What makes RNNs tick ??

Madan Ravi Ganesh (ECE)

Convolutional neural networks (CNNs), aka. dump and regress, are a formidable form of end to end gradient machines that are taking over rapidly in the various domains. Indeed, their boom has coincided with that of high end GPUs with rapid processing power. However, in their vanilla form, they stand unable to handle sequence data. These data form the next major barrier in various problem spaces and need to be handled effectively for CNNs to take the next great leap. Thus, born out of such need was the Recurrent Neural Network (RNN) **??**.

In its vanilla form, a basic RNN is characterized using the following equations,

$$s_t = f(Ux_t + Ws_{t-1}) \tag{1}$$

$$o_t = softmax(Vs_t) \tag{2}$$

$$E(y, \hat{y}) = -\sum_t y_t log(\hat{y_t}) \tag{3}$$

Notice the time dependence on each of these parameters, marked by the t or $t-1$. This recursive formulation does not lend itself to an easy or simple back-propagation formula. Get ready to put your calculus hats on people!!

The complete RNN system is parameterized by $U, V \text{and} W$. Hence, we need to find the gradients w.r.t. these parameters. Based on 3, we need to find $\frac{dE}{dV}, \frac{dE}{dW} \text{and} \frac{dE}{dU}$. Here, I dropped the time dependence parameters to make it easier to read.

The first partial derivative that can be easily handled is $\frac{dE}{dV}$.

$$\begin{aligned} \frac{dE_t}{dV} &= y_t * log(\hat{y_t}), \text{where} \\ \hat{y_t} &= softmax(Vs_t) \\ if Vs_t &= z_t \\ \hat{y_t} &= \frac{\exp z_t^i}{\sum k = 1K z_t^k} \end{aligned}$$

Here, t is assumed to a specific time-step while z is an alias for Vs_t.

$$\frac{dE_t}{dV} = \frac{dE_t}{d\hat{y_t}} * \frac{d\hat{y_t}}{dz_t} * \frac{dz_t}{dV} \tag{4}$$

Expanding the first term yields,

$$\frac{dE_t}{d\hat{y_t}} = \frac{y_t}{\hat{y_t}} \tag{5}$$

Expanding the second term yields,

REFERENCES