Lecture 5 Quiz

5/6 points (83%)

Quiz, 6 questions

✔ **Congratulations! You passed!**

Next Item

Lecture 5 Quiz

0 / 1 points

5/6 points (83%)

Quiz, 6 questions

1.

For which of the following tasks can we expect that the problem of "dimension hopping" will occur (given that the data is input correctly)? Check all that apply.

- [ ] Determining whether a wave has high frequency or low frequency. The input is a set of time values along with their corresponding vertical displacements.

This should be selected

- [ ] Determining whether a given image shows a bike or a car. The bike or car might appear anywhere in the image. The input is the whole set of pixels for the image.

Correct

Dimension hopping occurs when one can take the information contained in the dimensions of some input, and move this between dimensions while not changing the target. The canonical example is taking an image of a handwritten digit and translating it within the image. The dimensions that contain "ink" are now different (they have been moved to other dimensions), however the label we assign to the digit has not changed. Note that this is not something that happens consistently across the dataset, that is we may have a dataset containing two handwritten digits where one is a translated version of the other, however this still does not change the corresponding label of the digits.

- [ ] Estimating the market price of a house given the number of rooms in the house, the number of schools and the average income of the surrounding neighbourhood, and the average sale price of the surrounding houses.

Lecture 5 Quiz

1 / 1 points

5/6 points (83%)

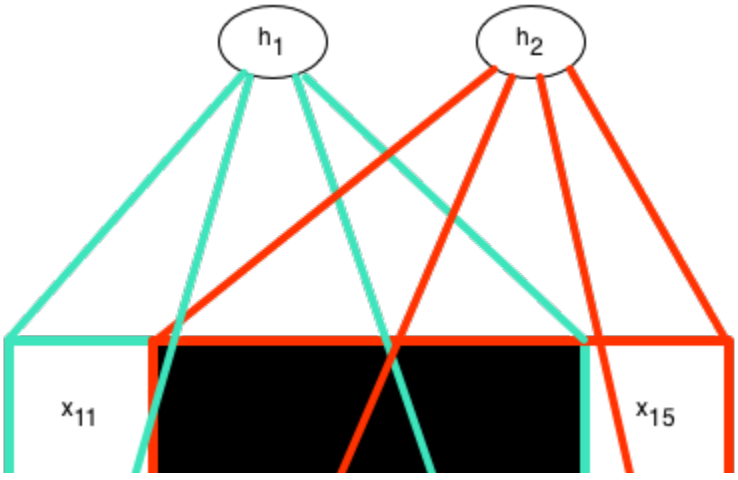
Quiz, 6 questions

2.

We have a convolutional neural network for images of 5 by 5 pixels. In this network, each hidden unit is connected to a different 4 x 4 region of the input image:

- The first hidden unit, h_1, is connected to the upper left 4x4 portion of the input image (as shown below).
- The second hidden unit, h_2, is connected to the upper right 4x4 portion of the input image (as shown below).
- The third hidden unit, h_3, is connected to the lower left 4x4 portion of the input image (not shown in the diagram).
- The fourth hidden unit, h_4, is connected to the lower right 4x4 portion of the input image (not shown in the diagram).

Because it's a convolutional network, the weights (connection strengths) are the same for all hidden units: the only difference between the hidden units is that each of them connects to a different part of the input image. In the second diagram, we show the array of weights, which are the same for each of the four hidden units.



Lecture 5 Quiz

1 / 1 points

5/6 points (83%)

Quiz, 6 questions

3.

Recall that pooling is the process of combining the outputs of several hidden units to create a single hidden unit. This introduces some invariance to local transformations in the input image.

In this example we are pooling hidden units h_1, h_2, and h_3.

Let's denote the output of hidden unit h_i as y_i. The hidden unit

that we're creating by pooling is called h_{pool}, with output y_{pool}.

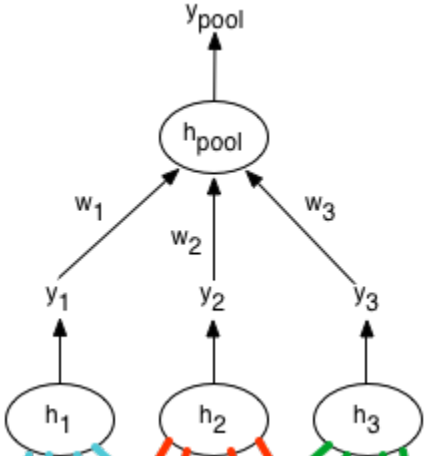
Sum pooling is defined as $y_{\text{pool}} = y_1 + y_2 + y_3$.

This form of pooling can be equivalently represented by

making h_{pool} a regular hidden unit that takes the output of the

other three hidden units, multiplies them by some weights w_1,

w_2 and w_3, adds up the resulting numbers and then outputs some function of this sum. This is illustrated in the figure below.



Lecture 5 Quiz

1 / 1 points

5/6 points (83%)

Quiz, 6 questions

4.

Suppose that we have a vocabulary of 3 words, "a", "b", and "c", and we want to predict the next word in a sentence given the previous two words. For this network, we don't want to use feature vectors for words: we simply use the local encoding, i.e.

a 3-component vector with one entry being 1 and the other

two entries being 0.

In the language models that we have seen so far, each of the context words has its own dedicated section of the network, so we would encode this problem with two 3-dimensional inputs. That makes for a total of 6 dimensions. For example, if the two preceding words (the "context" words) are "c" and "b", then the

input would be $(0, 0, 1, 0, 1, 0)$. Clearly, the more context words we want to include, the more input units our network must have. More inputs means more parameters, and thus increases the risk of overfitting. Here is a proposal to reduce the number of parameters in the model:

Consider a single neuron that is connected to this input, and call the weights that connect the input to this neuron

w_1, w_2, w_3, w_4, w_5, and w_6. w_1 connects the neuron to the first

input unit, w_2 connects it to the second input unit, etc. Notice how for every neuron, we need as many weights as there are input dimensions (6 in our case), which will be the number of words times the length of the context. A way to reduce the number of parameters is to *tie* certain weights together, so that they share a parameter. One possibility is to tie the weights coming from input units that correspond to the same word but at different context positions. In our example that

would mean that $w_1 = w_4$, $w_2 = w_5$, and $w_3 = w_6$

Lecture 5 Quiz

1 / 1 points

5/6 points (83%)

Quiz, 6 questions

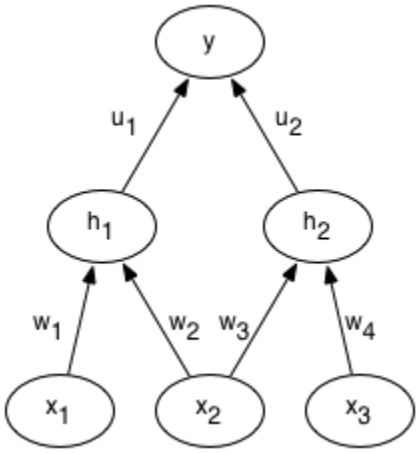
5.

Let's look at what weight tying does to gradients, computed using the backpropagation algorithm. For this question, our network has three input units,

x_1, x_2, x_3, two *logistic* hidden units h_1, h_2, four input to hidden

weights w_1, w_2, w_3, w_4, and two hidden to output weights

u_1, u_2. The output neuron y is a *linear neuron*, and we are using the *squared error* cost function.



Consider a single training case with target output t. The sequence of computations required to compute the error (this is called forward

propagation) are as follows:

$$z_1 = w_1 x_1 + w_2 x_2$$

$$z_2 = w_3 x_2 + w_4 x_3$$

Lecture 5 Quiz

1 / 1 points

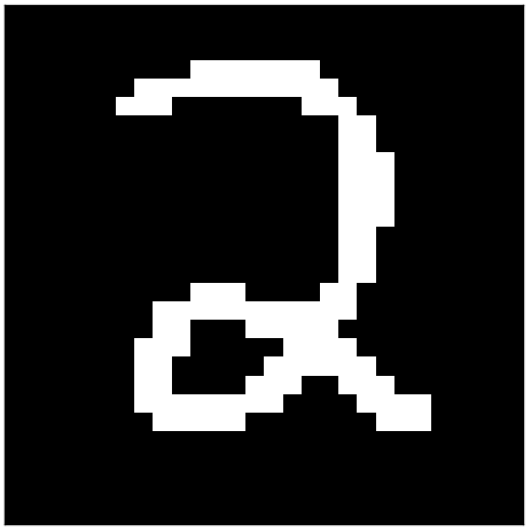
5/6 points (83%)

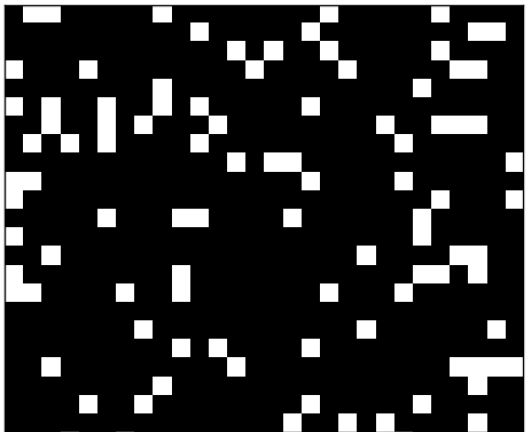
Quiz, 6 questions

6.

Oh no! Brian's done it again! Claire had a dataset of 28 x 28 pixel handwritten digits nicely prepared to train a neural network, but Brian has gone and accidentally scrambled the images by re-ordering the pixels in some totally meaningless way, and now they can't get the original dataset back! Below is an image of a handwritten '2' before and after being scrambled by Brian.

BeforeAfter





Lecture 5 Quiz

5/6 points (83%)

Quiz, 6 questions