**How do the Fixed-sparsity Distributed Representations advocated and implemented by Numenta work**

1 Answers

1. It's difficult to answer this concisely since the concepts are quite deep. I'll try to give a relatively brief response but it will help if you dive into the video referenced at the end. The fixed sparsity distributed representations (SDR) used by Numenta rely on several properties:1) Extremely high capacity. If you have a high dimensional vector with 2% of the bits on, the number of possible patterns is huge. From the standpoint of a learning algorithm, you don't lose anything by using a sparse representation.2) For any given binary vector size, say you randomly choose 2% of the bits to be on, you can compute the probability that two such vectors have some number of overlapping bits. This probability vanishes very fast as the number of dimensions increases.  The chance of two random patterns colliding are extremely small. (Bloom filters also rely on this property.) 3) Numenta's algorithms create SDR's where similar inputs have similar outputs. Thus individual bits have some semantic meaning. For example, a vertical line and a slightly tilted line are likely to have very similar representations. (This is different from Bloom filters.)  In general, because of property 2, if more than a few bits overlap in two different patterns, it can be treated as significant. The chance of this happening randomly is tiny, hence these inputs are likely to be similar and can be "pooled" together. Various parts of the algorithm use this property to make similar predictions for similar inputs. 4) Multiple SDR representations can be OR'ed together without much chance of conflict. (Bloom filters also rely on this property.) Numenta's temporal pooler relies on this property: it allows the temporal pooler to make simultaneous predictions about the future in a fixed width representation.  For example, suppose ABC and ABD are both common sequences. When the temporal pooler has seen A and then B, it can simultaneously predict C and D by OR'ing together the representations for C and D. 5) You can randomly subsample the on bits and still retain the above properties. This allows for more efficient implementations. I have also seen situations where random subsampling actually helps tremendously by increasing diversity. This in turn increases the chance of finding good representations. I don't fully understand that aspect yet though.Taken together these properties allow the learning algorithm to create interesting spatial and temporal representations of the inputs. In my opinion the real beauty of the system is the temporal pooler and how it uses SDR's to encode sequences. In terms of numbers, in some implementations we have roughly 2000 dimensions with about 2% sparsity. There is now a great keynote talk by Jeff Hawkins describing these intuitions in more detail. You can find the talk here:https://www.numenta.com/blog/the...Hope this helps!

Subutai Ahmad