Machine Learning: Yoshinari Fujinuma

University of Colorado Boulder

LECTURE 17

Slides adapted from Chenhao Tan, Jordan Boyd-Graber

Logistics

- Project proposal guideline will be released today

Learning Objectives

- Recurrent Neural Network
- Convolutional Neural Network

Outline

Recurrent Neural Networks (RNNs)

Convolutional Neural Network

Motivation

- We used bag-of-words features for spam classification
- ...but what should we do when we want to distinguish “cats like dogs” vs. “dogs like cats”?

Structured data

Sequential information
“My words fly up, my thoughts remain below: Words without thoughts never to heaven go.”

—Hamlet

Structured data

Sequential information
“My words fly up, my thoughts remain below: Words without thoughts never to heaven go.”

—Hamlet

- language
- activity history

Structured data

Sequential information
"My words fly up, my thoughts remain below: Words without thoughts never to heaven go."

—Hamlet

- language
- activity history

Above sentence can be regarded as a sequence $\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$, starting from left ($x_0 =$ "My") to right ($x_T =$ "go").
e.g.,

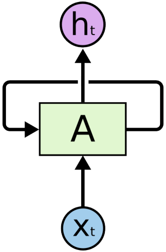
$$\boldsymbol{x}_{\text{“My words fly up, my thoughts remain below”}} = (\boldsymbol{x}_{\text{My}}, \ldots, \boldsymbol{x}_{\text{below}})$$

Recurrent Neural Networks

Sharing parameters along a sequence
At time t, the hidden representation h_t is given as

$$h_t = f(x_t, h_{t-1})$$

i.e., h_t is dependent on the hidden representation at the previous step t.



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

Recurrent Neural Networks

In (vanilla) RNN, there are following training parameters

- parameter matrix on recurrent input W_{rec}
- parameter matrix on input W_{in}
- bias vector b_{rec} and b_{in}

$$h_t = f(x_t, h_{t-1})$$

where

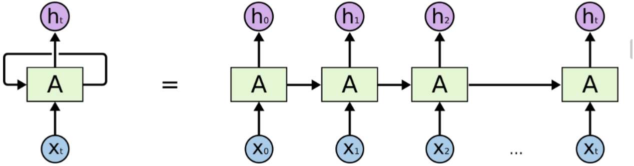
$$f(x_t, h_{t-1}) = g(W_{\text{in}} x_t + b_{\text{in}} + W_{\text{rec}} h_{t-1} + b_{\text{rec}})$$

and g is a non-linear activation function

Recurrent Neural Networks

RNNs can be unrolled over time, good at capturing long-term dependencies.

E.g., open and closed brackets in C++ programming “if (condition) {...}”.



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

How to train this?

“Back propagation over time” and helps capture long-term dependencies

Long short-term memory (LSTM)

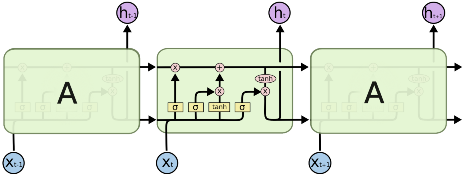
Problem with (vanilla) RNN:
Vanishing gradient [Pascanu et al., 2013], hard to capture extremely long-term dependencies

Long short-term memory (LSTM)

Problem with (vanilla) RNN:
Vanishing gradient [Pascanu et al., 2013], hard to capture extremely long-term dependencies

One hack: have “gates” to help remember/forget parameters

- $i_t \in \mathbb{R}^n$: input gate at time t
- $f_t \in \mathbb{R}^n$: forget gate at time t
- $o_t \in \mathbb{R}^n$: output gate at time t

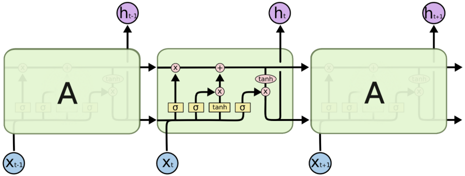


Long short-term memory (LSTM)

Problem with (vanilla) RNN:
Vanishing gradient [Pascanu et al., 2013], hard to capture extremely long-term dependencies

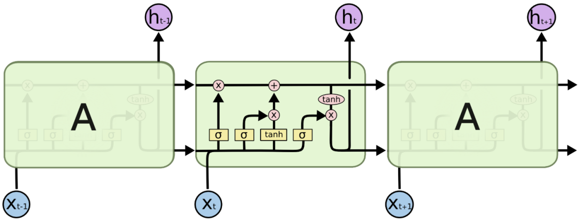
One hack: have "gates" to help remember/forget parameters

- $i_t \in \mathbb{R}^n$: input gate at time t
- $f_t \in \mathbb{R}^n$: forget gate at time t
- $o_t \in \mathbb{R}^n$: output gate at time t



Long short-term memory (LSTM)

Intuitively, “gates” are vectors that controls the amount of inputs that can go through e.g., if we have a forget gate f at time t as $f_t = (0, 0, 0, 0.1, 0)$, then $f_t \odot (1, 2, 3, 4, 5) = (0, 0, 0, 0.4, 0)$



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$
$$h_t = o_t \odot tanh(C_t)$$

Pros and Cons of RNNs

- Pros
  - Can handle inputs with arbitrary length
- Cons
  - Slow due to going through one input after another
  - (and this is why Transformers [Vaswani et al., 2017] are popular these days)

Outline

Recurrent Neural Networks (RNNs)

Convolutional Neural Network

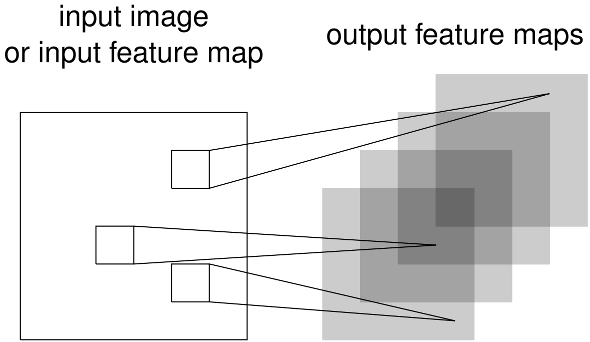
Structured data

Spatial information

https://www.reddit.com/r/aww/comments/6ip2la/before_and_after_she_was_told_she_was_a_good_girl/

Convolutional Layers

Sharing parameters across patches



$$a_{i'j'} = \sum_{i=1}^{k} \sum_{j=1}^{k} w_{ij} x_{ij}$$

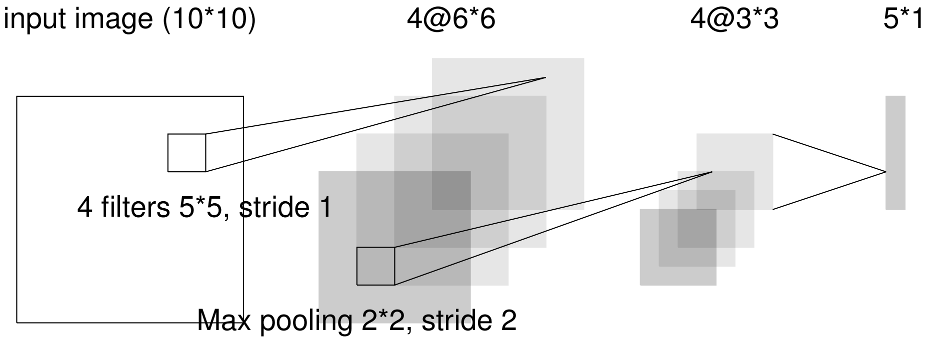
- Number of filters
- Filter shape
- Stride size

https://github.com/davidstutz/latex-resources/blob/master/tikz-convolutional-layer/convolutional-layer.tex

CNN

- Convolutional layer: extracting features, trainable parameters
- Pooling layer: downsampling, no parameters involved
- Full-connected layer: for optimizing on the objective function

E.g., convolutional layer with 4 filters, max pooling, and a fully-connected layer)



Wrap up

Neural Network architecture

- Recurrent Neural Networks
- Convolutional Neural Networks

References

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1310–1318, 2013. URL http://proceedings.mlr.press/v28/pascanu13.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.