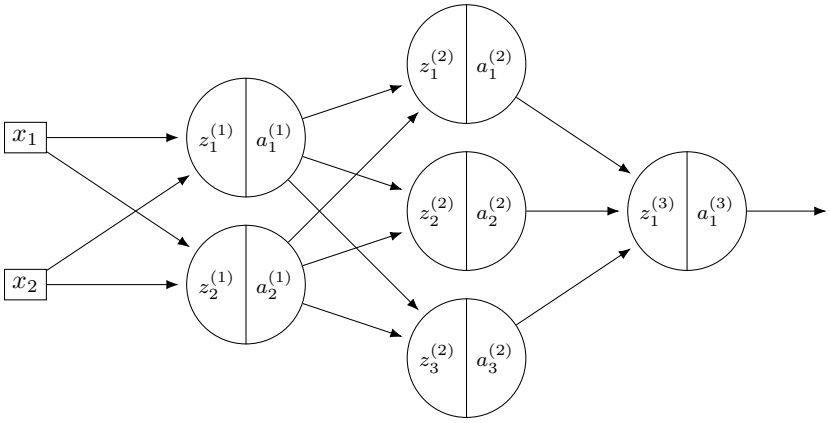
DSC 190 - Homework 08

Due: Wednesday, May 31

Write your solutions to the following problems by either typing them up or handwriting them on another piece of paper. Unless otherwise noted by the problem's instructions, show your work or provide some justification for your answer. Homeworks are due via Gradescope at 11:59 PM.

Problem 1.

Consider the neural network H shown below. You may assume for simplicity that there are no bias weights. Assume that the hidden layers use the ReLU as their activation function, and that the output layer uses the linear activation.



Suppose that $\vec{x} = (2, 1)^T$ and that the weights of this network are:

$$W^{(1)} = \begin{pmatrix} 2 & -2 \\ 3 & 1 \end{pmatrix} \qquad W^{(2)} = \begin{pmatrix} 2 & 1 & 2 \\ -3 & -1 & 0 \end{pmatrix} \qquad W^{(3)} = \begin{pmatrix} 4 \\ 1 \\ 2 \end{pmatrix}$$

Carry out backpropagation in order to fill in the tables of partial derivatives below. Each entry should be a number. Show your work.

Hint: some of the entries of the tables have been filled in for you – use these to check your work.

$\frac{\partial H}{\partial W_{11}^{(1)}}$	
$\frac{\partial H}{\partial W_{12}^{(1)}}$	
$\frac{\partial H}{\partial W_{21}^{(1)}}$	13
$\frac{\partial H}{\partial W_{22}^{(1)}}$	

$\frac{\partial H}{\partial W_{11}^{(2)}}$	
$\frac{\partial H}{\partial W_{12}^{(2)}}$	7
$\frac{\partial H}{\partial W_{13}^{(2)}}$	
$\frac{\partial H}{\partial W_{21}^{(2)}}$	
$\frac{\partial H}{\partial W_{22}^{(2)}}$	
$\frac{\partial H}{\partial W_{23}^{(2)}}$	

$\frac{\partial H}{\partial W_{11}^{(3)}}$	
$\frac{\partial H}{\partial W_{21}^{(3)}}$	
$\frac{\partial H}{\partial W_{31}^{(3)}}$	14

Solution:

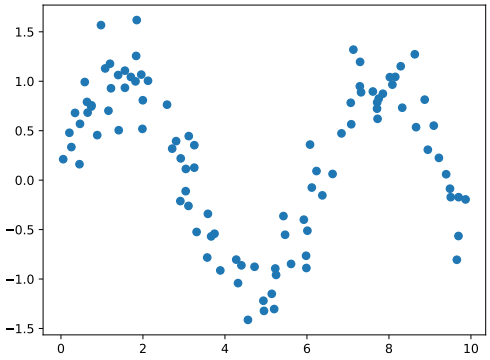
$\frac{\partial H}{\partial W_{11}^{(1)}}$	26
$\frac{\partial H}{\partial W_{12}^{(1)}}$	0
$\frac{\partial H}{\partial W_{21}^{(1)}}$	13
$\frac{\partial H}{\partial W_{22}^{(1)}}$	0

$\frac{\partial H}{\partial W_{11}^{(2)}}$	28
$\frac{\partial H}{\partial W_{12}^{(2)}}$	7
$\frac{\partial H}{\partial W_{13}^{(2)}}$	14
$\frac{\partial H}{\partial W_{21}^{(2)}}$	0
$\frac{\partial H}{\partial W_{22}^{(2)}}$	0
$\frac{\partial H}{\partial W_{23}^{(2)}}$	0

$\frac{\partial H}{\partial W_{11}^{(3)}}$	14
$\frac{\partial H}{\partial W_{21}^{(3)}}$	7
$\frac{\partial H}{\partial W_{31}^{(3)}}$	14

Problem 2.

This problem will use the data shown below:



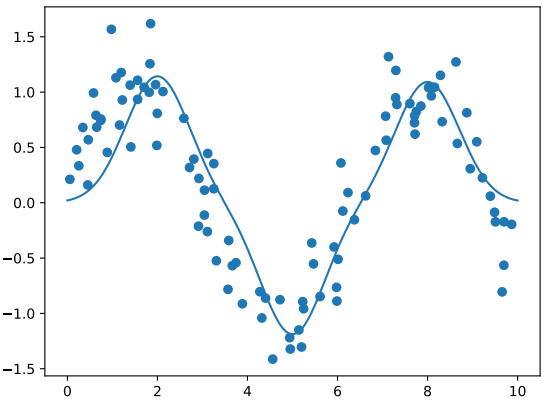
You find this data at:

`https://f000.backblazeb2.com/file/jeldridge-data/008-noisy_sin/data.csv`

a) First, train a Gaussian RBF network H on this data using three Gaussians, centered at 2, 5, and 8, and each with a width parameter of $\sigma = 1$. Plot $H(x)$ for x from 0 to 10, on top of a scatter plot of the data to visualize the accuracy of its predictions.

Hint: See Discussion 06.

Solution:




```
xx = np.linspace(0, 10, 200)
```

```
def make_phi(center, sigma):
    def phi(x):
        return np.exp(-(x - center)**2 / sigma**2)
    return phi

phis = [make_phi(mu, 1) for mu in [2, 5, 8]]

Phi = np.column_stack([phi(x) for phi in phis])

w = np.linalg.lstsq(Phi, y)[0]

Z = np.column_stack([phi(xx) for phi in phis])

plt.plot(xx, Z @ w)
plt.scatter(x, y)
```

b) Next, train a neural network H on the same data. The network should have two hidden layers: each containing 5 nodes. Use sigmoid activations for the hidden layers, and linear activation for the output layer. Train your network by minimizing the mean squared error.

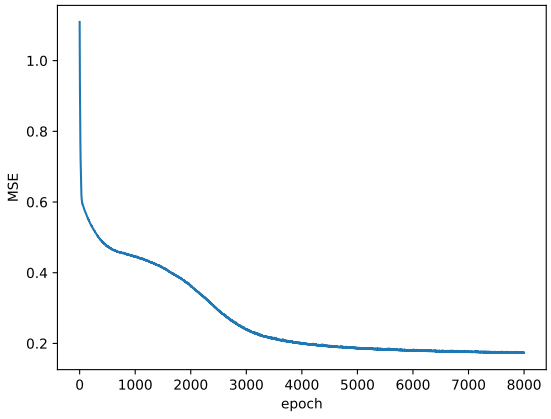
Plot two things: 1) the mean squared error as a function of the training epoch; and 2) $H(x)$ for x from 0 to 10, on top of a scatter plot of the data.

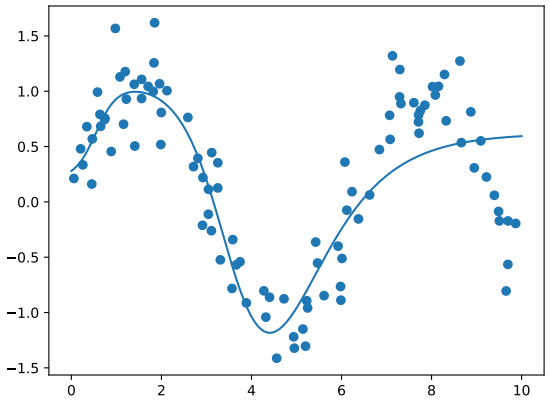
Hint: See Discussion 07. Not covered in that discussion was how to retrieve the mean squared error after each epoch. It turns out that the `.fit()` method returns a `History` object, which itself has a `.history` attribute that is a dictionary. One of the keys of this dictionary is `loss`; its corresponding value is the empirical risk after each epoch of SGD (or whatever optimization algorithm is used).

You will need to determine how many epochs of SGD to run; you can use the plot to do this. If you choose the number of epochs large enough, the plot should be mostly flat towards the end – otherwise, you could decrease MSE substantially by running SGD longer. Conversely, if your plot of $H(x)$ does not appear to fit the data, you might need to run for more epochs (or just try running again – you might be stuck in a local minimum). You should be able to attain an MSE of less than 0.25.

You might find that your NN does not fit the data quite as well as the RBF network – that's OK.

Solution:






```
inputs = keras.Input(shape=1)
hidden_layer_1 = keras.layers.Dense(3, activation='sigmoid')(inputs)
hidden_layer_2 = keras.layers.Dense(2, activation='sigmoid')(hidden_layer_1)
outputs = keras.layers.Dense(1, activation='linear')(hidden_layer_2)
model = keras.Model(inputs=inputs, outputs=outputs)

model.compile(
    loss=keras.losses.MeanSquaredError()
)

history = model.fit(np.expand_dims(x, axis=1), y, epochs=12000, verbose=0)

plt.plot(history.history['loss'])
plt.xlabel('epoch')
plt.ylabel('MSE')

plt.figure()
z = model(np.expand_dims(xx, axis=1))
plt.plot(xx, z.numpy().flatten())
plt.scatter(x, y)
```