Lecture 5 Quiz

Quiz, 6 questions

1/6 points (16%)

✖ Try again once you are ready.

Required to pass: 80 % or higher

You can retake this quiz up to 3 times every 8 hours.

Back to Week 5

Retake

✖ 0 / 1 points

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.

- [ ] Estimating the risk that a patient will develop heart disease given their age, weight, blood pressure, and cholesterol level.

Un-selected is correct

- [ ] 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.

This should be selected

- [ ] 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

☐ 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.

Un-selected is correct

---

0 / 1
points

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:

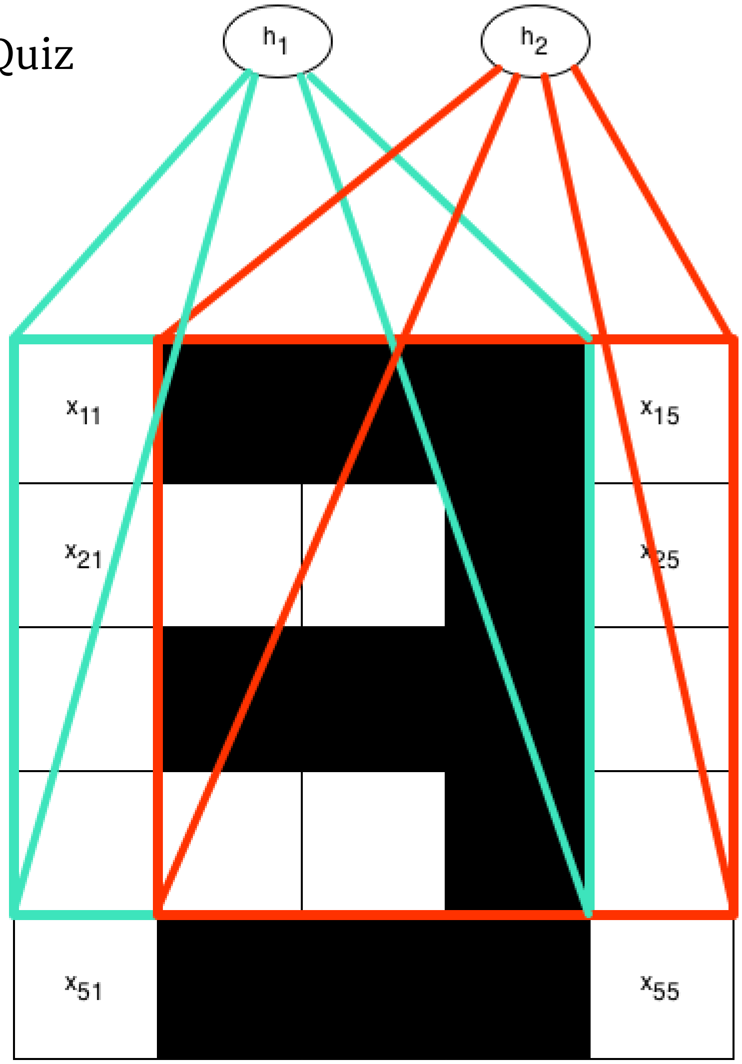
- The first hidden unit, h_1 , is connected to the upper left 4x4 portion of the input image (as shown below).
- The second hidden unit, h_2 , is connected to the upper right 4x4 portion of the input image (as shown below).
- The third hidden unit, h_3 , is connected to the lower left 4x4 portion of 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

Quiz, 6 questions

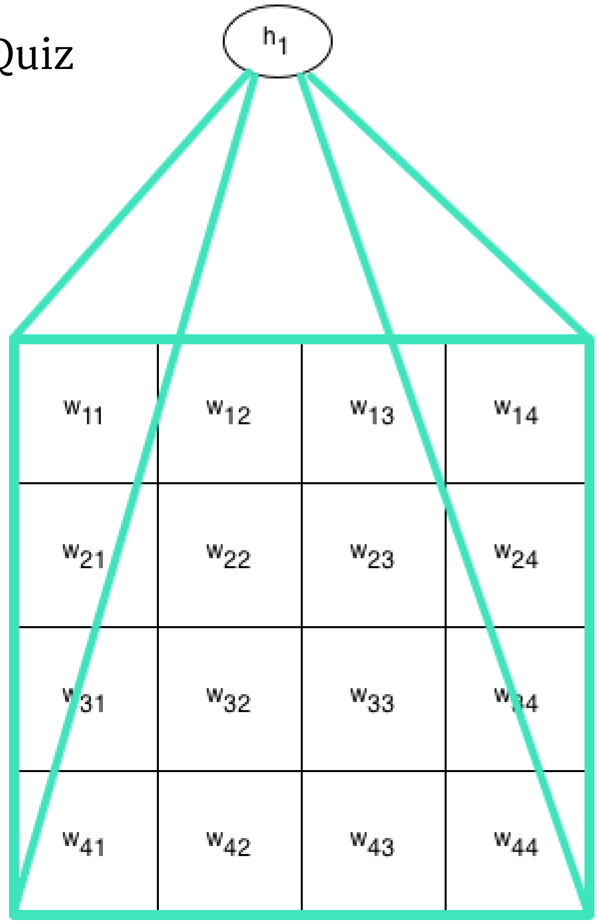
1/6 points (16%)



Lecture 5 Quiz

Quiz, 6 questions

1/6 points (16%)



For h_1 , weight w_{11} is connected to the top-left pixel, i.e. x_{11} , while for hidden unit h_2 , weight w_{11} connects to the pixel that is one to the right of the top left pixel, i.e. x_{12} .

Imagine that for some training case, we have an input image where each of the black pixels in the top diagram has value 1, and each of the white ones has value 0. Notice that the image shows a "3" in pixels.

The network has no biases. The weights of the network are given as follows:

$$
\begin{array}{llll}
w_{11}=1 & w_{12}=1 & w_{13}=1 & w_{14}=0 \\
w_{21}=0 & w_{22}=0 & w_{23}=1 & w_{24}=0 \\
w_{31}=1 & w_{32}=1 & w_{33}=1 & w_{34}=0 \\
w_{41}=0 & w_{42}=0 & w_{43}=1 & w_{44}=0
\end{array}
$$

The hidden units are *logistic*.

For the training case with that "3" input image, what is the output of each of the four hidden units?

- $h_1 = 0.881$, $h_2 = 0.982$, $h_3 = 0.982$, $h_4 = 1.000$
- $h_1 = 0.982$, $h_2 = 1.000$, $h_3 = 0.881$, $h_4 = 0.982$
- $h_1 = 0.982$, $h_2 = 0.982$, $h_3 = 0.982$, $h_4 = 0.982$
- $h_1 = 0.982$, $h_2 = 0.881$, $h_3 = 1.000$, $h_4 = 0.982$

✖ 0 / 1 points

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.

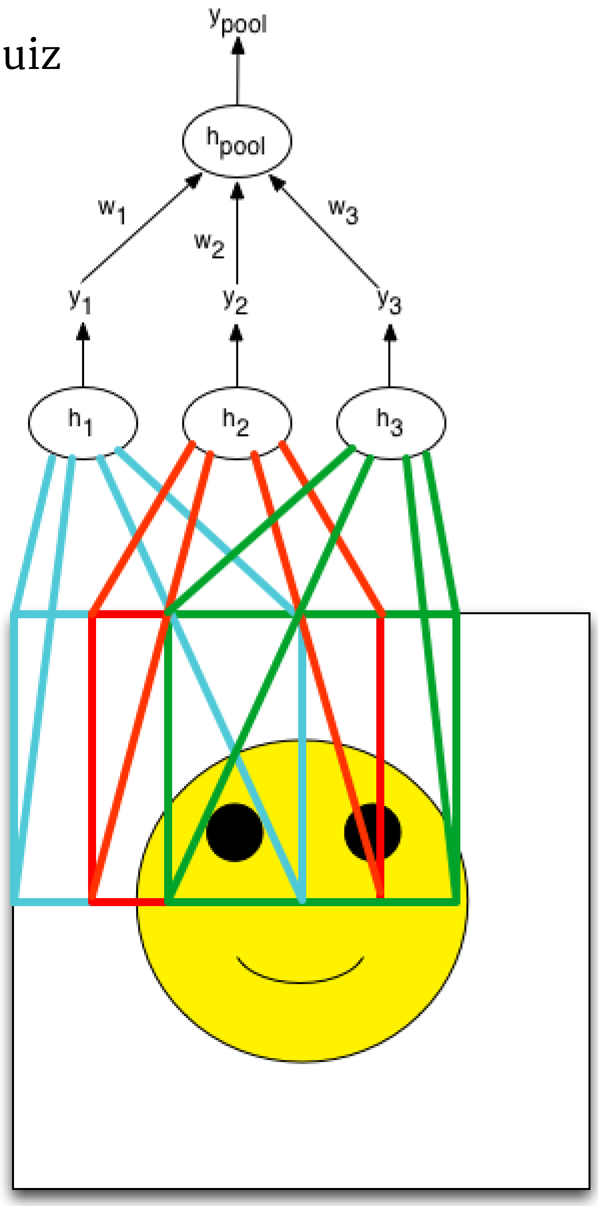
In this example we are pooling hidden units h_1 , h_2 , and h_3 . Let's denote the output of hidden unit h_i as y_i . The hidden unit that we're creating by pooling is called h_{pool} , with output y_{pool} . **Sum pooling** is defined as $y_{\text{pool}} = y_1 + y_2 + y_3$.

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

Quiz, 6 questions

1/6 points (16%)



For this to be equivalent to **sum pooling**, what type of neuron should h_{pool} be, and what should the weights be?

- $w_1 = \frac{1}{3}$, $w_2 = \frac{1}{3}$, $w_3 = \frac{1}{3}$, and h_{pool} is a linear neuron.
- $w_1 = 1$, $w_2 = 1$, $w_3 = 1$, and h_{pool} is a logistic neuron.
- $w_1 = 1$, $w_2 = 1$, $w_3 = 1$, and h_{pool} is a linear neuron.
- $w_1 = \frac{1}{3}$, $w_2 = \frac{1}{3}$, $w_3 = \frac{1}{3}$, and h_{pool} is a logistic neuron.

0 / 1
points

1/6 points (16%)

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:

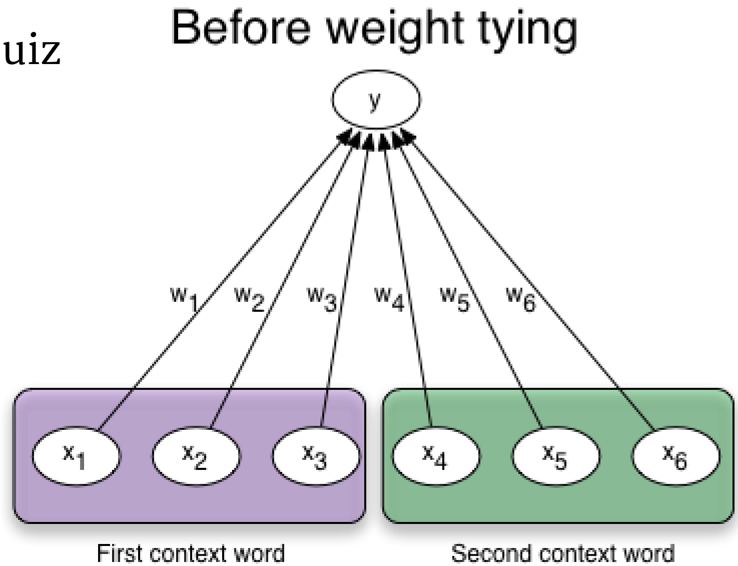
Consider a single neuron that is connected to this input, and call the weights that connect the input to this neuron w_1, w_2, w_3, w_4, w_5 , and w_6 . w_1 connects the neuron to the first input unit, w_2 connects it to the second input unit, etc. Notice how for every neuron, we need as many weights 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 $w_3 = w_6$

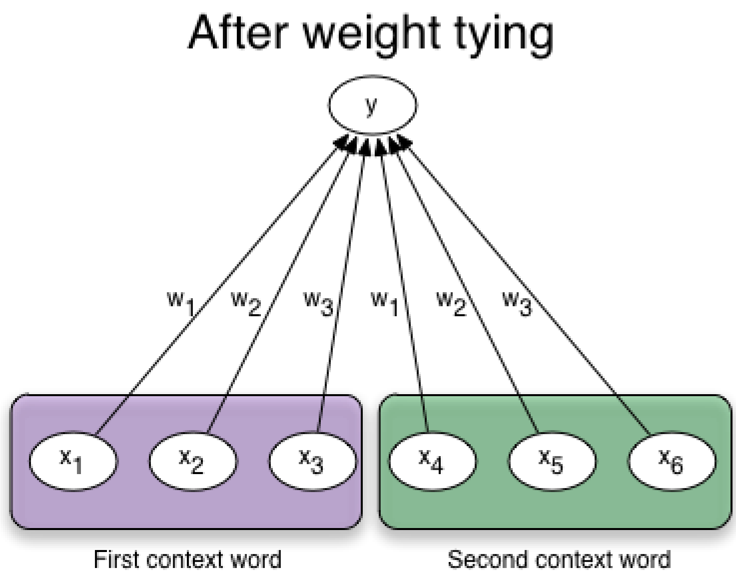
(see the "after" diagram below).

Lecture 5 Quiz

Quiz, 6 questions

1/6 points (16%)





Are there any significant problems with this approach?

- ◯ No: the new model after weight tying is an example of a convolutional neural network, and these are more powerful than a non-convolutional network because they are invariant to small transformations in the data.
- ◯ Yes: weight tying only makes sense when we are working with images.
- ◯ No: this method is an appropriate solution in that it will reduce the number of parameters and therefore always improve generalization.
- ◯

Yes: the network loses the knowledge of the location at which a context word occurs, and that is valuable knowledge.

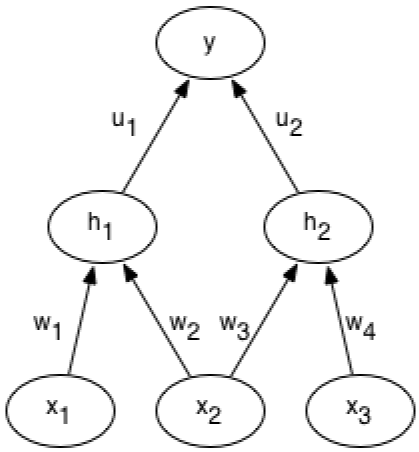
1 / 1
points

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,

, two *logistic* hidden units , four input to hidden weights , and two hidden to output weights

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



Consider a single training case with target output . The sequence of computations required to compute the error (this is called forward propagation) are as follows:

is the error. Reminder:

and are called the *total inputs* to hidden units and respectively.

Suppose we now decide to tie the weights so that . What is the derivative of the error with respect to ?

Hint: forget about weight tying for a moment and simply compute the derivatives and . Now set .

- ( ) $\frac{\partial E}{\partial w_{\text{tied}}} = -2(t-y)(u_1 h_1(1-h_1)x_2)$
- (•) $\frac{\partial E}{\partial w_{\text{tied}}} = -(t-y)u_1 h_1(1-h_1)(x_1+x_2)$

Correct

Starting from the error, backpropagation works by repeated application of the chain rule. Let's look at $\frac{\partial E}{\partial w_2}$ and remember that we are ignoring weight tying for now:

$$\frac{\partial E}{\partial w_2} = \frac{\partial E}{\partial y} \frac{\partial y}{\partial h_1} \frac{\partial h_1}{\partial z_1} \frac{\partial z_1}{\partial w_2}$$
$$\frac{\partial E}{\partial y} = -(t-y)$$
$$\frac{\partial y}{\partial h_1} = u_1$$
$$\frac{\partial h_1}{\partial z_1} = h_1(1-h_1)$$
$$\frac{\partial z_1}{\partial w_2} = x_2$$

So $\frac{\partial E}{\partial w_2} = -(t-y)u_1h_1(1-h_1)x_2$

We can similarly compute $\frac{\partial E}{\partial w_1} = -(t-y)u_1h_1(1-h_1)x_1$

To compute the derivative with tied weights, we now just need to add the derivatives:

$$\begin{aligned}\frac{\partial E}{\partial w_{\text{tied}}} &= \frac{\partial E}{\partial w_1} + \frac{\partial E}{\partial w_2} \\ &= -(t-y)u_1h_1(1-h_1)x_1 - (t-y)u_1h_1(1-h_1)x_2 \\ &= -(t-y)(u_1h_1(1-h_1))(x_1+x_2)\end{aligned}$$

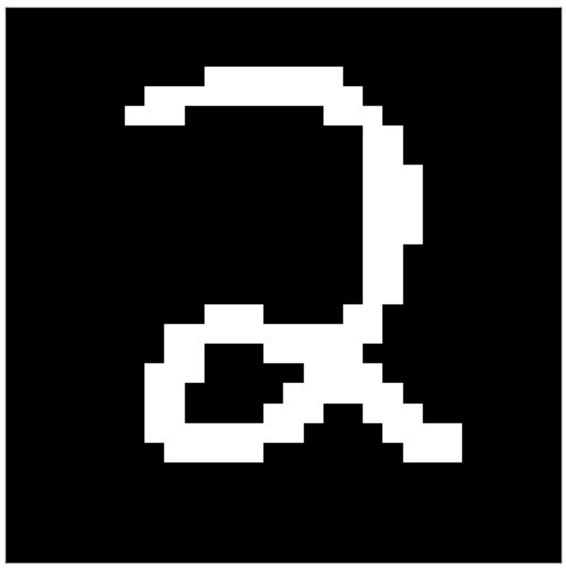
How did we know which sequence of derivatives we would need for backpropagation? One answer is to look at the equations, the other is to look at the picture. Take w_2 for example, and draw a path up the tree to the root. Notice how this path goes through h_1 and not h_2 ?

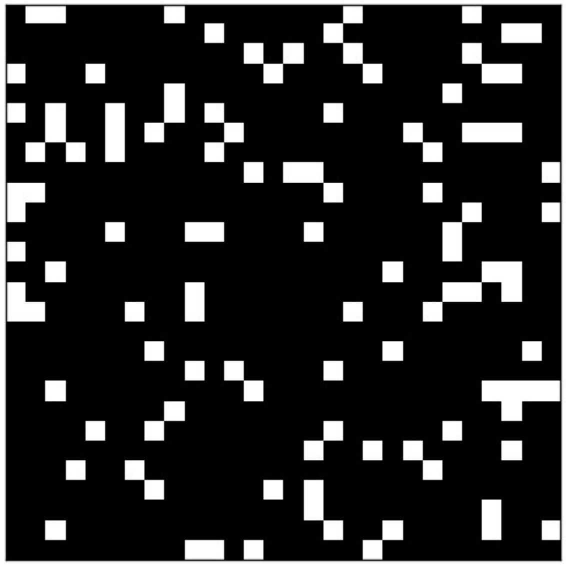
- ○ $\frac{\partial E}{\partial w_{\text{tied}}} = -(t-y)(u_1h_1(1-h_1) + u_2h_2(1-h_2))x_2$
- ○ $\frac{\partial E}{\partial w_{\text{tied}}} = -2(t-y)(u_1h_1(1-h_1)x_1)$

0 / 1
points

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





Luckily, all of the images (in both the training set and the test set) were changed in the same way. For example, if pixels number 1 and number 3 switched places in one image, then they switched places in every other image as well. Because of that, Claire thinks that perhaps she can train a network after all.

Whether Claire is right or not depends largely on the type of neural network that she has in mind. Which of the following neural networks will be at a **disadvantage** because of Brian's mistake? Check all that apply.

- [ ] A feed-forward neural network with no hidden layer and linear units (and no convolution).

Un-selected is correct

- [ ] A feed-forward neural network with no hidden layer and logistic units (and no convolution).

Un-selected is correct

- [ ] A convolutional neural network where the size of each weight filter is 10 x 10.

This should be selected

- [ ] A convolutional neural network where the size of each weight filter is 8 x 8.

This should be selected