Deep Learning III: memory-based layers for learning sequences

Moacir Ponti
ICMC, Universidade de São Paulo

Contact: www.icmc.usp.br/~moacir — moacir@icmc.usp.br

Rio de Janeiro/Brazil – April, 2019
Agenda

When data sequence matters

Basic recurrent layer (RNN)

Long Short Term Memory (LSTM)

Concluding remarks
Non-sequence processing

- Dense and convolutional layers only consider the current example to compute their output.
- In every iteration, the input moves forward, until reaching the output.
What if some output of a layer at iteration $t$ is used as an additional input to the same layer at iteration $t + 1$?
When sequence is important

- This way the output depends not only on the current input, but previous outputs of the same layers
Different configurations of sequences

- One input, sequence output
- Sequence input, one output
- Sequence input, sequence output
Different configurations of sequences

▪ One input, sequence output: e.g. one image is provided and the network outputs a sequence of words describing it
Different configurations of sequences

- **Sequence input, one output**: e.g. sentence (text) is given and the output is a sentiment analysis, into positive or negative content.

\[
\begin{align*}
  x(t) & \rightarrow \text{Layer 1} \rightarrow x_1^{(t)} & \rightarrow \text{Layer 2} \rightarrow x_2^{(t)} & \rightarrow \text{Layer 3} \\
  x(t+1) & \rightarrow \text{Layer 1} \rightarrow x_1^{(t+1)} & \rightarrow \text{Layer 2} \rightarrow x_2^{(t+1)} & \rightarrow \text{Layer 3} \\
  x(t+2) & \rightarrow \text{Layer 1} \rightarrow x_1^{(t+2)} & \rightarrow \text{Layer 2} \rightarrow x_2^{(t+2)} & \rightarrow \text{Layer 3} \rightarrow x_3^{(t+2)}
\end{align*}
\]
Different configurations of sequences

- **Sequence input, sequence output**: e.g. machine translation of sentences in different languages (it may or not have a delay)
When data sequence matters

Basic recurrent layer (RNN)

Long Short Term Memory (LSTM)

Concluding remarks
Recurrent layer: to remember or to forget?
Recurrent layer: to remember or to forget?

\[ h_t = \tanh(W_h h_{t-1} + W_x x_t + b_h) \]
\[ y = (W_y h_t + b_y) \]
Recurrent layer: to remember or to forget?

\[ h_t = \tanh \left( W_h h_{t-1} + W_x x_t + b_h \right) \]

\[ y = (W_y h_t + b_y) \]
Example: predicting next character

Let us define a one-hot vector for characters so that:

- \( h = [1, 0, 0, 0] \)
- \( e = [0, 1, 0, 0] \)
- \( l = [0, 0, 1, 0] \)
- \( o = [0, 0, 0, 1] \)

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Example: predicting next character

```
target chars:  "e"    "l"    "l"    "o"
output layer
            1.0  0.5  0.1  0.2
            2.2  0.3  0.5  -1.5
            -3.0 -1.0  1.9 -0.1
             4.1  1.2 -1.1  2.2

hidden layer
            0.3  1.0  0.1  -0.3
            -0.1 0.3 -0.5  0.9
             0.9  0.1 -0.3  0.7

input layer
            1  0  0  0
            0  1  0  0
            0  0  1  0
            0  0  1  0

input chars: "h"  "e"  "l"  "l"
```

http://karpathy.github.io/2015/05/21/rnn-effectiveness/
Agenda

When data sequence matters

Basic recurrent layer (RNN)

Long Short Term Memory (LSTM)

Concluding remarks
Long Short Term Memory Unit (LSTM)

Understanding LSTM Networks

\[
\begin{align*}
    f_t &= \sigma (W_f \cdot [h_{t-1}, x_t] + b_f) \\
    i_t &= \sigma (W_i \cdot [h_{t-1}, x_t] + b_i) \\
    \tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \\
    C_t &= f_t \ast C_{t-1} + i_t \ast \tilde{C}_t \\
    o_t &= \sigma (W_o \cdot [h_{t-1}, x_t] + b_o) \\
    h_t &= o_t \ast \tanh(C_t)
\end{align*}
\]
Long Short Term Memory Unit (LSTM)

This and following figures are from http://colah.github.io/posts/2015-08-Understanding-LSTMs/
LSTM network: Cell line

- Runs down the entire chain, with minor linear interactions
- LSTM may remove or add information to the cell state, via 3 gates
LSTM network: forget gate

- decide what to cancel out from the cell state
- outputs values between 0 (forget) and 1 (keep entirely) for each value of the cell state vector

\[ f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f) \]
LSTM network: input and update gate

- first, it combines previous output $h_{t-1}$ and the input $x_t$
- then, it filters out those by learning $\tilde{C}_t$, which are candidate values for updating the cell state

\[
i_t = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i)
\]
\[
\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)
\]
LSTM network: update Cell state

\[ C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \]

- now the previous and current cell state are combined
LSTM network: output gate

- decide what to output
- the output is based on the computed cell state $C_t$, which weights the vector formed by the recurrence $h_{t-1}$ and input $x_t$

\[
o_t = \sigma \left( W_o \left[ h_{t-1}, x_t \right] + b_o \right) \\
h_t = o_t \ast \tanh \left( C_t \right)
\]
Concluding remarks

- Recurrent layers are essential when sequential data is concerned.
- It is paramount to format the input to as simple as possible configurations.
- Example: one-hot vectors for words or characters.
Further reading

- Try to look for the Attention Networks: the idea is to let every step of an RNN pick information to look at from some larger collection of information.
- For example, a recurrent net to output caption of an image, it might pick a different part of the image to decide every word it outputs.
References

  http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- C. Olah. Understanding LSTM Networks 
  http://colah.github.io/posts/2015-08-Understanding-LSTMs/