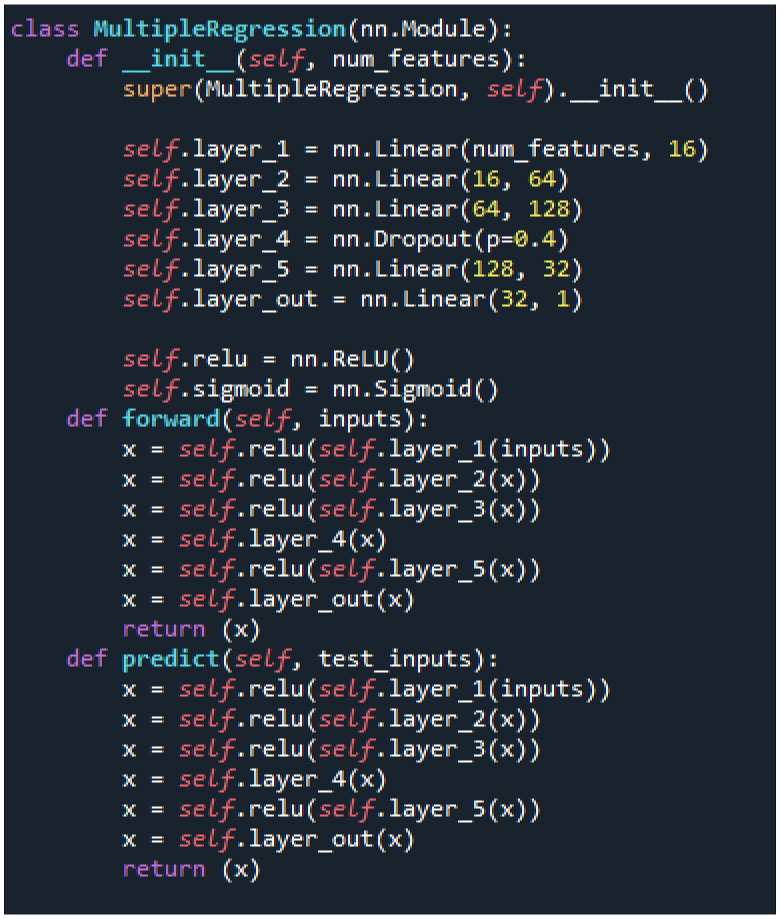
Ralph Gleaton

CSCE 587 – 001

Lab 01 Report

resgression_pytorch_sample.py Default Network					
Learning Rate	Training Epochs	Batch Size	Optimizer	MSE	R^2
0.001	150	64	Adam	0.407	0.37
0.01	150	64	Adam	0.43	0.33
0.001	200	100	Adam	0.409	0.367
0.0005	250	25	Adam	0.409	0.366
0.001	250	50	SGD	0.432	0.33
0.005	400	64	SGD (momentum = 0.4)	0.397	0.385
0.008	400	64	SGD (momentum = 0.5)	0.393	0.39

regression_pytorch_sample.py Custom Network (Pictured below)						
Learning Rate	Training Epochs	Batch Size	Optimizer	Momentum	MSE	R^2
0.001	250	64	Adam	N/A	0.397	0.384
0.001	500	64	SGD	0.5	0.418	0.352
0.01	500	64	SGD	0.2	0.387	0.4



```python
class MultipleRegression(nn.Module):
    def __init__(self, num_features):
        super(MultipleRegression, self).__init__()

        self.layer_1 = nn.Linear(num_features, 16)
        self.layer_2 = nn.Linear(16, 64)
        self.layer_3 = nn.Linear(64, 128)
        self.layer_4 = nn.Dropout(p=0.4)
        self.layer_5 = nn.Linear(128, 32)
        self.layer_out = nn.Linear(32, 1)

        self.relu = nn.ReLU()
        self.sigmoid = nn.Sigmoid()
    def forward(self, inputs):
        x = self.relu(self.layer_1(inputs))
        x = self.relu(self.layer_2(x))
        x = self.relu(self.layer_3(x))
        x = self.layer_4(x)
        x = self.relu(self.layer_5(x))
        x = self.layer_out(x)
        return (x)
    def predict(self, test_inputs):
        x = self.relu(self.layer_1(inputs))
        x = self.relu(self.layer_2(x))
        x = self.relu(self.layer_3(x))
        x = self.layer_4(x)
        x = self.relu(self.layer_5(x))
        x = self.layer_out(x)
        return (x)
```

mnist.py					
Learning Rate	Training Epochs	Batch Size	Optimizer	Avg Loss	Accuracy (Out of 10,000)
1.0	3	640	Adadelta	0.0441	9,858
0.5	4	640	Adadelta	0.0487	9,830
0.1	4	1280	Adaelta	0.1999	9,428
1.0	3	1280	Adam	2.3028	1028