Week 6 Lab

Week 6 Lab

(1.0 points possible)

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:

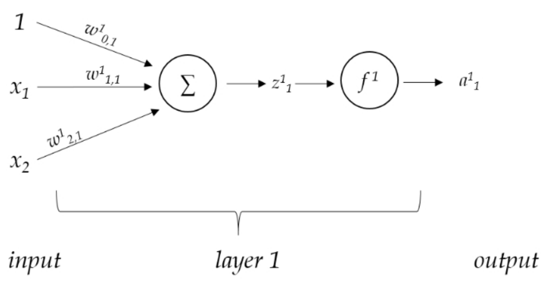
- The initialization of weights in a neural network is random, so subsequent runs might produce different results.
- `iters` indicates how many single steps of SGD to run, each using one training point.
- Pay attention to the decision boundaries, shown in the answers after you run the code. Also note 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 `t1` 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 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 t1(iters=5000, lr=0.02)

```

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

You have infinitely many submissions remaining.

Your score on your most recent submission was: 100.00%

Test Results:

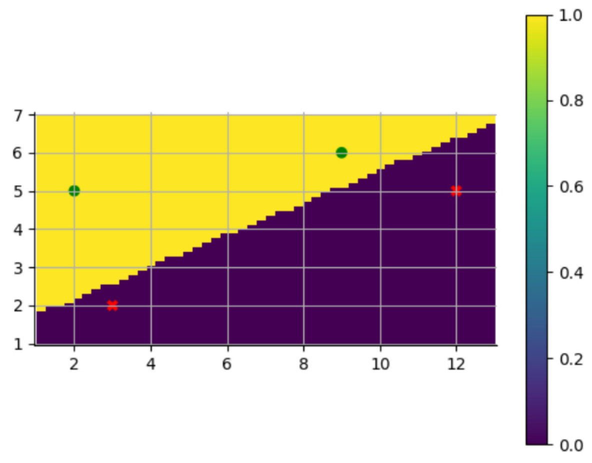
Test 01

The test case was:

```
#Your Code Here
ans=run()
```

Our solution produced the following value for `ans`:

Your submission produced the following value for `ans`:
Accuracy=1.000000



✓

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) Assume we add a hidden layer with ReLU activation units, connected to sigmoid unit in the output layer. Draw by hand the smallest network that can separate the data. Show all the weights (and biases) for the network units. Explain how it works (with reference to your earlier drawing of the Not-XOR data in x_1 and x_2 space), and explain why a smaller network won't work. (Try to spend just 10 minutes on this, then ask for tips!)

1.2.C) 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). If not, try larger networks. Explain what's going on.

The `t2` 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 (pretty please, don't pass in anything other than a positive integer!), the number of iterations, and the learning rate.

```
def run():
    return t2(hidden_units=2, iters=4999, lr=0.01)

```

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

You have infinitely many submissions remaining.

Your score on your most recent submission was: 100.00%

Test Results:

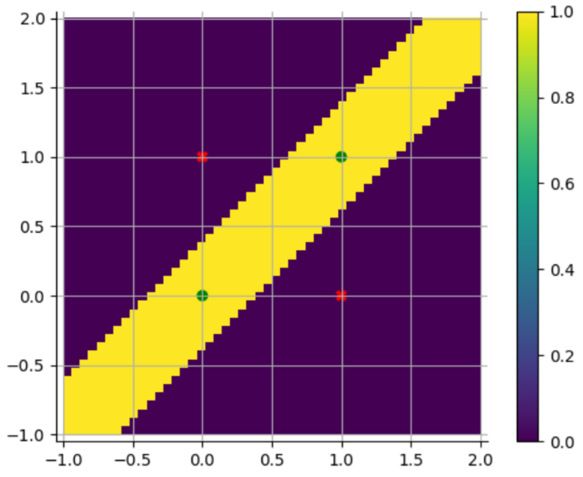
Test 01

The test case was:

```
#Your Code Here
ans=run()
```

Our solution produced the following value for `ans`:

Your submission produced the following value for `ans`:
Accuracy=1.000000



✓

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 well (at least 95% accuracy)? I f 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 `t3` function below, you can specify the number of hidden units in each hidden layer, the number of iterations, and the learning rate.

```
def run():
    return t3(hidden_units=2, iters=5000, lr=0.025)

```

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

You have infinitely many submissions remaining.

Your score on your most recent submission was: 100.00%

Your score on your most recent submission was: 100.00%

Test Results:

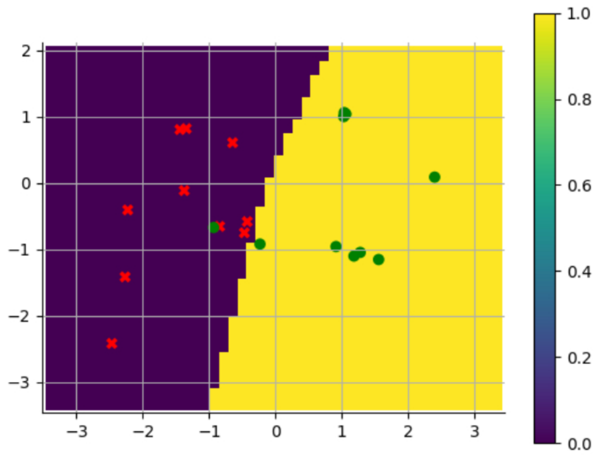
Test 01

The test case was:

```
#Your Code Here
ans=run()
```

Our solution produced the following value for `ans`:

Your submission produced the following value for `ans`:
Accuracy=0.950000



✓

2) Crime and Punishment

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, we have studied:

- Activation functions: linear, ReLU, sigmoid, softmax
- Loss functions: hinge, 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!**

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

2.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.

2.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.

2.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.

Previous Next