Machine learning
Language models

Katya Artemova

CS HSE

February 25, 2019

Overview

Language model

1. Compute the probability of a sequence of words:

$$P(w_1, w_2, \ldots, w_n)$$

2. Predict next word:

$$P(w_n | w_1, w_2, \ldots, w_{n-1})$$

LMs help are used to:

- Machine translation: choose best translation
- Spell checking: find incorrect word
- Speech recognition: choose best transcription
- Predict next word in your smartphone
- Generate poems, summaries, answers, etc.

Overview

А. А. Марков (1886).

Markov assumptions

1. Chain rule:

$$P(W) = P(w_1, w_2, \dots, w_n) = \prod_i P(w_i | w_1, \dots, w_{i-1})$$

2. Maximum likelihood estimates of probabilities:

$$P(w_i | w_1, \dots, w_{i-1}) = \frac{\text{count}(w_1, w_2, \dots, w_i)}{\text{count}(w_1, \dots, w_{i-1})}$$

3. Markov assumption (k-th order):

$$P(w_i | w_1, \dots, w_{i-1}) \approx P(w_i | w_{i-k}, \dots w_{i-1})$$

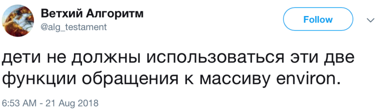
n-gram models

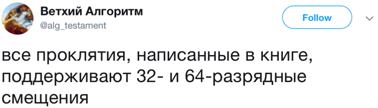
1. Unigram models: $P(W) = P(w_1, \ldots, w_n) \approx \prod_i P(w_i)$
2. Bigram models: $P(W) = P(w_1, \ldots, w_n) \approx \prod_i P(w_i|w_{i-1})$
3. Perplexity: $PP(W) = P(w_1, w_2, \ldots w_n)^{-\frac{1}{N}}$
   The lower perplexity, the better the model predicts an unseen test
4. Smoothing: $P(w_i|w_1, \ldots, w_{i-1}) = \frac{\text{count}(w_1,w_2,\ldots,w_i)+1}{\text{count}(w_1,\ldots,w_{i-1})+\alpha|V|}$, where $|V|$ is the size of dictionary
5. Interpolation: $\hat{P}(w_i|w_{i-1}) = \lambda P_{MLE}(w_i|w_{i-1}) + (1-\lambda)P_{MLE}(w_i)$

n-gram models for text generation

Given w_i:

1. choose the next most probable w_{i+i}
2. randomly select sample from this probability distribution of next words





`https://twitter.com/alg_testament`

n-gram models for IR

Given documents D and query q, estimate the probability of generating the query text from a document language model:

- Rank documents by the probability that the query could be generated by the document model;
- Calculate $P(d|q)$ to rank the documents: $P(d|q) \propto P(q|d)P(d)$
- Assuming prior is uniform, unigram model: $P(q|d) = \prod_i P(q_i|d)$
- MLE: $P(q_i|d) = \frac{\text{count}(q_i, d)}{|d|}$

Reading

1. Stuart Russell, Peter Norvig. Artificial Intelligence: A Modern Approach, Ch. 15
2. Dan Jurafsky, James H. Martin. Speech and Language Processing, Ch. 3, Ch. 8

Sequential data

1. Time series
   - Financial data analysis: stock market, commodities, Forex
   - Healthcare: pulse rate, sugar level (from medical equipment and wearables)
2. Text and speech: speech understanding, text generation
3. Spatiotemporal data
   - Self-driving and object tracking
   - Plate tectonic activity
4. Physics: jet identification
5. etc.

Sequence modelling I

Sequence labelling

1. $\boldsymbol{x} = x_1, x_2, \dots, x_n,\ x_i \in V$, - objects
2. $\boldsymbol{y} = y_1, y_2, \dots, y_n,\ y_i \in \{1, \dots, L\}$ - labels
3. $\{(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}), (\boldsymbol{x}^{(2)}, \boldsymbol{y}^{(2)}), \dots, (\boldsymbol{x}^{(m)}, \boldsymbol{y}^{(m)})\}$ – training data
4. exponential number of possible solutions : if $\texttt{length}(x) = n$, there are L^n possible solutions

Classification problem: $\gamma : \boldsymbol{x} \rightarrow \boldsymbol{y}$

1. Speech recognition: x – spoken words, y – transcription
2. Genome annotation: x – DNA, y – genes

Sequence modelling II

Sequence classification

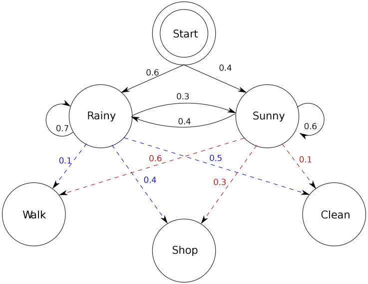
1. $\boldsymbol{x} = x_1, x_2, \dots, x_n,\ x_i \in V$, - objects
2. $y \in \{1, \dots, L\}$ - labels
3. $\{(\boldsymbol{x}^{(1)}, y_1), (\boldsymbol{x}^{(2)}, y_2), \dots, (\boldsymbol{x}^{(m)}, y_m)\}$ – training data

Classification problem: $\gamma : \boldsymbol{x} \to y$

1. Activity recognition: x – pulse rate, y – activity (walking, running, peace)
2. Opinion mining: x – sentence, y – sentiment (positive, negative)
3. Trading: x – stock market, y – action (sell, buy, do nothing)

Traditional ML approaches to sequence modelling

- Hidden Markov Models (HMM)
- Conditional Random Fields (CRF)
- Local classifier: for each x define features, based on x_{-1}, x_{+1}, etc, and perform classification n times



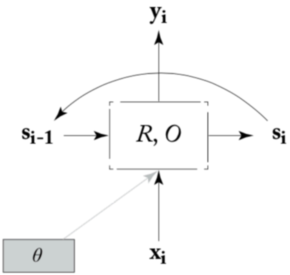
Problems:

1. Markov assumption: fixed length history
2. Computation complexity

1. Language modeling task
2. Count-based language models
3. Neural language models
   - Recurrent neural network
   - Gated architectures
   - RNN generators
4. Bonus: RecNN

Recurrent neural network

- Input: sequence of vectors
- $x_{1:n} = x_1, x_2, \ldots, x_n,\ x_i \in \mathbb{R}^{d_{in}}$
- Output: a single vector
  $y_n = RNN(x_{1:n}),\ y_n \in \mathbb{R}^{d_{out}}$
- For each prefix $x_{i:j}$ define an output vector y_i:
  $y_i = RNN(x_{1:i})$
- RNN^* is a function returning this sequence for input sequence $x_{1:n}$:
  $y_{1:n} = RNN^*(x_{1:n}),\ y_i \in \mathbb{R}^{d_{out}}$



Sequence modelling with RNN

1. Sequence labelling
   Produce an output y_i for each input RNN reads in. Put a dense layer on top of each output to predict the desired class of the input

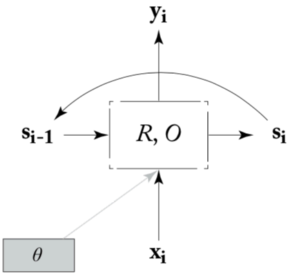
$$p(l_j|\boldsymbol{x}_j) = \texttt{softmax}(RNN(\boldsymbol{x}_{1:j}) \times W + b)_{[j]}$$

2. Sequence classification
   Put a dense layer on top of RNN to predict the desired class of the sequence after the whole sequence is processed

$$p(l_j|\boldsymbol{x}_{1:n}) = \texttt{softmax}(RNN(\boldsymbol{x}_{1:n}) \times W + b)_{[j]}$$

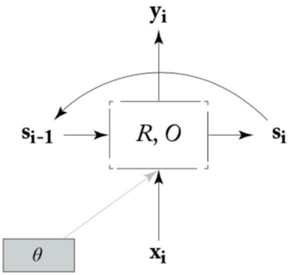
More details on RNN

- $RNN^*(x_{1:n}, s_0) = y_{1:n}$
- $y_i = O(s_i)$ – simple activation function
- $s_i = R(s_{i-1,x_i})$, where R is a recursive function, s_i is a state vector
- s_0 is initialized randomly or is a zero vector
- $x_i \in \mathbb{R}^{d_{in}}$, $y_i \in \mathbb{R}^{d_{out}}$, $s_i \in \mathbb{R}^{f(d_{out})}$
- θ – shared weights

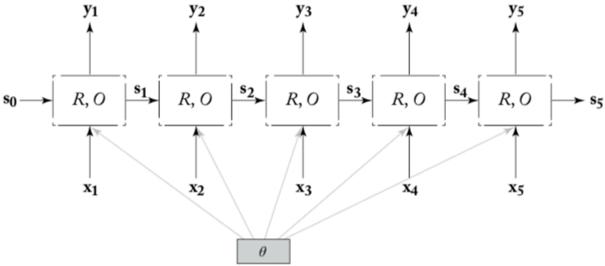


More details on RNN

- $s_i = R(x_i, s_{i-1}) = g(s_{i-1}W^s + x_iW^x + b)$
- $y_i = O(s_i) = s_i$
- $y_i, s_i, b \in \mathbb{R}^{d_{out}}$, $x_i \in \mathbb{R}^{d_{in}}$
- $W^x \in \mathbb{R}^{d_{in} \times d_{out}}$, $W^s \in \mathbb{R}^{d_{out} \times d_{out}}$



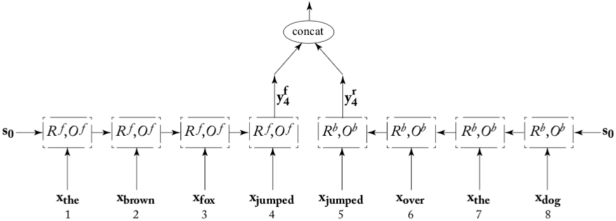
RNN unrolled



$$s_4 = R(s_3, x_4) = R(R(s_2, x_3), x_4) = R(R(R(s_1, x_2), x_3), x_4) =$$
$$= R(R(R(R(s_0, x1), x_2), x_3), x_4)$$

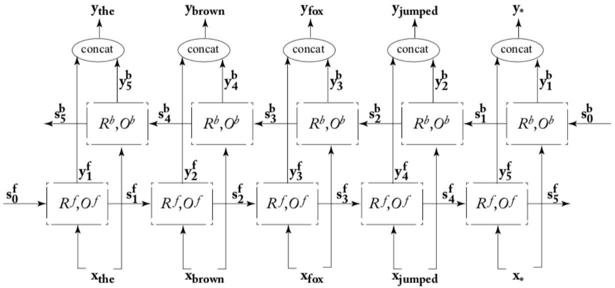
Bidirectional RNN (Bi-RNN)

The input sequence can be read from left to right and from right to left. Which direction is better?



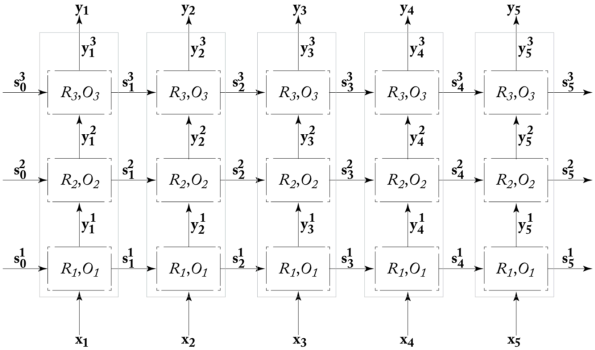
$$biRNN(x_{1:n}, i) = y_i = [RNN^f(x_{1:i}); RNN^r(x_{n:i})]$$

Bi-RNN



$$biRNN^*(x_{1:n}, i) = y_{1:n} = biRNN(x_{1:n}, 1) \ldots biRNN(x_{1:n}, n)$$

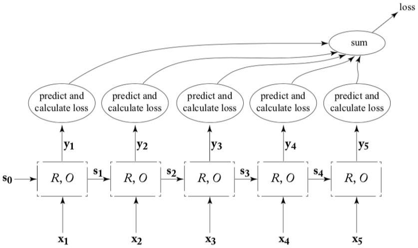
Multilayer RNN



Connections between different layers are possible too: $y_1^2 = \texttt{concat}(x_1, y_1^1)$

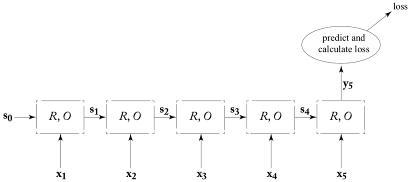
Sequence labelling

- Output $\hat{t}_i$ for each input $x_{1,i}$
- Local loss: $L_{local}(\hat{t}_i, t_i)$
- Global loss: $L(\hat{t}_n, t_n) = \sum_i L_{local}(\hat{t}_i, t_i)$
- L can take any form: cross entropy, hinge, margin, etc.

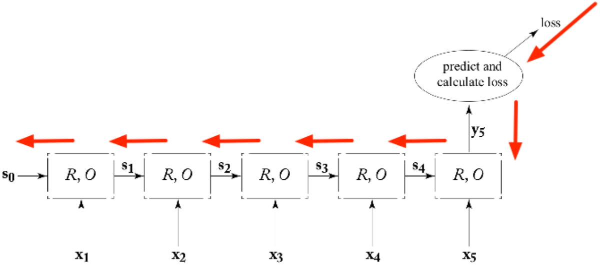


Sequence classification

- $\hat{y}_n = O(s_n)$
- $\texttt{prediction} = MLP(\hat{y}_n)$
- Loss: $L(\hat{y}_n, y_n)$
- L can take any form: cross entropy, hinge, margin, etc.



Backpropogation through time



$s_i = R(x_i, s_{i-1}) = g(s_{i-1}W^s + x_i W^x + b)$

Chain rule: $\frac{\partial L}{\partial w} = \frac{\partial L}{\partial p(\hat{y_5})} \frac{\partial p(\hat{y_5})}{\partial s_4} \left(\frac{\partial s_4}{\partial w} + \frac{\partial s_4}{\partial s_3} \frac{\partial s_3}{\partial w} + \frac{\partial s_4}{\partial s_3} \frac{\partial s_3}{\partial s_2} \frac{\partial s_2}{\partial s_w} + \ldots \right)$

Vanishing gradient problem

Chain rule: $\frac{\partial L}{\partial w} = \frac{\partial L}{\partial p(\hat{y_5})} \frac{\partial p(\hat{y_5})}{\partial s_4} (\frac{\partial s_4}{\partial w} + \frac{\partial s_4}{\partial s_3} \frac{\partial s_3}{\partial w} + \frac{\partial s_4}{\partial s_3} \frac{\partial s_3}{\partial s_2} \frac{\partial s_2}{\partial s_w} + \dots)$

g – sigmoid

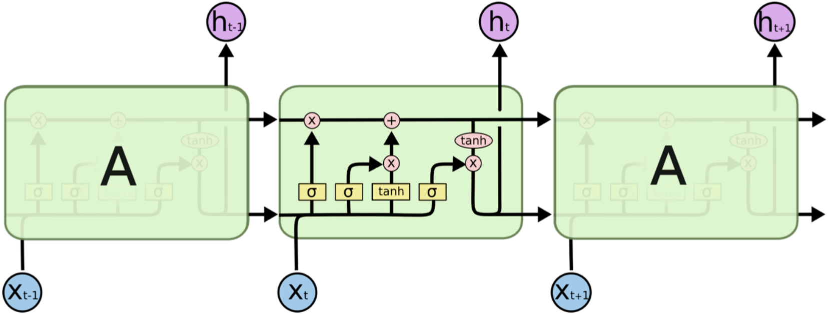
1. Many sigmoids near 0 and 1
   - Gradients $\rightarrow 0$
   - Not training for long term dependencies
2. Many sigmoids > 1
   - Gradients $\rightarrow + \inf$
   - Not training again

Solution: gated architectures (LSTM and GRU)

Controlled memory access

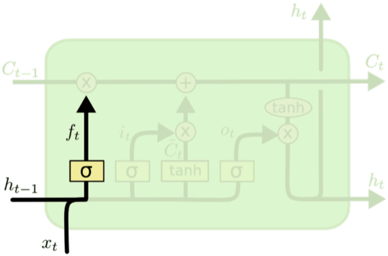
- Entire memory vector is changed: $s_{i+1} = R(x_i, s_i)$
- Controlled memory access: $s_{i+1} = g \odot R(x_i, s_i) + (1 - g)s_i$
  $g \in [0, 1]^d, s, x \in \mathbb{R}^d$
- Differential gates: $\sigma(g), g' \in \mathbb{R}^d$
- This controllable gating mechanism is the basis of the LSTM and the GRU architectures

Long short term memory



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

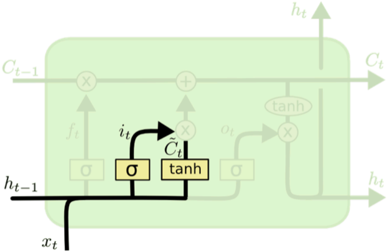
Long short term memory



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

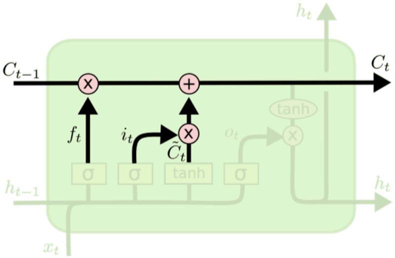
Long short term memory



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

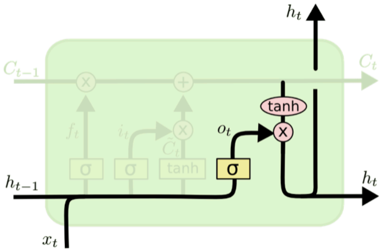
Long short term memory



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

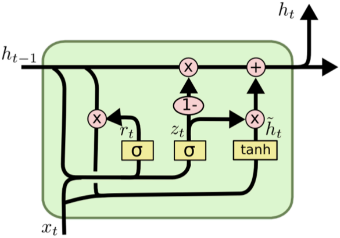
Long short term memory



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

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Gated recurrent unit

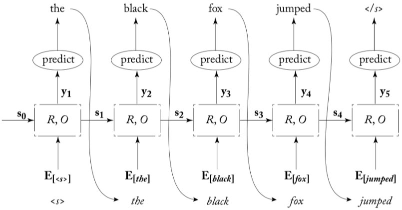


$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

http://colah.github.io/posts/2015-08-Understanding-LSTMs/

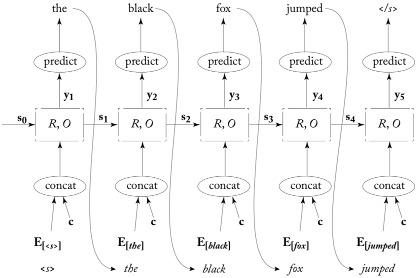
Sequence generation

- Teacher forcing:
  $x :=< s > x, y := x < /s >$
  $x :< s > x_1 x_2 \dots x_n$
  $y : x_1 x_2 \dots x_n < /s >$
- $\hat{t}_{j+1} \sim p(t_{j+1} = k | t_{1:j})$
- $p(t_{j+1} = k | t_{1:j}) = f(RNN(\hat{t}_{1:j}))$
  $f(x) = \texttt{softmax}(MLP(x))$



Conditioned sequence generation

- $\hat{t}_{j+1} \sim p(t_{j+1} = k | t_{1:j}, c)$
- $p(t_{j+1} = k | t_{1:j}) = f(RNN(\hat{v}_{1:j}))$
  $v_i = [\hat{t}_i, c]$



Sequence generation

- Examples of generated texts:
  http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- Examples of generated MIDI music:
  https://towardsdatascience.com/
  how-to-generate-music-using-a-lstm-neural-network-in-keras

Conclusion

Topics covered:

1. RNN is a powerful tool for sequence modeling
2. RNN usage scenarios: sequence labelling, sequence classification, sequence generation
3. RNN layers can be reversed $\rightarrow$ bidirectional RNN
4. RNN layers can be stacked $\rightarrow$ deep RNN
5. RNN suffers from gradient vanishing problem $\rightarrow$ LSTM, GRU

Topics not covered:

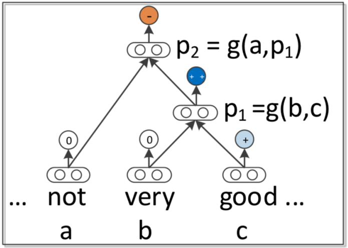
1. seq2seq models
2. Attention mechanism in RNN
3. Recursive neural networks

Modeling trees with Recursive NN

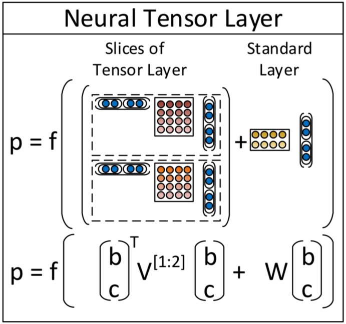
- Input: $x_1, x_2, \ldots, x_n$
- A binary tree T can be represented as a unique set of triplets (i, k, j), s.t. $i < k < j$, $x_{i:j}$ is parent of $x_{i:k}, i_{k+1,j}$
- RecNN takes as an input a binary tree and returns as output a corresponding set of inside state vectors $\boldsymbol{s^A_{i:j}} \in \mathbb{R}^d$
- Each state vector $\boldsymbol{s^A_{i:j}}$ represents the corresponding tree node $q^A_{i:j}$ and encodes the entire structure rooted at that node

RecNN

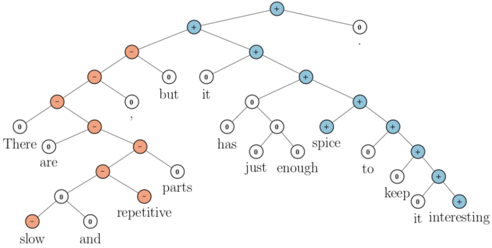
- Input: $x_1, x_2, \ldots, x_n$ and a binary tree T
- $RecNN(x_1, x_2, \ldots, x_n, T) =$
  $= \{\mathbf{s^A_{i:j}} \in \mathbb{R}^d | q^A_{i:j} \in T\}$
- $\mathbf{s^A_{i:i}} = v(x_i)$
- $\mathbf{s^A_{i:j}} = R(A, B, C, \mathbf{s^B_{i:k}}, \mathbf{s^C_{k+1:j}})$,
  $q^B_{i:k} \in T, q^C_{k+1:j} \in T$
- $R(A, B, C, \mathbf{s^B_{i:k}}, \mathbf{s^C_{k+1:j}}) =$
  $g([\mathbf{s^B_{i:k}}, \mathbf{s^C_{k+1:j}}]W)$



RecNN [SPW+13]



Neural Tensor Layer
Slices of Tensor Layer
Standard Layer
p = f
+
p = f
b c T V[1:2] b c + W b c



There
are
slow
and
repetitive
parts
,
but
it
has
just
enough
spice
to
keep
it
interesting
.

Reading

1. Yoav Goldberg. Neural Network Methods for Natural Language Processing, Ch. 14-19
2. Ian Goodfellowm Yoshua Bengio, Aaron Courville. Deel learning, Ch. 10

References I

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts, *Recursive deep models for semantic compositionality over a sentiment treebank*, Proceedings of the 2013 conference on empirical methods in natural language processing, 2013, pp. 1631–1642.