Deep Recurrent Networks

Sargur Srihari
srihari@buffalo.edu

Topics

- Recurrent Neural Networks

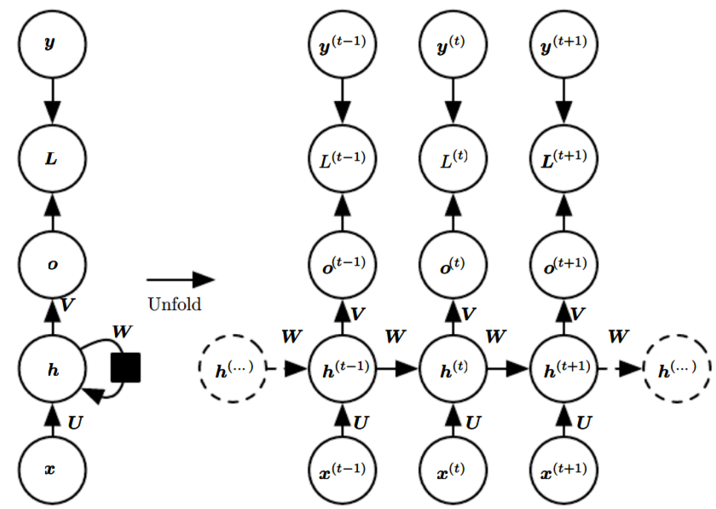
1. Unfolding Computational Graphs
2. Recurrent Neural Networks
3. Bidirectional RNNs
4. Encoder-Decoder Sequence-to-Sequence Architectures
5. Deep Recurrent Networks
6. Recursive Neural Networks
7. The Challenge of Long-Term Dependencies
8. Echo-State Networks
9. Leaky Units and Other Strategies for Multiple Time Scales
10. LSTM and Other Gated RNNs
11. Optimization for Long-Term Dependencies
12. Explicit Memory

Computation in RNNs: parameter blocks

- The computation in most recurrent neural networks can be decomposed into three blocks of parameters and associated transformations:
  1. From the input to the hidden state
  2. From the previous hidden state to the next hidden state
  3. From the hidden state to the output

Blocks of parameters as a shallow transformation

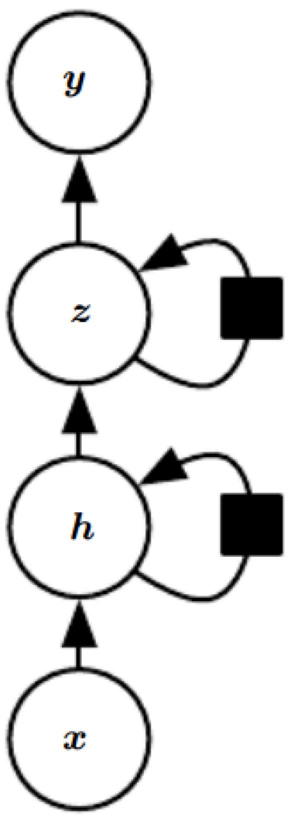
- With the RNN architecture shown each of these three blocks is associated with a single weight matrix, i.e.,
  - When the network is unfolded, each of these corresponds to a shallow transformation.
  - By a shallow Transformation we mean a transformation that would be represented a single layer within a deep MLP.



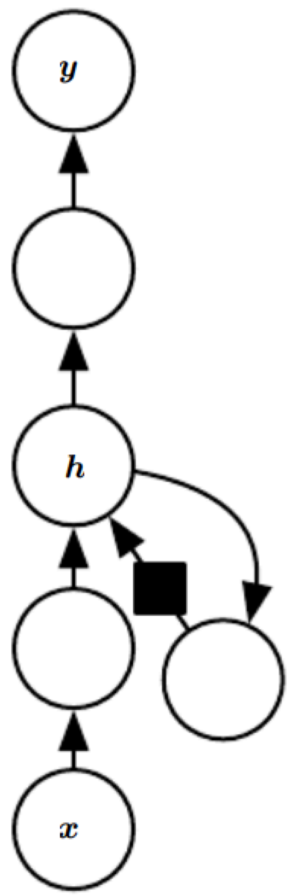
  - Typically this is a transformation represented by a learned affine transformation followed by a fixed nonlinearity
- Would it be advantageous to introduce depth into each of these operations?
  - Experimental evidence strongly suggests so.
    - That we need enough depth in order to perform the required transformations

Ways of making an RNN deep

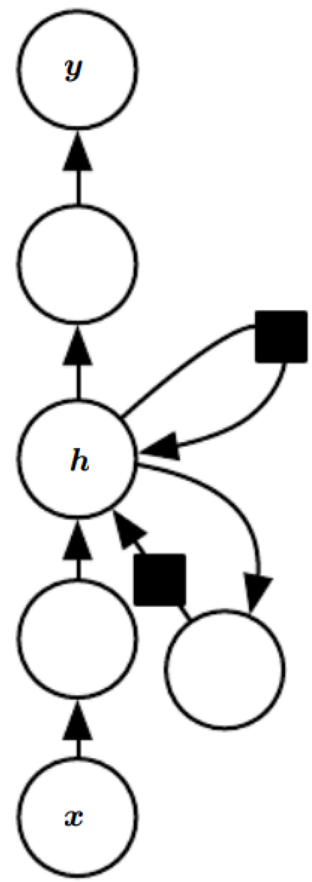
1. Hidden recurrent state can be broken down into groups organized hierarchically



2. Deeper computation can be introduced in the input-hidden, hidden-hidden and hidden-output parts. This may lengthen the shortest path linking different time steps

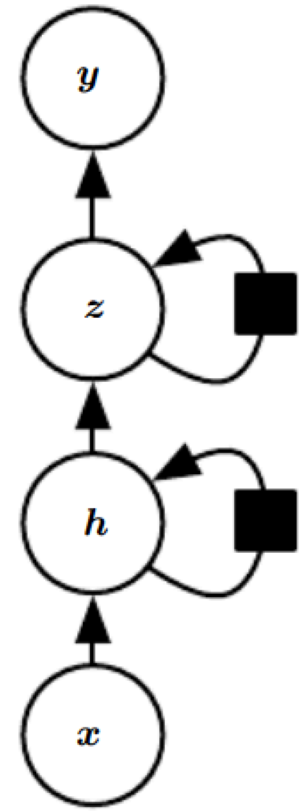


3. The path-lengthening effect can be mitigated by introducing skip connections.



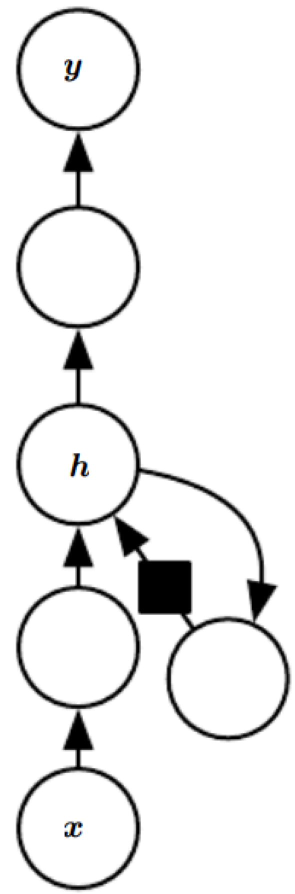
1. Recurrent states broken down into groups

We can think of lower levels of the hierarchy play a role of transforming the raw input into a representation that is more appropriate at the higher levels of the hidden state



2. Deeper computation in hidden-to-hidden

- Go a step further and propose to have a separate MLP (possibly deep) for each of the three blocks:
  1. From the input to the hidden state
  2. From the previous hidden state to the next hidden state
  3. From the hidden state to the output
- Considerations of representational capacity suggest that to allocate enough capacity in each of these three steps
  - But doing so by adding depth may hurt learning by making optimization difficult
  - In general it is easier to optimize shallower architectures
  - Adding the extra depth makes the shortest time of a variable from time step t to a variable in time step $t+1$ beome longer



3. Introducing skip connections

- For example, if an MLP with a single hidden layer is used for the state-to-state transition, we have doubled the length of the shortest path between variables in any two different time steps compared with the ordinary RNN.
- This can be mitigated by introducing skip connections in the hidden-to-hidden path as illustrated here