Letter

Representing sets as summed semantic vectors

Douglas Summers-Stay\textsuperscript{a,⁎}, Peter Sutor\textsuperscript{b}, Dandan Li\textsuperscript{a}

\textsuperscript{a} U.S. Army Research Laboratory, Adelphi, MD 20783, USA
\textsuperscript{b} University of Maryland, College Park, MD 20742, USA

\textbf{A R T I C L E  I N F O}

\textbf{Keywords:}
Hyperdimensional computing
Memory
Vector symbolic architectures

\textbf{A B S T R A C T}

Representing meaning in the form of high dimensional vectors is a common and powerful tool in biologically inspired architectures. While the meaning of a set of concepts can be summarized by taking a (possibly weighted) sum of their associated vectors, this has generally been treated as a one-way operation. In this paper we show how a technique built to aid sparse vector decomposition allows in many cases the exact recovery of the inputs and weights to such a sum, allowing a single vector to represent an entire set of vectors from a dictionary. We characterize the number of vectors that can be recovered under various conditions, and explore several ways such a tool can be used for vector-based reasoning.

\textbf{Introduction}

Representing concepts as high-dimensional vectors for reasoning is one of the most powerful ideas to come out of biologically-inspired computing. Unlike purely symbolic approaches, high-dimensional vectors have a structure that is rich enough to model the subtle and complex associations that concepts in memory have with one another, allowing generalization, analogy, and the combination of ideas.

There are various approaches to generate such high-dimensional vectors, including modeling semantic primitives with random vectors and using the weights of artificial neural networks. One of the most successful (in terms of immediate applications) sources of vectors has come out of the statistical linguistics community: creating vectors for words by ensuring that words which share a similar context in text corpora are mapped to similar vectors. Throughout this paper we will work with such distributional semantic vectors, but the ideas here can be applied to any architecture that uses high-dimensional vectors to represent meaning. As such we will usually refer to these vectors as representing words or concepts, but without intending to imply that distributional semantic vectors are themselves complete models of the meaning of words or concepts.

An operation common to many such approaches is vector addition or averaging to represent the meaning of a set of vectors. The meaning of this vector is in some way between the meanings associated with the vectors that make it up. For example, the word vector “astronomer” averaged with the word vector “physicist” will typically result in a vector close (in a suitable metric) to the word vector “astrophysicist.”.

However, this has generally been treated as a kind of summarization of meaning that inevitably loses track of the vectors that make up the sum. It would be convenient to be able to reverse this process: to take a vector which is a weighted sum of meaningful vectors, and find what the weights and inputs were. To a certain extent this is possible simply by looking for the nearest neighbors of the averaged vector in the dictionary. Because of inherent properties of high-dimensional Euclidean spaces and the small size of the dictionary compared to the size of the space, an averaged vector which is the mean of two word vectors will almost always be closer to these two word vectors than to any other word vector. But for more than a few words averaged together, the ability to recover words by looking at near neighbors fails rapidly. In this paper, we show how sparse vector decomposition techniques can be applied to overcome this limitation of summed vectors. This allows them to be directly useful for many kinds of tasks which would have been impractical in a vector-based representation previously. To our knowledge, this is the first use of such sparse decomposition techniques for recovering exact weights for individual word vectors from a vector sum.

What we show in this paper is the following surprising result: a single summed vector can be used to represent the entire set of words from which it was derived, as long as the set is smaller than the limits outlined in Section "Experiments: Recovery of weights and word vec-
tors”. Rather than thinking of a summed vector as representing a single meaning, we can think of it as picking out an entire set of meanings. For example, the Wikipedia page “list of fruits” lists 90 different fruits. A single 600-dimensional vector\(^1\) can be used to recall this entire list, provided each fruit is assigned a vector in the dictionary. Beyond that, each fruit on the list can be assigned a weight that it is possible to recover exactly.

Related work

Pentti Kanerva’s “hyperdimensional computing” is mainly concerned with large binary vectors, but also can include floating-point valued vectors. Kanerva (2009) Kanerva writes, “A set of vectors can be combined by componentwise addition, resulting in a vector of the same dimensionality...[T]he arithmetic-sum-vector is normalized, yielding a mean vector. It is this mean-vector that is usually meant when we speak of the sum of a set of vectors. The sum (and the mean) of random vectors has the following important property: it is similar to each of the vectors being added together. The similarity is very pronounced when only a few vectors are added and it plays a major role in artificial neural-net models. The sum-vector is a possible representation for the set that makes up the sum [...]".

To recover elements of the set, the paper recommends searching the dictionary for the nearest neighbor, subtracting it off, and searching for the nearest neighbor of the remainder. He notes that only small sets can be decomposed in this way.\(^2\) These ideas are explored further in the subfield called Vector Symbolic Architectures (Levy & Ross, 2008).

In the Hierarchical Temporal Memory model introduced by Jeff Hawkins, concepts are represented by sparse high-dimensional binary vectors, and the “union” of concepts is formed by applying the logical OR operation to these. If the logical AND operation is applied to a vector and this union and the original vector is returned, it belongs to the set with a probability that is easily calculated. Taylor (2016) This method gives no way of weighting concepts in the sum, however, and is not easily applied to floating-point valued vectors.

The notion of representing the meaning of a set by the simplex whose vertices are the set members in a semantic vector space was explored by Gardenfors (2000). A practical application of this with vectors derived from the weights of a neural network was discussed in Bechberger and Kuhnberger (2017).

Dominic Widdows and Trevor Cohen discuss a logic of projection operators in vector spaces (Widdows & Cohen, 2014). In their system, "Each projection operator projects onto a (linear) subspace; the conjunction of two operators projects onto the intersection of these subspaces; their disjunction projects onto the linear sum of these subspaces; and the negation is the projection onto the orthogonal complement." When these subspaces have the restriction that weights on semantic vectors must sum to one and be positive, they are cut down to simplices and the intersection and union operations he introduces become those discussed in this paper.

Our previous work discusses the possibility of representing a concept by a summed vector of synonyms of the concept (Summers-Stay, 2017a) as discussed in Section “Deductive logic”. In that work we used a fixed \(\lambda\) parameter for LASSO and didn’t attempt an exact recovery of the associated weights.

Decomposition of vector sums

Distributional semantic vector spaces represent words as (typically normalized) high-dimensional Euclidean vectors. A set of word vectors is often represented by a sum or average of each word in the set. We will refer to this as a summed vector or an averaged vector (if it has been normalized) as opposed to the word vectors listed in the dictionary (a mapping between the strings for words and their associated vectors).

The problem of finding a weighted sum of vectors that adds to another vector is a linear regression problem, equivalent to solving a system of linear equations, where the weights are the solutions to each variable and the word vectors are the columns of coefficients on each respective variable. When the number of word vectors in the dictionary is larger than the dimensionality of the vectors, the solution is not unique. In order to choose from among these solutions, we impose the restriction that the weights on most words will be zero (sparsity). This is known as a sparse sum problem, and can be solved with tools such as the Least Absolute Shrinkage and Selection Operator (LASSO) (Tibshirani, 1996), which selects a few vectors to have non-zero weight in order to approximate the sum.

LASSO balances sparsity against exactness in finding sparse sums with a parameter, \(\lambda\). It can be difficult to choose the correct lambda, so we use a screening method called DPP (Dual Polytope Projection) (Wang et al., 2013) to efficiently test over the full range of \(\lambda\) values from 0 to 1, and gather all the candidate non-zero weighted vectors from this range. With a dictionary size less than or equal to the dimensionality of the vectors, it is possible to simply solve the resulting system of linear equations exactly. That is, if it is possible to narrow down the list of possible non-zero weighted vectors to no more than the dimensions of the embedding space, all the weights and associated word vectors can be exactly recovered. Because of this, we choose the \(n\) vectors given the highest weight, where \(n\) is the dimensionality of the vectors, and solve exactly with these vectors as the dictionary. As long as the correct vectors are included among these \(n\) vectors, the exact weights will be recovered.

There is good reason to believe that the brain makes use of some kind of sparse decomposition in order to make sense of complex inputs (Beyeler, Rounds, Carlson, Dutt, & Krishnan, 2017). It has been speculated that this could be achieved through competition among neural units in a winner-take-all architecture (Coultrap, Granger, & Lynch, 1999). Others have found that sparsity can be achieved by appropriate thresholding (Rozell, Johnson, Baraniuk, & Olshausen, 2008). Note that in the experiments in this paper, the individual word vectors are not sparse. The sparsity is in which words are chosen from the dictionary (see Table 1).

Experiments: Recovery of weights and word vectors

When trying to recover sparse solutions, there is a phase transition between problems where the solution can be recovered exactly, and problems where this rarely is possible. Problems whose sparsity (non-zero weights / dimensionality) is low enough and whose undersampling ratio (dimensionality / dictionary size) are high enough can be shown to be exactly solvable.

The theoretical probability of a solution being recoverable has to do with the dual of the LASSO problem. It can be calculated by the following formula (see Donoho & Tanner, 2010 for a derivation):

\[
f_k\left(\frac{|\mathcal{C}|^N}{|\mathcal{C}^N|}\right)
\]

where \(f_k\) is the number of \(k\)-dimensional faces on a polytope, \(|\mathcal{C}|^N\) is the