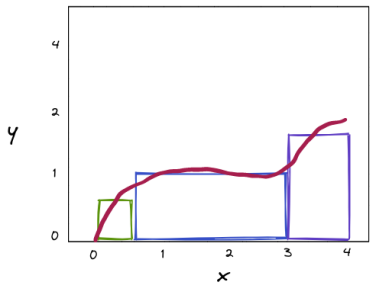
Introduction to Machine Learning
Problem Set: Deep and Convolutional Neural Networks

Summer 2021

1. **Neural network as a universal approximator.** A fully connected neural network with one hidden layer can approximate a continuous function to a desired level of accuracy, by using the hidden units to create "bumps" at the desired heights and positions in the feature space.

   Can you use this approach to approximate the function drawn in red here, in the range $0 \leq x \leq 4$ as shown, using three "bumps" as in the illustration?

   Figure 1: Approximating a continuous function with three "bumps".

   Draw a network that is capable of this approximation. Your network should have an input unit, a hidden layer with a sigmoid activation function at each hidden unit, and an output unit (with a linear activation). Indicate *actual, numeric values* of the weights on each edge and the bias input to each node.

   Then, write an expression for $\hat{y}$, the output of the network, as a function of the input. The only variable in this expression should be x - the weights and biases should be specified as numeric values, not variables.

   Hint: if you are not sure what to do, try working through this interactive tutorial which will show you exactly how to make "bumps" using sigmoid units.

Solution:

The output of the network should be:

$$\begin{aligned}\hat{y} =& 0.5\sigma(k(x)) - 0.5\sigma(k(x-0.5)) + \\ & 1\sigma(k(x-0.5)) - 1\sigma(k(x-3)) + \\ & 1.5\sigma(k(x-3)) - 1.5\sigma(k(x-4))\end{aligned}$$

where I use $k = 100$, but any large number will work. (The bigger k is, the steeper the sigmoid slope.) Here's a Colab notebook where you can play around with this.

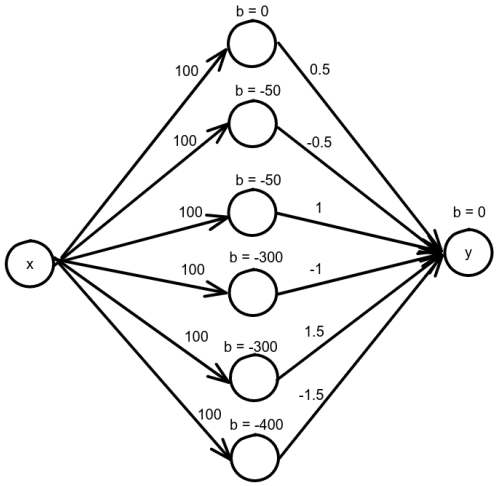


Figure 2: Neural network with $k = 100$, but any other large k will work.

2. This animation shows how the output volume of a convolutional layer of a neural network is computed for a specific input. Every time you refresh the page, it will show you a new, randomly generated example.

   Compute the output of the center pixel at each depth in the output volume, i.e. the missing squares in the image below, for the following example:

Input Volume (+pad 1) (7x7x3)

`x[:,:,0]`

0	0	0	0	0	0	0
0	1	2	0	2	1	0
0	2	2	1	2	1	0
0	1	1	2	1	1	0
0	0	1	1	1	1	0
0	1	2	2	2	1	0
0	0	0	0	0	0	0

`x[:,:,1]`

0	0	0	0	0	0	0
0	0	1	2	1	0	0
0	0	0	2	1	2	0
0	1	2	1	2	2	0
0	0	2	1	1	1	0
0	1	0	2	2	2	0
0	0	0	0	0	0	0

`x[:,:,2]`

0	0	0	0	0	0	0
0	2	1	0	1	1	0
0	2	2	1	1	1	0
0	2	0	0	0	2	0
0	0	0	2	0	1	0
0	0	2	0	2	0	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

`w0[:,:,0]`

1	0	1
-1	0	1
-1	1	-1

`w0[:,:,1]`

1	0	0
0	0	1
1	-1	1

`w0[:,:,2]`

-1	0	1
1	1	-1
-1	-1	0

Filter W1 (3x3x3)

`w1[:,:,0]`

-1	0	-1
-1	-1	1
1	0	0

`w1[:,:,1]`

-1	1	1
-1	-1	0
-1	1	0

`w1[:,:,2]`

0	1	0
1	-1	-1
-1	-1	-1

Output Volume (3x3x2)

`o[:,:,0]`

3	-5	-3
11		3
2	7	3

`o[:,:,1]`

-6	-3	-3
-3		-8
-1	-4	-5

Bias b0 (1x1x1)

`b0[:,:,0]`

1

Bias b1 (1x1x1)

`b1[:,:,0]`

0

Solution:

Input Volume (+pad 1) (7x7x3)

`x[:,:,0]`

0	0	0	0	0	0	0
0	1	2	0	2	1	0
0	2	2	1	2	1	0
0	1	1	2	1	1	0
0	0	1	1	1	1	0
0	1	2	2	2	1	0
0	0	0	0	0	0	0

`x[:,:,1]`

0	0	0	0	0	0	0
0	0	1	2	1	0	0
0	0	0	2	1	2	0
0	1	2	1	2	2	0
0	0	2	1	1	1	0
0	1	0	2	2	2	0
0	0	0	0	0	0	0

`x[:,:,2]`

0	0	0	0	0	0	0
0	2	1	0	1	1	0
0	2	2	1	1	1	0
0	2	0	0	0	2	0
0	0	0	2	0	1	0
0	0	2	0	2	0	0
0	0	0	0	0	0	0

Filter W0 (3x3x3)

`w0[:,:,0]`

1	0	1
-1	0	1
-1	1	-1

`w0[:,:,1]`

1	0	0
0	0	1
1	-1	1

`w0[:,:,2]`

-1	0	1
1	1	-1
-1	-1	0

Filter W1 (3x3x3)

`w1[:,:,0]`

-1	0	-1
-1	-1	1
1	0	0

`w1[:,:,1]`

-1	1	1
-1	-1	0
-1	1	0

`w1[:,:,2]`

0	1	0
1	-1	-1
-1	-1	-1

Output Volume (3x3x2)

`o[:,:,0]`

3	-5	-3
11	5	3
2	7	3

`o[:,:,1]`

-6	-3	-3
-3	-7	-8
-1	-4	-5

Bias b0 (1x1x1)

`b0[:,:,0]`

1

Bias b1 (1x1x1)

`b1[:,:,0]`

0

Figure 3: Output of a convolutional layer with stride 2.

3. Suppose the input to a 2D max pooling layer with 2x2 filters is as follows:

$$X = \begin{bmatrix} 4 & 3 & 2 & 1 \\ 3 & 3 & 2 & 1 \\ 3 & 2 & 1 & 0 \\ 2 & 1 & 0 & 0 \end{bmatrix}$$

   (a) Show the output of the pooling layer with stride 1.
   Solution:

$$\begin{bmatrix} 4 & 3 & 2 \\ 3 & 3 & 2 \\ 3 & 2 & 1 \end{bmatrix}$$

   (b) Show the output of the pooling layer with stride 2.
   Solution:

$$\begin{bmatrix} 4 & 2 \\ 3 & 1 \end{bmatrix}$$

4. Transfer learning notebook

   Please refer to the Colab notebook on the course site.