An analysis and verification of the efficacy of using Fast Weights with RNNs

CSE847 (Machine Learning) Project Final Report

Eric Alan Wayman[1] Adi Mathew[1]

[1]Department of Computer Science and Engineering
Michigan State University

April 27, 2017

RNN basics

Definition

A *standard RNN* is defined by the following equations:

$$
\begin{aligned}
a(t) &= b + Wh(t-1) + Ux(t) \\
h(t) &= \mathrm{activ}(a(t)) \\
o(t) &= c + Vh(t) \\
\widehat{y}(t) &= \mathrm{softmax}(o(t))
\end{aligned}
$$

where we choose to minimize the cross-entropy cost function since we are performing a classification task:

$$E = -\ln(p) = -\sum_{\tau=1}^{t} \sum_{j=1}^{k} y_{\tau k} \ln\left(\widehat{y_{\tau j}}\right) = \sum_{\tau=1}^{t} E_t$$

RNN basics

For the associative retrieval task, we wish to only look at the end-of-sequence y so the error function becomes

$$E = -\ln(p) = \sum_{j=1}^{k} y_{tk} \ln\left(\widehat{y_{tj}}\right) = E_t$$

RNN basics

$$
\begin{aligned}
a(t) &= b + Wh(t-1) + Ux(t) \\
h(t) &= \mathrm{activ}(a(t)) \\
o(t) &= c + Vh(t) \\
\widehat{y}(t) &= \mathrm{softmax}(o(t))
\end{aligned}
$$

Note that W and U when chosen with gradient-training procedures will choose W and U that influence h_t such that E_t over the entire batch is minimized (i.e. the "average error") is small. Can we do better?

Associative memory basics

Consider an associative memory which learns the single key pattern f and value g where both are column vectors (Anderson, *An Introduction to Neural Networks*, 1995). We let the system be the matrix

$$A = \eta g f^T$$

The system performs perfectly:

$$g' = Af = \eta g f^T f \propto g$$

since the g' that is recalled is proportional to the value g associated with the input f.

Associative memory basics

Now consider a set of key patterns f_i and associated values g_i where all f_i are orthogonal (we write $f_i \to g_i$ to denote the associations). Letting

$$A_i = g_i {f_i}^T, \qquad A = \sum_i A_i$$

we see that again A performs recall perfectly since for all j,

$$\begin{aligned} Af_j &= \sum_i A_i f_j = \sum_{k \neq j} A_k f_j + A_j f_j \\ &= \sum_{k \neq j} g_k f_k^T f_j + \eta g_j \propto g_j \end{aligned}$$

Associative memory basics

Using outer products to create memory storage is referred to "the generalization of Hebb's postulate of learning" (Haykin, *Neural Networks and Learning Machines* 2009) since weight updates in Hebbian learning are calculated with outer products.

Note that generally, not all sets of key patterns would be orthogonal; we discuss the implications of this when we consider the associative memory structure from the Fast Weights paper.

Dynamic systems approach

RNN specification

RNN training