Convolutional- & Recurrent Neural Networks

Notes for Topics in Social Data Science at UCPH

March 6, 2019

Thor Donsby Noe

1 Convolutional Nets

Issues with vanilla feed-forward neural network

- Data squeezed into a 1xN vector.
- Single operation on whole input.
- No attention to **spatial adjacency**.

Solution: 'Slide' weight matrix over a part of the input. E.g.

- Input image: 32x32x3 (i.e. RGB-colors)
- Filter: 5x5x3 $\rightarrow$ dot product $\rightarrow$ 1 number
- Activation map: 28x28x1 (convolved over all spatial locations)
  - One activation map for each filter. These are stacked.
  - E.g. 28x28x6 for 6 5x5x3 filters.
    * An additional activation map can be added by using a filter on the activation map of the former layer.
    * E.g. a new activation map layer: 24x24x10 by using 10 5x5x3 filters.

Stride

- How many pixels are moved each time, e.g. 1.

Padding

- The no. zeroes added at the borders of the input image before applying the filter.
- Full padding: If you don't want the image to shrink.
- Half padding: The image shrinks less, e.g. padding = 1.

Width of the activation map will be

$$W = \frac{N + 2P - F}{S} + 1$$

E.g. for the example above with padding=2

$$W = \frac{32 + 2 \cdot 2 - 5}{1} + 1 = 32$$

Pooling: Condense the information by downsampling.

- Speeds up the learning process.
- 'max pool' where only largest value is saved for each quadrant as they are expected to contain the more important signal.
- 'average pooling'
- E.g. 16x16 $\rightarrow$ 4x4
- Doesn't shrink it in the depth direction.

2 Recurrent Nets

Optimal for **sequential data** but useful for non-sequential data as well.

Issue for **all** feed forward neural networks

- Input and output must be of hard-assigned dimensions.

Backpropagation: Compute the sum of losses.

- Can get complicated as the number becomes very high up with long chains.
- Can break it into parts.

Elements

- f: Forget vector (0 = forget, 1 = remember) → what to forget from c-vector
  - c: New hidden vector, i.e. long term memory
  - c is not converted, → not multiplied by weight → gradient doesn't explode → much quicker
- i, g: adds to memory vector, c
- o: output vector