Learning Distributed Representations by Mapping Concepts and Relations into a Linear Space

Alberto Paccanaro ALBERTO@GATSBY.UCL.AC.UK
Geoffrey E. Hinton HINTON@GATSBY.UCL.AC.UK
Gatsby Computational Neuroscience Unit, UCL, 17 Queen Square, London WC1N 3AR, UK

Abstract

Linear Relational Embedding is a method of learning a distributed representation of concepts from data consisting of binary relations between concepts. Concepts are represented as vectors, binary relations as matrices, and the operation of applying a relation to a concept as a matrix-vector multiplication that produces an approximation to the related concept. A representation for concepts and relations is learned by maximizing an appropriate discriminative goodness function using gradient ascent. On a task involving family relationships, learning is fast and leads to good generalization.

1. Introduction

Given data which consists of concepts and relations among concepts, our goal is to correctly predict unobserved instances of relationships between concepts. We do this by representing each concept as a learned vector in a Euclidean space and each relationship between concepts as a learned matrix that maps the first concept into an approximation to the second concept.

To illustrate the approach, we start with a very simple task which we call the *number problem*. The data consists of concepts which are integers and relations which are operations among integers. In the *modular* number problem the numbers are integers in the set $V = [0 \ldots m-1]$ and the set of operations is $\mathcal{R} = \{+1, -1, +2, -2, +3, -3, +4, -4, +0\}_m$, where the subscript indicates that the operations are performed modulo m. The data then consists of all or some of the triplets (num_1, op, num_2) where $num_1, num_2 \in V$, $op \in \mathcal{R}$, and num_2 is the result of applying operation op to number num_1; for example, for $m = 10$, $\{(1, +1, 2), (4, +3, 7), (9, +3, 2), \cdots\}$.

The main idea in Linear Relational Embedding (LRE) is to represent concepts using n-dimensional vectors, relations as $(n \times n)$ matrices, and the operation of applying a relation to a concept (to obtain another concept) as a matrix-vector multiplication[1]. Within this framework, one could easily hand-code a solution for the number problem with $n = 2$ and $m = 10$, where the numbers are represented by vectors having unit length and disposed as in fig.1a, while relations are represented by rotation matrices $R(\theta)$, where the rotation angle θ is a multiple of $2\pi/10$ (first row of table 1). The result of applying, for example, operation $+3$ to number 4, is obtained by multiplying the corresponding matrix and vector, which amounts to rotating the vector located at 144 degrees by 108 degrees, thus obtaining the vector at 252 degrees, which corresponds exactly to the vector representing the number 7.

In this paper, we show how LRE finds an equivalent solution, which is presented in fig. 1b and in the second row of table 1. LRE can find this solution when many of the triplets are omitted from the training set and once it has learned this way of representing the concepts and relationships it can complete all of the omitted triplets correctly. Moreover, LRE works well not only on toy problems like the one presented above, but also in other symbolic domains where the task of generalizing to unobserved triplets is non-trivial.

Our ultimate aim is to be able to take a large set of facts about a domain expressed as tuples of arbitrary symbols in a simple and rigid syntactic format and to be able to infer other "common-sense" facts without having any prior knowledge about the domain. For example, given a large set of facts about animals and their interactions with each other we would like to learn representations for specific animals and specific types of interaction that make the tuple *(cow eat*

[1]The data is represented using arbitrary symbols for the concepts and relationships. LRE maps these arbitrary symbols into non-arbitrary vectors and matrices which we call the representations of the concepts and relations.

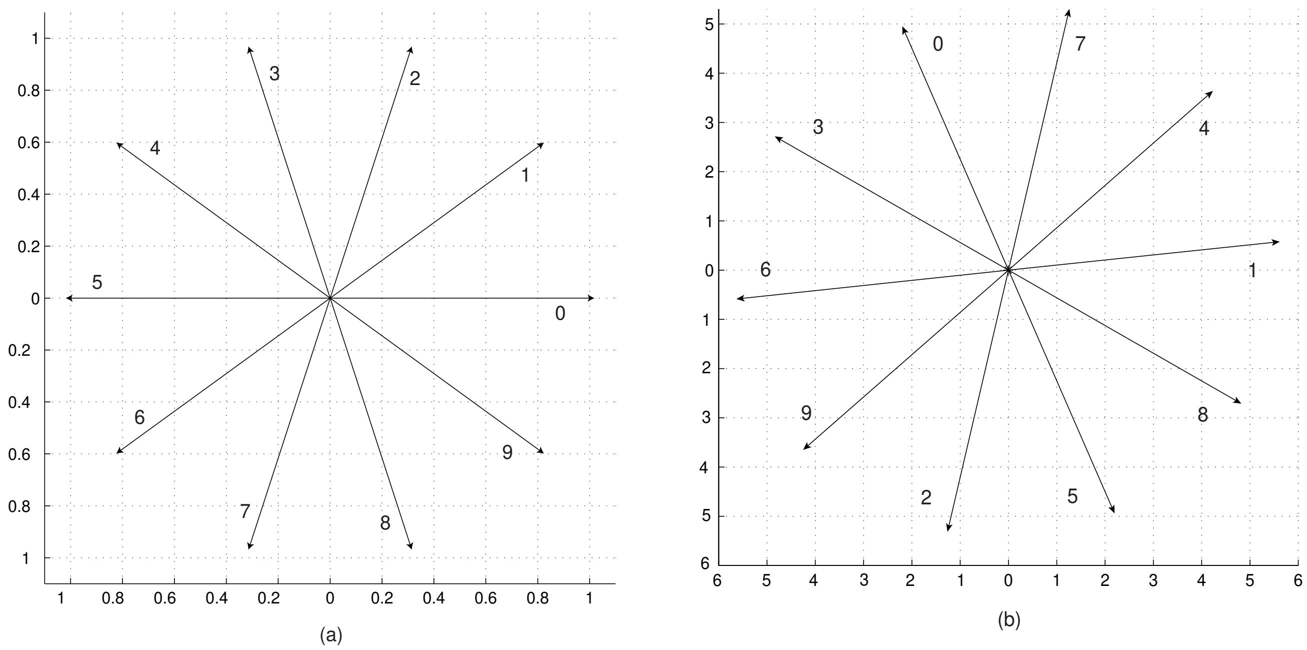

Figure 1. (a) vectors of the hand-coded solution for the number problem when $n = 2$ and $m = 9$ (b) vectors of the solution found by Linear Relational Embedding. Only 70 out of the 90 possible triplets were used for training. During testing, the system was able to correctly complete all the triplets.

Table 1. Angles, expressed in degrees, of the rotation matrices of the solutions to the number problem with $n = 2$ and $m = 10$, corresponding to the vectors in fig.1. The small errors of the LRE solution are due to the fact that only 70 triplets randomly chosen out of the 90 were used during training.

OPERATION	-4	-3	-2	-1	+0	+1	+2	+3	+4
Hand-coded Solution	-144	-108	-72	-36	0	36	72	108	144
LRE Solution	72.00	-35.97	-144.01	108.01	0.00	-108.02	144.02	35.98	-71.97

sheep) very implausible. Although we have not yet demonstrated a system that can do this for a large and diverse set of facts, LRE can already solve problems of this type in very limited domains.

Several methods already exist for learning sensible distributed representations from relational data[2]. Multidimensional Scaling (Kruskal, 1964; Young & Hamer, 1987) finds a representation of concepts as vectors in a multi-dimensional space, in such a way that the dissimilarity of two concepts is modeled by the Euclidean distance between their vectors. Unfortunately, dissimilarity is the only relationship used by multidimensional scaling so it cannot make use of the far more specific information about concepts contained in a triplet like "John is the father of Mary".

Latent Semantic Analysis (LSA) (Deerwester et al., 1990; Landauer & Dumais, 1997; Landauer et al., 1998) assumes that the meaning of a word is reflected in the way in which it co-occurs with other words. LSA finds features by performing singular value decomposition on a large matrix and taking the eigenvectors with the largest eigenvalues. Each row of the matrix corresponds to a paragraph of text and the entry in each column is the number of times a particular word occurs in the paragraph or a suitably transformed representation of this count. Each word can then be represented by its projection onto each of the learned features, and words with similar meanings will have similar projections. Again, LSA is unable to make use of the specific relational information in a triplet.

[2] We use the term "distributed representation" to stand for a particular kind of relationship between two descriptive languages. Individual symbols in one language correspond to conjunctions of symbols in the other language. For example, the concept represented by the symbol "Pat" in one language might be represented by the conjunction of the features "pedantic", "young", and "female" in the other language.

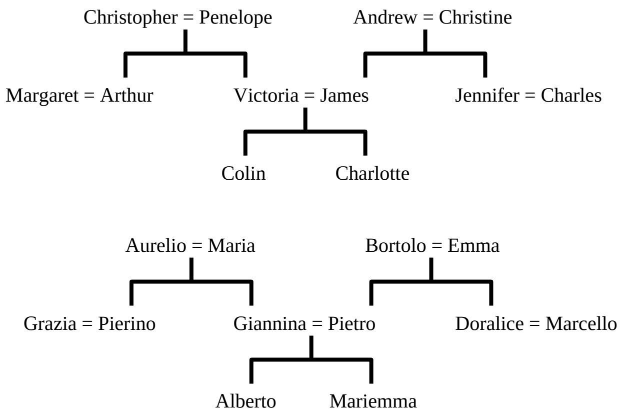

Figure 2. Two isomorphic family trees. The symbol "=" means "married to".

Hinton (1986) showed that a multilayer neural network trained using backpropagation (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. This was shown on a simple task called the *family tree problem*. In this problem, the data consists of people and relations among people belonging to two families, one Italian and one English, shown in figure 2. 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 such triplets in the two trees.

The next section presents the details of Linear Relational Embedding, while section 3 presents the results obtained using LRE on the number problem and the family tree problem, as well as the results obtained on a much larger version of the family tree problem that uses data from a real family tree. Finally section 4 concludes by indicating ways in which LRE could be extended.

2. Linear Relational Embedding

Let us assume that our data consists of C triplets (*concept1, relation, concept2*) containing N distinct concepts and M binary relations. As anticipated in section 1, the main idea of Linear Relational Embedding is to represent each concept with an n-dimensional vector, and each relation with an $(n \times n)$ matrix. We shall call: $V = \{\mathbf{v}_1, ..., \mathbf{v}_N\}$ the set of vectors, $\mathcal{R} = \{R_1, ..., R_M\}$ the set of matrices and $\mathcal{D} = \{(\mathbf{a}^c, R^c, \mathbf{b}^c)\}_{c=1}^{C}$ the set of all the triplets, where $\mathbf{a}^c, \mathbf{b}^c \in V$ and $R^c \in \mathcal{R}$. The operation that relates a pair $\{\mathbf{a}^c, R^c\}$ to a vector $\mathbf{b}^c$ is the matrix-vector multiplication, $R^c \cdot \mathbf{a}^c$, which produces an approximation to $\mathbf{b}^c$.

The goal of learning is to find suitable vectors and matrices such that for each triplet $(\mathbf{a}^c, R^c, \mathbf{b}^c) \in \mathcal{D}$, $\mathbf{b}^c$ is the vector closest to $R^c \cdot \mathbf{a}^c$. The obvious approach is to minimize the squared distance between $R^c \cdot \mathbf{a}^c$ and $\mathbf{b}^c$ but this is no good because it causes all of the vectors or matrices to collapse to zero. In addition to minimizing the squared distance to $\mathbf{b}^c$ we must also maximize the squared distances to the other concept vectors. This can be achieved by imagining that $R^c \cdot \mathbf{a}^c$ is a noisy version of one of the concept vectors and maximizing the probability that it is a noisy version of the correct answer, $\mathbf{b}^c$, rather than any of the other possibilities. We imagine that a concept has an average location in the space but that each "observation" of the concept is a noisy realization of this average location. If we assume spherical Gaussian noise with a variance of $1/2$ on each dimension, the probability that a realization of concept i would occur at $R^c \cdot \mathbf{a}^c$ is proportional to $\exp(-||R^c \cdot \mathbf{a}^c - \mathbf{v}_i||^2)$. So the posterior probability that $R^c \cdot \mathbf{a}^c$ matches concept $\mathbf{b}^c$ given that it must match one of the known concepts is:

$$\frac{e^{-||R^c \cdot \mathbf{a}^c - \mathbf{b}^c||^2}}{\sum_{\mathbf{v}_i \in V} e^{-||R^c \cdot \mathbf{a}^c - \mathbf{v}_i||^2}}$$

A discriminative goodness function that corresponds to the log probability of getting the right answer, summed over all training triplets is:

$$G = \sum_{c=1}^{C} \log \frac{e^{-||R^c \cdot \mathbf{a}^c - \mathbf{b}^c||^2}}{\sum_{\mathbf{v}_i \in V} e^{-||R^c \cdot \mathbf{a}^c - \mathbf{v}_i||^2}} \tag{1}$$

The goodness function, G, ensures that the learning does not stop when the concept closest to $R^c \cdot \mathbf{a}^c$ is the correct answer. Instead, the learning continues until the correct answer is several standard deviations closer than any other, if that is possible.

The results which we present in the next section were obtained by maximizing G using gradient ascent. All the vector and matrix components were updated simultaneously at each iteration. One effective method of performing the optimization is scaled conjugate gradient (Møller, 1993). Learning was fast, usually requiring only a few hundred updates and learning virtually ceased as the probability of the correct answer approached 1 for every data point. We have also developed an alternative optimization method which is less likely to get trapped in local optima when the task is difficult. The objective function is modified to include a temperature that divides the exponents in eq. 1. The temperature is annealed during the optimization. This method uses a line search in the direction of steepest ascent of the modified objective function. A small amount of weight decay helps to ensure that the exponents in eq. 1 do not cause numerical problems when the temperature becomes small.

In general, different initial configurations and optimization algorithms caused the system to arrive at different solutions, but these solutions were almost always equivalent in terms of generalization performance.

3. Results

We shall first present the results obtained applying LRE to the number problem and to the family tree problem. After learning a representation for matrices and vectors, we checked, for each triplet c, whether the vector with the smallest Euclidean distance from $R^c \cdot \mathbf{a}^c$ was indeed $\mathbf{b}^c$. We checked both how well the system learned the training set and how well it generalized to unseen triplets. Unless otherwise stated, in all the experiments we optimized the goodness function using scaled conjugate gradient. Two conditions had to be met simultaneously in order for the algorithm to terminate: the changes in the components of the vectors and matrices at two successive steps had to be less than 10^{-4} and the value of the objective function had to change by less than 10^{-8}. All the experiments presented here were repeated several times, starting from different initial conditions and randomly splitting training and test data[3]. In general the solutions found were equivalent in terms of generalization performance. The algorithm usually converged within a few hundred iterations, and rarely got stuck in poor local minima.

3.1 Results on the Number Problem

Let us consider the modular number problem which we saw in section 1. With numbers $[0 \dots 9]$ and operations $\{+1, -1, +2, -2, +3, -3, +4, -4, +0\}_{10}$, there exist 90 triplets *(num1, op, num2)*. LRE was able to learn all of them correctly using 2-dimensional vectors and matrices ($n = 2$). Figure 1 shows a typical solution that we obtained after training with 70 triplets randomly chosen out of the 90. The scaled conjugate gradient algorithm converged within the desired tolerance in 125 iterations which took 8.2 seconds running the Matlab code on a 667MHz alpha-EV6. We see that all the vectors have about the same length, and make an angle of about $2\pi/10$ with each other. The matrices turn out to be approximately orthogonal, with all their row and column vectors having about the same length. Therefore each can be approximately decomposed into a constant factor which multiplies an orthonormal matrix. The degrees of rotation of each orthonormal matrix are shown in the second row of table 1. The matrices' multiplicative factor causes the result of the rotation to be longer than the second vector of the triplet. Because the concept vectors lie at the vertices of a regular polygon centered at the origin, this lengthening increases the squared distance from the incorrect answers by more than it increases the squared distance from the correct answer, thus improving the discriminative goodness function in Eq. 1.

Let us now consider a *non-modular* version of the number problem with numbers $[1 \dots 50]$ and operations $\{+1, -1, +2, -2, +3, -3, +4, -4, +0\}$. When the result of the operation is outside $[1 \dots 50]$ the corresponding triplet is simply omitted from the data set. In two dimensions LRE was able to find the correct solution for all the 430 valid triplets of the problem, after training on 330 randomly chosen triplets for a few hundred iterations. Figure 3 shows a typical vector configuration after learning. For the non-modular number problem, LRE increases the separation between the numbers by using different lengths for the concept vectors so that the numbers lie on a spiral. In the figure we also indicated with a cross the result of multiplying $R \cdot \mathbf{a}$ when the result of the operation is outside $[1 \dots 50]$. Notice how the crosses are clustered, on the "ideal" continuation of the spiral - the answer to $49 + 3$ is located at almost exactly the same point as the answers to $48 + 4$, $50 + 2$, and so on.

Now consider the non-modular numbers problem with numbers $[1 \dots 50]$ and operations $\{+1, -1, +2, -2, +3, -3, +0, \div 3, \div 2, \times 2, \times 3\}$. When we tried to solve it in 2 dimensions LRE could not find a solution that satisfied all the triplets. Using gradient ascent to optimize the modified goodness function while annealing the temperature, LRE found a solution that gave the correct answer for all the addition and subtraction operations but the matrices representing multiplications and divisions mapped all vectors to the origin. In 3 dimensions, however, LRE is able to find a perfect solution in a few hundred iterations.

[3] In principle, if the random splitting caused all the triplets containing a certain concept or relation to be missing from the training set, then that concept or relation could not possibly be learned, and the system would perform poorly when tested on these triplets.

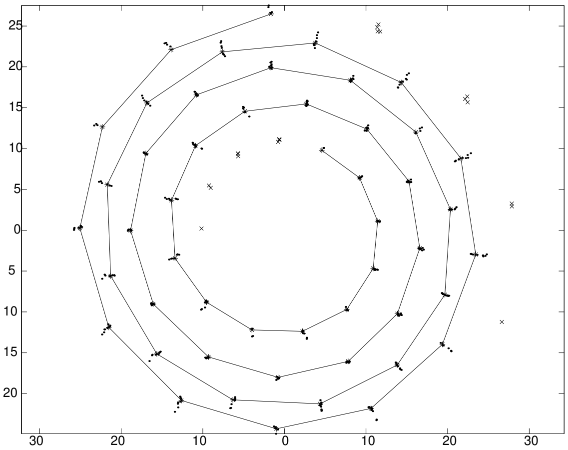

Figure 3. Vectors obtained after learning the non-modular number problem with numbers $[1 \ldots 50]$, operations $\{+1, -1, +2, -2, +3, -3, +4, -4, +0\}$ in two dimensions. Vector endpoints are marked with stars and a solid line connects the ones representing consecutive numbers. Smaller numbers are in the center of the spiral. The dots are the result of the multiplication $R^c \cdot \mathbf{a}^c$ for each triplet, c. The crosses are the result of the multiplication $R \cdot \mathbf{a}$ when the result of the operation is outside $[1 \ldots 50]$.

LRE is able to generalize well. The solutions shown in figure 1 and 3 answer correctly all the triplets in their problem databases even though they were trained on only 70/90 and 330/430 triplets respectively. It is worth pointing out that in order to do this the system had to discover structure implicit in the data.

3.2 Results on the Family Tree Problem

We used LRE in 3 dimensions on the *family tree problem.* When trained on all the data, LRE could correctly complete all 112 triplets and the resulting concept vectors are shown in figure 4. We can see that the Italian and English families are symmetric with respect to the origin and are linearly separable. When more than one answer was correct (as in the aunts of Colin) the two concept vectors corresponding to the two correct answers were always the two vectors closest to $R^c \cdot \mathbf{a}^c$.

LRE generalized perfectly when 12 triplets were held out during training. In particular, even when all the information on "who are the aunts of Colin" (i.e both triplets *(Colin, aunt, Jennifer)* and *(Colin, aunt, Margaret)*) was held out during training, the system was still able to answer correctly. Notice how, in order to do this, the system had first to use the implicit information in the other triplets to figure out both the meaning of the relation *aunt* and the relative position of *Colin*, *Margaret* and *Jennifer* in the tree, and then

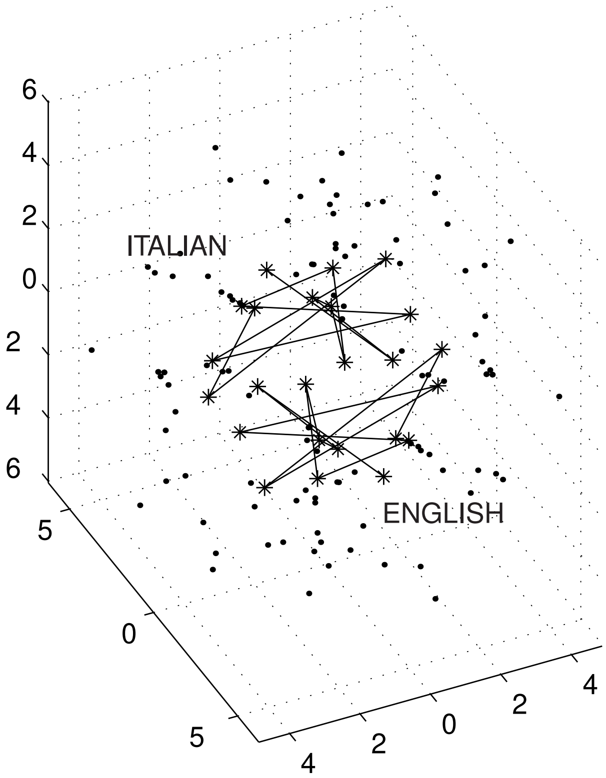

Figure 4. Layout of the vectors in 3D space obtained for the family tree problem. Vectors are represented by *, the ones in the same family tree are connected to each other. The dots are the result of the multiplication $R^c \cdot \mathbf{a}^c$ for each triplet, c. The solution shown here was obtained using gradient ascent to optimize the modified goodness function while the temperature was annealed.

use this information to make the correct inference.

The generalization achieved by LRE is much better than the neural networks of Hinton (1986) and O'Reilly (1996) which typically made one or two errors even when only 4 cases were held out during training.

3.3 Results on the Family Tree Problem with Real Data

We have used LRE to solve a much bigger family tree task. The tree is a branch of the real family tree of one of the authors containing 49 people over 5 generations. Using the 12 relations seen earlier it generates a data set of 644 triplets. The goodness function in Eq.1 was modified to behave more sensibly when there were many alternative correct completions of a triplet. In such cases each completion was given a weight inverse-

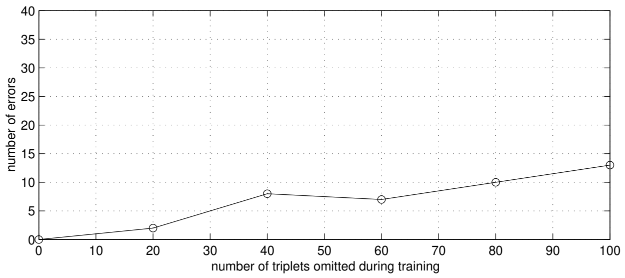

Figure 5. Plot of the errors made by the system when tested on the whole set of 644 triplets vs. the number of triplets which were omitted during training, using 10 dimensions. Omitted triplets were chosen randomly.

ly proportional to the number of alternatives. LRE was usually able to learn the tree in 6 dimensions using scaled conjugate gradient, taking about 4 minutes to run the Matlab code on a 667MHz alpha-EV6.

The generalization performance was very good. Figure 5 is the plot of the number of errors made by the system when tested on the whole data set after being trained on a subset of it using 10 dimensions. Triplets were held out randomly from the training set and the system was run for 5000 iterations, or until the convergence criteria were met. The results shown are the *median* of the number of errors over 3 different runs, since the system very occasionally failed to converge. We can see that the performance degrades slowly as an increasing number of triplets is omitted from the training data.

4. Discussion and further Developments

Linear Relational Embedding is a new method for discovering distributed representations of concepts and relations from data consisting of binary relations between concepts. On the task on which we tried it, it was able to learn sensible representations of the data, and this allowed it to generalize well.

In the *family tree* task, the great majority of the generalization errors were of a specific form. The system appears to believe that "brother of" means "son of parents of". It fails to model the extra restriction that people cannot be their own brother. This failure nicely illustrates the problems that arise when there is no explicit mechanism for variable binding.

A minor modification, which we have not tried yet, should allow the system to make use of negative data of the form "Christopher is not the father of Colin". This could be handled by minimizing G instead of maximizing it, while using Christopher as the "correct answer".

Another minor modification would be required to handle erroneous answers in the training data. Instead of assuming Gaussian noise on each concept vector we could assume a mixture of Gaussian and uniform noise. This would mean that every concept had some small probability of being chosen as the answer however far away it was from $R^c \cdot \mathbf{a}^c$. So if the allegedly correct answer, $\mathbf{b}^c$, was very far from $R^c \cdot \mathbf{a}^c$ the derivative of the goodness function would exert very little force on $\mathbf{b}^c$. LRE would then be unwilling to modify its representations to incorporate assertions that it found extremely implausible.

One limitation of the version of LRE presented here is that it always picks some answer to any question even if the correct answer is not one of the concepts presented during training. This limitation can be overcome by modifying the denominator of eq. 1 to include an extra learned term that is specific to each relationship:

$$G = \sum_{c=1}^{C} \log \frac{\exp(-||R^c \cdot \mathbf{a}^c - \mathbf{b}^c||^2)}{\exp(-r_{R^c}^2) + \sum_{\mathbf{v}_i \in V} \exp(-||R^c \cdot \mathbf{a}^c - \mathbf{v}_i||^2)} \tag{2}$$

where each relationship matrix, $R_i \in \mathcal{R}$, has its own extra parameter r_{R_i} so that the value of i specified by R^c determines r_{R^c}. The effect of this modification is that the probabilities of the known concepts add to less than 1. On training examples for which the correct answer is "unknown", the term $\exp(-r_{R^c}^2)$ is also used as the numerator. LRE is presumed to give the answer "don't know" if the most probable known concept is further than r_{R^c} away from $R^c \cdot \mathbf{a}^c$. Preliminary experiments with the non-modular number problems have been very successful. If, for example, the largest known number is 10, LRE learns to make the answer to $9+3$ be further than the threshold distance from all the known numbers. Moreover it locates the answer to $9+3$ at almost exactly the same point as the answers to $10+2$ and $8+4$. In a sense, it has constructed a new concept. See figure 6.

It is interesting to compare LRE to a system like FOIL (Quinlan, 1990). FOIL assumes that relational information can be represented as a set of predicates, i.e. mappings from k-tuples of concepts (constants) onto truth values. Given data consisting of these mappings for a particular set of concepts (and under the closed-world assumption) FOIL learns a definition of each predicate in terms of the other ones and itself. This is particularly interesting when the data contain a set

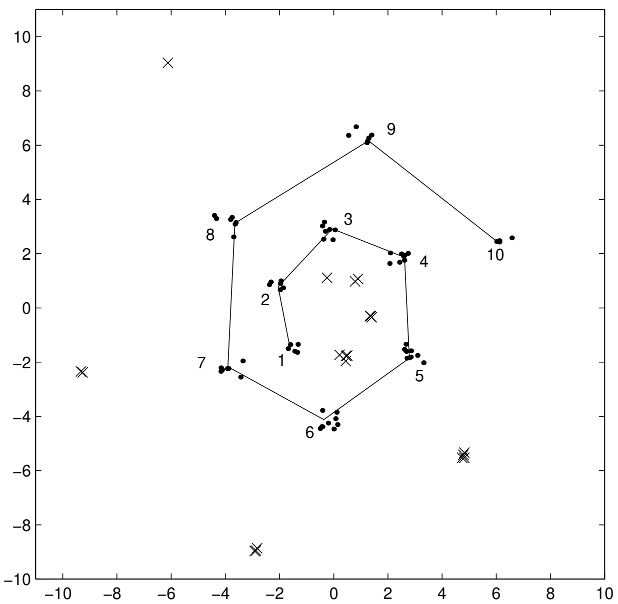

Figure 6. Vectors obtained after learning the number problem with numbers $[1 \dots 10]$, operations $\{+1, -1, +2, -2, +3, -3, +4, -4, +0\}$. The dots are the result of the multiplication $R^c \cdot \mathbf{a}^c$ for each triplet c such that the answer is among the known concepts. The crosses are the result of the multiplication $R^k \cdot \mathbf{a}^k$ for each triplet k such that the correct answer is "don't know". In all cases the system answered correctly to all the questions in the data set. All the triplets were used for training.

of basic predicates: FOIL is then able to learn other predicates that can be expressed as a combination of the basic ones. The definitions which are learned are very similar to Horn clauses and will then hold for any other set of data.

The approach taken by LRE is quite different, since it learns a representation for both relations and concepts for a given training set. Relations are now seen as functions rather than predicates: the result of applying a relation to a concept is another concept. Relations are represented as linear mappings between concepts in some vector space. The idea behind the scenes is that the distributed representation should make explicit the semantic features implicit in the data: each concept is a combination of semantic features, and relations compute an approximation to the features of one concept from the features of another.

Let us see how FOIL and LRE compare in terms of generalization performance and learning. Given a set of data, LRE is able to learn a representation that allows perfect generalization on that set of data with many more missing triplets than FOIL. [4] On the other hand, the definitions for the predicates which are found by FOIL are general, not specific to the set of data from which they are learned. Thus it is effortless to use them on new data, while LRE needs additional learning for the new concept vectors.

As regards learning, FOIL can learn the definition of a predicate only when it is possible to define that predicate in terms of the predicates available. Thus, if in the family tree example the data contains only the predicates: *parent*, *child* and *spouse*, FOIL could not possibly learn a definition for *mother* or *uncle* since the predicates available do not carry information about the sex of a person, which is fundamental in order to define *mother* and *uncle*. Similarly, in the number problem, if only relation +1 was given, there is nothing that FOIL could possibly learn. This is irrelevant to LRE, which is able to learn a predicate, no matter what the other predicates are.

It is interesting to consider how LRE could be improved by incorporating the idea of a set of basis relations. At present a separate matrix is needed for each relation and this requires a lot of parameters because the number of parameters in each matrix is the square of the dimensionality of the concept vector space. When there are many different relations it should be advantageous to model their matrices as combinations of a smaller set of learned basis matrices. It is fairly straightforward to modify LRE so that the relationship matrices are modeled as linear combinations of a learned set of basis matrices. This has some similarity to the work of Tenenbaum and Freeman (1996). It is much more difficult to see how LRE can be modified so that it learns basis matrices which are multiplied together to form the relationship matrices because the order of multiplication matters and orderings do not form a continuous space.

In this paper, we have assumed that the concepts and relations are presented as arbitrary symbols so there is no inherent constraint on the mapping from concepts to the vectors that represent them. LRE can also be applied when the "concepts" already have a rich and informative representation. Consider the task of mapping pre-segmented intensity images into the pose parameters of the object they contain. This mapping is non-linear because the average of two intensity images is not an image of an object at the average of the positions, orientations and scales of the objects in the two images. Suppose we have a discrete sequence

[4] The generalization results of LRE presented here on the Family Tree problem are better than the 78 out of 80 achieved by Quinlan (1990) on the same problem.

of images $I(1) \ldots I(t) \ldots I(T)$ of a stationary object taken with a moving camera and we know the camera motion $M(t, t+1)$ between each successive image pair.

In an appropriately parameterized space of pose parameters, the camera motion can be represented as a transformation matrix, $R(t, t+1)$, that converts one pose vector into the next:

$$R(t, t+1)v(t) \;=\; v(t+1) \tag{3}$$

The central assumption of LRE is therefore exactly satisfied by the representation we wish to learn. So it should be possible to learn a mapping from intensity images to pose vectors and from sensory representations of camera motions to transformation matrices by backpropagating the derivatives obtained from Eq. 1 through a non-linear function approximator such as a multilayer neural network. Preliminary simulations by Sam Roweis (personal communication) show that it is feasible to learn the mapping from preprocessed intensity images to pose vectors if the mapping from camera motions to the appropriate transformation matrices is already given.

Acknowledgments

The authors would like to thank Peter Dayan, Sam Roweis, Carl van Vreeswijk, Hagai Attias and Marco Buiatti for many useful discussions.

References

Deerwester, S., Dumais, S. T., Furnas, G., Landauer, T. K., & 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*, 1–12. NJ: Erlbaum.

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

Landauer, T. K., & 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., & Foltz, P. (1998). Learning human-like knowledge by singular value decomposition: A progress report. In M. I. Jordan, M. J. Kearns and S. A. Solla (Eds.), *Advances in neural processing information systems 10*, 45–51. Cambridge Massachusetts: The MIT Press.

Møller, M. (1993). A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, *6*, 525–533.

O'Reilly, R. C. (1996). *The leabra model of neural interactions and learning in the neocortex.* Doctoral dissertation, Department of Psychology, Carnegie Mellon University, Pittsburgh, PA.

Quinlan, J. R. (1990). Learning logical definitions from relations. *Machine Learning*, *5*, 239–266.

Rumelhart, D. E., Hinton, G. E., & Williams, R. (1986). Learning internal representation by error propagation. In D. E. Rumelhart, J. L. McClelland and the PDP research Group (Eds.), *Parallel distributed processing*, vol. 1, 283–317. The MIT Press.

Tenenbaum, J. B., & Freeman, W. T. (1996). Separating style and content. In M. C. Mozer, M. I. Jordan and T. Petsche (Eds.), *Advances in neural processing information systems 9*, 662–668. Cambridge Massachusetts: The MIT Press.

Young, F. W., & Hamer, R. M. (1987). *Multidimensional scaling: History, theory and applications.* Hillsdale, NJ: Lawrence Erlbaum Associates, Publishers.