

A brief survey of techniques for learning distributed representations

Tony A. Plate

September 30, 2003

This is an extract from Chapter 2 in the book “Holographic Reduced Representation: Distributed Representation for Cognitive Structures”, Tony A. Plate, CSLI Publications, 2003. This extract surveys research on learning distributed representations.

1 Learning distributed representations

One of the basic problems with distributed representations is coming up with the codes for the particular objects to be represented. A distributed representation that is useful for solving a particular problem is usually a re-representation of the raw input (possibly sensory data). It represents, in a salient manner, the *semantic* features of the input (i.e., features that are relevant to solving the problem). The raw inputs to a neural network often carry few or no clues, in terms of simple dot-product similarity of activity in the input units, as to which inputs should be treated as similar or identical, and which should be treated as different. For a feedforward neural network such as the one in Figure 1 to solve such a problem in a parsimonious fashion the inputs must be re-represented, in the hidden layers, in a manner such that semantically similar, but neurally dissimilar, inputs evoke neurally similar activation patterns in hidden layers. E.g., if input patterns A and B are neurally dissimilar (i.e., have low dot-product similarity) but should produce the same output from the network, then one way for a network to do that is to have the weights entering the hidden layer transform the neurally dissimilar input layer patterns A and B into the neurally similar hidden-layer patterns A' and B' .

The techniques used in the literature for developing distributed representations can be classified into five broad categories: (1) random; (2) hand constructed; (3) learned in a supervised neural network from examples of input-output mappings; and (4) learned in an unsupervised neural network from examples of typical inputs (with no outputs presented); and (5) derived mathematically from summary statistics of data. Unfortunately, there have not been any successful attempts to systematically learn, on a large scale, distributed representations for objects with complex structure.

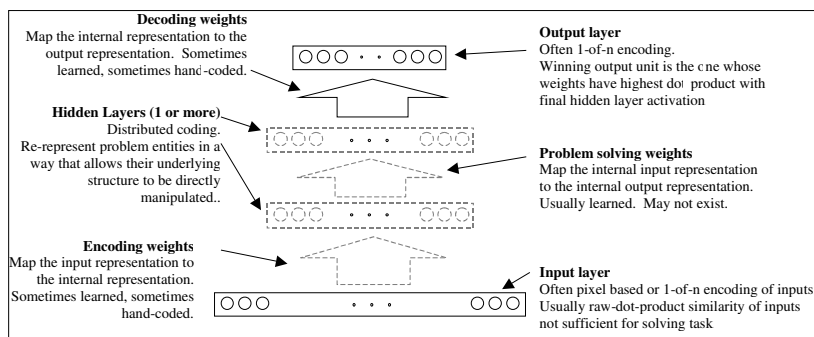


Figure 1: Levels of representation in an archetypal feedforward neural network.

1.1 Hand-constructed representations

Many researchers have constructed distributed representation by hand, often based on features of inputs that are known or suspected to be useful. For example, Rumelhart and McClelland's (1986) past-tense acquisition model used a hand-constructed distributed representation based on *Wickelfeatures*. To code a word in Wickelfeatures one first builds an unordered set of its Wickelphones, which are the triples of three consecutive phonemes in the word. For example, the word *came* contains three Wickelphones: #kA, kAm, and Am# (its three phonemes are k, A, and m, and # is used as a start and end symbol). The second step in representing a word was to turn on all the Wickelfeatures that responded to any of the Wickelphones in the word. Rumelhart and McClelland devised a set of 460 Wickelfeatures. Each responded to Wickelphones that had a certain subset of features. For example, one particular Wickelfeature responded to Wickelphones whose preceding context phoneme was an interrupted consonant, whose central phoneme was a vowel, and whose following context phoneme was an interrupted consonant. This Wickelfeature would respond to the Wickelphones kAm, bid, mAp, and many others. The set of 460 Wickelfeatures was chosen so that each Wickelphone would activate 16 Wickelfeatures. This meant that a word with three phonemes would activate 48 or fewer Wickelfeatures, because some Wickelphones might activate the same Wickelfeatures. Order of phonemes within a word is only represented implicitly in both Wickelphone and Wickelfeature representations. Although it is possible to find pairs of words that have the same set of Wickelphones (and/or Wickelfeatures), these are very unusual and did not occur in vocabulary of the model. The patterns of activation over the 460 Wickelfeatures formed a distributed coarse conjunctive code that captured much of the phonemic structure of words that was important to mapping the phonemic root form of an English verb to its phonemic past-tense form. The code was conjunctive because Wickelfeatures responded to conjunctions of features in adjacent phonemes, and was coarse because any particular Wickelfeature responded to many different Wickelphones.

In this book, I use of combination of random and manually constructed dis-

tributed representations in order to demonstrate the feasibility of HRRs. A concept (e.g., Fido) is represented by a small set of features (e.g., animal, mammal, dog, the name *Fido*). Each feature is represented by a vector of numbers randomly chosen from a Gaussian distribution. A concept is represented as the normalized superposition of its features. This gives representations for concepts that have many of the statistical properties of random Gaussian vectors, but which also reflect underlying similarities of the domain.

Developing a good distributed code by hand is often time consuming and is generally regarded as a stopgap measure suitable only for demonstrating the potential of a technique.

1.2 Representations derived from statistical summaries of data

Latent Semantic Analysis (LSA) (Deerwester et al., 1990) has achieved some remarkable successes in extracting the hidden structure of text documents. LSA can be viewed as a statistical technique for developing distributed representations for words and documents. LSA works by first constructing a term-document frequency matrix X for a collection of documents. The matrix has one row for each document and one column for each word in the collection. Element X_{ij} of the matrix is the count of the number of times word j appears in document i (or some monotonically-nondecreasing function of the count). This results in a very large matrix with many zeros. The next step in LSA is to compress the information in this matrix by applying Singular Value Decomposition and discarding all but the k (typically several hundred) most important dimensions. This results in a k -dimensional vector (pattern) representation for words and for documents. In general, similar words are represented by similar patterns, as are similar documents. The representation is dense, in that each element of the pattern usually takes on a real value from a Gaussian distribution. It also has superpositional properties, in that the pattern representing a document is approximately equal to the superposition of the patterns representing the words the document contains. By compressing the information in the term-document frequency matrix, LSA succeeds in discarding the noise regarding exactly which words were used to express the meanings of each document while retaining the underlying semantic similarity structure of the terms and documents. LSA has been applied to many tasks with a surprising degree of success, from document retrieval (by picking documents with patterns most closely matching the superposition of patterns for words in the query), to learning material from a textbook (it passed the exam!) to grading essays.

1.3 Backpropagation for learning representations

One of the most intriguing properties of the backpropagation algorithm applied to multilayer neural networks (such as the one in Figure 1) is its ability to learn distributed representations in its hidden layer that capture the underlying structure of the domain. Hinton's (1989) family trees network is an excellent example

of this. For training, this network was presented with an incomplete set of examples of person/relationship/person (e.g., Penelope/Daughter-of/Margaret) from two isomorphic English and Italian family trees, each involving 12 people from three generations. The objective was to see whether, using backpropagation, the network could learn appropriate features in the hidden layers that would allow it to generalize to the relations not included in the training set. There were two groups of input units, one for person (24 units) and one for relationship (12 units), and one group of output units for person (24 units). Each input and output group used a 1-of- n encoding, thus betraying no information about how a particular input should be treated. The network had multiple hidden layers, with fewer units than in the input layer in order to encourage a componential encoding, and was trained using backpropagation with weight decay. After training, the network was able to generalize to unseen relations. Examination of the representations the network developed for the hidden layers indicated that it had used units in the hidden layers to encode underlying important features such as sex, nationality and generation, which were not explicitly present in the input or output.

Other good demonstrations of backpropagation’s ability to learn useful internal distributed representations are Le Cun et al. (1989) (handwritten digit recognition), Elman (1991) (learning the structure of sentences generated by a simple artificial grammar for a subset of English), and Pollack (1990) (learning to represent hierarchical data structures).

1.4 Learning sparse representations

For some types of input with rich structure, e.g., natural images, useful distributed representations may be learned using unsupervised techniques. These typically work by learning an internal neural code from which the inputs can be reconstructed with a low degree of error. A typical architecture for such a net is shown in Figure 2. Early work with these types of autoencoder networks used the backpropagation algorithm and fewer hidden units than there were input units. This forced the network to learn a compact code to represent the inputs. Bourlard and Kamp (1988) and Baldi and Hornik (1989) showed that for a linear autoencoder network, training to minimize mean-square reconstruction error is equivalent to finding the first k principal components (where k is the number of hidden units). This makes linear autoencoders of limited interest, as it proves that the codes they will develop depend only on the pairwise covariance structure of the input. For example, they will develop the same codes for the two-dimensional data sets illustrated in Figure 3, which obviously have quite different underlying structure; (a) is Gaussian; (b) and (c) are non Gaussian.

For higher-dimensional data, e.g., images with binary pixels, this limitation is crippling. Even simple features such as lines and edges of varying contrast and orientation cannot be extracted with a linear mapping from input activations to hidden activations, which is the same as saying they cannot be identified through their pairwise covariance statistics.

If one imposes some constraint on the internal codes, i.e., the activations in

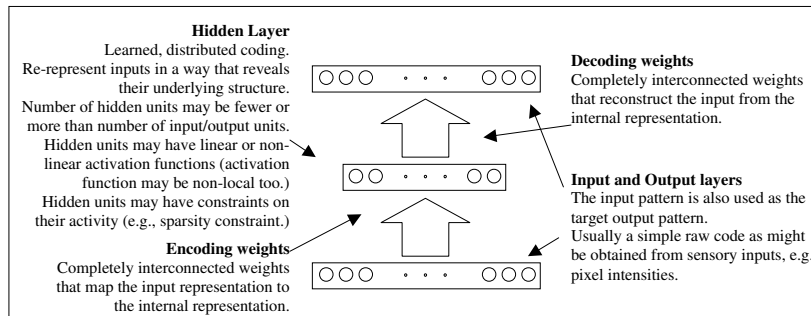


Figure 2: Typical autoencoder network. The network attempts to learn weights that map the input patterns an internal representation and then back again to patterns identical to the inputs.

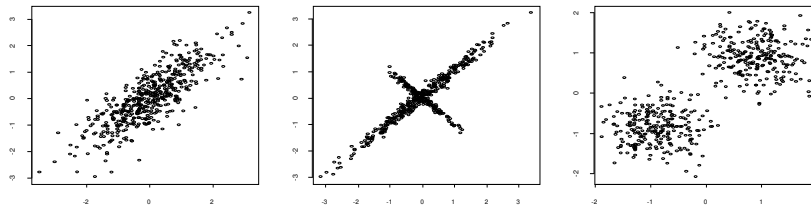


Figure 3: Three data sets that have the same pairwise statistics. Each example in each data set is a point in 2-d space. These have the same principal components, and a linear autoencoder network will learn identical encodings for them.

the hidden layer, and allows hidden-layer activations to be a nonlinear function of input activations, more interesting results can be obtained. One potentially useful constraint is that internal codes be sparse, i.e., that most activation values in the internal code be zero. This allows the use of overcomplete codes, in which there are more units in the internal code than are used in the input representation. Overcomplete codes require nonlinear computation of hidden-layer activation values from inputs (the reconstruction weights from the hidden layer to the outputs can still be linear). Olshausen and Field (1996) used a network in which the incoming weights to a hidden unit are the same as its outgoing weights, augmented with connections among hidden units and having a settling phase during which the hidden units compete for the right to represent the inputs. The settling phase is in effect a search for a set of hidden-unit activations that maximizes an objective function that trades off sparsity of hidden-unit activation against accuracy of output reconstruction. Outputs are computed from hidden-unit activations in a straightforward linear fashion. Their training algorithm slowly modified the hidden-to-output weights to better allow the network to find sparse hidden-layer activations that could accurately reconstruct a set of natural images. This resulted in an overcomplete set of image features (the

weights from one hidden unit to all the output units) that could describe the natural images. This set of image features had the characteristics of localized, oriented, bandpass receptive fields (i.e., receptive fields for short edges or bars of different sizes at different angles). The advantage of being an overcomplete set of image features is that it allows any particular image to be represented with a relatively small set of features well suited to representing that particular image. In this way it also captures some of the higher-order statistical structure of typical inputs that cannot be captured by a more compact principal components representation.

Olshausen and Field's (1996) approach is closely related to Independent Component Analysis (Hyvärinen and Oja, 2000). Hinton and Ghahramani (1997) and Zemel and Hinton (1995) explore other approaches to unsupervised learning of sparse distributed representations that efficiently encode inputs and capture the higher-order statistical structure in a set of inputs. Hinton and Ghahramani see sparse representations as a point on the spectrum between localist hidden-unit representations and principal-component hidden-unit representations. Localist hidden-unit representations have the advantage that they can capture any statistics of the data (each distinct input pattern can be assigned to a single hidden unit) and can represent multiple concepts simultaneously but have the twin disadvantages that they are a very inefficient representation and do not support generalization. Principal-component hidden-unit representations have the advantages that they are compact and efficient and support generalization, but have the disadvantages that they can only capture pairwise input statistics and can only represent a single concept at once. Sparse representations have the potential for blending the advantages of both approaches.

1.5 Kernel-based learning methods

Although kernel-based learning methods (Haussler, 1999, Collins and Duffy, 2002) are not commonly considered to use distributed representations, they have enough in common with distributed representations to deserve a mention in this section. In kernel-based methods, entities are represented by high-dimensional vectors. For example, a representation for sequences might use a vector that has an element for every possible subsequence. The dimensionality of the representational space is usually extremely large, and often infinite. Kernel-based methods use clever algorithms to avoid ever explicitly enumerating even the non-zero features of the representational space. The high-dimensional vectors that represent entities could be considered as a variety of sparse distributed representation. Learning with kernel-based methods involves computing statistics over examples and learning which features of the representational space are important. The relationship between kernel-based methods and HRRs is further explored in (Plate, 2003).

References

- Baldi, P. and K. Hornik. 1989. Neural networks and principal components analysis: Learning from examples without local minima. *Neural Networks* 2:53–58.
- Bourlard, H. and Y. Kamp. 1988. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics* 59:291–294.
- Collins, M. and N. Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 263–270.
- Deerwester, S., S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the Society for Information Science* 41(6):391–407.
- Elman, J. 1991. Distributed representations, simple recurrent networks and grammatical structure. *Machine Learning* 7(2&3):195–226.
- Haussler, D. 1999. Convolution kernels on discrete structures. Tech. Rep. UCS-CRL-99-10, UC Santa Cruz, Santa Cruz, CA. <http://www.cse.ucsc.edu/haussler>.
- Hinton, G. E. 1989. Learning distributed representations of concepts. In R. G. M. Morris, ed., *Parallel Distributed Processing: Implications for Psychology and Neurobiology*, pages 46–61. Oxford, UK: Oxford University Press. Reprint of a paper of the same title in Proceedings of the Eighth Annual Conference of the Cognitive Science, Hillsdale, NJ: Erlbaum.
- Hinton, G. E. and Z. Ghahramani. 1997. Generative models for discovering sparse distributed representations. *Philosophical Transactions of the Royal Society of London, B* 352:1177–1190.
- Hyvärinen, A. and E. Oja. 2000. Independent component analysis: Algorithms and applications. *Neural Networks* 13(4-5):411–430.
- Le Cun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. 1989. Back-propagation applied to handwritten zipcode recognition. *Neural Computation* 1(4):541–551.
- Olshausen, B. A. and D. J. Field. 1996. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature* 381:607–609.
- Plate, T. 2003. *Holographic Reduced Representation: Distributed Representation for Cognitive Structures*. CSLI Publications.
- Pollack, J. B. 1990. Recursive distributed representations. *Artificial Intelligence* 46(1-2):77–105.

- Rumelhart, D. E. and J. L. McClelland. 1986. On learning the past tenses of English verbs. In J. L. McClelland, D. E. Rumelhart, and the PDP Research Group, eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 2, pages 216–271. Cambridge, MA: MIT Press.
- Zemel, R. S. and G. Hinton. 1995. Learning population codes by minimizing description length. *Neural Computation* 7(3):549–564.