notice how the solid line connecting the vectors forms a spiral, and that the cross clusters are placed on the "continuation" of the spiral, along both directions. The system anticipates where the vectors for those numbers ought to be placed, if it had some information about them. To verify this, let us look at what happens if we take out the information about a particular number in our set, i.e. if we eliminate all the triplets in $\mathcal{N}$ containing that number, and we transform all the triplets in $\mathcal{P}$ containing that number into triplets in $\mathcal{N}$. Figure 2b shows the vectors which are found, together with the answers of the system, after we eliminated all the information about number 7 in the training data. Notice how a cluster of crosses indicates where number 7 should go. If we now add the information about number 7 to the training data and continue the training, we see that in very few iterations the vector representing number 7 is placed exactly where it is supposed to go (fig.2c). It was interesting to verify that only one triplet in $\mathcal{P}$ is infact enough for the system to correctly position number 7. Figure 2d shows the solution obtained starting from the solution shown in fig.2b and training using only triplet $(6, +1, 7)$.

Family Tree Problem - LRE is able to find a solution for the Family Tree Problem that satisfies all the 296 triplets (112 in $\mathcal{P}$ and 184 in $\mathcal{N}$) using five dimensions. Diagrams of the solution vectors representing each persons

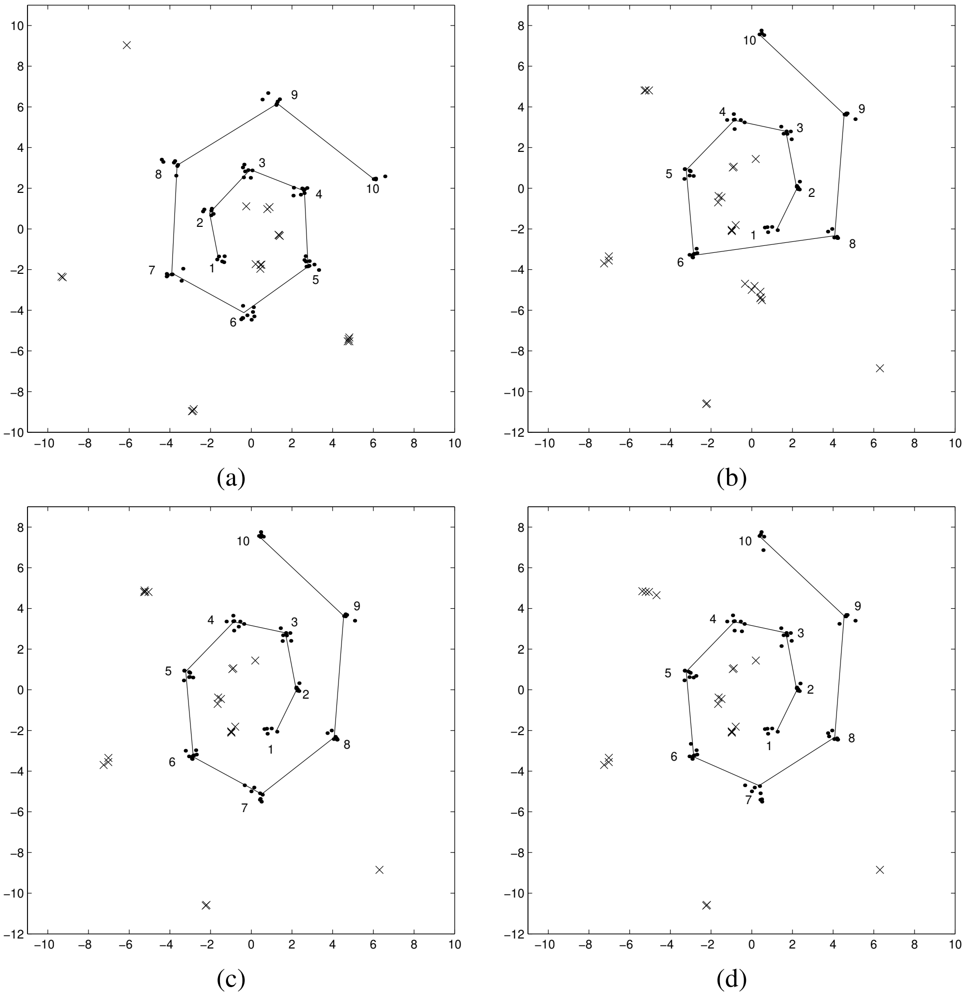

Figure 2: Vectors obtained after learning the number problem with numbers $[1 \ldots 10]$, operations $\{+1, -1, +2, -2, +3, -3, +4, -4, +0\}$. The dots are the result of the multiplication $R_j \cdot \vec{v}_i$ for each triplet $\{\vec{v}_i, R_j, \vec{v}_k\} \in \mathcal{P}$. The crosses are the result of the multiplication $R_j \cdot \vec{v}_i$ for each triplet $\{\vec{v}_s, R_t, \perp\} \in \mathcal{N}$. In all cases the system answered correctly to all the questions in the data set. (a) All the 90 triplets were used for training. (b) The information about number 7 was omitted. (c) The system was trained using the complete data set, starting from the configuration shown in (b). (d)The system was trained using only the triplet $(6, +1, 7)$ starting form the configuration shown in (b).

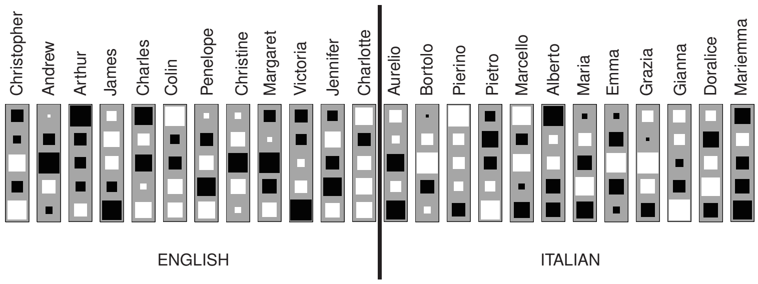

Figure 3: Diagrams of the vectors obtained for the family tree problem.

are shown in fig.3. White rectangles stand for positive components, black for negative ones, and the area of the rectangle encodes the magnitude of the component. Notice how the vectors representing the Italian people are perfectly symmetric to the ones representing their English counterparts.

Generalization is very good: the system is able to generalize perfectly when 26 cases are randomly omitted from the 296 triplets during training. It is thus able to spell out that `Charlotte` has *exactly* two uncles, and that they are `Arthur` and `Charles`, even if it had never been taught that `Charles` is an uncle of `Charlotte`. It is worth pointing out that in order to do this the system has to discover structure implicit in the data since correct generalization is not possible using only the pairwise correlations between terms in the triplets. It has to use the implicit information in the other triplets to figure out both the meaning of the relation `uncle` and the relative position of `Charlotte`, `Charles` and `Arthur` in the family tree, and then use this information to make the correct inference.

4 Conclusions

In this paper we introduced a refinement of LRE which can extract a distributed representation of concepts and relations from data making use of both positive and negative information. This new version also handles data in which there are multiple correct answers. The method is quite simple and generalizes well, even if learning is not as straightforward as in the original LRE case.

Its biggest limitation is that it deals only with binary relations. In order to overcome this problem we are going to use higher-dimensional tensors instead of vectors and matrices.

5 Acknowledgments

This research was founded by the Gatsby Charitable Foundation. We thank Peter Dayan, Zoubin Ghahramani, Carl van Vreeswijk, Hagai Attias and Marco Buiatti for helpful discussions.

References

Deerwester, S., Dumais, S. T., Furnas, G., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407.

Hinton, G. E. (1986). Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, pages 1–12. Erlbaum, NJ.

Kruskal, J. B. (1964). Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29, 1:1–27.

Landauer, T. K. and Dumais, S. T. (1997). A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104, 2:211–240.

Landauer, T. K., Laham, D., and Foltz, P. (1998). Learning human-like knowledge by singular value decomposition: A progress report. In Jordan, M. I., Kearns, M. J., and sara A. Solla, editors, *Advances in Neural Processing Information Systems 10*, pages 45–51. The MIT Press, Cambridge Massachusetts.

Paccanaro, A. and Hinton, G. E. (1999). Extracting distributed representation of concepts from relational data using linear relational embedding. *submitted to IEEE Trans. on Knowledge and Data Engineering - special issue on 'Connectionists Models for Learning in Structured Domains'*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. (1986). Learning internal representation by error propagation. In Rumelhart, D. E., McClelland, J. L., and the PDP research Group, editors, *Parallel Distributed Processing*, volume 1, pages 283–317. The MIT Press.

Young, F. W. and Hamer, R. M. (1987). *Multidimensional Scaling: History, Theory and Applications*. Hillsdale, NJ: Lawrence Erlbaum Associates, Publishers,.