```
clear all;
close all;
```

Perceptron Algorithm Sample Complexity

```
% Setup & initialise variables:
n = 100; % largest number of dimensions.
m = 500; % number of test data point used for each error estimation.
loop = 100; % number of runs per dimension.
GenErrLim = 0.1; % point where 10% error reached as per handout
 definition
NumSamples = zeros(n, loop); % initialise Sample Complexity variable

for i = 1:n   % change size of column of data set

    for l = 1:loop

        % generate test and training data  from a uniformly randomly
        % generated data set ([-1,1]^n)

        trainX = randi([0 1], m, i); %uniform random set of [0,1]
        testX = randi([0 1], m, i);

        %convert data into [-1 ,1]
        trainX = trainX.*2 - ones(size(trainX));
        testX = testX.*2 - ones(size(testX));


        trainY = trainX(:, 1); %select first column
        testY = testX(:, 1);

        % initialise error and counter values:
        Counter = 0;  % sample complexity counter
        GenErr = 1000; % start by setting at an artificially high
 number

        while(GenErr > GenErrLim)
           % Keep track of the number of training samples used in the
 loop:
           Counter = Counter + 1;

           % Train perceptron on data (calculate weight alpha)
           alpha = perceptrontrain(trainX(1:Counter,:),...
               trainY(1:Counter,:),1);

           % Calculate predictions on test set
           yP = sign(alpha'*(trainX(1:Counter,:)*testX'));
           yP = yP';

           % count miss-classification instances:
           loss = sum(yP ~= testY);
```

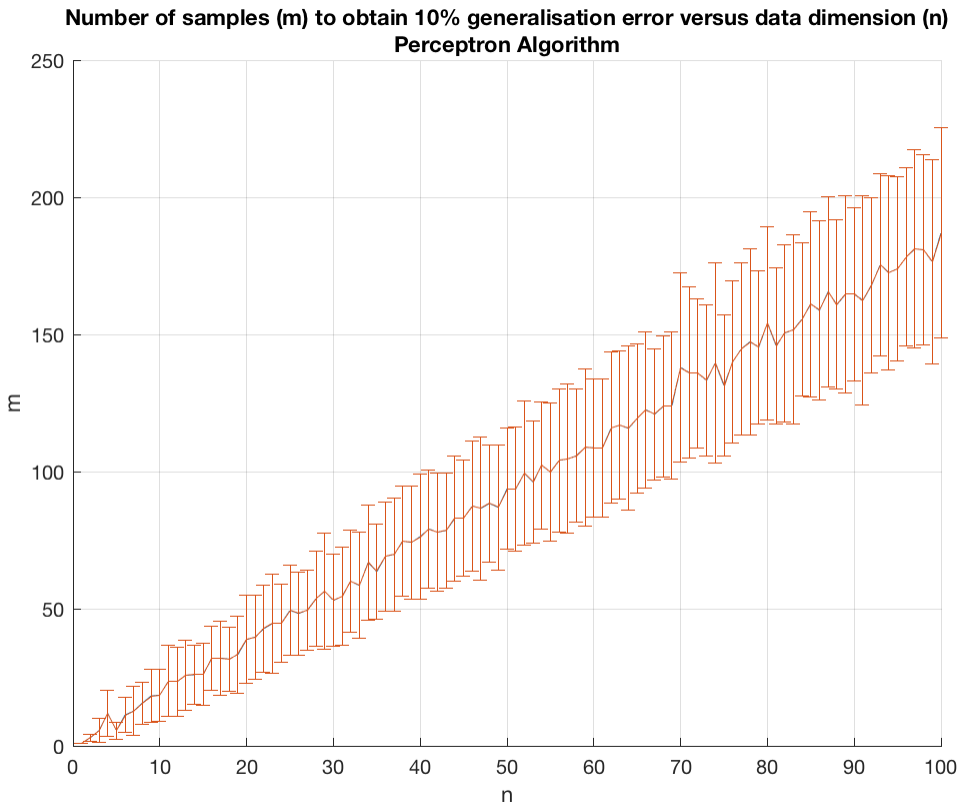
```
            % calculate the generalisation error estimate as a
 proportion
            % of the amount of test data instances (m):
            GenErr = (loss)/m;

          end

        % sample complexity as a function of the n dimension of data:
        NumSamples(i,l) = Counter;
    end
end


% calculate means and standard deviations of number of samples needed
meanNumSamples = mean(NumSamples,2);
sdNumSamples = std(NumSamples,[],2);

figure;
hold on;
plot(meanNumSamples)
title({'Number of samples (m) to obtain 10% generalisation error
 versus data dimension (n)','Perceptron Algorithm'});
ylabel('m');
xlabel('n');
errorbar(1:n, meanNumSamples, sdNumSamples);
hold off;
grid on;
```



Published with MATLAB® R2016b