Chapter 10

Recurrent Neural Networks

Contents

10 Recurrent Neural Networks 1
- 10.1 Motivation . . . 2
- 10.2 Neurons with Recurrent . . . 2
- 10.3 Recurrent Neural Network . . . 3
- 10.4 RNN Implementation . . . 3
- 10.5 Unrolling the RNN . . . 3
- 10.6 Sequence Modeling . . . 3
- 10.7 Backpropagation Through Time . . . 4
- 10.8 LSTM Cell . . . 4
- 10.9 LSTM Memory . . . 4
- 10.10 Forget Gate . . . 4
- 10.11 Input Gate . . . 4
- 10.12 Updating Cell State . . . 5
- 10.13 Output Gate . . . 5
- 10.14 Training LSTM . . . 5
- 10.15 LSTM BPTT . . . 5
- 10.16 LSTM Example . . . 6
- 10.17 Gated Recurrent Unit . . . 6
- 10.18 Bidirectional RNN . . . 6
- 10.19 Seq2Seq . . . 6
- 10.20 Limitations of RNN . . . 6
- 10.21 Desired Capabilities of RNN . . . 6

10.1 Motivation

- When the model generates "people", we need a way to tell the model that "several" has already been generated and similarly for the other words.

10.2 Neurons with Recurrent

$$y^t = f(x^t, h^{t-1}) \tag{10.1}$$

10.3 Recurrent Neural Network

- Apply a recurrence relation at every time step t to process a sequence.
- The feedback connection allows information to persist.
- RNNs have hidden state h^t ....

10.4 RNN Implementation

- ...
- Loop through all individual words in the sentence.
- Inside the loop, each word is fed into RNN model along with previous hidden state.
- This generates ...
- Prediction for the final word is the RNN's output after all the prior work have been fed in through the model.

10.5 Unrolling the RNN

$$\begin{aligned} h^t &= g(w_{xh}x^t + w_{hh}h^{t-1}) \\ y^t &= g(w_{hy}h^t) \end{aligned} \tag{10.2}$$

$$L = \sum_{t=0}^{k} L^t \tag{10.3}$$

10.6 Sequence Modeling

- Sequence models have been motivated by the analysis of sequential data such as text sentences, time-series and other discrete sequences data.
- These models are designed to handle sequential information in the same way that CNN are adapted to handle spatial data.
- The key point for sequence models is that the data we are processing are not independently and identically distributed samples and the data carry some dependency due to their sequential ordering.
- Applications of sequence models ...
- Sequence modeling design criteria:

- Handle variable-length sequences.
- Track long-term dependencies.
- Maintain information about the order.
- Share parameters across the sequence.

- RNNs satisfy all of the design criteria for sequence modeling.

10.7 Backpropagation Through Time

10.8 LSTM Cell

10.9 LSTM Memory

- The key of an LSTM is the cell state, the horizontal line running through the top of the diagram.
- The cell state is like a conveyor belt. It's very easy for information to just flow along it unchanged.

10.10 Forget Gate

- It looks at h^{t-1} and x^t, and outputs a number between 0 and 1 for each number in the cell state c^t.
- 1 represents "completely keep this" while a 0 represents "completely get rid of this".

$$f^t = \sigma\left(w_{xf}x^t + w_{hf}h^{t-1} + b_f\right) \tag{10.4}$$

10.11 Input Gate

- First, a sigmoid layer called the "input gate layer" decides which values to update.
- Second, a tanh layer creates a vector of new candidate values c^t, that could be added to the state.

$$\begin{aligned} i^t &= \sigma\left(w_{xi}x^t + w_{hi}h^{t-1} + b_i\right) \\ c'^t &= \tanh\left(w_{xc}x^t + w_{hc}h^{t-1} + b_c\right) \end{aligned} \tag{10.5}$$

10.12 Updating Cell State

- Next, update the old cell state c^{t-1} into the new cell state c^t.
- We use the Hadamard product $\circ$ (element-wise product).

$$c^t = f^t \circ c^{t-1} + i^t \circ c'^t \tag{10.6}$$

10.13 Output Gate

- A sigmoid layer decides what parts of the cell state are going to be output.
- The cell state is then put through tanh (to make the values $\in [-1, 1]$) and this is element-wise multiplied by the output of the sigmoid gate.

$$o^t \tag{10.7}$$

10.14 Training LSTM

- An LSTM network is trained with BPTT.
- In a vanilla RNN, if $h^t = \left(w_{xh}x^t + w_{hh}h^{t-1}\right)$, then its derivative is –

$$\frac{\partial h^t}{\partial h^{t-1}} = w_{hh}\sigma(\cdot)\left(1 - \sigma(\cdot)\right)$$

- However, for LSTM, we have the cell state $c^t = f^t \circ c^{t-1} + i^t \circ c'^t$ and its derivative is –

$$\frac{\partial c^t}{\partial c^{t-1}} = f^t = \sigma\cdot$$

- This helps LSTM preserve a constant error when it is back-propagated.
- The cells learn when to allow data to enter, leave or delete through the iterative process of back-propagating errors and adjusting weights.

10.15 LSTM BPTT

$$o^t = \sigma\left(w_{xo}x^t + w_{ho}\right)$$

10.16 LSTM Example

- Cell state and hidden state represents long-term and short-term memory.
- Forget gate will decide how much of the long-term memory to pass on.

10.17 Gated Recurrent Unit

- Similar performance as LSTM with less computation.
- GRUs have fewer parameters than LSTM, as they lack an output gate.

10.18 Bidirectional RNN

- Output at time t may not only depend on the previous elements in the sequence, but also future elements.
- They are just two RNNs stacked on top of each other. The output is then computed based on the hidden state of both RNNs.

10.19 Seq2Seq

- Consists of two RNNs:
  - The encoder reads the input sequence and outputs a vector.
  - The decoder reads the vector and produces the output sequence.
- Primarily used in NLP applications.

10.20 Limitations of RNN

- Encoding bottleneck
- Slow, no parallelization
- Not long memory

10.21 Desired Capabilities of RNN

- Continuous stream
- Parallelization
- Long memory