Sparse, Adaptive Hypernetworks

July 5, 2017

Abstract

Many neural network architectures benefit from sparse connections between layers This sparseness can be designed (as in convolutional neural networks), but resent experiments have shown that it can also be learned, even as part of the end-to-end learning by gradient descent. Unfortunately, such sturcture learning requires all potential weights to be stored in memory (usually in a dense matrix structure): few weights will be nonzero, but we do not know which ones, so all zero values must be stored explicitly. To remedy this problem, we introduce sparse, adaptive hypernetworks. We use a second, dense network to store the weights of the sparse layer—a hypernetwork—and construct the sparse layer dynamically. Moreover, we provide the data as input to the hypernetwork so the structure of the sparse layer can change adaptively. We show that such sparse, adaptive layers can be used to build models that achieve state-of-the-art performance in several domains where large, sparse structures and transformations are important, with minimal memory footprint.

1 Introduction

Many forms of data in modern machine learning are highly structured: vector images, (knowledge) graphs, natural language, etc. Such data is usually best stored in sparse format: usually using one-hot coding for the individual elements, and something akin to an adjacency matrix for the relations holding them together. Consequently, the transformations we want to learn for such instances are also highly sparse.

Consider, for instance the example of learning a graph-to-graph transformation for directed graphs. We will allow self-loops, but not multiple connections, so that each graph can be represented by a square, binary adjacency matrix. Let us also assume that we want to learn a simple graph transformation (possibly as part of a large network). The most straightforward approach would be to use a single, fully-connected layer from one adjacency matrix to another. Let x be the input adjacency matrix, W the wight matrix, and y the output.

In this scenario, our input is likely very sparse: for large, natural graphs, most elements of the adjacency will likely be zero. For good transformations, most weights will likely also be zero: an effective transformation should exploit the locality of the data as suggested by the graph structure. Unfortunately, the structure of our data is not regular. When processing images, we know in advance that each instance will consist of a grid of pixels, and we can wire our network accordingly for the whole dataset. In our case, each instance contains a new structure, which is encoded in the input itself. We must therefore learn the sparse structure of our transformation adaptively.

In a naive implementation, this would still require a dense matrix of weights, even though the matrix we store will be sparse. We don't know if a particular weight will be zero or not for any given instance, so all zero weights must be stored explicitly. Assuming that the output of our transformation is as large as the input, the number of weights of our layer grows as the square of the size of the input. Since the size of the input is already the square of the number of nodes in the network, this is clearly an unfeasible approach for large graphs.

We solve this problem with *sparse, adaptive hypernetworks*. Hypernetworks [**?**] operate by using a secondary, small network to produce the weights of the primary network layer. We make this approach adaptive, by supplying the input of the layer to the hypernetwork as well, allowing the hypernetwork to change the wiring of the main network on the fly. We apply a regulariser to steer the optimisation towards sparse W, combined with a clipping of all but the highest k weights produced by the hypernetwork.

We test the performance of hypernetworks in three domains. First, we show that an image autoencoder with sparse adaptive layers can match the performance of one with a similar number of parameters and weights. Additionally, we show that the adaptive wiring acts as a kind of attention mechanism, allowing the network to devote more parameters to areas that require more attention. Second, we test the hypothesis that multilayer sparse networks can be used to learn complex invariant representations of graphs. We design a graph classification task which cannot be solved without successfully detecting isomorphic graphs as an intermediate step. We train a model end-to-end and show that after training, the embeddings learning for graphs in the same isomorphism class show strong clustering in the embedding space. Finally, we show that sparse, adaptive layers can be used in *generative models*. We use a variational autoencoder [**?**] consisting of sparse, adaptive layers to model a collection of MIDI files stored as knowledge graphs (using the midi-ld standard). We show that the knowledge graphs sampled from the resulting distribution adhere to the required standard (i.e. they can be parsed into music) and result in generated music that is judged to be more likely to have been produced by a human being than several baseline music generators.

2 Model

3 Experiments

4 Conclusion