Problem (b)

```
function net = approximator(f, T)

% Adjust T
K = ceil(1/T);
T = 1/K;

% First layer
k = repmat(0:K, 3, 1);
v1 = repmat([-1 0 1]', K+1, 1);
net(1).gain = 1/T * ones(3*(K+1), 1);
net(1).bias = -k(:) - v1;
net(1).h = @(a) max(0, a);

% Second layer
a = repmat(arrayfun(f, 0:T:1), 3, 1);
v2 = repmat([1 -2 1], 1, K+1);
net(2).gain = a(:)' .* v2;
net(2).bias = zeros(1, 1);
net(2).h = @(a) a;
```

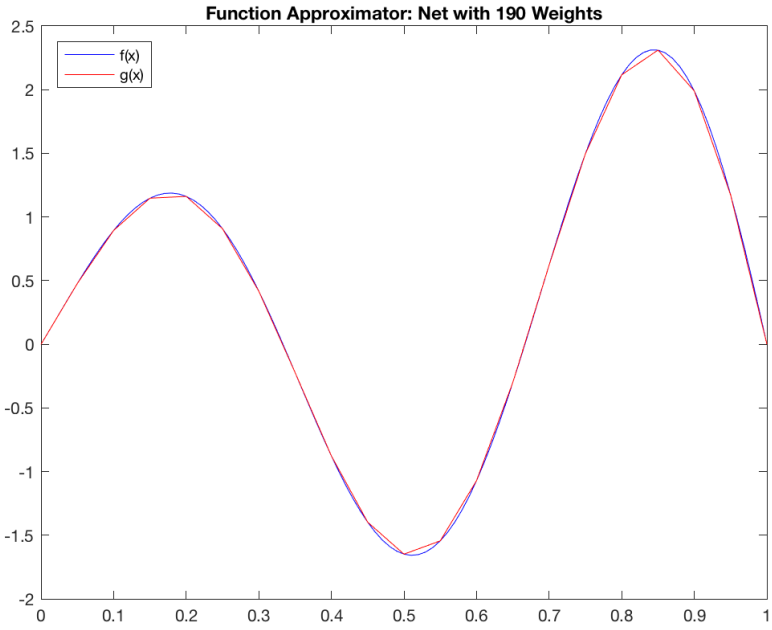

Figure 2: Two-layer function approximator

Designing and Training a Small Neural Network

Problem (a)

```
function [layers, options] = good(checkpointPath)

layers = [imageInputLayer([1 2])
    fullyConnectedLayer(4)
    reluLayer
    fullyConnectedLayer(4)
    reluLayer
    fullyConnectedLayer(2)
    softmaxLayer
    classificationLayer()];

% Learning rate schedule, based on the following relation:
% finalRate = initialRate * factor ^ (epochs/period)
miniBatchSize = 175;
initialRate = 0.25;
finalRate = 0.001;
epochs = 1000;
dropFactor = 0.1;
if initialRate <= finalRate
    dropPeriod = epochs;
else
    dropPeriod = round(epochs * log(dropFactor) / log(finalRate/initialRate));
end

options = trainingOptions('sgdm', ...
    'MaxEpochs', epochs, ...
    'MiniBatchSize', miniBatchSize, ...
    'InitialLearnRate', initialRate, ...
    'LearnRateSchedule', 'piecewise', ...
    'LearnRateDropFactor', dropFactor, ...
    'LearnRateDropPeriod', dropPeriod, ...
    'CheckpointPath', checkpointPath);
```

Our network structure is:

- Input layer
- Fully connected layer (4 neurons)
- ReLU Layer
- Fully connected layer (4 neurons)
- ReLU Layer
- Fully connected layer (2 neurons)
- Softmax layer
- Classification Layer

We added an extra fully connected layer to increase the depth and thus allow a better division of the feature space. In addition, the number of neurons for the first two fully connected layers were increased to 4, to give the net more tunable parameters. However, we did not choose a higher number to avoid having more parameters than training samples. Also, we tried adding a dropout layer, but it did not perform as well. This is likely due to the net being relatively shallow and thus the layer is not appropriate.

Our training parameters and rationale are summarized below.

- `MaxEpochs` = 1000: Increased to allow for longer convergence.
- `MiniBatchSize` = 175: Increased to a majority as to best represent the training data without getting stuck in local minima.
- `InitialLearnRate` = 0.25: Increased so as to initially converge faster, quickly reaching a decent solution.
- `FinalLearnRate` = 0.001: Decreased so that a decent solution can carefully be refined in later epochs.
- `LearnRateSchedule` = piecewise: Remained the same, so that the learning rate can change.
- `LearnRateDropFactor` = 0.1: Decreased so that the initial learn rate has time to reach a decent solution.

Problem (b)

The net's accuracy is 94.56%.

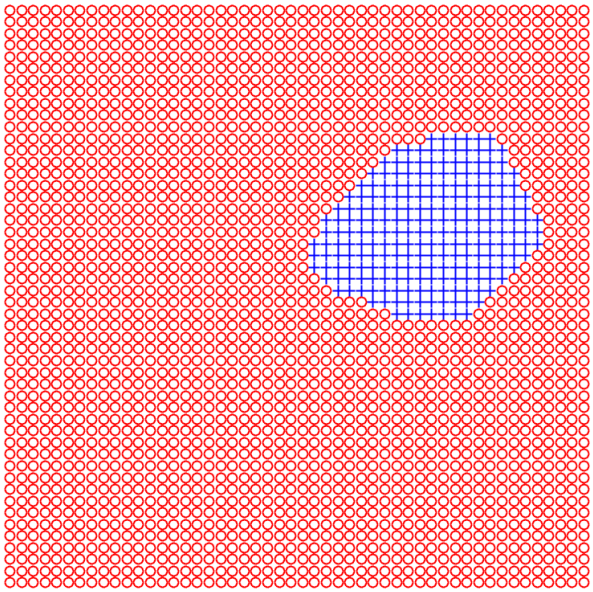

Figure 3: Labeled test data

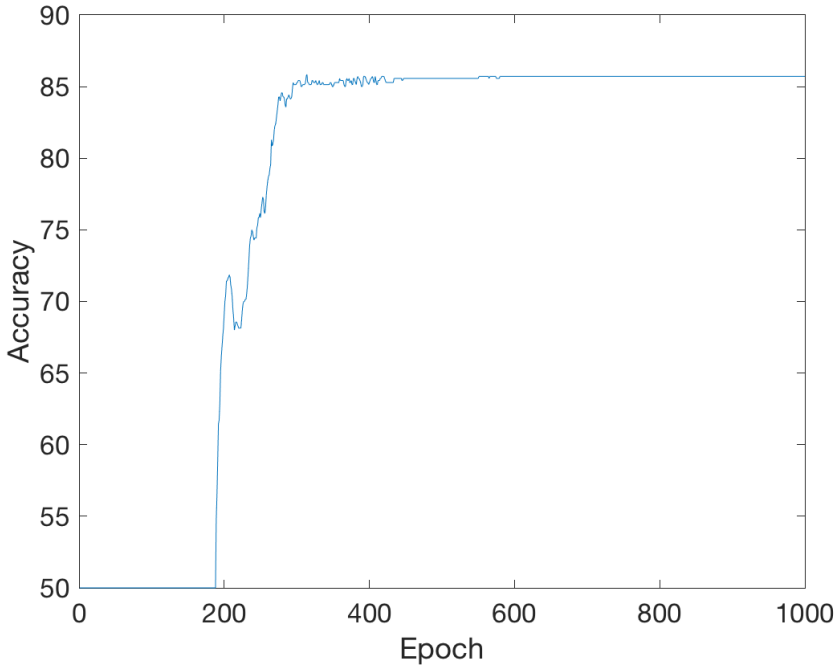

Figure 4: Validation accuracy vs. epoch number