

# COMP9444 Neural Networks and Deep Learning

## Term 2, 2020

### Solutions to Exercise 5: Hidden Units and Convolution

This page was last updated: 06/22/2020 15:47:48

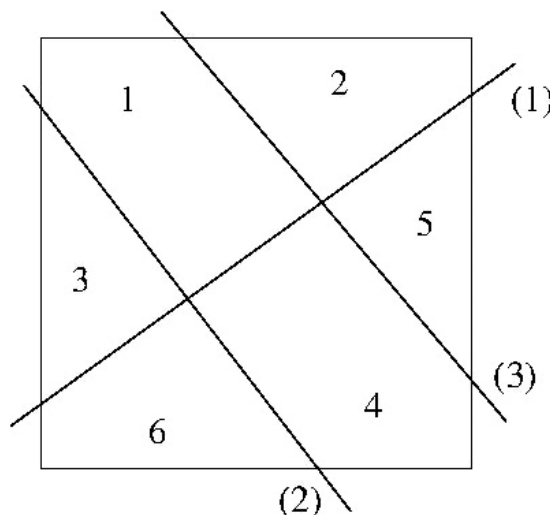
#### 1. Hidden Unit Geometry

Consider a fully connected feedforward neural network with 6 inputs, 2 hidden units and 3 outputs, using tanh activation at the hidden units and sigmoid at the outputs. Suppose this network is trained on the following data, and that the training is successful.

Item	Inputs	Outputs
----	-----	-----
	123456	123
----	-----	-----
1.	100000	000
2.	010000	001
3.	001000	010
4.	000100	100
5.	000010	101
6.	000001	110

Draw a diagram showing:

- for each input, a point in hidden unit space corresponding to that input, and
- for each output, a line dividing the hidden unit space into regions for which the value of that output is greater/less than one half.



#### 2. Softmax

Recall that the formula for Softmax is

$$\text{Prob}(i) = \exp(z_i) / \sum_j \exp(z_j)$$

Consider a classification task with three classes 1, 2, 3. Suppose a particular input is presented,

producing outputs

$$z_1=1.0, z_2=2.0, z_3=3.0$$

and that the correct class for this input is Class 2. Compute the following, to two decimal places:

a.  $\text{Prob}(i)$ , for  $i = 1, 2, 3$

$$\text{Prob}(1) = e^1 / (e^1 + e^2 + e^3) = 2.718/30.193 = 0.09$$

$$\text{Prob}(2) = e^2 / (e^1 + e^2 + e^3) = 7.389/30.193 = 0.24$$

$$\text{Prob}(3) = e^3 / (e^1 + e^2 + e^3) = 20.086/30.193 = 0.67$$

b.  $d(\log \text{Prob}(2))/dz_j$ , for  $j = 1, 2, 3$

$$d(\log \text{Prob}(2))/dz_1 = d(z_2 - \log \sum_j \exp(z_j))/dz_1 = -\exp(z_1)/\sum_j \exp(z_j) = -0.09$$

$$d(\log \text{Prob}(2))/dz_2 = d(z_2 - \log \sum_j \exp(z_j))/dz_2 = 1 - \exp(z_2)/\sum_j \exp(z_j) = 1 - 0.24 = 0.76$$

$$d(\log \text{Prob}(2))/dz_3 = d(z_2 - \log \sum_j \exp(z_j))/dz_3 = -\exp(z_3)/\sum_j \exp(z_j) = -0.67$$

Note how the correct class (2) is pushed up, while the incorrect class with the highest activation (3) is pushed down the most.

### 3. Convolutional Network Architecture

One of the early papers on Deep Q-Learning for Atari games (Mnih et al, 2013) contains this description of its Convolutional Neural Network:

"The input to the neural network consists of an  $84 \times 84 \times 4$  image. The first hidden layer convolves 16  $8 \times 8$  filters with stride 4 with the input image and applies a rectifier nonlinearity. The second hidden layer convolves 32  $4 \times 4$  filters with stride 2, again followed by a rectifier nonlinearity. The final hidden layer is fully-connected and consists of 256 rectifier units. The output layer is a fully-connected linear layer with a single output for each valid action. The number of valid actions varied between 4 and 18 on the games we considered."

For each layer in this network, compute the number of

- weights per neuron in this layer (including bias)
- neurons in this layer
- connections into the neurons in this layer
- independent parameters in this layer

You should assume the input images are gray-scale, there is no padding, and there are 18 valid actions (outputs).

First Convolutional Layer:

$$J = K = 84, L = 4, M = N = 8, P = 0, s = 4$$

$$\text{weights per neuron: } 1 + M \times N \times L = 1 + 8 \times 8 \times 4 = 257$$

$$\text{width and height of layer: } 1 + (J - M)/s = 1 + (84 - 8)/4 = 20$$

neurons in layer:  $20 \times 20 \times 16 = 6400$   
connections:  $20 \times 20 \times 16 \times 257 = 1644800$   
independent parameters:  $16 \times 257 = 4112$

### Second Convolutional Layer:

$J = K = 20, L = 16, M = N = 4, P = 0, s = 2$

weights per neuron:  $1 + M \times N \times L = 1 + 4 \times 4 \times 16 = 257$   
width and height of layer:  $1 + (J - M) / s = 1 + (20 - 4) / 2 = 9$   
neurons in layer:  $9 \times 9 \times 32 = 2592$   
connections:  $9 \times 9 \times 32 \times 257 = 666144$   
independent parameters:  $32 \times 257 = 8224$

### Fully Connected Layer:

weights per neuron:  $1 + 2592 = 2593$   
neurons in layer: 256  
connections:  $256 \times 2593 = 663808$   
independent parameters: 663808

### Output Layer:

weights per neuron:  $1 + 256 = 257$   
neurons in layer: 18  
connections:  $18 \times 257 = 4626$   
independent parameters: 4626

---