Some notes on CNN

November 2019

1 Introduction

A collection of important things about convolutional neural networks (CNN). As was the case with the 'lectures' I gave, these will probably be extremely confusing. So remember: Google is your friend.

2 CNN

- For the classification of images, CNN in general work better than DNN. The reason for this is that, if we want to feed an image of 64x64 pixels with 3 color channels to a DNN, we would have to transform the 64x64x3 tensor to a 12.228x1 Numpy array. Not that this wouldn't work, but it's better to keep the spatial structure of the image intact. The more detailed the data, the better the network can perform. Another advantage of CNN over DNN for image classification is that CNN are invariant under translations. This means that, if the CNN is successful in classifying cats, it doesn't matter where in the image the cat is.
- Probably the most important object of a CNN is the convolutional filter. A convolutional filter is just a matrix of which the entries constitute the weights of the network. This filter is then convoluted with the image. This is a 4-step process:

  a) Put the filter on the top left corner of the image

  b) Calculate the element-wise product between the elements of the filter and the corresponding elements of the image

  c) Sum the obtained products. This sum represents the output for the particular pixel we were considering

  d) Move the filter with some particular step-size (= stride) and repeat.

  The result of this is a new matrix of different dimensionality. For example, imagine that the filter is

$$\begin{pmatrix} 1 & 2 \\ -1 & 0 \end{pmatrix} \tag{1}$$

and the image was represented by (the numbers represent the intensity of the color channel, eg for grayscale lower values could mean more black while larger values could mean more gray)

$$\begin{pmatrix} 5 & 2 & 2 & 3 \\ 3 & 1 & 2 & 1 \\ 9 & 0 & 4 & 5 \end{pmatrix}. \tag{2}$$

Putting the filter in the top-left corner then yields as output 6. Moving the filter one place to the left gives as output 5 etc. It is possible that not all pixels are accessible to the filter near the edge. To solve this, we use padding; this is just adding extra columns to the image. Often one uses zero-padding (just adding zero-columns) or same-padding (adding the last column of the image).

- In a CNN, you of course don't just have one filter: you have a stack of them. Each filter then encodes a specific aspect of the input data. For example, if you want to classify cats, 1 filter could encode the presence of a tail; another the presence of paws etc.

- Knowing this, a convolutional layer is then just a stack of filters that turn the input image into a different output image. The number of weights in such a layer is the size of the filter times the number of filters. So, a layer using 4 filters of the same form as the one in (1) would have 16 weights. In practice, often 3x3, 5x5 and 7x7 filters are used.

- Neighboring pixels in an image usually contain similar information, which in turn means similar outputs after convolution. So, it will be redundant to keep all this information. The way to avoid this is called pooling. Two popular methods are max-pooling (if we have a AxB filter, we go over the image and replace each AxB patch by the highest value in that patch) and average-pooling (replace each patch by its average).