Deep learning: sequence predictions

Benoit Favre <benoit.favre@univ-amu.fr>

Aix-Marseille Université, LIF/CNRS

last generated on January 15, 2018

What is sequence prediction

- Make a sequence of decisions dependent on an input
  - Decisions are interdependent
  - Sometimes input synchroneous (tagging), aligned (machine translation), or lously coupled (chatbot)

$$X = x_1 x_2 x_3 \dots x_n$$
$$Y = y_1 y_2 \dots y_m$$
$$Y = \text{neural\_network}(X)?$$

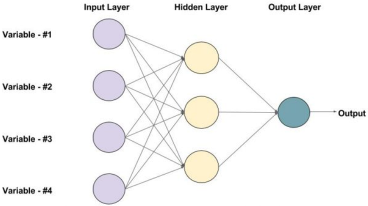
- Main deep learning benefits
  - Automatic differentiation
  - Various architectures
  - Representation learning (automate feature extraction)

Multilayer perceptron

- Predict y_i from x_i

$$y_i = softmax(tanh(W_2 z_i + b_2)) \qquad softmax_k(x) = \frac{e^{x_k}}{\sum_j e^{x_j}}$$

$$z_i = tanh(W_1 x_i + b_1)$$



An example of a Feed-forward Neural Network with one hidden layer (with 3 neurons)

- No dependency on neighboring predictions
  - x_i can include a range of features around i
  - Can include forward dependency $y_{i-1} \subset x_i$

Basic RNNs

- Back to the $Y = \text{neural\_network}(X)$ notation
  - $x = x_1 \dots x_n$ is a sequence of observations
  - $y = y_1 \dots y_n$ is a sequence of labels we want to predict
  - $h = h_0 \dots h_n$ is a hidden state (or history for language models)
  - t is discrete time (so we can write x_t for the t-th timestep
- We can define a RNN as

$$h_0 = 0 \quad (1)$$

$$h_t = tanh(Wx_t + Uh_{t-1} + b) \quad (2)$$

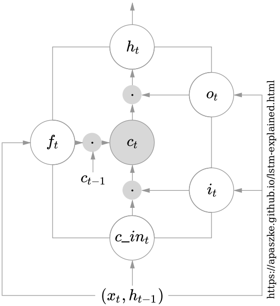
$$y_t = softmax(W_o h_t + b_o) \quad (3)$$

Long-short term memory (LSTM)

- Vanishing gradient
  - In RNNs, the gradient gets very small as it propagates backward
  - Idea: allow shortcuts towards far past
- Gating mechanism

$$g = f(x_t, h_t) \in [0, 1]$$
$$x_{\text{gated}} = g \odot x_t$$

- LSTMs have two hidden states: h and c



LSTM Math

- LSTM

$$
\begin{aligned}
i_t &= \sigma(W_i x_t + U_i h_t + b_i) && \text{input} \\
f_t &= \sigma(W_f x_t + U_f h_t + b_f) && \text{forget} \\
o_t &= \sigma(W_o x_t + U_o h_t + b_o) && \text{output} \\
c'_t &= \tanh(W_c x_t + U_c h_t + b_c) && \text{cell state} \\
c_{t+1} &= f_t \odot c_t + i_t \odot c'_t \\
h_{t+1} &= o_t \odot \tanh(c_{t+1}) \\
\mathrm{LSTM}(x_t, h_t, c_t) &= h_{t+1}
\end{aligned}
$$

- LSTMs output their hidden state like simple RNNs
  - Need to add a dense layer to predict labels

LSTM: how can it memorize things?

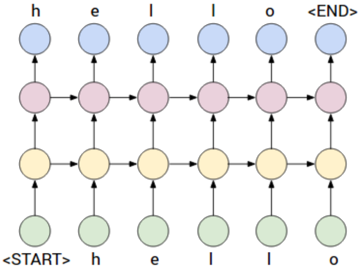
- Let's have a closer look at the gated output

$$\text{cell}_{t+1} = \text{forget}_t \odot \text{cell}_t + \text{input}_t \odot \text{cell}'_t$$
$$\text{hidden}_{t+1} = \text{output}_t \odot \tanh(\text{cell}_{t+1})$$

- Interpretation
  - if $\text{forget}_t = 1$ and $\text{input}_t = 0$: previous cell state is used
  - if $\text{forget}_t = 0$ and $\text{input}_t = 1$: previous cell state is ignored
  - if $\text{output}_t = 1$: output is set to cell state
  - if $\text{output}_t = 0$: output is set to 0

Stacked RNNs

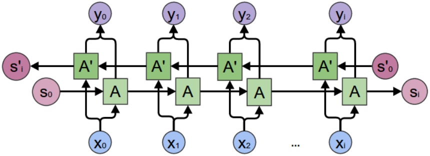
- Increasing hidden state size is very expensive
  - U is of size $(hidden \times hidden)$
  - Can feed the output of a RNN to another RNN cell
  - $\rightarrow$ Multi-resolution analysis, better generalization



- Highway connections create shortcuts between layers
  - $gate_l = \sigma(W_g h_{l-1})$
  - $h_l = \mathrm{LSTM}(h_{l-1}) \circ gate_l + h_{l-1} \circ (1 - gate_l)$

Bidirectional networks

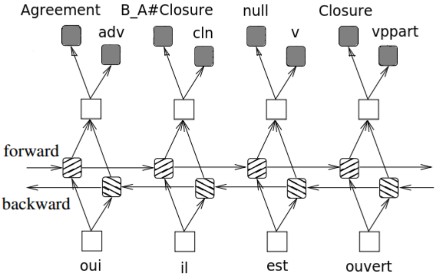
- RNN make predictions independent of the future observations
  - Need to look into the future
- Idea: concatenate the output of a forward and backward RNN
  - The decision can benefit from both past and future observations
  - Only applicable if we can wait for the future to happen



- Not strictly necessary (Time-Delay Neural Networks)

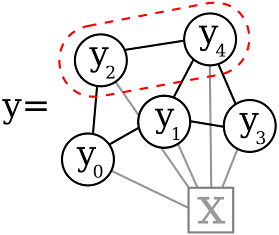
Multi-task learning

- Can we build better representations by training the NN to predict different things?
  - Share the weights of lower system, diverge after representation layer
  - Also applies to feed forward neural networks
- Example: semantic tagging from words
  - Train system to predict low-level and high-level syntactic labels, as well as semantic labels
  - Need training data for each task
  - At test time only keep output of interest



Conditional Random Fields

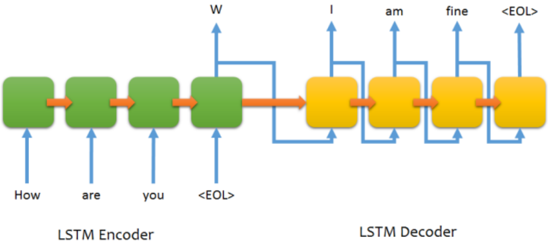
- Conditional random fields as a loss function
  - Generate emissions with NN
  - Generate transitions with NN
  - Backpropagate CRF loss through NNs



[Artieres et al, 2010; Huang et al, 2015...]

Encoder-decoder framework

- Generalisation of the conditioned language model
  - Build a representation, then generate sentence
  - Also called the seq2seq framework

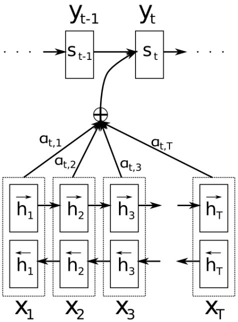


- Basis of neural machine translation (NMT)
  - *Les carottes étaient cuites* → *It was game over*
  - Output can be synchroneous
- Can model non-sequential phenomena
  - "Grammar As a Foreign Language" [Vynials et al, 2014]

Attention mechanisms

- Loosely based on human visual attention mechanism
  - Let neural network focus on aspects of the input to make its decision
  - Learn what to attend based on what it has produced so far
  - More of a mechanism for memorizing the input

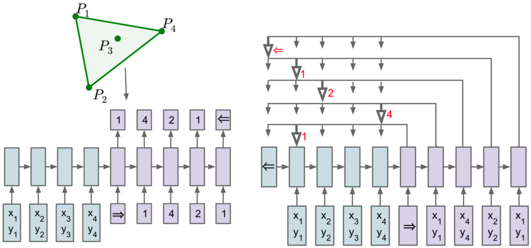
$$
\begin{aligned}
enc_j &= \text{encoder hidden state} \\
dec_t &= \text{decoder hidden state} \\
u_t^j &= v^T tanh(W_e enc_j + W_d dec_t) \\
&\quad \forall j \in [1..n] \\
\alpha_t &= softmax(u_t) \\
s_t &= dec_t + \sum_j \alpha_t^j enc_j \\
y_t &= softmax(W_o s_t + b_o)
\end{aligned}
$$



- Structured attention networks [Kim et al, 2017]
  - Latent variables over the input (segments, trees...)

Pointer networks

- Decision is an offset in the input
  - Number of classes dependent on the length of the input
  - Decision depends on hidden state in input and hidden state in output
  - Can learn simple algorithms, such as finding the convex hull of a set of points



(a) Sequence-to-Sequence (b) Ptr-Net

- Multilingual Language Processing From Bytes [Gillick et al, 2015]

Do we really need RNNs

- "Attention is all you need" [Vaswani et al, 2017]
  - Multiple layers of attention
- Position encoding
  - For position i, dimension j (total d, $k = 10000$)
  - $\mathrm{pe}_{i+k} = Linear(\mathrm{pe}_i)$

$$\mathrm{pe}_{i,2j} = sin\left(\frac{i}{k^{\frac{2j}{d}}}\right)$$

$$\mathrm{pe}_{i,2j+1} = cos\left(\frac{i}{k^{\frac{2j}{d}}}\right)$$

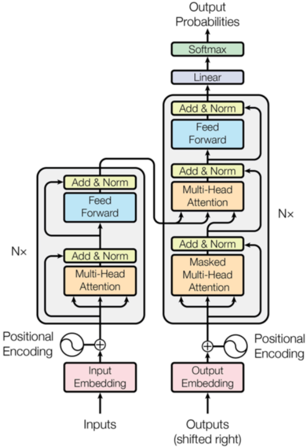
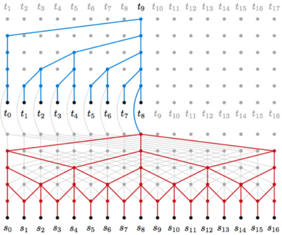


Figure 1: The Transformer - model architecture.

Explore other structures?

- WaveNet architecture
  - Extract long-term relations



- Account for parse tree
  - Generate annotations of the tree node

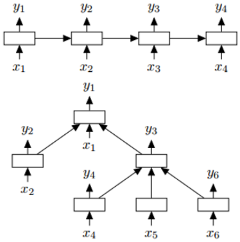


Figure 1: **Top:** A chain-structured LSTM network. **Bottom:** A tree-structured LSTM network with arbitrary branching factor.