Recurrent Neural Networks 1: Modelling sequential data

Hakan Bilen
Machine Learning Practical — MLP Lecture 9
12 November 2019
Sequential Data

The man at bat readies to swing at the pitch while the umpire looks on.
Sequential Data

- We often wish to model data that is a sequence or trajectory through time, for instance, text (sequences of characters/words), audio signals, currency exchange rates, motion of a vehicle.
- What should a good model of sequential data include?

- Track long-term dependencies
  - Eg "Hinton is one of the fathers of deep learning. He was co-author of a highly cited paper ..."

- Learn invariances across time
  - Eg "I went to Nepal in 2009" and "In 2009, I went to Nepal"

- Handle variable-length sequences
  - Convolutional networks model invariances across space – can we do something similar to model invariances across time?
    - Yes - time-delay neural networks
We often wish to model data that is a sequence or trajectory through time, for instance, text (sequences of characters/words), audio signals, currency exchange rates, motion of a vehicle.

What should a good model of sequential data include?

- Track long-term dependencies
  Eg “Hinton is one of the fathers of deep learning. He was co-author of a highly cited paper ...”
- Learn invariances across time
  Eg “I went to Nepal in 2009” and “In 2009, I went to Nepal”
- Handle variable-length sequences

Convolutional networks model invariances across space – can we do something similar to model invariances across time?

- Yes - time-delay neural networks
Modelling sequences

- Imagine modelling a time sequence of 3D vectors
Modelling sequences

- Imagine modelling a time sequence of 3D vectors
- Can model fixed context with a feed-forward network with previous time input vectors added to the network input
Imagine modelling a time sequence of 3D vectors.

Can model fixed context with a feed-forward network with previous time input vectors added to the network input.

Model using 1-dimension convolutions in time - time-delay neural network (TDNN).
Modelling sequences

- Imagine modelling a time sequence of 3D vectors
- Can model fixed context with a feed-forward network with previous time input vectors added to the network input
- Model using 1-dimension convolutions in time - time-delay neural network (TDNN)
- Network takes into account a finite context
TDNNs in action – Speech-to-text


• TDNN operating on 23 frames of context
• Without sub-sampling (blue+red)
• With sub-sampling (red)
https://arxiv.org/abs/1609.03499
CNNs can scale to images with large width and height
Some CNNs can process images of variable size
RNNs vs CNNs

- CNNs can scale to images with large width and height
- Some CNNs can process images of variable size
- RNNs can scale to much longer sequences than would be practical for networks without sequence-based specialization
- RNNs can also process sequences of variable length
Dynamical systems

- State of the system \( s^{(t)} = f(s^{(t-1)}; W) \)
- Unfolding \( s^{(3)} = f(s^{(2)}; W) = f(f(s^{(1)}; W); W) \)
- Assumption: The rules of the universe that govern the dynamic system does not change over time
- Property 1: Regardless of the sequence length, the learned model always has the same input size
- Property 2: It uses the same transition function \( f \) with the same weights \( W \) at every time step
Unfolding computational graphs

- Observation as input $x$
- State units are not visible, now hidden $h(t) = f(h(t-1), x(t); W)$
- State units as memory – remember things for (potentially) an infinite time
- State units as information compression – compress the history (sequence observed up until now) into a state representation
Simple RNN: Recurrent hidden units

\[
\begin{align*}
    y(t-1) & \quad E(t-1) & \quad o(t-1) & \quad h(t-1) \\
    y(t) & \quad E(t) & \quad o(t) & \quad h(t) \\
    y(t+1) & \quad E(t+1) & \quad o(t+1) & \quad h(t+1)
\end{align*}
\]
Simple RNN with recurrent hidden unit

- Hidden state $h^{(t)}$

$$h^{(t)} = \tanh \left( W_{hx} x^{(t)} + W_{hh} h^{(t-1)} \right)$$

- Output $o^{(t)}$

$$o^{(t)} = \text{softmax} \left( W_{oh} h^{(t)} \right)$$

- Error $E^{(t)}$ for ground-truth $y^{(t)}$

$$E^{(t)} = CE \left( o^{(t)}, y^{(t)} \right)$$
Training simple RNN with recurrent hidden unit

- View an RNN for a sequence of $\tau$ inputs as a $\tau$-layer network with shared weights
- Train an RNN by doing backprop through this unfolded network
Training simple RNN with recurrent hidden unit

\[
\begin{align*}
\frac{\partial E^{(t+1)}}{\partial W_{oh}} &= \frac{\partial E^{(t+1)}}{\partial o^{(t+1)}} \frac{\partial o^{(t+1)}}{\partial W_{oh}} \\
\frac{\partial E^{(t+1)}}{\partial W_{hx}} &= \frac{\partial E^{(t+1)}}{\partial o^{(t+1)}} \frac{\partial o^{(t+1)}}{\partial W_{hx}} \left( \frac{\partial h^{(t+1)}}{\partial W_{hx}} + \frac{\partial h^{(t+1)}}{\partial h^{(t)}} \frac{\partial h^{(t)}}{\partial W_{hx}} + \ldots \right) \\
\frac{\partial E^{(t+1)}}{\partial W_{hh}} &= \frac{\partial E^{(t+1)}}{\partial o^{(t+1)}} \frac{\partial o^{(t+1)}}{\partial W_{hh}} \left( \frac{\partial h^{(t+1)}}{\partial W_{hh}} + \frac{\partial h^{(t+1)}}{\partial h^{(t)}} \frac{\partial h^{(t)}}{\partial W_{hh}} + \frac{\partial h^{(t+1)}}{\partial h^{(t-1)}} \frac{\partial h^{(t-1)}}{\partial W_{hh}} + \ldots \right)
\end{align*}
\]
Back-propagation through time (BPTT)

• We can train a network by unfolding and back-propagating through time, summing the derivatives for each weight as we go through the sequence

• More efficiently, run as a recurrent network
  • cache the unit outputs at each timestep
  • cache the output errors at each timestep
  • then backprop from the final timestep to zero, computing the derivatives at each step
  • compute the weight updates by summing the derivatives across time

• Expensive – backprop for a 1,000 item sequence equivalent to a 1,000-layer feed-forward network

• Truncated BPTT – backprop through just a few time steps (e.g. 20)
Gradient clipping

Gradient clipping can make gradient descent perform more reasonably in the vicinity of extremely steep cliffs

\[ \text{if } \|g\| > v, \text{ then } g \leftarrow \frac{vg}{\|g\|} \]

- Can be used with RMS-Prop and Adam

Image credit: Goodfellow et al. 2016
Recurrence only through the output

\[ y \xrightarrow{W_{oh}} E \xrightarrow{W_{ho}} o \]
\[ h \xrightarrow{W_{hx}} x \]

Unfold

\[ o^{(t-1)} \xrightarrow{W_{ho}} o^{(t)} \xrightarrow{W_{ho}} o^{(t+1)} \]
\[ E^{(t-1)} \xrightarrow{W_{ho}} E^{(t)} \xrightarrow{W_{ho}} E^{(t+1)} \]
\[ h^{(t-1)} \xrightarrow{W_{hx}} h^{(t)} \xrightarrow{W_{hx}} h^{(t+1)} \]
\[ x^{(t-1)} \xrightarrow{W_{hx}} x^{(t)} \xrightarrow{W_{hx}} x^{(t+1)} \]
Recurrence only through the output

- The feedback connection is from the output to the hidden layer

\[ h(t) = \tanh \left( W_{hx} x(t) + W_{ho} h(t-1) \right) \]

- More restricted hidden recurrent network, o is the only information it is allowed to send to the future
Teacher forcing in output recurrent network

- Teacher forcing: the model receives the ground truth output $y^{(t)}$ during training
- Each loss function at time $t$ is decoupled
- Training can thus be parallelized: the gradient for each step $t$ computed in isolation
- Disadvantage: Train and test time difference
- One solution is to train with both teacher-forced inputs ($y$) and free-running inputs ($o$)
Sequence input, single output ("seq2vec")

- RNN with a single output at the end of the sequence
- Summarize a sequence and produce a fixed-size representation used as input for further processing
- Eg sentiment analysis
Single input, sequence output ("vec2seq")

- RNN with single input and sequence output
- Each element $y(t)$ of the observed output sequence serves both as input (for the current time step) and, during training, as target (for the previous time step)
- Eg image captioning
We want to translate a sentence from one language to another one. Is the architecture below suitable for this problem?
• seq2seq = seq2vec + vec2seq
• Encoder takes in a sequence of inputs and compresses it into a context vector $c$
• Decoder takes in a context vector and outputs a sequence
• Eg machine translation

Bidirectional RNN

- So far, “causal” relation, the state at time $t$ captures only information from the past, $x^{(1)}, \ldots, x^{(t-1)}$ and the present input $x^{(t)}$
- In some applications, we want the output $o^{(t)}$ to depend on the whole sequence
- Bidirectional RNN – combine an RNN moving forward in time, with one moving backwards in time
- State units provide a combined representation that depends on both the past and the future
Recurrent neural networks
examples from ancient history
(using “vanilla” RNNs and BPTT)
Example 1: speech recognition with recurrent networks

Example 2: recurrent network language models

T Mikolov et al (2010). “Recurrent Neural Network Based Language Model”, Interspeech
Summary

- Model sequences using finite context using feed-forward networks with convolutions in time (TDNNs, Wavenet)
- Model sequences using infinite context using recurrent neural networks (RNNs)
- Unfolding an RNN gives a deep feed-forward network with shared weights
- Train using back-propagation through time
- (Historical) examples on speech recognition and language modelling
- Reading: Goodfellow et al, chapter 10 (sections 10.1, 10.2, 10.3) http://www.deeplearningbook.org/contents/rnn.html
- Next lecture: LSTM, sequence-to-sequence models, attention