Machine Learning: Chenhao Tan

University of Colorado Boulder

LECTURE 15

Slides adapted from Jordan Boyd-Graber

Logistics

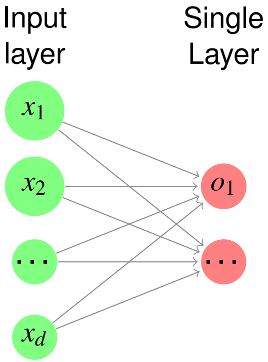
- HW3 available on Github, due on October 18
- Final project team formation due by October 9

Outline

Recap

Layers for Structured Data

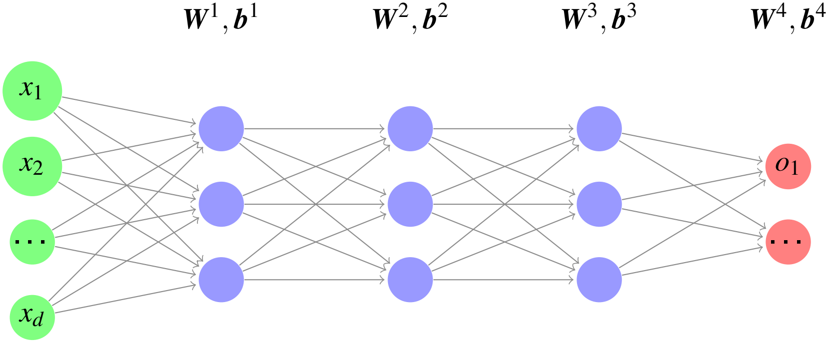
Logistic Regression as a Single-layer Neural Network



- What is the activation used in logistic regression?
- What is the objective function used in logistic regression?

Forward propagation algorithm

How do we make predictions based on a multi-layer neural network?
Store the biases for layer l in $\boldsymbol{b}^l$, weight matrix in $\boldsymbol{W}^l$



Forward propagation algorithm

Suppose your network has L layers

Make prediction for an instance $\boldsymbol{x}$

1: Initialize $\boldsymbol{a}^0 = \boldsymbol{x}$

2: **for** $l = 1$ to L **do**

3: $\quad \boldsymbol{z}^l = \boldsymbol{W}^l \boldsymbol{a}^{l-1} + \boldsymbol{b}^l$

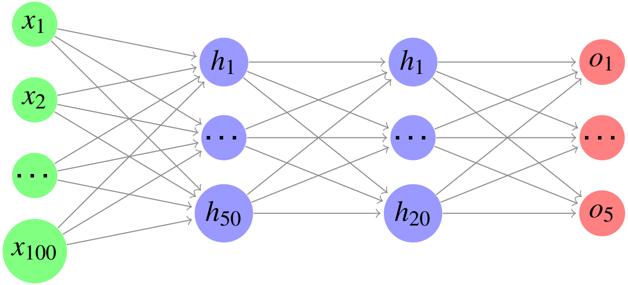
4: $\quad \boldsymbol{a}^l = g^l(\boldsymbol{z}^l)$ // g^l represents a nonlinear activation

5: **end for**

6: The prediction $\hat{y}$ is simply $\boldsymbol{a}^L$

Quiz 1

How many parameters are there in the following feed-forward neural networks (assuming no biases)?



A. 100 * 50 + 50 * 20 + 20 * 5

B. 100 * 50 + 50 + 50 * 20 + 20 + 20 * 5 + 5

Neural networks in a nutshell

- Training data $S_{\text{train}} = \{(\boldsymbol{x}, y)\}$
- Network architecture (model)

$$\hat{y} = f_w(\boldsymbol{x})$$

- Loss function (objective function)

$$\mathcal{L}(y, \hat{y})$$

- Learning (next lecture)

Quiz 2

Which of the following statements is true? (Suppose that training data is large.)

A. In training, K-nearest neighbors takes shorter time than neural networks.

B. In training, K-nearest neighbors takes longer time than neural networks.

C. In testing, K-nearest neighbors takes shorter time than neural networks.

D. In testing, K-nearest neighbors takes longer time than neural networks.

Outline

Recap

Layers for Structured Data

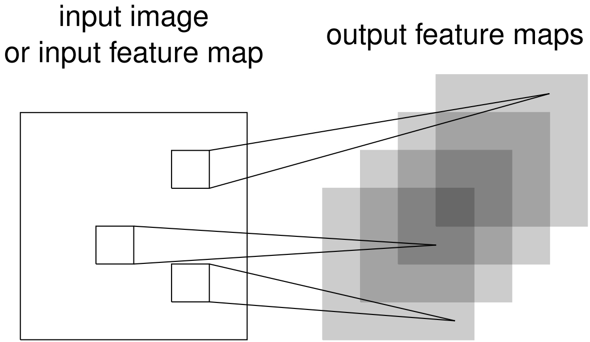
Structured data

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



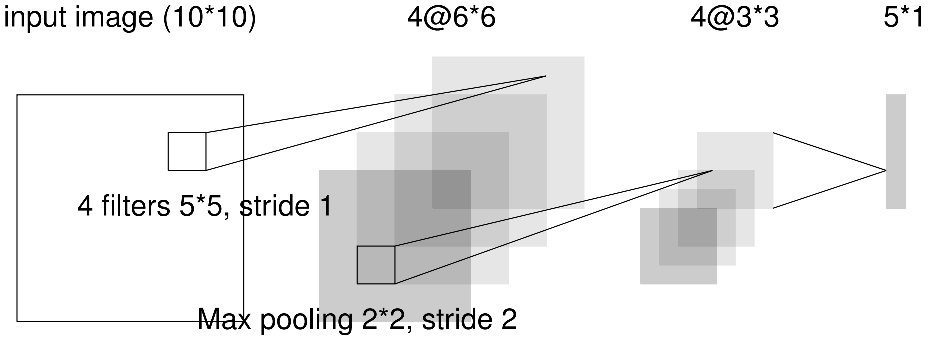
$$a_{i'j'} = \sum_{i=1}^{k} \sum_{j=1}^{k} w_{ij} x_{ij}$$

- Number of filters
- Filter shape
- Stride size

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

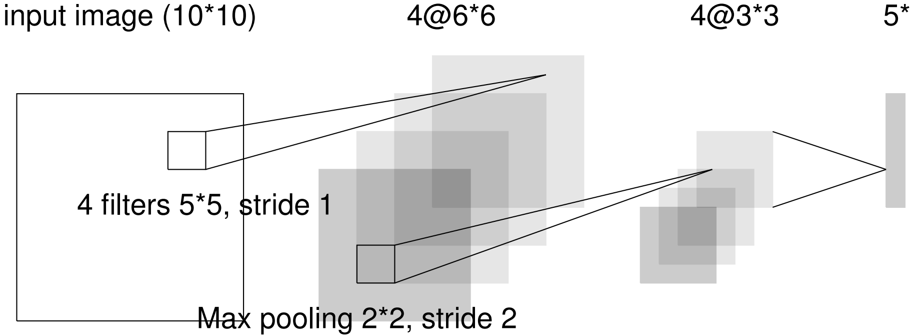
Quiz 3

A concrete example (assuming no biases, convolution with 4 filters, ReLU, max pooling, and finally a fully-connected softmax layer)



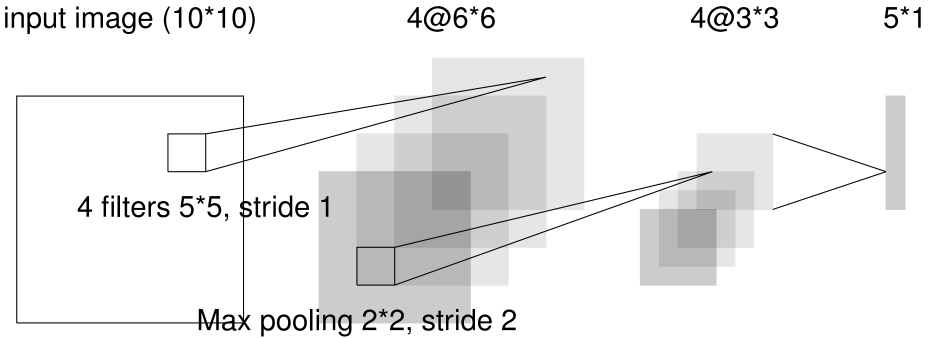
Quiz 3

How many parameters are there in the following convolutional neural networks? (assuming no biases, convolution with 4 filters, ReLU, max pooling, and finally a fully-connected softmax layer)



Quiz 3

How many ReLU operations are performed on the forward pass? (assuming no biases, convolution with 4 filters, ReLU, max pooling, and finally a fully-connected softmax layer)



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

$$\boldsymbol{x} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$$

Recurrent Layers

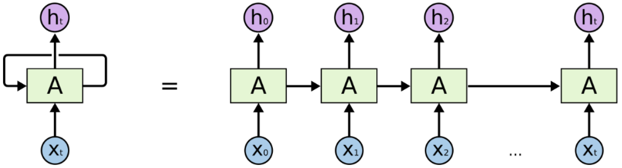
Sharing parameters along a sequence

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

Recurrent Layers

Sharing parameters along a sequence

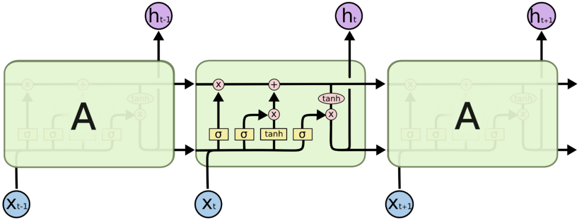
$$h_t = f(x_t, h_{t-1})$$



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

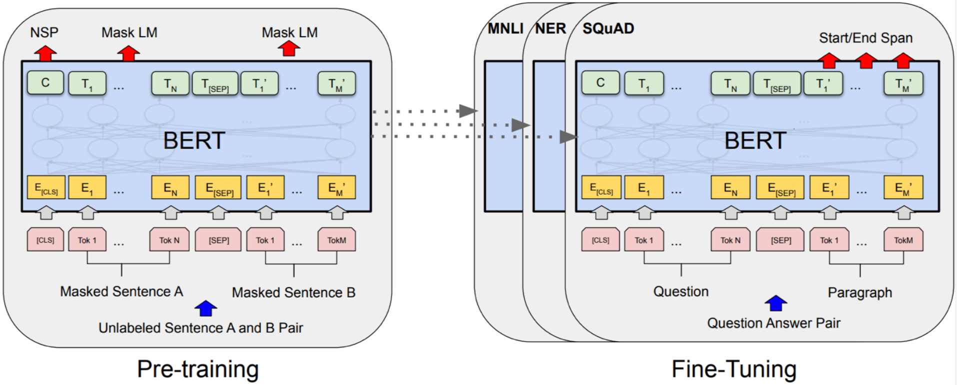
Long short-term memory

A commonly used recurrent neural network in natural language processing

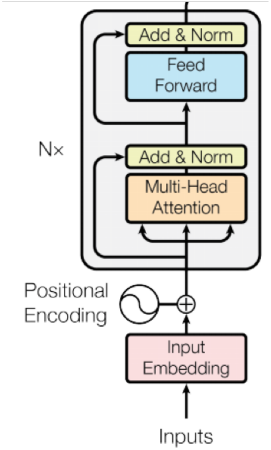


$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$h_t = o_t * tanh(C_t)$$

Transformer



Transformer



Wrap up

Neural networks

- Network architecture (a lot more then fully connected layers)
  - Convulutional layer
  - Recurrent layer
- Loss function

What is missing?

- How to find good weights?
- How to make the model work (regularization, even more architecture, etc)?