Lab -- Neural Networks

Group information

Create Your Group

Once you and your partner(s) are in a breakout room, you need to create a group.

Instructions:

(1) **One** of you should click here to create a group.

(2) Everyone else should then enter the given new group name (including trailing numbers) into the box below (and press enter when done).

Name of group to join:

Reload this page anytime to refresh your group status (and to get a delete button, if you want to remove yourself from a group you've joined).

Groups are limited to 3 persons max each, so that checkoff discussions can be inclusive.

You are not currently in any group

Neural Networks

For this lab, you will need to understand the material in the notes on Neural Networks up through and including section 6 on loss functions.

1) Exploring neural networks

Notes on the functions in the code boxes below:

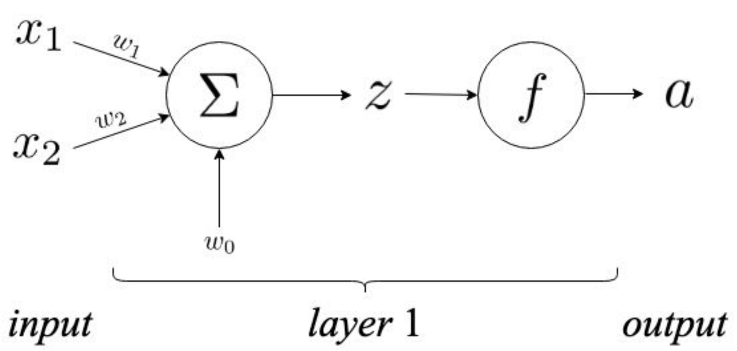
- The initialization of weights in a neural network is random, so subsequent runs might produce different results.
- `iters` indicates how many single steps of SGD to run, each using one training point.
- Pay attention to the decision boundaries, shown in the answers after you run the code. Also note that the accuracy of the network on the training data is printed above the graph.

1.1) Simple separable data set

Here's a very simple data set.

```
X = np.array([[2, 3, 9, 12],
              [5, 2, 6, 5]])
Y = np.array([[1, 0, 1, 0]])
```

The code below runs a very simple neural network composed of a single sigmoid unit with two inputs, using negative log likelihood (NLL) loss:



The function `sigmoid_nn` in the code box below plots the classifier found after training for the specified number of iterations with the specified "learning rate" (step size). The classifier predicts label +1 if the output of the sigmoid, a, is greater than 0.5 and label 0 otherwise -- remember that the range of the sigmoid function is $(0, 1)$.

1.1.A) What is the shape of the separator produced by this network? Explain.

1.1.B) Are you able to get relatively consistent 100% accuracy with some combination of `iters` and learning rate `lr`?

```
def run():
    return sigmoid_nn(iters=100, lr=0.05)

```

Run Code Save Submit View Answer Ask for Help

As staff, you are always allowed to submit. If you were a student, you would see the following:
You have infinitely many submissions remaining.

1.2) Not-XOR

The XOR dataset is a classic example showing the need for more than one layer of non-linear units. Here, we will consider a function outputting the **negation** of XOR, **Not-XOR**:

```
X = np.array([[0, 1, 0, 1],
              [0, 1, 1, 0]])
Y = np.array([[1, 1, 0, 0]])
```

1.2.A) Draw a plot showing the locations of these data points in x_1, x_2 coordinate space, with the corresponding labels. Now consider a network with no hidden layers as in part 1.1 above, which just has input units connected (via weights) to an output sigmoid unit. Can this network learn a separator for the given dataset? Explain.

1.2.B) Consider the following truth table representing the logical AND and OR operators. We would like to represent the AND function as a neural network, using the same structure as from Problem 1.1. Find weights w_1, w_2, and w_0 that represent the (i) AND operation and (ii) OR operation.

x_1	x_2	AND	OR
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

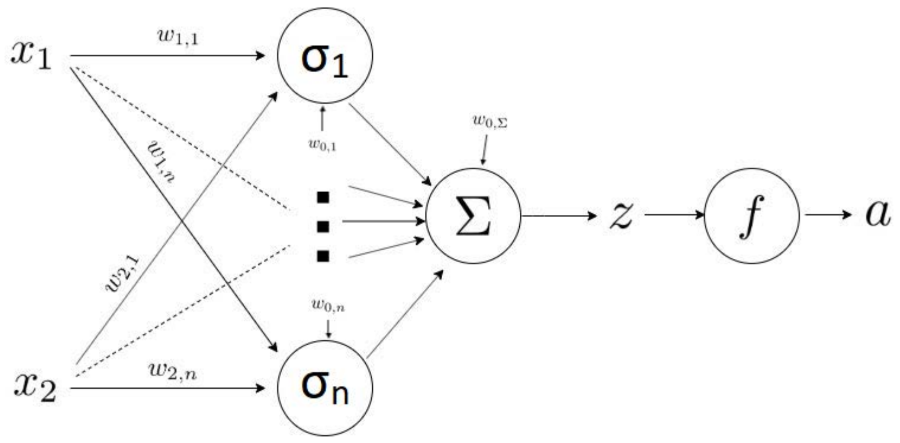
1.2.C) We can express Not-XOR using only AND, OR, and NOT operations. Which expression corresponds to x_1 Not-XOR x_2?

- ○ (x_1 AND x_2) AND (NOT x_1 AND NOT x_2)
- ○ (x_1 OR x_2) AND (NOT x_1 OR NOT x_2)
- ○ (x_1 AND x_2) OR (NOT x_1 AND NOT x_2)
- ○ (x_1 OR x_2) OR (NOT x_1 OR NOT x_2)

Save Submit View Answer Ask for Help

As staff, you are always allowed to submit. If you were a student, you would see the following:
You have infinitely many submissions remaining.

1.2.D) Consider the following network with the hidden layer of Sigmoid (σ) units, where f is a sigmoid unit and $w_{0,1} \dots w_{0,n}, w_{0,\Sigma}$ are the offsets.



We can use this network to represent the Not-XOR function. What is the minimal number of units we need in the hidden layer?

Enter the minimal number of units we need in the hidden layer:

Save Submit Clear Answer Ask for Help

As staff, you are always allowed to submit. If you were a student, you would see the following:
You have infinitely many submissions remaining.

Solution: 2

Explanation:

Discuss in checkoff

1.2.E) Now that we have our architecture, find the weights and offsets corresponding to the units.

Explain how the network works with reference to your earlier drawing of the Not-XOR data in x_1 and x_2 space. (Try to spend just 10 minutes on this, then ask for tips!)

1.2.F) In the run box below, first try to see if you can get a network of the size you found above to separate the data reliably (several times in a row, keeping `randomInit = True`). If not, try larger networks. Try each value of hidden units a few different times. Experiment with different learning rates and iterations as well. Explain what's going on. Keep in mind that the network is being trained using batch gradient descent.

Now, change `randomInit = False`. Try different values of hidden units. Try these values a few different times. Comment on how the behavior is different.

The `single_layer_nn` function in the code below runs on the dataset using a network with a single hidden layer of ReLU units, a sigmoid output layer, and NLL loss. You can specify the number of hidden units, the number of iterations, and the learning rate.

```
def run():
    return single_layer_nn(hidden_units=1, iters=1000, lr=0.05, randomInit=True)

```

Run Code Save Submit View Answer Ask for Help

As staff, you are always allowed to submit. If you were a student, you would see the following:
You have infinitely many submissions remaining.

1.3) Hard data set

In this next example we use a much harder data set that is **barely** separable (although not linearly). We'll try a two-layer network architecture.

1.3.A) Running the code below, can you get this architecture to separate the data set with at least 95% accuracy? How about 100% accuracy (Don't spend too long looking for 100%)? If you can't, explain why not. If you can, explain why. Make sure your accuracy is reliable by running the code several times.

1.3.B) Do you think it's a good idea to try to find a "perfect" separator for this data? Explain.

The network here has two hidden layers of ReLU units and a sigmoid output unit with NLL loss. In the `two_layer_nn` function call below, you can play around with the number of hidden units in each hidden layer, the number of iterations, and the learning rate.

```python
def run():
    return two_layer_nn(hidden_units=2, iters=100, lr=0.05)

```

Run Code Save Submit View Answer Ask for Help

As staff, you are always allowed to submit. If you were a student, you would see the following:
You have infinitely many submissions remaining.

2) Checkoff

Checkoff 1:
Have a check-off conversation with a staff member, to explain your answers.

Ask for Help Ask for Checkoff

3) Activation and Loss Function Applications

One important part of designing a neural network application is understanding the problem domain and choosing

- A representation for the input
- The number of output units and what range of values they can take on
- The loss function to try to minimize, based on actual and desired outputs

We have studied input representation (featurization) in a previous lab, so in this problem we will concentrate on the number of output units, activation function on the output units, and loss function. These should generally be chosen jointly.

Just as a reminder, among different loss functions and activation functions (see Sections 3 and 6 of the Chapter 8 notes), we have studied:

- Activation functions: linear, ReLU, sigmoid, softmax
- Loss functions: negative log likelihood (NLL a.k.a. cross-entropy), quadratic (mean squared)

For each of the following application domains, specify good choices for the number of units in the output layer, the activation function(s) on the output layer, and the loss function. When you choose to use multiple output units, be very clear on the details of how you are applying the activation and the loss. **Please write your answers down!**

3.A) Map the words on the front page of the New York Times to the predicted (numerical) change in the stock market average.

3.B) Map a satellite image centered on a particular location to a value that can be interpreted as the probability it will rain at that location sometime in the next 4 hours.

3.C) Map the words in an email message to which one of a user's fixed set of email folders it should be filed in.

3.D) Map the words of a document into a vector of outputs, where each index represents a topic, and has value 1 if the document addresses that topic and 0 otherwise. Each document may contain multiple topics, so in the training data, the output vectors may have multiple 1 values.

4) Social Utility

Over the last several labs, we've explored different forms of biases with problematic outcomes: historical bias, representation bias, aggregation bias. This week, we'll start discussing potential benefits and drawbacks of intentionally biasing your model, and how this can affect the model's utility.

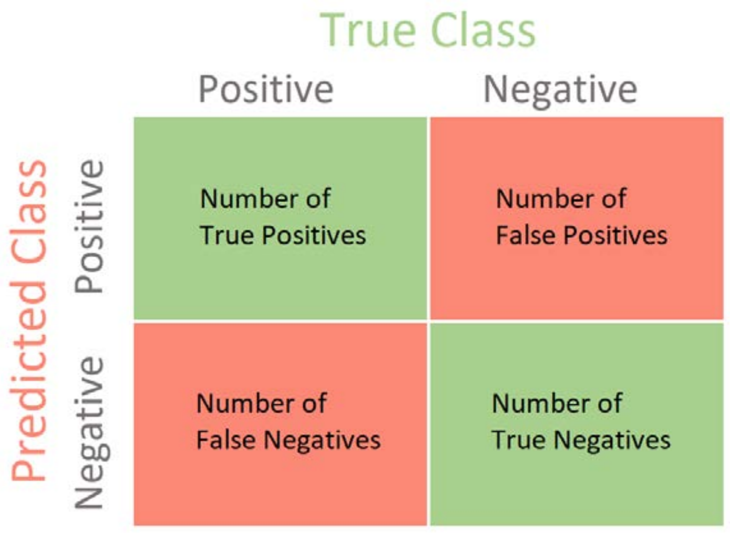
Dr. Niyu Ralnette's research group is investigating the use of machine learning to support radiologists.

2% of the population have a rare, deadly disease which can be detected with x-rays. The research group built a model that achieves 98% accuracy on the test set of 100,000 people and a 99% accuracy on the training set of 1 million individuals.

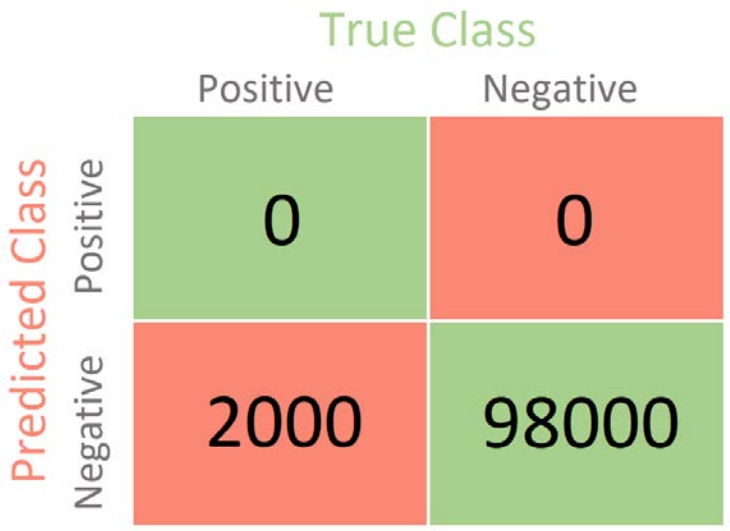
The Ralnette group starts to test the model in the wild, but when Dr. Ralnette takes a look at the results, she finds that the model never produces a positive diagnosis of the disease. Dr. Ralnette is confused by this since the model seemed to perform well on both the training data and test data.

4.A) Can you think of a possible explanation for why the model had high accuracies on the training data and test data, but seems to be unable to correctly diagnose the disease in the wild?

We now introduce another evaluation tool, used in addition to accuracy measurements, called **confusion matrices**. A confusion matrix is a table used to provide more detailed information on the performance of a classification model.



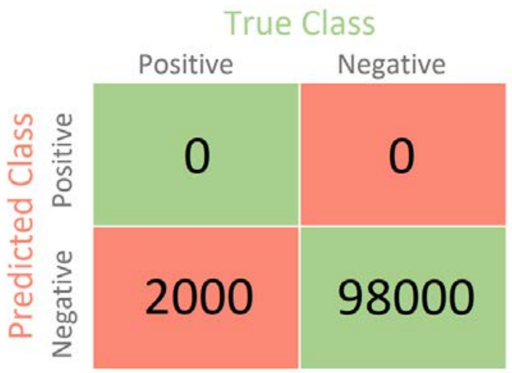
Our model that achieved a 98% accuracy on the test set, had the following confusion matrix.



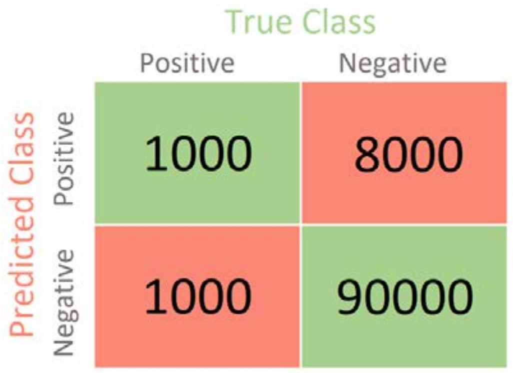
		True Class	
		Positive	Negative
Predicted Class	Positive	0	0
	Negative	2000	98000

It seems the model learned to always predict “False” for the diagnosis! Dr. Ralnette’s group trains a few more models to see if we get different performance

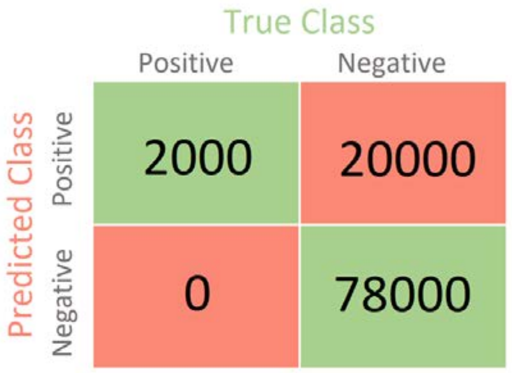
4.B) The confusion matrices of the developed models are below. Calculate the accuracy of each model.



		True Class	
		Positive	Negative
Predicted Class	Positive	0	0
	Negative	2000	98000



		True Class	
		Positive	Negative
Predicted Class	Positive	1000	8000
	Negative	1000	90000



		True Class	
		Positive	Negative
Predicted Class	Positive	2000	20000
	Negative	0	78000

4.C) Which of the following models would you prefer to deploy, and why? In the case of diagnosing this rare, deadly disease, do you prefer false negatives, false positives, or do you not have a preference for the type of failures? Why?

Note: In case you are curious, one way achieve models with different combinations of false positive and false negative rates is to use an asymmetric loss (as seen in the Chapter 1 notes).

4.D) This disease can be treated if detected, but the treatment has extremely harsh, non-lethal side effects. Does this knowledge change the model you would choose? Would you choose a different model if the model were going to diagnose the patient without a radiologist making the final recommendation?

5) Checkoff

Checkoff 2:
Have a check-off conversation with a staff member, to explain your answers.

Ask for Help Ask for Checkoff