Recursive Neural Networks Review

Hazem Nomer
Mathematics and Computer Science Department, Faculty of Science, Alexandria University
hazemahmed@alexu.edu.eg

Abstract

In this review, we propose a generalization of recursive neural networks that operates over recursive structures. We discuss major contributions of researchers in applying recursive neural networks to different problems. Due to many theoretical implications, recursive neural networks are believed to be powerful models. In addition, recursive networks learn high-level representations from explicit inputs thus, it is successful in many deep learning tasks where the input is a structure. We also tend in this review to connect convolution neural network and recurrent neural network to a more general model that stand differently from feed-forward neural networks.

1. Introduction

Regular feed-forward neural networks are composed of two basic operations: affine transformation and non-linear application of an activation function to this transformation. A hidden layer in neural network transforms the input into a new – usually compact- representation in a new space *h*. This is the actual ability of neural networks to fit the training data and generalize to all inputs. Neural networks operate usually on vectors of real numbers in *n* dimension.

Other neural networks as convolution ones, are heavily used in computer vision applications, operate on a grid or a 2-D vector. A convolution neural network defines a receptive field that operates on part of the grid. While CNN takes advantage of parameter sharing it does not consider the spatial structure of the grid. This means that each receptive field is computed isolated from the other.

A neural network that operates on a structure (a grid, string, graph, tree, sequence, etc...) is called a recursive neural network. It takes advantage of parameter sharing as in CNN and the spatial structure of the input. For example, given a sentence parse tree, the recursive neural network operates on each internal node of the tree computing its activations as a function of its children, which may be an internal node or a vector representing the word at the leaves. At each node, the neural network uses the same set of weights to compute the activations.

Similarly, a recurrent neural network operates on sequences of vectors: $x^{(1)}$, $x^{(2)}$, $x^{(3)}$, …, $x^{(t)}$. This introduces a notion of time in the model, as each vector is located in a slot of time in a time space (τ). At each time step, the recurrent neural network computes a function of the current input and the activation or the output from the

previous time step. In this way, the recurrent neural network exploits the time structure of a sequence using parameter sharing concept.

In this review, we introduce recurrent neural network as a type of recursive networks. In general, recursive networks operates on structured data and thus find applications in many domains as natural language processing, protein sequence prediction, scene understanding, and any other domain that include structured data.

2. Parameter sharing

Recursive neural networks process structured data of different forms. For example, natural language sentences have different lengths and structures with the same meaning. A model operating on sentences should generalize to all types of their different forms. Parameter sharing allows recursive neural networks to generalize to different forms of data not seen before in training data.

CNN uses the idea of parameter sharing in images. Each CNN feature map takes only *nxn* of an image, which is called a receptive field. The weights connecting the receptive field and each neuron in the feature map are called kernel. For each receptive field in the input image, we use the same set of weights.

A recurrent neural network that operates on sequences applies the same set of weights on each vector $x^{(t)}$ and activations from previous time steps. A feed forward neural network would have different weights for each vector $x^{(t)}$.

For a recursive neural network operating on a binary tree, we apply the same set of weights on each parent node and its left and right children. In recursive neural networks, parameter sharing is only applied to different lengths or size of the data and not different structures. For example, a recursive network that works on binary trees will be able to generalize to different binary tree heights but will not generalize to ternary trees or other trees structures.

3. Formal description

A direct acyclic graph is a pair (V, E), V is the set of vertices or nodes and E is the set of edges between them. For a node, pa[v] is the set of parents of node v and ch[v] is the set of children of v. Each node is a vector that encodes information useful for the task. A graph may be used to model different kinds of information, an edge between two nodes models a logical relationship between two them.
A recursive neural network is two functions f and g:

$$h(v^i) = f\left(\sum_{j=0}^{m} W_j . x_j^i\right)$$

$$o(v^i) = g\left(U . h(v^i)\right)$$

where $W_j$is the weight matrix connecting node $v^i$with its m children and x_j^i is the j_{th} child of $v^i$and U is the weight matrix connecting the hidden activation with final

output at the node. According to parameter sharing concept, theses weight matrices are the same for any node in the graph.

The function ***f*** receives as input the activations from ch[v] and multiply each by a weight matrix to compute an activation for the parent node. The function ***g*** labels each node v^i using its hidden activation. To process a graph G we apply the network recursively until we reach the source or root node. Note that a node may not contain an input, but an initial input must be represented in the graph as in parse trees of sentences, the input is the leaves of the tree. The output function can only be applied at the source node labeling the whole graph.

3.1 Recurrent neural network

A recurrent neural network operating over time sliced sequences is a recursive neural network:

$$h^{(t)} = f\left(W^{hx}x^{(t)} + W^{hh}h^{(t-1)} + b\right)$$

In the same way, a recurrent neural network receives both an input from the sequence and the hidden activation from the previous time step. There are many variations of recurrent neural networks but the main concepts of parameter sharing and recursive application of the network over the input structure remain the same.

3.2 Deep Recursive neural network

We represented so far recursive neural networks that only contain one hidden layer h, but a recursive neural network can also be deep in the representation space. In deep feed forward networks every hidden layer is a more abstract representation of the input. A deep recursive neural network as proposed by (Irsoy and Cardie, 2014) is constructed by stacking multiple layers of recursive neural networks. The model was evaluated on the task of fine-grained sentiment classification.

A general description of deep recursive neural networks is:

$$h^l\left(v^i\right) = f\left(V^l h^{l-1}\left(v^i\right) + \sum_{j=0}^{m} W^l{}_j . x_j^i\right)$$

$$o\left(v^i\right) = g\left(U . h^L\left(v^i\right)\right)$$

Where $V^l$is the weight matrix that connects layer $l-1$ with the next layer l and the output layer is connected to the final hidden layer$L$.

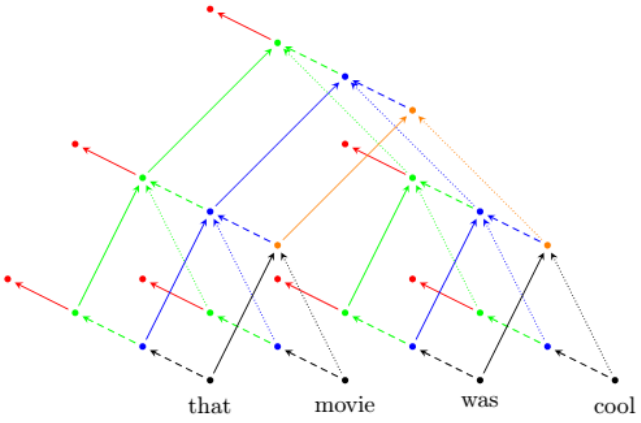

Figure 1. Deep recursive neural network operating on parse tree of sentences as described by (Irsoy & Cardie, 2014)

4. Recursive Long short term memory

A long short-term memory network is a recurrent neural network introduced by Hochreiter and Schmidhuber (1997) to overcome the vanishing gradients problem. Each node in LSTM is replaced by a memory cell. It has been applied to many problems as speech recognition, machine translation and handwriting recognition (Graves et al., 2009).

Each memory cell receives the activation from the previous time step and the current input vector from the sequence. A memory cell is built from simpler nodes that operate in a specific order using multiplicative nodes between each two nodes. One type of nodes is the gates, which are nodes that receive the input from the current sequence and activation from previous time step.

In a recursive LSTM applied on a graph, a memory cell at a node receives input vectors or activations from its children. The S-LSTM (Zhu et al., 2015) is a model extended to structure as parse trees. The model considers the flow of information from the children of a node to the parent thus, it reflects the history of memories of children and descent nodes, for example, a memory cell at a certain node can learn when to block or allow certain information from the descent nodes.

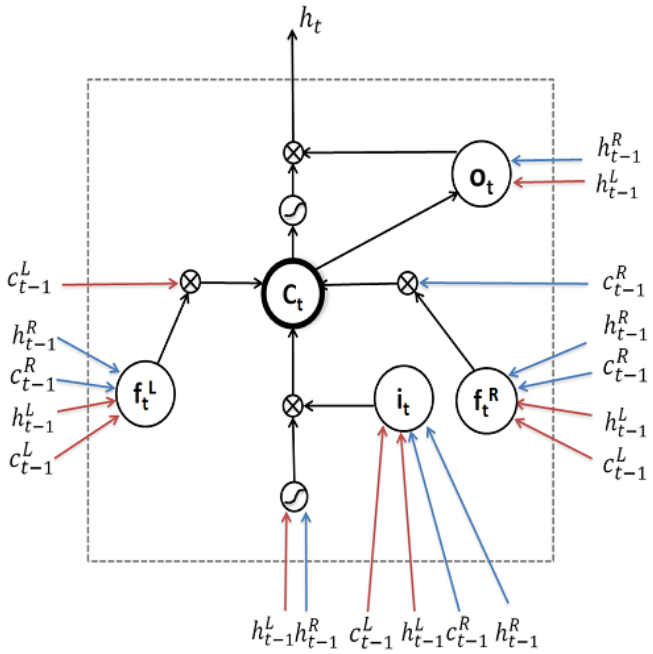

Figure 2. An S-LSTM memory block (Zho et al., 2015), consisting of an input gate, , two forget gates and an output gate. ⊗ denotes a multiplicative node between the gates. The design choice to have two forget gates is because the model was applied to binary trees and can be extended to *n-ary* trees by adding more forget gates.

An S-LSTM memory block has input gate, forget gates, an output gate, and a cell state. An input node receives activation from the left and right children of the node (the model is applied to binary trees).

The gates (except the output gate) receive four sources of information the cell states of right and left children and the activations of the right and left children. The cell state at the current node receives cell states of both left and right children. The output gate receives activations of right and left children of the node.

The number of forget gates depend on the number of children of a node to determine how much information can pass from each child to the parent node.

5. Backpropagation through structure

The back propagation algorithm is modified to compute gradients for recursive neural networks. Usually, a back propagation scheme starts by unfolding the network for a single iteration (or epoch). The unfolding of the network is copying each instance of the network through the structure with the shared parameters. Now the network is a regular feed-forward neural network that has the same topology as the structure of an input.

In General, to compute an error term vector for a node in this unfolded network, the error of the parent node is propagated down to its children (or subgraphs). The weight change is then computed as the sum of all subgraphs in the structure. Understanding back propagation is essential to design a variant for recursive neural networks. Many

variants of the original algorithm exist (Goller & Kuchler, 1996) as this proposed by (Socher et al., 2013) and for recursive LSTM by (Zhu et al., 2015).

6. Applications

Almost recent applications are in applying recursive neural networks in natural language processing. In (Socher et al., 2011) they described a recursive network (RvNN) in scene understanding. The model described was used to combine image segments and labeling them. Legrand and Collobert leveraged neural networks over greedy syntactic parsing (Pinheiro & Collobert, 2014.). In (Hermann & Blunsom, 2013) composition was performed with the consideration of linguistic motivations specifically from the combinatory categorial grammar (CCG) formalism.

7. Conclusion

Recursive neural networks is a generalization for networks that operates over structures using parameter sharing algorithms. Researchers try to simplify and constraint the structures which the network operates on, but in general, recursive networks are very powerful models. They have been applied successfully to many tasks in natural language processing as sentiment analysis and scene understanding. Recursive neural networks are expected to gain success in many deep learning tasks as it captures information from structured explicit data. Recursive neural networks have been applied to protein structure prediction problem. It is expected that large computational power delivered by modern GPUs will help tackle this problem using recursive networks.

References

Chinea, Alejandro, and Elka Korutcheva. "Complexity Analysis of Vario-eta through Structure." *arXiv preprint arXiv:1106.1113* (2011).

Goller, Christoph and Kuchler, Andreas. Learning taskdependent distributed representations by backpropagation through structure. In *Proceedings of ICNN*, pp. 347–352, Bochum, Germany, 1996.

Hermann, Karl Moritz and Blunsom, Phil. The Role of Syntax in Vector Space Models of Compositional Semantics. In *Proceedings of ACL*, pp. 894–904, Sofia, Bulgaria, August 2013.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, (8):1735–1780, 1997.

Irsoy, Ozan and Cardie, Claire. Deep recursive neural networks for compositionality in language. In *Proceedings of NIPS*, pp. 2096–2104. 2014.

Socher, Richard, Lin, Cliff C., Ng, Andrew Y., and Manning, Christopher D. Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *Proceedings of ICML*, 2011.

Socher, Richard, Perelygin, Alex, Wu, Jean Y., Chuang, Jason, Manning, Christopher D., Ng, Andrew Y., and Potts, Christopher. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, 2013.

Zhu, Xiaodan, Parinaz Sobhani, and Hongyu Guo. "Long short-term memory over tree structures." *arXiv preprint arXiv:1503.04881* (2015).