Introduction to Recurrent Neural Networks

Jakob Verbeek
Modeling sequential data with Recurrent Neural Networks

- Compact schematic drawing of standard multi-layer perceptron (MLP)
Modeling sequential data

- So far we considered “one-to-one” prediction tasks
  - Classification: one image to one class label, which digit is displayed 0...9
  - Regression: one image to one scalar, how old is this person?

- Many prediction problems have a sequential nature to them
  - Either in input, in output, or both
  - Both may vary in length from one example to another
Modeling sequential data

- One-to-many prediction
- Image captioning
  - Input: an image
  - Output: natural language description, variable length sequence of words

<table>
<thead>
<tr>
<th>one to one</th>
<th>one to many</th>
<th>many to one</th>
<th>many to many</th>
<th>many to many</th>
</tr>
</thead>
</table>

A brown dog is running through the grass
Modeling sequential data

- **Text classification**
  - **Input:** a sentence
  - **Output:** user rating

---

I liked it...

Author: JustMeOnline from Portugal

14 November 2014

I decided to watch it, because I liked the movie very much, and this TV Series was in the same level of quality, with the same environment, with more cold crimes and a good investigation...
Modeling sequential data

- Machine translation of text from one language to another
  - Sequences of different length on input and output
Modeling sequential data

- Part of speech tagging
  - For each word in sentence predict PoS label (verb, noun, adjective, etc.)

- Temporal segmentation of video
  - Predict action label for every video frame

- Temporal segmentation of audio
  - Predict phoneme labels over time
Modeling sequential data

- Possible to use a k-order autoregressive model over output sequences
  - Limits memory to only k time steps in the past
- Not applicable when input sequence is not aligned with output sequence
  - Many-to-one tasks, unaligned many-to-many

![Diagram showing one-to-one, one-to-many, many-to-one, many-to-many mappings]
Recurrent neural networks

- Recurrent computation of hidden units from one time step to the next
  - Hidden state accumulates information on entire sequence, since the field of view spans entire sequence processed so far
  - Time-invariant function makes it applicable to arbitrarily long sequences

- Similar ideas used in
  - Hidden Markov models for arbitrarily long sequences
  - Parameter sharing across space in convolutional neural networks
  - But has limited field of view: parallel instead of sequential processing
Recurrent neural networks

- Basic example for many-to-many prediction
  - Hidden state linear function of current input and previous hidden state, followed by point-wise non-linearity
  - Output is linear function of current hidden state, followed by point-wise non-linearity

\[
\begin{align*}
  z_t &= \phi(A x_t + B z_{t-1}) \\
  y_t &= \psi(C z_t)
\end{align*}
\]
Recurrent neural network diagrams

- Two graphical representations are used
  - “Unfolded” flow diagram
  - Recurrent flow diagram

  \[
  z_t = \phi \left( A x_t + B z_{t-1} \right) \\
  y_t = \psi \left( C z_t \right)
  \]

- Unfolded representation shows that we still have an acyclic directed graph
  - Size of the graph (horizontally) is variable, given by sequence length
  - Weights are shared across horizontal replications

- Gradient computation via back-propagation as before
  - Referred to as “back-propagation through time” (Pearlmutter, 1989)
Recurrent neural network diagrams

- Deterministic feed-forward network from inputs to outputs
- Predictive model over output sequence is obtained by defining a distribution over outputs given y
  - For example: probability of a word given via softmax of word score
- Training loss: sum of losses over output variables
  - Independent prediction of elements in output given input sequence

\[ L = - \sum_{t=1}^{T} \ln p(w_t|x_{1:t}) \]

\[ z_t = \phi(A x_t + B z_{t-1}) \]

\[ y_t = \psi(C z_t) \]

\[ p(w_t = k|x_{1:t}) = \frac{\exp y_{tk}}{\sum_{v=1}^{V} \exp y_{tv}} \]
More topologies: “deep” recurrent networks

- Instead of a recurrence across a single hidden layer, consider a recurrence across a multi-layer architecture
More topologies: multi-dimensional recurrent networks

- Instead of a recurrence across a single (time) axis, consider a recurrence across a multi-dimensional grid
- For example: axis aligned directed edges
  - Each node receives input from predecessors, one for each dimension
More topologies: bidirectional recurrent neural networks

- Standard RNN only uses left-context for many-to-many prediction
- Use two separate recurrences, one in each direction
  - Aggregate output from both directions for prediction at each time step
- Only possible on a given input sequence of arbitrary length
  - Not on output sequence, since it needs to be predicted/generated
More topologies: output feedback loops

- So far the element in the output sequence at time $t$ was independently drawn given the state at time $t$
  - State at time $t$ depends on the entire input sequence up to time $t$
  - No dependence on the output sequence produced so far
- Problematic when there are strong regularities in output, eg character or words sequences in natural language

\[
p(w_{1:T}|x_{1:T}) = \prod_{t=1}^{T} p(w_t|x_{1:t})
\]
More topologies: output feedback loops

- To introduce dependence on output sequence, we add a feedback loop from the output to the hidden state.

\[ p(y_{1:T}|x_{1:T}) = \prod_{t=1}^{T} p(y_t|x_{1:t}, y_{1:t-1}) \]

- Without output-feedback, the state evolution is a deterministic non-linear dynamical system.

- With output feedback, the state evolution becomes a stochastic non-linear dynamical system.
  - Caused by the stochastic output, which flows back into the state update.
How do we generate data from an RNN?

- RNN gives a distribution over output sequences

- Sampling: sequentially sample one element at a time
  - Compute state from current input and previous state and output
  - Compute distribution on current output symbol
  - Sample output symbol

- Compute maximum likelihood sequence?
  - Not feasible with feedback since output symbol impacts state

- Marginal distribution on n-th symbol
  - Not feasible: marginalize over exponential nr. of sequences

- Marginal probability of a symbol appearing anywhere in seq.
  - Not feasible: average over all marginals
Approximate maximum likelihood sequences

- Exhaustive maximum likelihood search exponential in sequence length
- Use Beam Search, computational cost linear in
  - Beam size $K$, vocabulary size $V$, (maximum) sequence length $T$
Ensembles of networks to improve prediction

- Averaging predictions of several networks can improve results
  - Trained from different initialization and using different mini-batches
  - Possibly including networks with different architectures, but not per se
  - For CNNs see eg [Krizhevsky & Hinton, 2012] [Simonyan & Zisserman, 2014]

- For RNN sequence prediction
  - Train RNNs independently
  - “Run” RNNs in parallel for prediction, updating states with common seq.
  - Average distribution over next symbol
  - Sample or beam-search based on av. distribution
How to train an RNN without output feedback?

- Compute full state sequence given the input (deterministic given input)
- Compute loss at each time step w.r.t. ground truth output sequence
- Backpropagation (“through time”) to compute gradients w.r.t. loss
How to train an RNN with output feedback?

- Compute state sequence given input and ground-truth output, deterministic due to known and fixed output.
- Loss at each time step wrt ground truth output seq, backprop through time.
- Note discrepancy between train and test:
  - Train: predict next symbol from ground-truth sequence so far.
  - Test: predict next symbol from generated sequence so far.
    - Might deviate from observed ground-truth sequences.
Scheduled sampling for RNN training

- Compensate discrepancy between train and test procedure by training from generated sequence [Bengio et al. NIPS, 2015]
  - Learn to recover from partially incorrect sequences
- Directly training from sampled sequences does not work well in practice
  - At the start randomly initialized model generates random sequences
  - Instead, start by training from ground-truth sequence, and progressively increase probability to sample generated symbol in the sequence
Scheduled sampling for RNN training

- Evaluation image captioning
  - Image in, sentence out
  - Higher scores are better
- Scheduled sampling improves baseline, also in ensemble case
- Uniform Scheduled Sampling: sample uniform instead of using model
  - Already improves over baseline, but not as much as using model
- Always sampling gives very poor results, as expected

<table>
<thead>
<tr>
<th>Approach vs Metric</th>
<th>BLEU-4</th>
<th>METEOR</th>
<th>CIDER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>28.8</td>
<td>24.2</td>
<td>89.5</td>
</tr>
<tr>
<td>Baseline with Dropout</td>
<td>28.1</td>
<td>23.9</td>
<td>87.0</td>
</tr>
<tr>
<td>Always Sampling</td>
<td>11.2</td>
<td>15.7</td>
<td>49.7</td>
</tr>
<tr>
<td>Scheduled Sampling</td>
<td>30.6</td>
<td>24.3</td>
<td>92.1</td>
</tr>
<tr>
<td>Uniform Scheduled Sampling</td>
<td>29.2</td>
<td>24.2</td>
<td>90.9</td>
</tr>
<tr>
<td>Baseline ensemble of 10</td>
<td>30.7</td>
<td>25.1</td>
<td>95.7</td>
</tr>
<tr>
<td>Scheduled Sampling ensemble of 5</td>
<td>32.3</td>
<td>25.4</td>
<td>98.7</td>
</tr>
</tbody>
</table>
Limitations recurrent networks

- Recurrent net can be unrolled as deep network with shared parameters
  - As deep as the number of time steps of the RNN
  - Very deep for very long sequences

- Gradients of “deep” layers (far from input) computed via chainrule as product of Jacobians between layers (time-steps)
  - Product of Jacobians tend to either “explode” to inf. or “vanish” to zero
  - Similar effect observed in non-recurrent networks

- Approaches to address this issue
  - Non-recurrent case: add skip connections from earlier layers towards output: Residual networks, dense networks
  - Introduction of “gates” that shield a hidden unit from input and/or output for several layers, effectively shortening the depth for that unit
Long short-term memory (LSTM) cells

- LSTM consist of hidden state $h$ and a “memory cell” $c$
- Gates are used to modulate the state updates

[Hochreiter & Schmidhuber, Neural Computation, 1997]
Long short-term memory (LSTM) cells

- Introduced by Hochreiter & Schmidhuber (Neural Computation, 1997)
- LSTM defines a dynamical system on hidden state \( h \) and a “memory cell” \( c \)
- Involves a number of additional processing elements
  - Cell update: can forget previous cell state, can ignore input

\[
C_t = f_t \ast C_{t-1} + i_t \ast \tilde{C}_t
\]
Long short-term memory (LSTM) cells

- Introduced by Hochreiter & Schmidhuber (Neural Computation, 1997)
- LSTM defines a dynamical system on hidden state $h$ and a “memory cell” $c$
- Involves a number of additional processing elements
  - Forget gate $f$: “remember” or “forget” previous cell state $c$

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
Long short-term memory (LSTM) cells

- Introduced by Hochreiter & Schmidhuber (Neural Computation, 1997)
- LSTM defines a dynamical system on hidden state $h$ and a “memory cell” $c$
- Involves a number of additional processing elements
  - Input gate $i$: controls flow of input to cell state
  - Input modulator $\tilde{C}$, maps input and previous state to cell state update

\[
\begin{align*}
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)
\end{align*}
\]
Long short-term memory (LSTM) cells

- Introduced by Hochreiter & Schmidhuber (Neural Computation, 1997)
- LSTM defines a dynamical system on hidden state $h$ and a “memory cell” $c$
- Involves a number of additional processing elements
  - Output gate $o$, controls flow of cell state to output
  - Output vector also passed to next time step of LSTM unit

\[ o_t = \sigma \left( W_o \left[ h_{t-1}, x_t \right] + b_o \right) \]
\[ h_t = o_t \times \text{tanh} \left( C_t \right) \]
Gated Recurrent Unit (GRU) cells

- GRU is simplified gated RNN as compared to LSTM
  [Cho et al., Empirical Methods in Natural Language Processing, 2014]
- Two gates, single state signal
  - Forget gate: $z$
  - Read gate: $r$

$$z_t = \sigma (W_z \cdot [h_{t-1}, x_t])$$
$$r_t = \sigma (W_r \cdot [h_{t-1}, x_t])$$
$$\tilde{h}_t = \tanh (W \cdot [r_t \ast h_{t-1}, x_t])$$
$$h_t = (1 - z_t) \ast h_{t-1} + z_t \ast \tilde{h}_t$$
Examples of character-level LSTM language model

- Training data: all Paul Graham essays, about 1 million characters
- Random sample from the trained model:

"The surprised in investors weren't going to raise money. I'm not the company with the time there are all interesting quickly, don't have to get off the same programmers. There's a super-angel round fundraising, why do you can do. If you have a different physical investment are become in people who reduced in a startup with the way to argument the acquirer could see them just that you're also the founders will part of users' affords that and an alternation to the idea. [2] Don't work at first member to see the way kids will seem in advance of a bad successful startup. And if you have to act the big company too."

- Learns to spell words, as well as long range grammatical dependencies

[examples taken from Andrej Karpathy]
Examples of character-level LSTM language model

- Training data: all of Shakespeak (4.4 MB)
- Random sample from the trained model:

  **PANDARUS:**
  Alas, I think he shall be come approached and the day
  When little srain would be attain'd into being never fed,
  And who is but a chain and subjects of his death,
  I should not sleep.

  **Second Senator:**
  They are away this miseries, produced upon my soul,
  Breaking and strongly should be buried, when I perish
  The earth and thoughts of many states.

  **DUKE VINCENTIO:**
  Well, your wit is in the care of side and that.

  **Second Lord:**
  They would be ruled after this chamber, and
  my fair nues begun out of the fact, to be conveyed,
  Whose noble souls I'll have the heart of the wars.

  **Clown:**
  Come, sir, I will make did behold your worship.

  **VIOLA:**
  I'll drink it.

- Specific style structure is also captured by the model
Examples of character-level LSTM language model

- **Training data:** linux source code (474 MB)
- **Very long range dependencies on bracket structure**

```c
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void) {
    int error;
    if (fd == MARN_EPT) {
        /*
         * The kernel blank will coeld it to userspace.
         */
        if (ss->segment < mem_total)
            unblock_graph_and_set_blocked();
        else
            ret = 1;
        goto bail;
    }
    segaddr = in_SB(in.addr);
    selector = seg / 16;
    setup_works = true;
    for (i = 0; i < blocks; i++) {
        seq = buf[i++];
        bpf = bd->bd.next + i * search;
        if (fd) {
            current = blocked;
        }
    }
    rw->name = "Getjbbregs";
    bprm_self_clearl(&iv->version);
    regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 12;
    return segtable;
}
```
Case study: image captioning

- Given image generate descriptive English sentence

a brown dog is running through the grass
Image captioning with encoder-decoder system

- **Encoder**: CNN takes image and maps it into a vector
  - For example, fully connected layer of VGG16 network
    - CNN pre-trained on ImageNet classification task (>1 million images)
Image captioning with encoder-decoder system

- Decoder: RNN takes CNN image vector to initialize RNN state
- Typical configuration:
  - Single layer of 512 GRUs
  - Output feedback to ensure coherent sentence
Image captioning with encoder-decoder system

- Example output (Vinyals et al., CVPR 2015)
Encoder-decoder machine translation

- Translation of a sentence into another language
  - Input and output of different length

I decided to watch it, because I liked the movie very much, and this TV Series was in the same level of quality, with the same environment, with more cold crimes and a good investigation.

Je décidai de le regarder, parce que j’ai aimé le film beaucoup, et que ce téléviseur série a été dans le même niveau de qualité, avec le même environnement, avec plus de crimes froides et une bonne enquête.
Encoder-decoder machine translation

- Read source sentence with **encoder** RNN (Sutskever et al., NIPS 2014)
  - Can use bidirectional RNN since input sequence is given
- Generate target sentence with **decoder** RNN
  - Uses a different set of parameters
  - Uses output feedback to ensure output coherency
- Meaning of source sentence encoded in the RNN state vector passed between encoder and decoder
  - For the captioning model, a CNN is used as an image encoder
Encoder-decoder machine translation

- Decoder learns word embedding matrix $G$ for output feedback
  
  \[ z_{i+1} = \phi(W z_i + G u_i) \]

- Decoder learns word embedding matrix $F$ for word probabilities via softmax normalization
  
  \[ p_i = \sigma(F z_i) \]

- Encoder learns a word embedding matrix $E$

- Columns contain word vector embedding
  
  \[ s_i = E w_i \]
Encoder-decoder machine translation

- Trained from “aligned” corpus of matching source-target sentences
- Encoder and decoder can be learned on multiple language pairs in parallel
  - (English to French) and (Dutch to French) use same decoder
  - (English to French) and (English to Dutch) use same encoder
- Generalizes to translation between new language pairs for which no aligned training corpus was available
Encoder-decoder machine translation

- PCA projection of LSTM encoder state after reading a sentence
  - Word order important for meaning, captured in encoder state vector
Attention mechanisms in RNNs

- Encoder-decoder based models compress the entire input into a single vector
  - Difficult to store all details as the sentences grow longer

- Sequential nature of RNN updates makes that start of sentence is less well encoded into the RNN state
  - Using bi-directional RNN helps, but not in the middle of the sentence...
Attention mechanisms in RNNs

- Let decoder attend to part of the input for each state update
  - Selectively: based on current state and input representation
  - Should work for input sequences of variable size
- Sub-network takes state and input encoding, computes attention weights
  - Soft-max over candidate positions in the input
- Feed weighted sum of inputs to the state update

\[
a_{ij} = \sigma(z_i, h_j)
\]

\[
c_i = \sum_{j=1}^{T} a_{ij} h_j
\]

\[
z_{i+1} = \phi(z_i, c_i, u_i)
\]

[Bahdanau et al., ICLR'15]
Attention mechanisms in RNNs

- Example correspondences identified by attention mechanism
  - Trained from sentence-level supervision, no word correspondences
Attention mechanisms in image captioning

- Without attention image content encoded into vector output of CNN
  - Decoder cannot “look back” at image
- Attention can be used to focus decoder model on parts of input [Xu et al, ICML’15]
Attention mechanisms in image captioning

- What are the image “parts” that we should be looking at? [Pedersoli et al, ICCV’17]

- Activation grid: the locations in a convolutional CNN layer

- Object proposals: plausible object locations predicted by external method

- Spatial transformer: regress deformation of default boxes at locations in activation grid
Attention mechanisms in image captioning

- Examples of generated sentences together with the attention regions
  - Region width proportional to attention weight [Pedersoli et al., ICCV’17]
Further reading

- “Pattern Recognition and Machine Learning”
  Chris Bishop.

- “Supervised Sequence Labelling with Recurrent Neural Networks”
  Alex Graves, 2012 (free online)

- “Deep Learning”
  Ian Goodfellow, Yoshua Bengio, Aaron Courville.
  http://www.deeplearningbook.org/