1 Problem 1

1.1 Part a
The accuracy of the nearest neighbor classifier was: 0.9, and the classifier took 2.55 seconds to run.

1.2 Part b
The results using random projections are shown below:

<table>
<thead>
<tr>
<th>Dim</th>
<th>Accuracy</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.397</td>
<td>0.363</td>
</tr>
<tr>
<td>100</td>
<td>0.857</td>
<td>0.739</td>
</tr>
<tr>
<td>200</td>
<td>0.875</td>
<td>1.143</td>
</tr>
</tbody>
</table>

1.3 Part c
The results using rescaling are shown below: The results using random projections are shown

<table>
<thead>
<tr>
<th>Dim</th>
<th>Accuracy</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No projection</td>
<td>0.913</td>
<td>2.868</td>
</tr>
</tbody>
</table>

We see that the results are slightly better than the results using unscaled observations.

2 Problem 2
The VC-dimension of axis-parallel rectangles in the space $R^n$ is $2n$. To justify this answer we will first show that the VC-dimension must be at least $2n$, and then show that it cannot be more than $2n$. 

1
To prove that the VC-dimension is at least $2n$, we need to show that for any $n$ there exists some set of $2n$ points in $R^n$ that is shattered by our axis-parallel rectangles concept class. In particular, we will show that a specific construction of points is always shatter-able. Our construction of points is as follows: for the $n$-dimensional case place two points on each axis, at coordinates $-1$ and $1$ for a total of $2n$ points.

To show that this construction is always shatterable by axis-parallel rectangles, consider the construction in an arbitrary number of dimensions $n$ and consider an arbitrary subset of these $2n$ points: $S$ (where, $0 \leq |S| \leq 2n$). We can construct an axis-parallel rectangle that exactly captures the points in $S$ (and no other points) by starting with an axis-parallel rectangle that extends from $-0.5$ to $0.5$ along each axis and extending to $(+/−1)$ along each axis if and only if the corresponding point at $(+/−1)$ is in $S$. Since any possible subset of our construction of $2n$ points in $n$ dimensions can be exactly captured by our axis-parallel rectangle concept class, we see that the construction is shatterable and therefore the VC dimension of axis-parallel rectangles in $n$ dimensions must be at least $2n$.

Now we need to justify why $2n + 1$ or more points are not shatterable. Assume we have some set $S$ of $>2n+1$ points in $R^n$. For each axis $a$, we can chose 2 points $p_{a-min}$ and $p_{a-max}$ that define the extremes for that axis, meaning that $p_{a-min}$ is the point with the minimum coordinate along axis $a$ and $p_{a-max}$ is the point with the maximum coordinate along axis $a$. Consider $S'$, the union of these $p_{a-min}$ and $p_{a-max}$ for all $n$ axes, which is a set of $\leq 2n$ points that define the extremes of $S$. In order capture all of the points in $S'$ with an axis-parallel rectangle, the rectangle must extend at least to the extremes of $S$ in each dimension, therefore any axis-parallel rectangle that captures the points in $S'$ must, by construction, also capture all other points in $S$. It follows that it is not possible to define an axis parallel rectangle that captures all the points in $S'$ but no other points in $S$, so clearly it is not possible to capture every possible subset of $S$ with the axis-parallel rectangle concept class. Thus we can conclude that the VC-dimension of of axis-parallel rectangles cannot be $>2n$. 


### 3 Code

#### 3.1 Code

```matlab
function [X, Y] = setupData( varargin )

% Preprocess a dataset into and appropriate form for % our SVM training/prediction code.
%
% varargin: A set of (N' x d) matrices where each % matrix contains all (N') observations for % a given class.
%
% X: Combined matrix of all observations (d x N)
% Y: Vector of labels for each observation (1 x N)

% Concatenate the 'X' matrices into one training set
X = double(vertcat(varargin{:}));

% Add the extra dimension
X = [X; ones(1, size(X, 2))];

% Create a vector with the label for each observation
Y = zeros(1, size(X,2));
start = 1;
for label = 1:nargin
    numlabel = size(varargin{label}, 1);
    Y(start:(start + numlabel - 1)) = label;
    start = start + numlabel;
end
end
```
function [ Xtrain, Xtest ] = randproject( Xtrain, Xtest, m )

% Reduce the dimensionality of training and test data
% using random projections
%
% Xtrain: Training observations (d/m x N)
% Xtest: Test observations (d/m x N)
% m: New dimensionality

    d = size(Xtrain, 1);
    A = 1 / sqrt(m) * randn(m, d);
    Xtrain = A * Xtrain;
    Xtest = A * Xtest;

end

function [ Xtrain, Xtest ] = rescale( Xtrain, Xtest )

% Rescales every observation in the training and test
% to have unit length
%
% Xtrain: Training data (d x N)
% Xtest: Test data (d x X)

    Xtrain = bsxfun(@rdivide, Xtrain, sqrt(sum(Xtrain .^ 2, 1)));
    Xtest = bsxfun(@rdivide, Xtest, sqrt(sum(Xtest .^ 2, 1)));

end
function [ Ytest ] = NN( Xtrain, Ytrain, Xtest )
% Run the nearest neighbor algorithm to classify
% test data
% Xtrain: Training observations (d x N)
% Ytrain: Training labels (N)
% Xtest: Test observations (d x N)
% Ytest: Predicted labels (N)
    dist = -2 * (Xtrain' * Xtest);
    dist = bsxfun(@plus, sum(Xtest .^ 2, 1), dist);
    dist = bsxfun(@plus, sum(Xtrain .^ 2, 1)', dist);
    [~, nearest] = min(dist, [], 1);
    Ytest = Ytrain(nearest);
end

function [ acc ] = score( Ytrue, Ypred )
% Computes the accuracy of a set of predictions
% Ytrue: True labels of data (1 x N)
% Ypred: Predicted labels of data (1 x N)
% acc: Accuracy
    acc = sum(double(Ytrue == Ypred)) / numel(Ytrue);
end
% hw7.mat
% Runs all of the experiments for homework 5

% Load and combine the training and test data matrices
load digits.mat
[Xtrain, Ytrain] = setupData(train0, train1, train2, train3, train4, train5, train6, train7, train8, train9);
[Xtest, Ytest] = setupData(test0, test1, test2, test3, test4, test5, test6, test7, test8, test9);

% Part a
tic;
Ypred = NN(Xtrain, Ytrain, Xtest);
time = toc;
acc = score(Ytest, Ypred);
fprintf('Standard NN, accuracy: %f, time: %f\n', acc, time);

% Part b
% Test and evaluate the NN algorithm for a range of dimensions
fprintf('NN with random projections:\n');
for m = [10, 100, 200]
tic;
[XtrainP, XtestP] = randproject(Xtrain, Xtest, m);
Ypred = NN(XtrainP, Ytrain, XtestP);
time = toc;
acc = score(Ytest, Ypred);
fprintf('\t%d dimensions, accuracy: %f, time: %f\n', m, acc, time);
end

% Part c
fprintf('Experiments with rescaling\n');
[Xtrain, Xtest] = rescale(Xtrain, Xtest);

tic;
Ypred = NN(Xtrain, Ytrain, Xtest);
time = toc;
acc = score(Ytest, Ypred);
fprintf('Standard NN, accuracy: %f, time: %f\n', acc, time);

% Test and evaluate the NN algorithm for a range of dimensions
fprintf('NN with random projections:
');
for m = [10, 100, 200]
    tic;
    [XtrainP, XtestP] = randproject(Xtrain, Xtest, m);
    Ypred = NN(XtrainP, Ytrain, XtestP);
    time = toc;
    acc = score(Ytest, Ypred);
    fprintf('%5d dimensions, accuracy: %f, time: %f
        ', m, acc, time);
end