This homework builds on the material in the notes on neural networks up through and including section 6 on loss functions.

In particular, in this homework we consider neural networks with multiple layers. Each layer has multiple inputs and outputs, and can be broken down into two parts:

- A **linear** module that implements a linear transformation: $z_j = (\sum_{i=1}^{m} x_i w_{i,j}) + w_{0\,j}$ specified by a weight matrix W and a bias vector W_0. The output is $[z_1, \dots, z_n]^T$.
- An **activation** module that applies an activation function to the outputs of the linear module for some activation function f, such as Tanh or ReLU in the hidden layers or Softmax (see below) at the output layer. We write the output as: : $[f(z_1), \dots, f(z_m)]^T$, although technically, for some activation functions such as softmax, each output will depend on all the z_i, not just one.

We will use the following notation for quantities in a network:

- Inputs to the network are $x_1, \dots, x_d$.
- Number of layers is L
- There are m^l inputs to layer l
- There are $n^l = m^{l+1}$ outputs from layer l
- The weight matrix for layer l is W^l, an $m^l \times n^l$ matrix, and the bias vector (offset) is W_0^l, an $n^l \times 1$ vector
- The outputs of the linear module for layer l are known as **pre-activation** values and denoted z^l
- The activation function at layer l is $f^l(\cdot)$
- Layer l activations are $a^l = [f^l(z_1^l), \dots, f^l(z_{n^l}^l)]^T$
- The output of the network is the values $a^L = [f^L(z_1^L), \dots, f^L(z_{n^L}^L)]^T$
- Loss function $Loss(a, y)$ measures the loss of output values a when the target is y

1) Loss functions and output activations: classification

When doing classification, it's natural to think of the output values as being discrete: +1 and -1. But it is generally difficult to use optimization-based methods without somehow thinking of the outputs as being continuous (even though you will have to discretize when it's time to make a prediction).

1.1) Hinge loss, linear activation

When we looked at the SVM objective for classification, we did this:

- Defined the output space to be $\mathbb{R}$
- Developed the hinge loss function

$$Loss(a, y) = L_h(ya) = \begin{cases} 0 & \text{if } ya > 1 \\ 1 - ya & \text{otherwise} \end{cases}$$

  where a is the continuous output (we're using a here to be consistent with the neural network terminology of *activation*) and y is the desired/target output
- Tried to find parameters θ of our model to minimize loss summed over the training data

Consider a single "neuron" with a linear activation function; that is, where $a_1^L = \sum_k w_{k,1}^L x_k + w_{0,1}^L$. In this case, we have $L = 1$ and $f^L(z) = z$.

1.1.A) Write a short program to compute the gradient of the loss function with respect to the weight vector (not the bias): $\nabla_{w^L} Loss(a^L_1, y)$ when $Loss(a, y) = L_h(ya)$.

- `x` is a column vector
- `y` is a number, a label
- `a` is a number, an activation

It should return a column vector.

```
def hinge_loss_grad(x, y, a):
  d, n = x.shape
  if a * y > 1:
```

Run Code | Submit | View Answer | Ask for Help **100.00%**

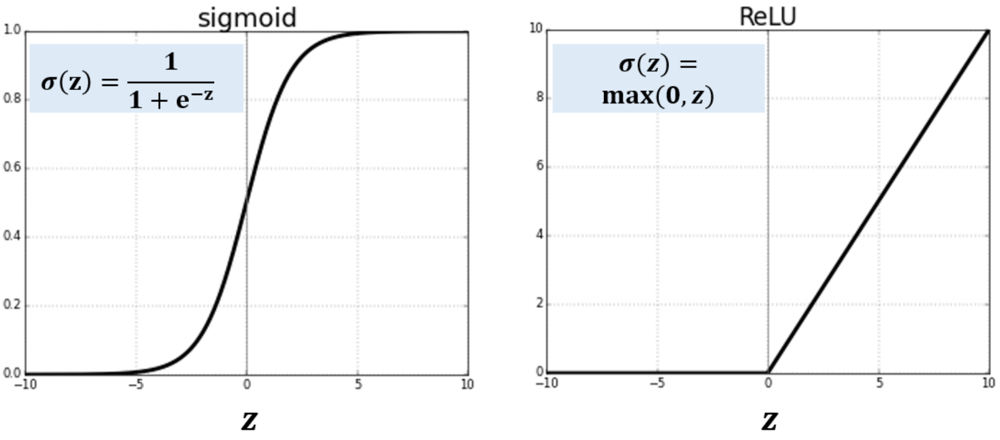
You have infinitely many submissions remaining.

1.2) Log loss, sigmoidal activation

Another way to make the output for a classifier continuous is to make it be in the range $(0, 1)$, which admits the interpretation of being the predicted **probability** that the example is positive. A convenient way to make the activation of a unit be in the range $(0, 1)$ is to use a *sigmoid* function:

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

The figure below shows a sigmoid activation function on the left, with the rectified linear (ReLU) activation function on the right for comparison.



1.2.A) What is an expression for the derivative of the sigmoid with respect to z, expressed as a function of z, its input?

Enter a Python expression (use ** for exponentiation) involving `e` and `z`:

(e**(-z))/(1+e**-z)**2

Check Syntax | Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

1.2.B) What is an expression for the derivative of the sigmoid with respect to z, but this time expressed as a function of $o = \sigma(z)$, its output?
Hint: Think about the expression $1 - \frac{1}{1+e^{-z}}$. (Here is a review of computing derivatives.)

Enter a Python expression (use ** for exponentiation) involving only `o` (note: `e` and `z` are not allowed, and remember $o = \sigma(z)$): (1-o)*o

Check Syntax | Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

In this model, we will consider positive points to have label +1, and negative points to have label 0.

We need a loss function that works well when we are predicting probabilities. A good choice is to ask what probability is assigned to the correct label. We will interpret the value outputted by our classifier as the probability that the example is positive. So, if the output value is a and the true label is $+1$, then the probability assigned to the true label is a; on the other hand, if the true label is 0, then the probability assigned to the true label is $1 - a$. Because we actually will be interested in the probability of the predictions on the whole data set, we'd want to choose weights to **maximize**

$$\prod_t P(a^{(t)}, y^{(t)})$$

where $P(a^{(t)}, y^{(t)})$ is the probability that the network predicts the correct label for data point (t).

Using a notational trick (which turns an *if* expression into a product) that might seem unmotivated now, but will be useful later, we can write the probability $P(a, y)$ as

$$P(a, y) = a^y (1 - a)^{(1-y)}.$$

1.2.C) What is the value of $P(a, y)$ when $y = 0$?

Enter an expression for $P(a, y)$ when $y = 0$ in terms of `a`: 1-a

Check Syntax | Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

1.2.D) What is the value of $P(a, y)$ when $y = 1$?

Enter an expression for $P(a, y)$ when $y = 1$ in terms of `a`: a

Check Syntax | Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

1.2.E) Find a simplified expression for $\log P(a, y)$ that does not use exponentiation. Note that we refer to the natural logarithm $\ln$ as $\log$ throughout this assignment, consistent with the lecture notes.

Enter an expression in terms of `y` and `a`; you can use `log(.)` to indicate natural log:

y*log(a) + (1-y)*log(1-a)

Check Syntax | Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

In fact, because log is a monotonic function, the same weights that maximize the product of the probabilities will minimize the *negative log likelihood* ("likelihood" is the same as probability; we just use that name here because the phrase is an idiom in machine learning, abbreviated NLL):

$$Loss(a, y) = NLL(a, y) = -y \log a - (1 - y) \log(1 - a).$$

Our objective function (over our n data points) will then be

$$\sum NLL(a^{(t)}, y^{(t)}) = -\sum_{t=1}^{n} \left[y^{(t)} \log a^{(t)} + (1 - y^{(t)}) \log(1 - a^{(t)}) \right].$$

Remember that $a^{(t)}$ is our model's output for training example t, and $y^{(t)}$ is the true label (+1 or 0).

Now, we can think about a single unit with a sigmoidal activation function, trained to minimize NLL. So, $a_1^L = \sigma(\sum_k w_{k,1}^L x_k + w_{0,1}^L)$. In this case, we have $L = 1$.

1.2.F) Write a formula for the gradient of the NLL with respect to the first weight, $\nabla_{w_{1,1}^L} NLL(a_1^L, y)$, for a single training example. Hint: consider using the chain rule; the final answer (expression) is very short.

Write an expression in terms of `x_1`, `a_1`, and `y`: (a_1 - y) * x_1

Check Syntax | Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

1.2.G) Write a formula for the gradient of the NLL with respect to the full weight vector, $\nabla_{W^L} NLL(a_1^L, y)$, for a single training example.

Enter an expression in terms of `x`, `a_1`, and `y`: (a_1 -y) *x

Check Syntax | Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

2) Multiclass classification

What if we needed to classify homework problems into three categories: enlightening, boring, impossible? We can do this by using a "one-hot" encoding on the output, and using three output units with what is called a "softmax" (SM) activation module. It's not a typical activation module, since it takes in all n_L pre-activation values z_j^L in $\mathbb{R}$ and returns n_L output values $a_j^L \in [0, 1]$ such that $\sum_j a_j^L = 1$. This can be interpreted as representing a probability distribution over the possible categories.

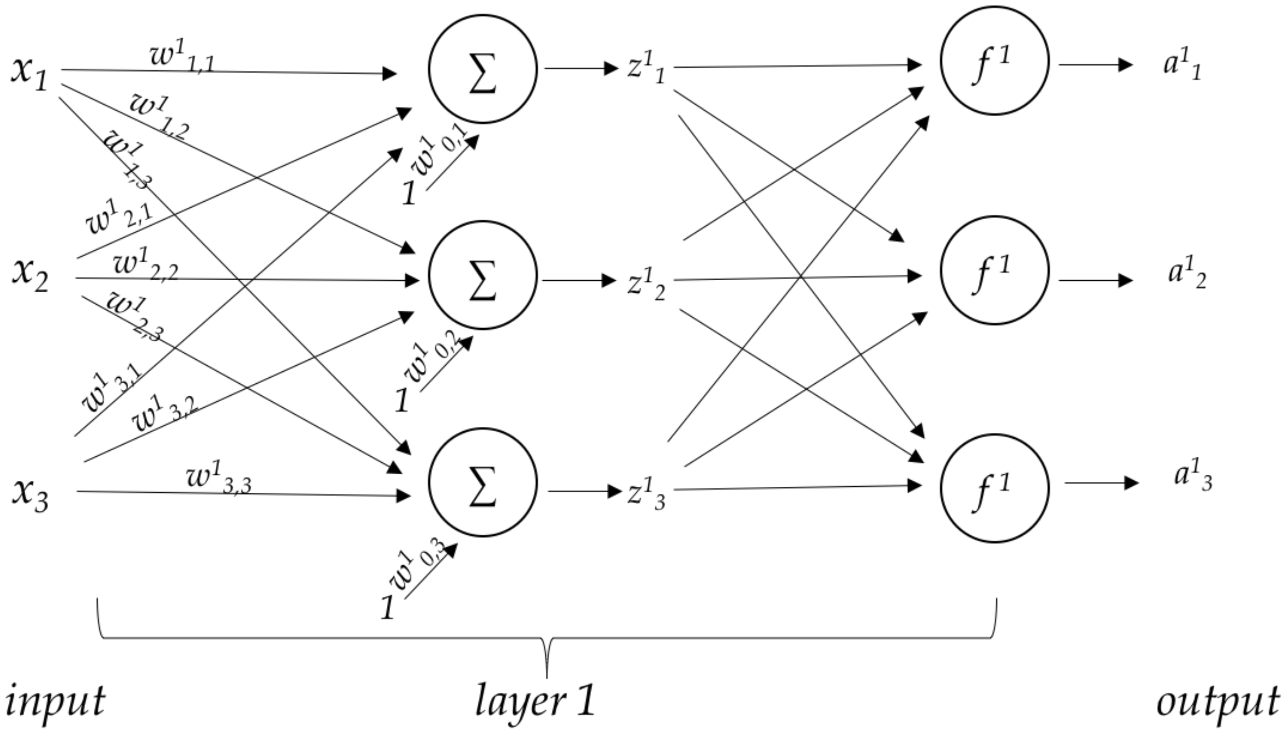
The individual entries are computed as

$$a_j = \frac{e^{z_j}}{\sum_{k=1}^{n^L} e^{z_k}}$$

We'll describe the relationship of the vector a on the vector z as

$$a = \mathrm{SM}(z)$$

The network below shows a one-layer network with a linear module followed by a softmax activation module.



2.A) What probability distribution over the categories is represented by $z^L = [-1, 0, 1]^T$?

Enter a distribution (a list of three numbers adding up to 1) for the three categories. Your answers should be numeric (please enter numbers, and do not use the symbol e):

[0.09, 0.24, 0.66]

Submit View Answer Ask for Help **100.00%**

You have infinitely many submissions remaining.

Now, we need a loss function $Loss(a, y)$ where a is a discrete probability distribution and y is a one-hot vector encoding of a single output value. It makes sense to use negative log likelihood as a loss function for the same reasons as before. So, we'll just extend our definition of NLL from earlier:

$$NLL(a, y) = -\sum_{j=1}^{n^L} y_j \ln a_j^L.$$

Note that the above expression is for multi-classes (number of class > 2). For two-classes, the expression reduce to what you saw after Problem 1.2.E.

2.B) If $a = [.3, .5, .2]^T$ and $y = [0, 0, 1]^T$, what is $NLL(a, y)$?

Enter an expression involving `log(.)` (for natural log) and constants: -log(0.2)

Check Syntax Submit View Answer Ask for Help **100.00%**

You have infinitely many submissions remaining.

Now, we can think about a single layer with a softmax activation module, trained to minimize NLL. The pre-activation values (the output of the linear module) are:

$$z_j^L = \sum_k w_{k,j}^L x_k + w_{0,j}^L$$

and $a^L = SM(z^L)$.

To do gradient descent, we need to know $\frac{\partial}{\partial w_{k,j}^L} NLL(a^L, y)$. We'll reveal the secret (that you might guess from Problem 1) that it has an awesome form! (Please consider deriving this, for fun and satisfaction!)

$$\frac{\partial}{\partial w_{k,j}^L} NLL(a^L, y) = x_k(a_j^L - y_j)$$

And of course, it's easy to compute the whole matrix of these derivatives, $\nabla_{W^L} NLL(a^L, y)$, in one quick matrix computation.

2.C) Suppose we have two input units and three possible output values, and the weight matrix W^L is

$$W^L = \begin{bmatrix} 1 & -1 & -2 \\ -1 & 2 & 1 \end{bmatrix}$$

or in Python form: `w = np.array([[1, -1, -2], [-1, 2, 1]])`.

Assume the biases are zero, the input $x = [1, 1]^T$ (e.g., `x = np.array([[1, 1]]).T`), and the target output $y = [0, 1, 0]^T$ (e.g., `y = np.array([[0, 1, 0]]).T`). What is the matrix $\nabla_{W^L} NLL(a^L, y)$? Hint: You might want to solve using Python and numpy, or using colab for calculation.

Enter the matrix as a list of lists, one list for each row of the matrix. Please enter values with a precision of three decimal points. [[0.244, -0.334,   0.09003057] , [0.244, -0.334, 0.09003057]]

Submit View Answer Ask for Help **100.00%**

You have infinitely many submissions remaining.

2.D) What is the predicted probability that x is in class 1, before any gradient updates? (Assume we have classes 0, 1, and 2.)

Enter a number: 0.6652

Submit View Answer Ask for Help **100.00%**

You have infinitely many submissions remaining.

2.E) Using step size 0.5, what is W^L after one gradient update step?

Enter the matrix as a list of lists, one list for each row of the matrix. Please enter values with a precision of three decimal points. [[0.87763576, -0.83262048, -2.04501529], [-1.12236424, 2.16737952, 0.95498471]]

Submit View Answer Ask for Help **100.00%**

You have infinitely many submissions remaining.

2.F) What is the predicted probability that x is in class 1, given the new weight matrix?

Enter a number: 0.77245284

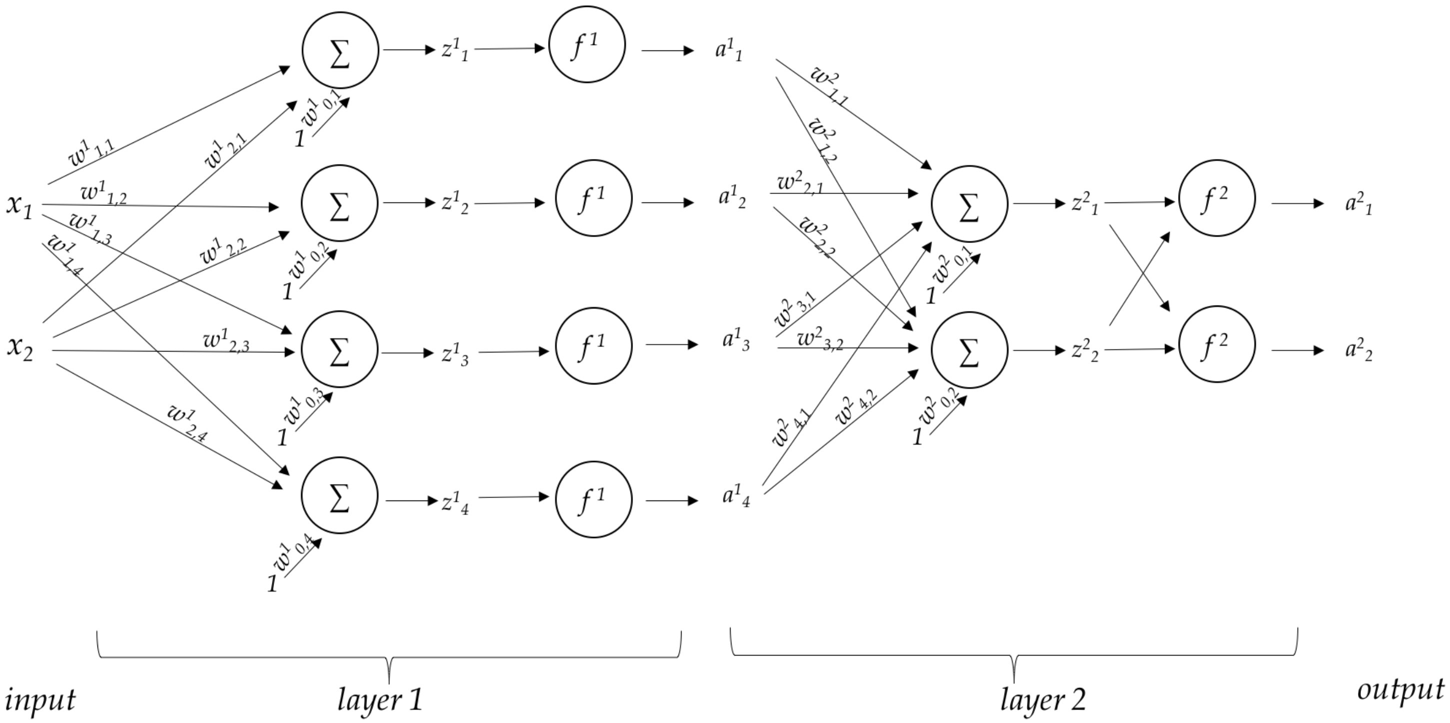
Submit View Answer Ask for Help **100.00%**

You have infinitely many submissions remaining.

3) Neural Networks

In this problem we will analyze a simple neural network to understand its classification properties. You might find the colab file useful. However, we encourage you to go through all the calculation by hand once, which should be a good practice.

Consider the neural network given in the figure below, with ReLU activation functions (f^1 in the figure) on all hidden neurons, and softmax activation (f^2 in the figure) for the output layer, resulting in softmax outputs (a_1^2 and a_2^2 in the figure).



Given an input $x = [x_1, x_2]^T$, the hidden units in the network are activated in stages as described by the following equations:

$$
\begin{aligned}
z_1^1 &= x_1 w_{1,1}^1 + x_2 w_{2,1}^1 + w_{0,1}^1 & a_1^1 &= \max\{z_1^1, 0\} \\
z_2^1 &= x_1 w_{1,2}^1 + x_2 w_{2,2}^1 + w_{0,2}^1 & a_2^1 &= \max\{z_2^1, 0\} \\
z_3^1 &= x_1 w_{1,3}^1 + x_2 w_{2,3}^1 + w_{0,3}^1 & a_3^1 &= \max\{z_3^1, 0\} \\
z_4^1 &= x_1 w_{1,4}^1 + x_2 w_{2,4}^1 + w_{0,4}^1 & a_4^1 &= \max\{z_4^1, 0\}
\end{aligned}
$$

$$z_1^2 = a_1^1 w_{1,1}^2 + a_2^1 w_{2,1}^2 + a_3^1 w_{3,1}^2 + a_4^1 w_{4,1}^2 + w_{0,1}^2$$
$$z_2^2 = a_1^1 w_{1,2}^2 + a_2^1 w_{2,2}^2 + a_3^1 w_{3,2}^2 + a_4^1 w_{4,2}^2 + w_{0,2}^2.$$

The final output of the network is obtained by applying the *softmax* function to the last hidden layer,

$$a_1^2 = \frac{e^{z_1^2}}{e^{z_1^2} + e^{z_2^2}}$$
$$a_2^2 = \frac{e^{z_2^2}}{e^{z_1^2} + e^{z_2^2}}.$$

In this problem, we will consider the following setting of parameters:

$$\begin{bmatrix} w_{1,1}^1 & w_{1,2}^1 & w_{1,3}^1 & w_{1,4}^1 \\ w_{2,1}^1 & w_{2,2}^1 & w_{2,3}^1 & w_{2,4}^1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}, \qquad \begin{bmatrix} w_{0,1}^1 \\ w_{0,2}^1 \\ w_{0,3}^1 \\ w_{0,4}^1 \end{bmatrix} = \begin{bmatrix} -1 \\ -1 \\ -1 \\ -1 \end{bmatrix},$$

$$\begin{bmatrix} w_{1,1}^2 & w_{1,2}^2 \\ w_{2,1}^2 & w_{2,2}^2 \\ w_{3,1}^2 & w_{3,2}^2 \\ w_{4,1}^2 & w_{4,2}^2 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 1 & -1 \\ 1 & -1 \\ 1 & -1 \end{bmatrix}, \qquad \begin{bmatrix} w_{0,1}^2 \\ w_{0,2}^2 \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}.$$

3.1) Output

Consider the input $x_1 = 3$, $x_2 = 14$.

3.1.A) What are the outputs of the hidden units, $(f^1(z_1^1), f^1(z_2^1), f^1(z_3^1), f^1(z_4^1))$.

Enter a Python list of 4 numbers: [2, 13, 0, 0]

Submit View Answer Ask for Help **100.00%**

You have infinitely many submissions remaining.

3.1.B) What is the final output (a_1^2, a_2^2) of the network?

Enter a Python list of 2 numbers: [1, 0]

Submit View Answer Ask for Help **100.00%**

You have infinitely many submissions remaining.

3.2) Unit decision boundaries

Let's characterize the *decision boundaries* in x-space, corresponding to the four hidden units. These are the regions where the input to the units $z_1^1, z_2^1, z_3^1, z_4^1$ are exactly zero.

Hint: You should draw a diagram of the decision boundaries for each unit in the x-space and label the sides of the boundaries with 0 and $+$ to indicate whether the unit's output would be exactly 0 or positive, respectively. (The diagram should be a 2D plot with x_1 and x_2 on each axis, with lines for $z_1^1 = 0$, $z_2^1 = 0$, $z_3^1 = 0$, $z_4^1 = 0$.)

3.2.A) What is the shape of the decision boundary for a single unit?

Choose one: Line

Submit View Answer Ask for Help **100.00%**

You have infinitely many submissions remaining.

3.2.B) Enter a 2 x 4 matrix where each column represents a (different) input vector $[x_1, x_2]^T$ each of which is on the decision boundary for the first unit, that is, for which $z_1^1 = 0$. (There are multiple possible answers.)

Enter a Python list of lists where each list is a row of the matrix.

[[1, 1, 1, 1], [1, 3, 234, 546853]]

Submit View Answer Ask for Help **100.00%**

You have infinitely many submissions remaining.

3.2.C) Consider the following input vectors: $x^{(1)} = [0.5, 0.5]^T, x^{(2)} = [0, 2]^T, x^{(3)} = [-3, 0.5]^T$. Enter a matrix where each column represents the outputs of the hidden units ($f(z_1^1), \ldots, f(z_4^1)$) for each of the input vectors. You can use your diagram of decision boundaries.

Enter a Python list of lists where each list is a row of the matrix.

[[0, 0, 0],[0, 1, 0],[0, 0, 2],[0, 0, 0]]

Submit View Answer Ask for Help **100.00%**

You have infinitely many submissions remaining.

3.3) Network outputs

In our network above, the output layer with two softmax units is used to classify into one of two classes. For class 1, the first unit's output should be larger than the other unit's output, and for class 2, the second unit's output should be larger. This generalizes nicely to k classes by using k output units.

(We have previously examined addressing two-class classification problems using a single output unit with a sigmoid activation; this is another way to address them.)

Let's characterize the region in x-space where this network's output indicates the first class (that is, a_1^2 is larger) or indicates the second class (that is, a_2^2 is larger). Your diagram from the previous part will be useful here.

What is the output value of the neural network in each of the following cases? Write your answer for a_i^2 as expressions, you can use powers of e, for example, `e**2 + 1`; the exponents can be negative, `e**(-2) + 1`.

Case 1) For $f(z_1^1) + f(z_2^1) + f(z_3^1) + f(z_4^1) = 0$

3.3.A)

$a_1^2 =$ e**0/(e**0 + e**2)

Check Syntax | Submit | View Answer | Ask for Help 100.00%

You have infinitely many submissions remaining.

3.3.B)

$a_2^2 =$ e**2/(e**0 + e**2)

Check Syntax | Submit | View Answer | Ask for Help 100.00%

You have infinitely many submissions remaining.

3.3.C)

Which class is predicted? Class 2

Submit | View Answer | Ask for Help 100.00%

You have infinitely many submissions remaining.

Case 2) For $f(z_1^1) + f(z_2^1) + f(z_3^1) + f(z_4^1) = 1$

3.3.D)

$a_1^2 =$ 1/2

Check Syntax | Submit | View Answer | Ask for Help 100.00%

You have infinitely many submissions remaining.

3.3.E)

$a_2^2 =$ 1/2

Check Syntax | Submit | View Answer | Ask for Help 100.00%

You have infinitely many submissions remaining.

3.3.F)

Which class is predicted? Boundary

Submit | View Answer | Ask for Help 100.00%

You have infinitely many submissions remaining.

Case 3) For $f(z_1^1) + f(z_2^1) + f(z_3^1) + f(z_4^1) = 3$

3.3.G)

$a_1^2 =$ e**3/(e**3+e**-1)

Check Syntax | Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

3.3.H)

$a_2^2 =$ e**-1/(e**3+e**-1)

Check Syntax | Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.

3.3.I)

Which class is predicted? Class 1

Submit | View Answer | Ask for Help **100.00%**

You have infinitely many submissions remaining.