Types of Recurrent Neural Networks



Figure 1: RNN that produce an output at each time step and have recurrent connections between hidden units.

**Sequence RNN:** This network generates a sequence of outputs based on a sequence of inputs (y is desired, O is network output, L is cost function)

W: changes during back propagation through time (BPTT)

h: represents multiple hidden layers

V: is weight used in feedback to hidden layers



Figure 2: RNN producing an output at each time step and have recurrent connections only from output at one time step to the hidden units at next time step.

* Previous output becomes input to next state



Figure 3: RNN with recurrent connections between hidden units that read an entire sequence and then produce a single output.

* Produce an output at a chosen time step

**Forward Propagation for RNN that produce an output at each time step and have recurrent connections between hidden units.**

**Step 1:**

Let the initial condition of the hidden state be

**Step 2:**

For each time step from t = 1 to t = τ, perform the following updates:

Where

b is the bias vector of the hidden layer,

c is the bias vector pf the output layer,

U is the input-to-hidden weight matrix,

V is the hidden-to-output weight matrix,

W is the hidden-to-hidden connections weight matrix

**Step 3:**

* Gradient computation of the loss function involves performing a forward propagation pass moving from left to right in Figure 1 through the unrolled graph, followed by a backward propagation pass moving from right to left through the unrolled graph.
* The forward propagation being inherently sequential will result in a runtime of *O(τ)* and cannot be reduced by parallelization as each step can be computed only after the previous step is completed.
* The ***h*** states computed during forward propagation has to be stored for reuse during the back pass resulting in a memory cost of *O(τ).*

Vanishing Gradient Problem:

Consider a simple recurrent neural network without any activation function and without inputs at each time step as shown below:



The recurrence relationship can be written as

(1)

The eqn. 1 describes the power method and can be simplified as

Performing eigen-decomposition on the weight matrix W with the assumption that the matrix W is real symmetric matrix, we have

Where,

Q is an orthogonal matrix with the eigenvectors of ***W*** as columns

is a diagonal matrix of the eigenvalues of ***W***.

* The eigenvalues are raised to the power of t causing the eigenvalues with magnitude less than one to decay to zero and eigenvalues with magnitude greater than one to explode.
* Therefore a component of *h(0*) not aligned with the largest eigenvector will eventually be discarded.
* Therefore, the product ***wt***will either vanish or explode depending on the magnitude of ***w***.
  + Exploding can be solved with clipping

RNN producing an output at each time step and have recurrent connections only from output at one time step to the hidden units at next time step.

* RNN’s that have connections from outputs leading back into the model can be trained with teacher forcing approach.
* Teaching forcing approach during training involves the model receiving the ground truth i.e., ground truth output as the input at time (*t*+1).





Conveyer Belt Visualization of RNN



Additional Carry Track to Address Vanishing Gradient Problem in RNN

* Long Short Term Memory (LSTM) Cell
* Gated Recurrent Unit (GRU) Cell
* Peek LSTM Cell