Sequences in Caffe

Jeff Donahue
CVPR Caffe Tutorial
June 6, 2015
Sequence Learning

• Instances of the form \( x = \langle x_1, x_2, x_3, \ldots, x_T \rangle \)

• Variable sequence length \( T \)

• Learn a transition function \( f \) with parameters \( W \):

• \( f \) should update hidden state \( h_t \) and output \( y_t \)

\[
\begin{align*}
    h_0 &:= 0 \\
    \text{for } t = 1, 2, 3, \ldots, T: & \\
    \langle y_t, h_t \rangle &= f_W(x_t, h_{t-1})
\end{align*}
\]
Sequence Learning

Equivalent to a T-layer deep network, unrolled in time
Sequence Learning

- What should the transition function $f$ be?

- At a minimum, we want something **non-linear** and **differentiable**
Sequence Learning

- A “vanilla” RNN:

  \[ h_t = \sigma(W_{hx}x_t + W_{hh}h_{t-1} + b_h) \]
  \[ z_t = \sigma(W_{hz}h_t + b_z) \]

- Problems
  - Difficult to train — vanishing/exploding gradients
  - Unable to “select” inputs, hidden state, outputs
Sequence Learning

Long Short-Term Memory (LSTM)
Proposed by Hochreiter and Schmidhuber, 1997
Sequence Learning

- Allows long-term dependencies to be learned
- Effective for
  - speech recognition
  - handwriting recognition
  - translation
  - parsing

LSTM
(Hochreiter & Schmidhuber, 1997)
Sequence Learning

Exactly remember previous cell state — discard input

LSTM
(Hochreiter & Schmidhuber, 1997)
Sequence Learning

Forget
previous
cell state —
only remember
input

LSTM
(Hochreiter &
Schmidhuber, 1997)
Activity Recognition

sequential input
CNN
LSTM
LSTM
LSTM
LSTM
sequential output
output
a
man
is
talking
EOS
Video Description

Sequential input & output
Sequence learning features now available in Caffe.
Check out PR #2033
“Unrolled recurrent layers (RNN, LSTM)”
Training Sequence Models

- At training time, want the model to predict the next time step given all previous time steps: \( p(w_{t+1} \mid w_{1:t}) \)
- Example: A *bee buzzes.*

<table>
<thead>
<tr>
<th>input</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>&lt;BOS&gt;</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>bee</td>
</tr>
<tr>
<td>3</td>
<td>buzzes</td>
</tr>
<tr>
<td></td>
<td>&lt;EOS&gt;</td>
</tr>
</tbody>
</table>
**Sequence Input Format**

- First input: “cont” (continuation) indicators \((T \times N)\)
- Second input: data \((T \times N \times D)\)

<table>
<thead>
<tr>
<th>N = 2, T = 6</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>batch 1</strong></td>
</tr>
<tr>
<td>a</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>cat</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td><strong>batch 2</strong>...</td>
</tr>
<tr>
<td>buzzes</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>tree</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>
Sequence Input Format

- Inference is exact over infinite batches
- Backpropagation approximate — truncated at batch boundaries

\[ N = 2, \ T = 6 \]

**batch 1**

<table>
<thead>
<tr>
<th>a</th>
<th>dog</th>
<th>fetches &lt;EOS&gt;</th>
<th>the</th>
<th>bee</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cat</th>
<th>in</th>
<th>a</th>
<th>hat</th>
<th>&lt;EOS&gt;</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**batch 2**

<table>
<thead>
<tr>
<th>buzzes</th>
<th>&lt;EOS&gt;</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>tree</th>
<th>falls</th>
<th>&lt;EOS&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Sequence Input Format

• Words are usually represented as one-hot vectors

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>the</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>am</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cat</td>
<td>3999</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bee</td>
<td>4000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dog</td>
<td>4001</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>run</td>
<td>7999</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

8000D vector; all 0s except index of word

“bee”
Sequence Input Format

• EmbedLayer projects one-hot vector

8000D vector; all 0s except index of word “bee”

4000 (index of the word “bee” in our vocabulary)

500D embedding of “bee”

layer {
  type: "Embed"
  embed_param {
    input_dim: 8000
    num_output: 500
  }
}
A female tennis player in action on the court.

A group of young men playing a game of soccer.

A man riding a wave on top of a surfboard.
A black and white cat is sitting on a chair.

A large clock mounted to the side of a building.

A bunch of fruit that are sitting on a table.
Video Description

N timesteps: watch video

M timesteps: produce caption

Venugopalan et al., “Sequence to Sequence -- Video to Text,” 2015.
http://arxiv.org/abs/1505.00487