Plotting Titanic

Plotting Titanic Prediction

This notebook is a tutorial of how we can modify the predictor python script to show a visualization of the data in the hidden space and how the predictor classifies each point. Each dimension of the hidden space corresponds to a neuron from the predictor. Because only two neurons are used for the titanic prediction model, we can plot it in the hidden space. *Note: This isn't part of Daimensions. We are going to manually edit the python script to output a png of the model's results in the hidden space.*

1. Build the Predictor

```
In [5]: ! btc -f NN titanic_train.csv -o vis_titanic.py -target Survived -rank -e 5
```

```
Brainome Daimensions(tm) 0.97 Copyright (c) 2019, 2020 by Brainome, Inc. All Rights
     Reserved.

Ranking attributes...done.

Attribute Ranking:
Using only the important columns: Sex Fare Parch

Data:
Number of instances: 891
Number of attributes: 3
Number of classes: 2
Class balance: 61.62% 38.38%

Learnability:
Best guess accuracy: 61.62%
Capacity progression (# of decision points): [11, 11, 12, 14, 14, 14]
Decision Tree: 225 parameters
Estimated Memory Equivalent Capacity for Neural Networks: 41 parameters

Risk that model needs to overfit for 100% accuracy...
using Decision Tree: 50.73%
using Neural Networks: 89.13%

Expected Generalization...
```

```
using Decision Tree: 3.94 bits/bit
using a Neural Network: 21.73 bits/bit

Recommendations:
Note: Enough data to generalize. [green]

Training...done.
Classifier Type:                    Neural Network
System Type:                        Binary classifier
Best-guess accuracy:                61.61%
Model accuracy:                     78.78% (702/891 correct)
Improvement over best guess:        17.17% (of possible 38.39%)
Model capacity (MEC):               11 bits
Generalization ratio:               63.81 bits/bit
Model efficiency:                   1.56%/parameter
System behavior
True Negatives:                     52.53% (468/891)
True Positives:                     26.26% (234/891)
False Negatives:                    12.12% (108/891)
False Positives:                    9.09% (81/891)
True Pos. Rate/Sensitivity/Recall: 0.68
True Neg. Rate/Specificity:         0.85
Precision:                          0.74
F-1 Measure:                        0.71
False Negative Rate/Miss Rate:      0.32
Critical Success Index:             0.55
Overfitting:                        No
```

2. Make Changes to Python Script

The previous line of code should output a python script 'vis_titanic.py' with the model's predictor. Next, we'll manually edit the code by adding a few lines so that it will output the desired png when we run -validate. I'll show you the section of code we'll change with the modified lines highlighted.

```
# Helper (save an import)
def argmax(l):
    f = lambda i: l[i]
    return max(range(len(l)), key=f)
# Classifier
def single_classify(row, ax):
    #inits
    x = row
    o = [0] * num_output_logits


    #Neuron Equations
    h_0 = max((((-0.23418278 * float(x[0]))+ (0.2055268 * float(x[1]))+ (-3.683856
        * float(x[2]))) + -0.5669335), 0)
    h_1 = max((((3.1762295 * float(x[0]))+ (0.29285425 * float(x[1]))+
        (-0.0113988025 * float(x[2]))) + 1.4681039), 0)
    o[0] = (0.6990876 * h_0)+ (-0.89847755 * h_1) + 2.2160573

    #visualization modification
    ax.scatter(h_0, h_1, marker=('o' if o[0] >= 0 else '^'), c=('r' if o[0] >= 0 else 'b'))


    #Output Decision Rule
    if num_output_logits==1:
        return o[0]>=0
    else:
        return argmax(o)


def classify(arr):
    #init
    w_h = np.array([[-0.23418277502059937, 0.20552679896354675,
        -3.6838560104370117], [3.176229476928711, 0.2928542494773865,
        -0.011398802511394024]])
    b_h = np.array([-0.5669335126876831, 1.4681038856506348])
    w_o = np.array([[0.6990876197814941, -0.8984775543212891]])
    b_o = np.array(2.216057300567627)

    #Hidden Layer
    h = np.dot(arr, w_h.T) + b_h

    relu = np.maximum(h, np.zeros_like(h))


    #Output
    out = np.dot(relu, w_o.T) + b_o
    if num_output_logits == 1:
        return (out >= 0).astype('int').reshape(-1)
    else:
        return (np.argmax(out, axis=1)).reshape(-1)



def Predict(arr,headerless,csvfile, get_key, classmapping):

    with open(csvfile, 'r') as csvinput:
        #readers and writers
        writer = csv.writer(sys.stdout, lineterminator=os.linesep)
        reader = csv.reader(csvinput)
        #print original header
        if (not headerless):
            writer.writerow(','.join(next(reader, None) + ["Prediction"]))


        #visualization modification
        import matplotlib.pyplot as plt
        fig = plt.figure()
        ax = fig.add_subplot(111)

        for i, row in enumerate(reader):
            #use the transformed array as input to predictor
            pred = str(get_key(int(single_classify(arr[i],ax)), classmapping))
            #use original untransformed line to write out
            row.append(pred)
            writer.writerow(row)

        #visualization modification
        ax.set_xlabel('h0')
        ax.set_ylabel('h1')
        plt.savefig('titanic_visual.png')
```

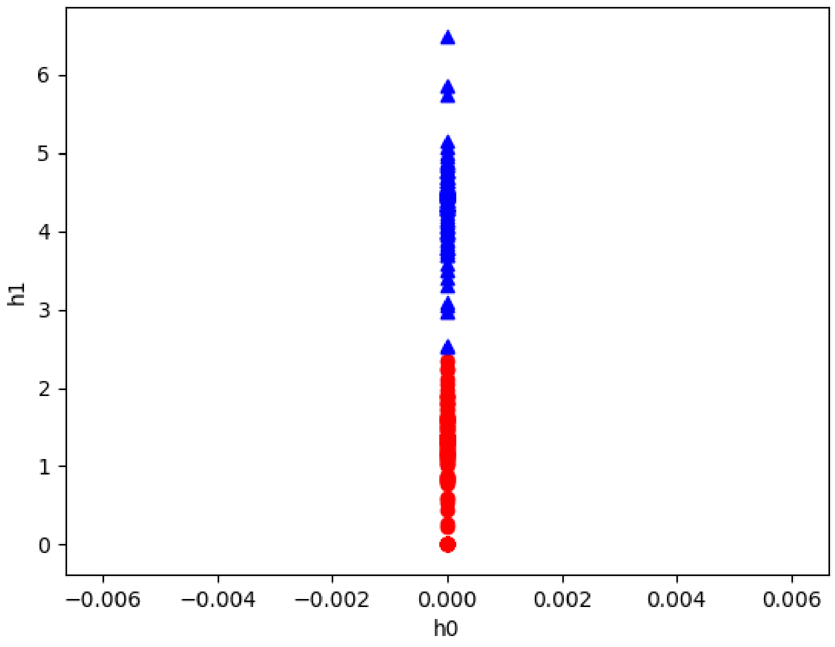
3. Run -validate

You can make this modification in your own python script most easily by replacing lines 359-422 with the code above. Below I'm using an already modified version of the script called 'vis_titanic_mod.py', but you would use your edited 'vis_titanic.py'. We simply have to run the following line of code and it will output our desired png.

```
In [18]: ! python3 vis_titanic_mod.py titanic_test.csv
```

```
Sex,Parch,Fare,Prediction
male,0,7.8292,0
female,0,7,1
male,0,9.6875,0
male,0,8.6625,0
female,1,12.2875,1
male,0,9.225,0
female,0,7.6292,1
male,1,29,0
female,0,7.2292,1
male,0,24.15,0
male,0,7.8958,0
male,0,26,0
...
male,0,8.05,0
male,1,22.3583,0
```

Your current directory should now have a png called 'titanic_visual.png' in it that looks like the following:



Here we see the 2D hidden space that corresponds to the two neurons the model uses. We can see a clear plane separating the two classifications. This is the plane that the predictor uses to distinguish between the two classes.