Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks

Introduction

Gated Recurrent Neural Network (RNN) have shown success in several applications involving sequential or temporal data. For example, they have been applied extensively in speech recognition, natural language processing, machine translation, etc.. Long Short-Term Memory (LSTM) RNN and the recently introduced Gated Recurrent Unit (GRU) RNN have been successfully shown to perform well with long sequence applications.

This paper focuses on the GRU RNN and explore three new gate-variants with reduced parameterization. The performance of the original and the variant GRU RNN on two public datasets are comparatively evaluated. Using the MNIST dataset, one generates two sequences. One sequence is obtained from each 28x28 image sample as pixelwise long sequence of length 28x28=784 (basically, scanning from the upper left to the bottom right of the image). Also, one generates a row-wise short sequence of length 28, with each element being a vector of dimension 28. The third sequence type employs the IMDB movie review dataset where one defines the length of the sequence in order to achieve high performance sentiment classification from a given review paragraph.

Gated Recurrent Unit (GRU) RNN

The GRU RNN reduce the gating signals into two from the LSTM RNN model. The two gates are called an update gate z_t and a reset gate r_t . The GRU RNN model is presented in the form:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot h_t \qquad (1)$$

$$h_t = g(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \qquad (2)$$

with the two gates presented as:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \qquad (3)$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \qquad (4)$$

One observes that the GRU RNN [Eqns (1)-(2)] is similar to the LSTM RNN, however with less external gating signal in the interpolation Eqn (1). This saves one gating signal and the associated parameters. In essence, the GRU RNN has 3-folds increase in parameters in comparison to the simple RNN. Specifically, the total number of parameters in the GRU RNN equals $3\times(n2 + nm + n)$.

In various studies, it has been noted that GRU RNN is comparable to, or even outperforms, the LSTM in most cases. Moreover, there are other reduced gated RNNs, e.g. the Minimal Gated Unit (MGU) RNN, where only one gate equation is used and it is reported that this (MGU) RNN performance is comparable to the LSTM RNN and the GRU RNN. This paper focuses on the GRU RNN model and evaluate new variants. Specifically, we retain the architecture of Eqns (1)-(2) unchanged, and focus on variation in the structure of the gating signals in Eqns (3) and (4). We apply the variations identically to the two gates for uniformity and simplicity.

The variant gru architectures

The gating mechanism in the GRU (and LSTM) RNN is a replica of the simple RNN in terms of parametrization. The weights corresponding to these gates are also updated using the backpropagation through time (BTT) stochastic gradient descent as it seeks to minimize a loss/cost function. Thus, each parameter update will involve information pertaining to the state of the overall network. Thus, all information regarding the current input and the previous hidden states are reflected in the latest state variable. There is a redundancy in the signals driving the gating signals. The key driving signal should be the internal state of the network. Moreover, the adaptive parameter updates all involve components of the internal state of the system. In this study, we consider three distinct variants of the gating equations applied uniformly to both gates:

Variant 1: called GRU1, where each gate is computed using only the previous hidden state and the bias.

$$z_t = \sigma(U_z h_{t-1} + b_z) \qquad (5\text{-a})$$

$$r_t = \sigma(U_r h_{t-1} + b_r) \qquad (5\text{-b})$$

Thus, the total number of parameters is now reduced in comparison to the GRU RNN by 2× nm.

Variant 2: called GRU2, where each gate is computed using only the previous hidden state.

$$z_t = \sigma(U_z h_{t-1}) \qquad (6\text{-a})$$

$$r_t = \sigma(U_r h_{t-1}) \qquad (6\text{-b})$$

Thus, the total number of parameters is reduced in comparison to the GRU RNN by 2× (nm+n).

Variant 3: called GRU3, where each gate is computed using only the bias.

$$z_t = \sigma(b_z) \qquad (7\text{-a})$$

$$r_t = \sigma(b_r) \qquad (7\text{-b})$$

Thus the total number of parameters is reduced in comparison to the GRU RNN by 2× (nm+n2).

An empirical study of the performance of each of these variants as compared to the GRU RNN is performed on, first, sequences generated from the MNIST dataset and then on the IMDB movie review dataset. Here we refer to the base GRU RNN model as GRU0 and the three variants as GRU1, GRU2, and GRU3 respectively. Our architecture consists of a single layer of one of the variants of GRU units driven by the input sequence and the activation function set as ReLU. (Initial experiments using g = tanh have produced similar results). For the MNIST dataset, we generate the pixel-wise and the row-wise sequences. The networks have been generated in Python using the Keras library with Theano as a backend library. As Keras has a GRU layer class, we modified this class to classes for GRU1, GRU2, and GRU3. All of these classes used the ReLU activation function. The RNN layer of units is followed by a softmax layer in the case of the MNIST dataset or a traditional logistic activation layer in the case of the IMDB dataset to predict the output category. The Root Mean Square Propagation (RMSprop) is used as the choice of optimizer that is known to adapt the learning rate for each of the parameters. To speed up training, we also decay the learning rate exponentially with the cost in each epoch

$$\eta = \eta_0 e^{cost} \qquad (8)$$

where η_0 represents a base constant learning rate and cost is the cost computed in the previous epoch.

Results

A. Application to MNIST Dataset – pixel-wise sequences

The MNIST dataset consists of 60000 training images and 10000 test images, each of size 28x28 of handwritten digits. The three variants against the original GRU model on the MNIST dataset is evaluated by generating the sequential input in one case (pixel-wise, one pixel at a time) and in the second case (row-wise, one row at a time). GRU1 and GRU2 perform almost as well as GRU0 on MNIST pixel-wise generated sequence inputs. While GRU3 does not perform as well for this (constant base) learning rate. Reducing the (constant base) learning rate to (0.0001) and below has enabled GRU3 to increase its (test) accuracy performance to 59.6% after 100 epochs, and with a positive slope indicating that it would increase further after more epochs.

B. Application to MNIST Dataset – row-wise sequences

The row-wise generated sequences can test short sequences (of length 28) with vector elements. All the four variants GRU0, GRU1, GRU2, and GRU3 appear to exhibit comparable accuracy performance over three constant base learning rates. GRU3 exhibits lower performance at the base learning rate of 1e-4 where, after 50 epochs, is still lagging. More epochs are likely to increase performance to comparable levels with the other variants. It is noted that in this experiment, GRU3

can achieve comparable performance with roughly one third of the number of (adaptively computed) parameters.

C. Application to the IMDB Dataset– natural sequence

The IMDB dataset is composed of 25000 test data and 25000 training data consisting of movie reviews and their binary sentiment classification. Each review is represented by a maximum of 80 (most frequently occurring) words in a vocabulary of 20000 words. We have trained the dataset on all 4 GRU variants using the two constant base learning rates of 1e-3 and 1e-4 over 100 epochs. In the training, 128-dimensional GRU RNN variants are employed and have adopted a batch size of 32. We have observed that, using the constant base learning rate of 1e-3, performance fluctuates visibly, whereas performance is uniformly progressing. The IMDB data experiments provide the most striking results. It can be clearly seen that all the 3 GRU variants perform comparably to the GRU RNN while using less number of parameters. The learning pace of GRU3 was also similar to those of the other variants at the constant base learning rate of 1e-4. More saving in computational load is achieved by all variant GRU RNN as the input is represented as a large 128-dimensional vector.