Transformers How To

Daniel Walther Berns

May 5, 2023

This article is divided into two parts; they are:

1. Introduction to RNN
   (a) Unfolding in time
   (b) Backpropagation through time algorithm
2. Different RNN architectures and variants

1 Introduction to RNN

A recurrent neural network (RNN) is a special type of artificial neural network adapted to work for time series data or data that involves sequences. Ordinary feedforward neural networks are only meant for data points that are independent of each other. However, if we have data in a sequence such that one data point depends upon the previous data point, we need to modify the neural network to incorporate the dependencies between these data points. RNNs have the concept of "memory" that helps them store the states or information of previous inputs to generate the next output of the sequence.

1.1 Unfolding a Recurrent Neural Network

A simple RNN has a feedback loop, as shown in the first diagram of the above figure. The feedback loop shown in the gray rectangle can be unrolled in three time steps to produce the second network of the above figure. Of course, you can vary the architecture so that the network unrolls k time steps. In the figure, the following notation is used:

1. $x_t \in \Re$ is the input at time step t. To keep things simple, we assume that x_t
2. is a scalar value with a single feature. You can extend this idea to a d-dimensional feature vector.
3. $y_t \in \Re$ is the output of the network at time step t . We can produce multiple outputs in the network, but for this example, we assume that there is one output.
4. $h_t \in \Re^m$ vector stores the values of the hidden units/states at time t. This is also called the current context. m is the number of hidden units. h_0 vector is initialized to zero.

5. w_x are weights associated with inputs in the recurrent layer.
6. w_h are weights associated with hidden units in the recurrent layer
7. w_y are weights associated with hidden units to output units
8. b_h is the bias associated with the recurrent layer
9. b_y is the bias associated with the feedforward layer

At every time step, we can unfold the network for k time steps to get the output at time step $k+1$. The unfolded network is very similar to the feedforward neural network. The rectangle in the unfolded network shows an operation taking place. So, for example, with an activation function f:

$$h_{t+1} = f(x_t, h_t, w_x, w_h, b_h) = f(w_x\, x_t + w_h\, h_t + b_h). \tag{1}$$

The output y at time t is computed as:

$$y_t = f(h_t, w_y) = f(w_y \cdot h_t + b.) \tag{2}$$

Hence, in the feedforward pass of an RNN, the network computes the values of the hidden units and the output after k time steps. The weights associated with the network are shared temporally. Each recurrent layer has two sets of weights: one for the input and the second for the hidden unit. The last feedforward layer, which computes the final output for the kth time step, is just like an ordinary layer of a traditional feedforward network.

1.2 The Activation Function

We can use any activation function we like in the recurrent neural network. Common choices are:

1. Sigmoid function: $a(x) = \frac{1}{1+e^{-x}}$.
2. Tanh function: $a(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$.
3. Relu function: $a(x) = max(0, x)$.

2 Training a Recurrent Neural Network

The backpropagation algorithm of an artificial neural network is modified to include the unfolding in time to train the weights of the network. This algorithm is based on computing the gradient vector and is called backpropagation in time or BPTT algorithm for short. The pseudo-code for training is given below. The value of k can be selected by the user for training. In the pseudo-code below, p_t is the target value at time step t:

1. Repeat till the stopping criterion is met:
   (a) Set all h to zero.
   (b) Repeat for $t = 0$ to $n - k$,

(c) Forward propagate the network over the unfolded network for k time steps to compute all h and y.

(d) Compute the error as: $e = y_{t+k} - p_{t+k}$.

(e) Backpropagate the error across the unfolded network and update the weights

3 Types of RNN

The encoder-decoder architecture for recurrent neural networks is the standard neural machine translation method that rivals and in some cases outperforms classical statistical machine translation methods.

The Encoder-Decoder architecture with recurrent neural networks has become an effective and standard approach for both neural machine translation (NMT) and sequence-to-sequence (seq2seq) prediction in general.

The key benefits of the approach are the ability to train a single end-to-end model directly on source and target sentences and the ability to handle variable length input and output sequences of text.

As evidence of the success of the method, the architecture is the core of the Google translation service: "Our model follows the common sequence-to-sequence learning framework with attention. It has three components: an encoder network, a decoder network, and an attention network."

4 Sutskever NMT Model

We will look at the neural machine translation model developed by Ilya Sutskever, et al. as described in their 2014 paper "Sequence to Sequence Learning with Neural Networks". We will refer to it as the "Sutskever NMT Model", for lack of a better name.

This is an important paper as it was one of the first to introduce the Encoder-Decoder model for machine translation and more generally sequence-to-sequence learning.

It is an important model in the field of machine translation as it was one of the first neural machine translation systems to outperform a baseline statistical machine learning model on a large translation task.

The model was applied to English to French translation, specifically the WMT 2014 translation task.

The translation task was processed one sentence at a time, and an end-of-sequence (`<EOS>`) token was added to the end of output sequences during training to signify the end of the translated sequence. This allowed the model to be capable of predicting variable length output sequences.

The model was trained on a subset of the 12 Million sentences in the dataset, comprised of 348 million French words and 304 million English words. This set was chosen because it was pre-tokenized.

The source vocabulary was reduced to the 160000 most frequent source English words and 80000 of the most frequent target French words. All out-of-vocabulary words were replaced with the "UNK" token.

4.1 The model

An Encoder-Decoder architecture was developed where an input sequence was read in entirety and encoded to a fixed-length internal representation.

A decoder network then used this internal representation to output words until the end of sequence token was reached. LSTM networks were used for both the encoder and decoder.

The idea is to use one LSTM to read the input sequence, one timestep at a time, to obtain large fixed-dimensional vector representation, and then to use another LSTM to extract the output sequence from that vector

The final model was an ensemble of 5 deep learning models. A left-to-right beam search was used during the inference of the translations.

4.2 Model Configuration

1. Input sequences were reversed.
2. A 1000-dimensional word embedding layer was used to represent the input words.
3. Softmax was used on the output layer.
4. The input and output models had 4 layers with 1,000 units per layer.
5. The model was fit for 7.5 epochs where some learning rate decay was performed.
6. A batch-size of 128 sequences was used during training.
7. Gradient clipping was used during training to mitigate the chance of gradient explosions.
8. Batches were comprised of sentences with roughly the same length to speed-up computation.

5 Cho NMT Model

In this section, we will look at the neural machine translation system described by Kyunghyun Cho, et al. in their 2014 paper titled "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation." We will refer to it as the "Cho NMT Model" model for lack of a better name.

Importantly, the Cho Model is used only to score candidate translations and is not used directly for translation like the Sutskever model above. Although extensions to the work to better diagnose and improve the model do use it directly and alone for translation.

5.1 Problem

As above, the problem is the English to French translation task from the WMT 2014 workshop.

The source and target vocabulary were limited to the most frequent 15,000 French and English words which covers 93 percent of the dataset, and out of vocabulary words were replaced with "UNK".

5.2 Model

The model uses the same two-model approach, here giving it the explicit name of the encoder-decoder architecture.

"called RNN Encoder–Decoder that consists of two recurrent neural networks (RNN). One RNN encodes a sequence of symbols into a fixed-length vector representation, and the other decodes the representation into another sequence of symbols."

5.3 Model Configuration

A 100-dimensional word embedding was used to represent the input words. The encoder and decoder were configured with 1 layer of 1000 GRU units. 500 Maxout units pooling 2 inputs were used after the decoder. A batch size of 64 sentences was used during training.