Summary

sagisk

February 6, 2022

1 Chapter 7: Recurrent Neural Networks (RNN)

While most of the neural architectures are designed to deal with mutlidimensional data there are some like RNN that deals with sequential data represented both by symbols and by real-valued variables. Some examples of sequential data are time-series, text, and biological data.

Bag of Words: Many seqeunces are processed as bag of words. In such cases the ordering of the words in the document is ignored and hence, the semantic interpretation of the sentences is important. Solution: Use a conventional neural network. Problems:

1. Different length of sentences. It is hard to fix the number of input nodes since if we fix too many the sentences with fewer words will have missing inputs. If the number of input nodes will be few than not all sentences can be fed into the network.

2. Ordering of words matters. Small changes in word ordering can lead to different connotations.

RNNs tackle this problems. Instead of variable number of inputs the network contains a variable number of layers with one-to-one correspondence between the layers of the network and the time-stamps. Moreover, each layer shares the set of parameters. Hence, the same layer is repeated the n times. When n is the sequence length. Hence, the name is recurrent. Inputs: At each time-stamp t the input is x_t, tth element of the sequence. The input can be absent at some time-stamps. Hidden state: $h_t = f(h_{t-1}, x_t) = \tanh(W_{xh}x_t + W_{hh}h_{t-1})$. It is a function of the hidden vector at time $(t-1)$ and the input vector at time t. The function is defined by the shared weight matrices. The recursive nature of the equation shows that $h_1 = f(h_0, x_1), h_2 = f(h_1, x_2) = f(f(h_0, x_1), x_2)...$ So, that $h_1$is a function of x_1 whereas, h_2 is a function of x_1, x_2. In general, h_t represents the p dimensional embedding of the text segment of t words. Weights:

- W_{xh} the weight between input and hidden layer.
- W_{hh} the weights between hidden and hidden layer. (Reccurent nature).
- W_{hy} the weight between hidden and output layer.

Output: The tth d dimensional output $y_t = W_{hy}h_t$ correspond to the tth input in the sequence. y_t is the vector of probabilities of the $(t+1)$th word. y_t can be absent for some time-stamps. Since y_t is a function of h_t than we can say that $y_t = F_t(x_1, x_2, ..., x_t)$.

Backpropagation: Output: At each time-stamp y_t we transform it to probabilities using the softmax function $[\hat{p}_t^1, ..., \hat{p}_t^d] = \text{Softmax}([\hat{y}_t^1, ..., \hat{y}_t^d])$. Loss function: The negative logarithms of the softmax probability of the **correct** words at various time-stamps are aggregated to create the loss function. If j_t is the index of the ground-truth word at time t then: $L = -\sum_{t=1}^{T} \log(\hat{p}_t^{j_t})$ Loss function derivative: $\frac{dL}{d\hat{y}_t^k} = \hat{p}_t^k - I(k, j_t)$ Training:

1. Run the input in the forward direction and compute the errors (negative-log loss of sfotmax layer) at each time-stamp.

2. Compute the gradients by making an assumption that the weights are not shared i.e. $\frac{dL}{dW_{xh}^t}, \frac{dL}{dW_{hh}^t}, \frac{dL}{dW_{hy}^t}$er to emphasize our assumption.

3. Add up the derivative for each time-step of the corresponding weight matrices to get the true partial derivatives: $\frac{dL}{dW_{xh}} = \sum_{t=1}^{T} \frac{dL}{dW_{xh}^t}, \frac{dL}{dW_{hh}} = \sum_{t=1}^{T} \frac{L}{dW_{hh}^t}, \frac{dL}{dW_{hy}} = \sum_{t=1}^{T} \frac{dL}{dW_{hy}^t}$.

Training issues: If the sequence is very long the number of layers will be large and the backpropogation can face convergence problems. The solution is to use truncated backpropagation by doing the updates only over segments of modest length. The problem of vanishing/exploding gradients is quite sharp in case of RNN. Since the RNN tends to have layers of varying depth dependent on the input sequence length the number of layers in RNN can be very big. The partial derivatives of the loss function in different layers has different values that can vary drastically. Moreover, the weigths are shared and the combination of theses can cause serious problems. Let $\Phi()$ be the tanh activation function, $\Phi'(h_t)$ its derivative at the hidden layer t then for the RNN with one unit in each layer we have: $\frac{dL}{dh_t} = \Phi'(h_t)w_{t+1}\frac{dL}{dh_{t+1}}$ where w_{t+1} is the fictional copy of the shared weights. Now we can see that if: $w_t = w < 1$ then the gradient will eventually vanish. Otherwise explode. If $\Phi'()$ is almost always less than 2 the gradient will tend to vanish. Possible solutions:

- Truncated backpropagation.
- Heavy regularization. However, it can limit the potential of the model.
- Gradient clipping: By the value of the gradient where the largest temporal components of the gradient are clipped before adding them. Norm-based clipping: the norm is re-scaled back to some threshold value.
- Higher-order gradients. However, it is computationally expensive. An alternative is to use approximations to the higher-order gradients.
- Layer-normalization: Normalize the activation values $a_t = W_{xh}x_t + W_{hh}h_{t-1}$ before using the activation tanh function. One computes the mean and std of pre-activation values: $u_t = \frac{\sum_{i=1}^{p} a_{ti}}{p}, \sigma_t = \sqrt{\frac{\sum_i^p a_{ti}^2}{p} - u_t^2}$. Then, compute $h_t = \tanh(\frac{\gamma_t}{\sigma_t} \otimes (a_t - u_t) + \beta_t)$ where γ_t is the gain parameter, β_t is the bias parameter and u_t is the vector of sampled means repeated p times.

Other Issues: Knows information only about the past inputs up to a certain point.

1.1 Bidirectional RNN

Knows the past input states as well as the future input states. Hidden states: h_t^f-is responsible for the forward direction, h_t^b-for the backward direction. The forward and backward hidden states does not interact between each other. Both get the same input vector x_t (one-hot encoded) and interact with the same output $\hat{y}_t$. Though this methods get more information about the semantic structure of the text the number of parameters in such a network is bigger. Weights: $W_{xh}^f, W_{hh}^f, W_{hy}^f$ are the forward weight matrices. $W_{xh}^b, W_{hh}^b, W_{hy}^b$ are the backward weight matrices. The recurrence is given as: $h_t^f = \tanh(W_{xh}^f x_t + W_{hh}^f h_{t-1}^f), h_t^b = \tanh(W_{xh}^b x_t + W_{hh}^b h_{t+1}^b), y_t = W_{hy}^f h_t^f + W_{hy}^b h_t^b$

Training: 1. Compute forward and backwards hidden states in independent and separate passes. 2. Compute output states from backwards and forward hidden states. 3. Compute partial derivatives of loss with respect to output states and each copy of the output parameters. 4. Compute partial derivatives of loss with respect to forward states and backwards states independently using backpropagation. Use these computations to evaluate par- tial derivatives with respect to each copy of the forwards and backwards parameters. 5. Aggregate partial derivatives over shared parameters.

1.2 Multilayer Recurrent Networks

The difference of the multilayer RNN in comparison with the usual RNN is that there are multiple hidden layers. Hidden states: h_t^k where k shows to which layer the state belongs to. Weights: W^k where k shows to which layer the weights belong to. The weights are shared among the k th layer but not between different layers. First hidden layer: Is special since it gets inputs both from the input layer at the given time-stamp and from the adjacent hidden state of the previous time-stamp. It's weights are given as $p \times (d+p)$ matrix $W^1 = [W_{xh}^1, W_{hh}^1]$ where we stacked the input-hidden and hidden-hidden weight matrices into one. The input vector can also be represented as the stacked one like $\begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$.

And $h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$ For the remaining layers we have $p \times (p+p)$ matrix $W^k = [W_{hh}^{k-1}, W_{hh}^k]$ and the input is $\begin{pmatrix} h_t^{k-1} \\ h_{t-1}^k \end{pmatrix}$. And $h_t^k = \tanh W^k \begin{pmatrix} h_t^{k-1} \\ h_{t-1}^k \end{pmatrix}$

References