Table of Contents

Preamble 1
Crop Individual Letters 1
Implement the Approach on the Training Set 4
Test Our Approach on the Test Set 8

Preamble

Andrew Walker AMATH 482 A Final Assignment - 3/19/14

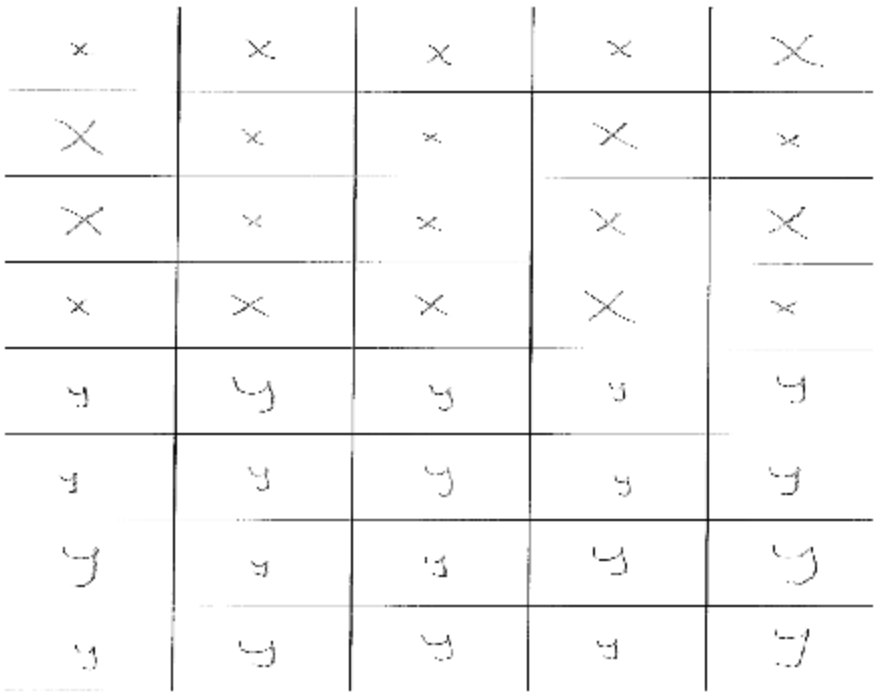
```
close all;
clear all;
```

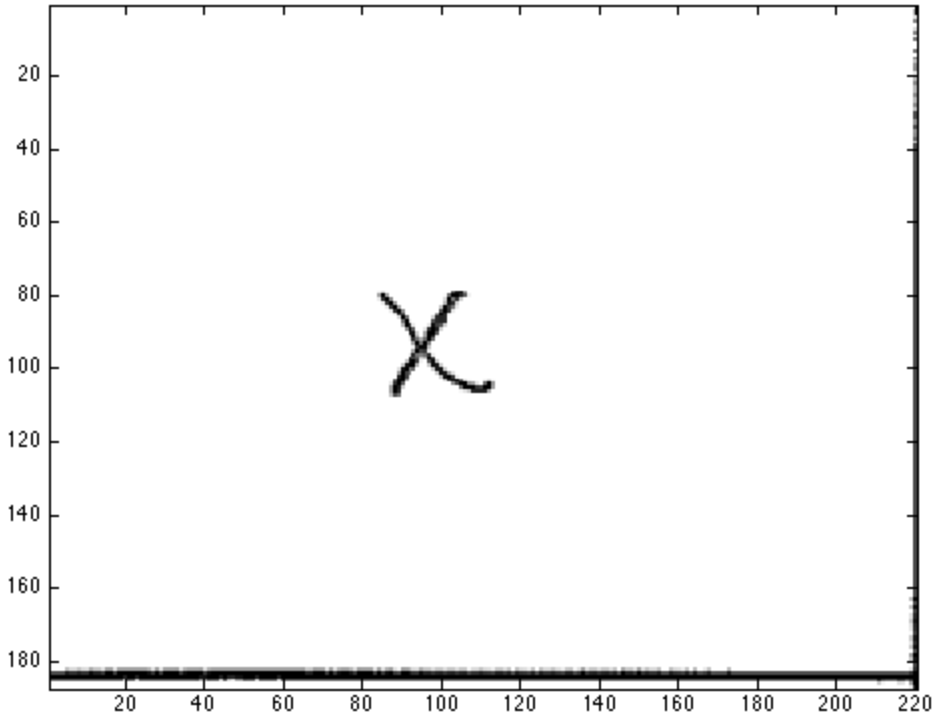
Crop Individual Letters

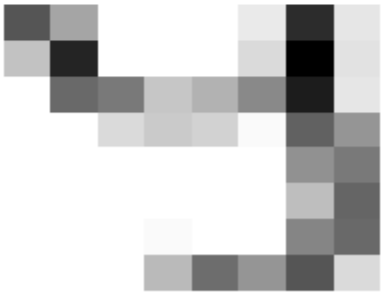
Note that the image commands are, for the most part, commented out. I will plot only a few from each section for demonstration purposes and as to not overload the pdf with images. However, I have plotted them all and they all look reasonable.

```
xytrain = imread('x_y_train.png'); % Load the training set
xytrain = xytrain(:,:,1);
xytrain = xytrain(73:end-82,75:end-101); % Crop to the main array
colormap gray;
image(xytrain,'CDataMapping','scaled'); % Check that it looks correct
axis off;

letter1 = xytrain(1:187,1:220);
figure;
colormap gray;
image(letter1,'CDataMapping','scaled'); % Check our first letter

% Do this for each letter in the array, and check that it looks okay,
% at this time we also crop a bit more to remove the black boarders
letter = cell(8,5);

for j=1:8
    for k=1:5
        letter{j,k} = xytrain(1+(187*(j-1)):187*j,1+(220*(k-1)):220*k);
        letter{j,k} = letter{j,k}(40:end-40,40:end-40);
        %figure; colormap gray;
        %image(letter{j,k},'CDataMapping','scaled');
    end
end

figure; colormap gray; subplot(2,2,1), image(letter{1,1},'CDataMapping','scaled');
axis off; subplot(2,2,2), image(letter{5,1},'CDataMapping','scaled'); axis off;

% Now we want to remove whitespace to have the letter filling a rectangular
```

```
% array. To do this we will find the smallest rectangle that will contain
% the letter. For convenience we want white to correspond to 0, currently
% it is 255, so we will color invert the image for the sake of simplicity.
% We change the coloring back to regular before resizing the image.
for j=1:8
    for k=1:5
        letter{j,k} = 255 - letter{j,k};

        rowSum = sum(letter{j,k},1);
        colSum = sum(letter{j,k},2);
        nonWhiteHor = find(rowSum);
        nonWhiteVert = find(colSum);
        letter{j,k} = letter{j,k}(nonWhiteVert(1):nonWhiteVert(end), ...
                                  nonWhiteHor(1):nonWhiteHor(end));
        letter{j,k} = 255 - letter{j,k};

        % Now we can use the imresize function
        letter{j,k} = imresize(letter{j,k}, [8 8]);

        %figure; colormap gray;
        %image(letter{j,k},'CDataMapping','scaled');
    end
end

subplot(2,2,3), image(letter{1,1},'CDataMapping','scaled'); axis off;
subplot(2,2,4), image(letter{5,1},'CDataMapping','scaled'); axis off;
```




20
40
60
80
100
120
140
160
180
20
40
60
80
100
120
140
160
180
200
220



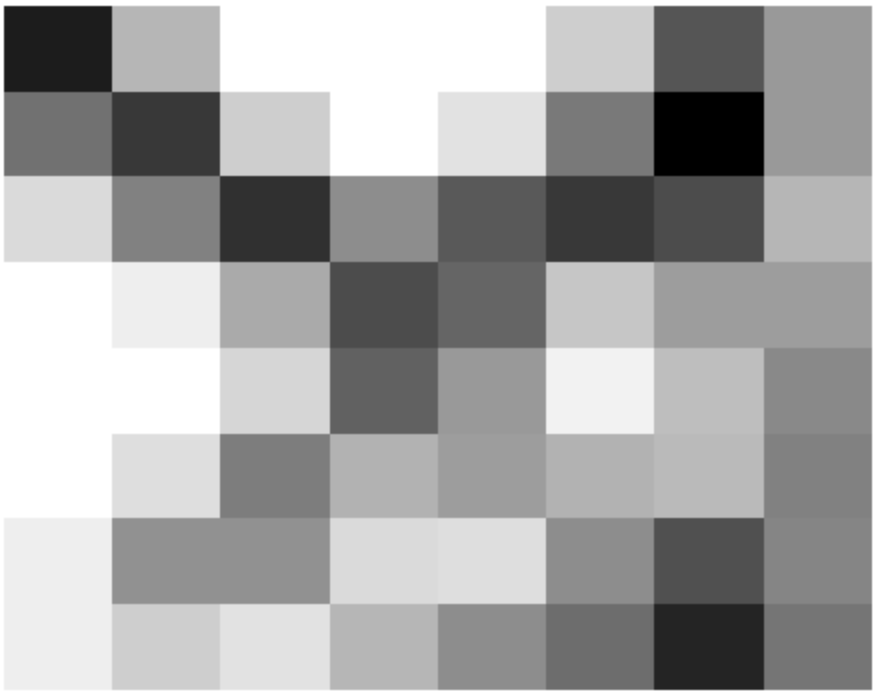
Implement the Approach on the Training Set

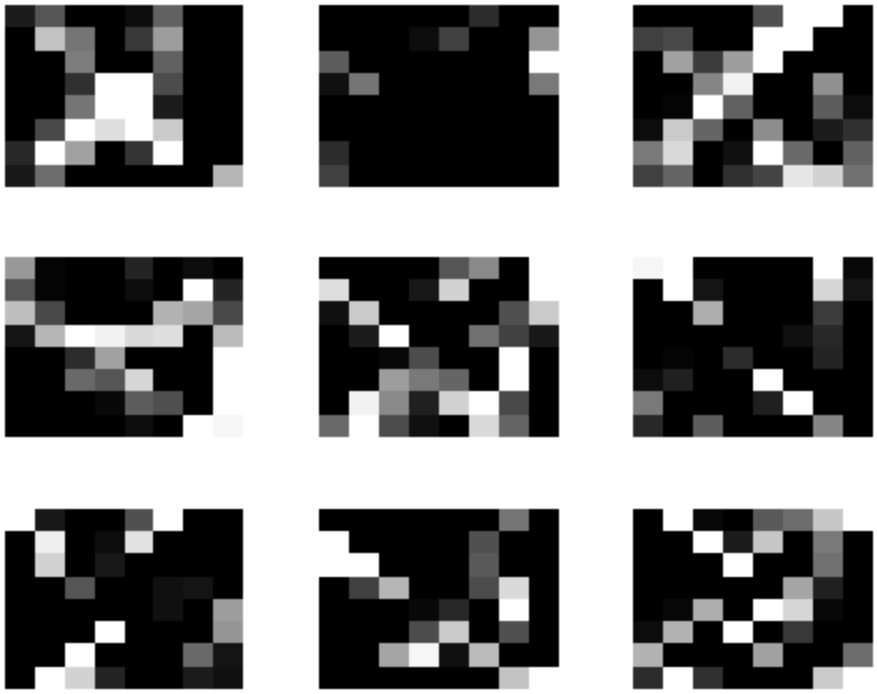
This approach is inspired by a lecture given by Professor Brunton in AMATH 301. It is an adaptation of the 'eigenface' approach. We will use PCA on a set of training images to come up with a basis of 'average letters' on which we can project images. In this projected space x's and y's cluster separately. In fact we will find that we only need a single basis 'vector' to differentiate the letters, this suggests that a simpler approach may have also worked, but in the initial conception of the solution I assumed that it would be difficult to tell letters apart and hence chose a more robust method.

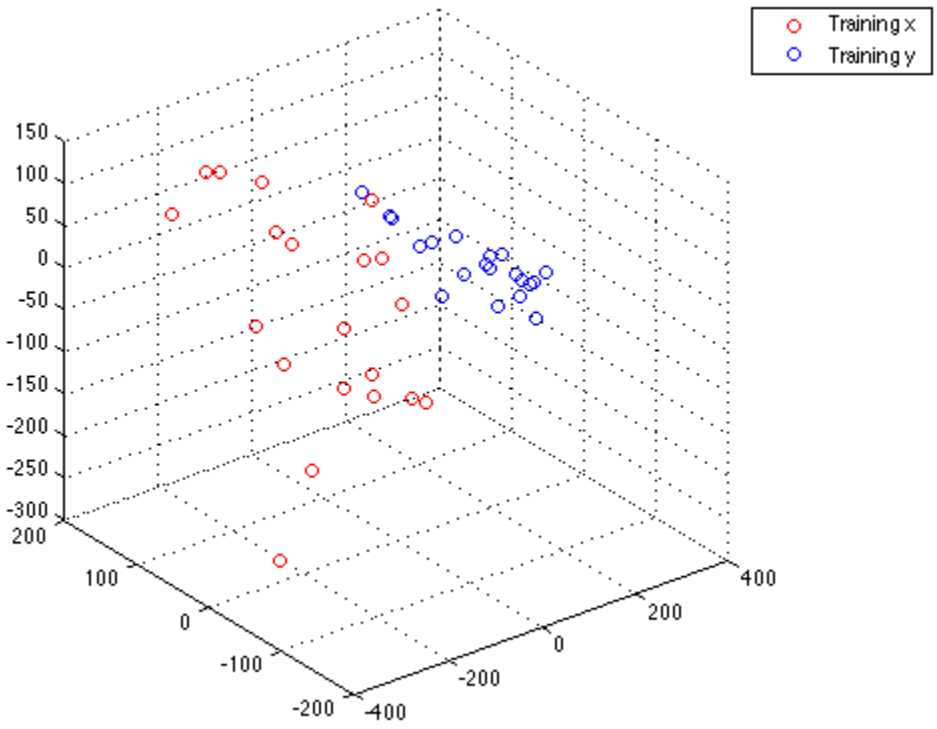
```
% Size of each picture
m = 8;
n = 8;

% Number of samples
N = 20;

avg = zeros(m*n,1);
A = [];

% Arrange our cropped and resize images
x = cell(1,20);
count = 1;
for j=1:4
    for k=1:5
        x{count} = letter{j,k};
        count = count + 1;
    end
end
y = cell(1,20);
count = 1;
for j=5:8
    for k=1:5
        y{count} = letter{j,k};
        count = count + 1;
    end
end

% Load x
count = 0;
for j = 1:N
    u = x{j};
    M=double(u);
    R = reshape(M,m*n,1);
    A = [A,R];
    avg = avg + R;
    count = count + 1;
end

% Load y
for j = 1:N
    u = y{j};
    M = double(u);
    R = reshape(M,m*n,1);
```

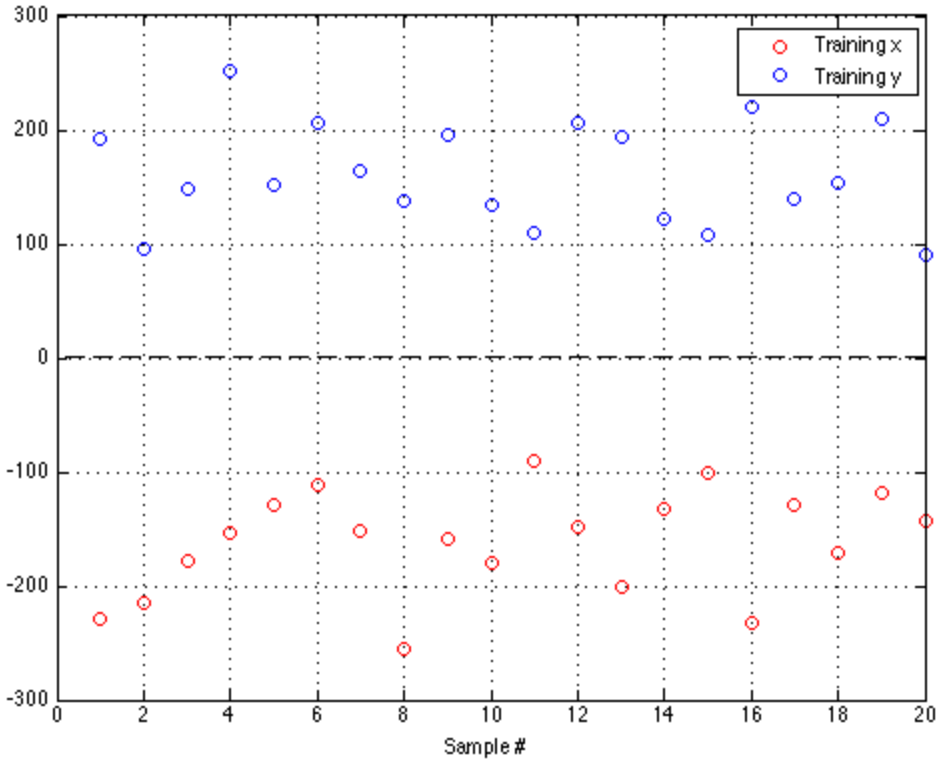
```
    A = [A,R];
    avg = avg + R;
    count = count + 1;
end
avg = avg /count;

% Calculate the average
avgLetter = uint8(reshape(avg,m,n));
figure; colormap gray; image(avgLetter,'CDataMapping','scaled');
axis off;

%De-mean the samples
for j = 1:2*N
    A(:,j) = A(:,j) - avg;
    R = reshape(A(:,j),m,n);
end

% SVD
[U,S,V] = svd(A,0);
Phi = U(:,1:2*N);
Phi(:,1) = -1*Phi(:,1);
figure; colormap gray;
count = 1;
% The nine most prominent basis 'vectors'
for j=1:9
    subplot(3,3,count)
    % The 1500* is for contrast purposes, otherwise we seen gray boxes
    image(uint8(reshape(1500*Phi(:,count),m,n)),'CDataMapping','scaled');
    count = count + 1;
    axis off;
end

for j = 1:N
    imvec = A(:,j);
    X(:,j) = imvec'*Phi(:,1:3);
end
for j = 1:N
    imvec = A(:,N+j);
    Y(:,j) = imvec'*Phi(:,1:3);
end

% Plot the positions in a space spanned by the three most prominent basis
% 'vectors,' we have 40, but we can only visualize three.
figure;

plot3(X(1,:),X(2,:),X(3,:),'ro');
hold on;
plot3(Y(1,:),Y(2,:),Y(3,:),'bo');
legend('Training x','Training y');
grid on;

% We see that when we project our training images onto the first three
% eigenbasis that there is a clear distinction between an x and a y. In
% fact this is even overkill, we can plot just the first basis and see
```

```
% the separation is just as clear, as pointed out above, this suggests a
% simple method may also be just as effective given that we are ignoring 39
% of the basis 'vectors' we computed.

figure;

plot(X(1,:),'ro');
hold on;
plot(Y(1,:),'bo');
legend('Training x','Training y');
grid on;
xlabel('Sample #');
plot([0 20],[0 0],'k--'); % The zero line

% We have 100% differentiation in the training set
```





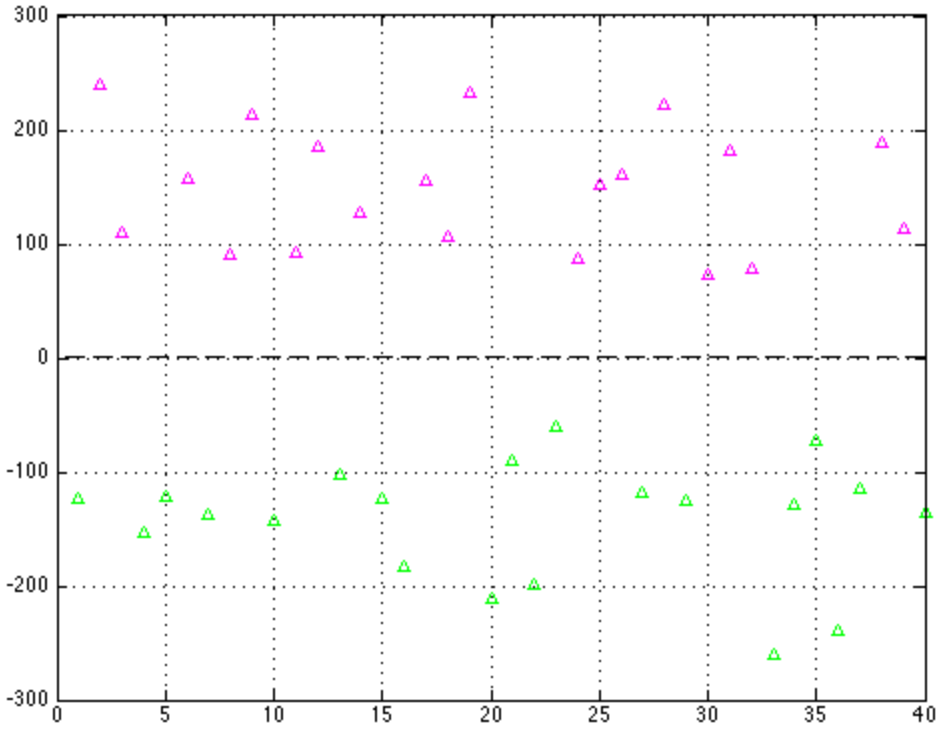

Training x
Training y
150
100
50
0
-50
-100
-150
-200
-250
-300
200
100
0
-100
-200
-400
-200
0
200
400



Test Our Approach on the Test Set

```
xytest = imread('x_y_test.png');
xytest = xytest(:,:,1);
xytest = xytest(73:end-82,75:end-101);

letterTest = cell(8,5);

for j=1:8
    for k=1:5
        letterTest{j,k} = xytest(1+(187*(j-1)):187*j,1+(220*(k-1)):220*k);
        letterTest{j,k} = letterTest{j,k}(40:end-40,40:end-40);
    end
end

for j=1:8
    for k=1:5
        letterTest{j,k} = 255 - letterTest{j,k};

        rowSum = sum(letterTest{j,k},1);
        colSum = sum(letterTest{j,k},2);
        nonWhiteHor = find(rowSum);
        nonWhiteVert = find(colSum);
        letterTest{j,k} = letterTest{j,k}(nonWhiteVert(1):nonWhiteVert(end), ...
                                          nonWhiteHor(1):nonWhiteHor(end));
```

```
        letterTest{j,k} = 255 - letterTest{j,k};

        letterTest{j,k} = imresize(letterTest{j,k}, [8 8]);
    end
end

test = cell(1,40);
count = 1;
for j=1:8
    for k=1:5
        test{count} = letterTest{j,k};
        count = count + 1;
    end
end

figure;
plot([0 40],[0 0],'k--');
hold on;
grid on;

% True values, 1=x, 0=y
isX = [1,0,0,1,1,0,1,0,0,1,0,0,1,0,1,1,0,0,0,...
       1,1,1,1,0,0,0,1,0,1,0,0,0,1,1,1,1,1,0,0,1];

for j=1:40
    u = double(test{j});
    u = reshape(u,m*n,1)-avg;
    proj = u'*Phi(:,1);
    % Plot a certain color depending on true value
    if isX(j)==1
        plot(j,proj,'g^');
    else
        plot(j,proj,'m^');
    end
end

% We see that it got 100% accuracy on the testing set as well, everything
% above the zero line is actually a y, and everything below is actually
% an x!
```



Published with MATLAB® 7.14