Table of Contents


```
clear all;
close all;
clc;

tic
```

Load in the MNIST Dataset

```
images = loadMNISTImages('C:\Users\PhD\Documents\MATLAB\Exploring-AXB-
using-the-MNIST-Database\MNIST-Dataset\train-images.idx3-ubyte');
labels = loadMNISTLabels('C:\Users\PhD\Documents\MATLAB\Exploring-AXB-
using-the-MNIST-Database\MNIST-Dataset\train-labels.idx1-ubyte');
imagesTest = loadMNISTImages('C:\Users\PhD\Documents\MATLAB\Exploring-
AXB-using-the-MNIST-Database\MNIST-Dataset\t10k-images.idx3-ubyte');
labelsTestAns = loadMNISTLabels('C:\Users\PhD\Documents\MATLAB
\Exploring-AXB-using-the-MNIST-Database\MNIST-Dataset\t10k-
labels.idx1-ubyte');

pDim = 28; %size of photos used in dataset
```

```
        2051

        2051
```

Using variousAX=Bsolvers, determine a mapping from the image space to the label space.

X = data matrix A = model parameters we are trying to find B = 0 to 1 and then loop for each of the nine number options

```
parfor iter = 1:10
    if iter==10
        BLabelsTrain(iter,:) = 0==labels.';
        BLabelsTest(iter,:) = 0==labelsTestAns.';
    else
        BLabelsTrain(iter,:) = iter==labels.';
        BLabelsTest(iter,:) = iter==labelsTestAns.';
    end
    %generate the different A values
```

```
    ApInv(:,:,iter) = pinv(images.')*BLabelsTrain(iter,:).';
    ABsl(:,:,iter) =  images.'\BLabelsTrain(iter,:).'; %fix matrix
 dimensions
    ALasso0(:,:,iter) =
 lasso(images.',double(BLabelsTrain(iter,:).'),'Lambda',0,'Options',statset('UsePa
    ALasso1(:,:,iter) =
 lasso(images.',double(BLabelsTrain(iter,:).'),'Lambda',0.001,'NumLambda',1,'Optio
    ALasso5(:,:,iter) =
 lasso(images.',double(BLabelsTrain(iter,:).'),'Alpha',0.003,'NumLambda',1,'Option
    ARobust(:,:,iter) =
 robustfit(images(2:end,:).',BLabelsTrain(iter,:).');
    ARidge(:,:,iter) =
 ridge(BLabelsTrain(iter,:).',images(2:end,:).',0.5,0);
    %Build the labels
    labelsTestL0(iter,:) = round(ALasso0(:,:,iter)*imagesTest);
    labelsTestL1(iter,:) = round(ALasso1(:,:,iter)*imagesTest);
    labelsTestL5(iter,:) = round(ALasso5(:,:,iter)*imagesTest);
    labelsTestpInv(iter,:) =
 round(ApInv(:,:,iter).'*imagesTest); %note if we have multiple
 answers above .5 we get the wrong answer.
    labelsTestBsl(iter,:) = round(ABsl(:,:,iter).'*imagesTest);
    labelsTestRo(iter,:) = round(ARobust(:,:,iter).'*imagesTest);
    labelsTestRi(iter,:) = round(ARidge(:,:,iter).'*imagesTest);
    %Calculate the errors
    numWrongL0(iter) = sum(BLabelsTest(iter,:)-labelsTestL0(iter,:));
    numWrongL1(iter) = sum(BLabelsTest(iter,:)-labelsTestL1(iter,:));
    numWrongL5(iter) = sum(BLabelsTest(iter,:)-labelsTestL5(iter,:));
    numWrongpInv(iter) = sum(BLabelsTest(iter,:)-
labelsTestpInv(iter,:));
    numWrongBsl(iter) = sum(BLabelsTest(iter,:)-
labelsTestBsl(iter,:));
    numWrongRo(iter) = sum(BLabelsTest(iter,:)-labelsTestRo(iter,:));
    numWrongRi(iter) = sum(BLabelsTest(iter,:)-labelsTestRi(iter,:));

    totalPossible(iter) = sum(BLabelsTest(iter,:));
    ApInvShow = reshape(ApInv(:,:,iter),pDim,pDim);
    subplot(3,4,iter);pcolor(ApInvShow);
    disp(iter);
end
```

```
Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 12).
Warning: Rank deficient, rank = 712, tol =  2.270267e-09.
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
Warning: Rank deficient, rank = 712, tol =  2.270267e-09.
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
Warning: Rank deficient, rank = 712, tol =  2.270267e-09.
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
Warning: Rank deficient, rank = 712, tol =  2.270267e-09.
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
```

```
Warning: Rank deficient, rank = 712, tol =  2.270267e-09.
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
Warning: Rank deficient, rank = 712, tol =  2.270267e-09.
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
Warning: Rank deficient, rank = 712, tol =  2.270267e-09.
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
Warning: Rank deficient, rank = 712, tol =  2.270267e-09.
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
Warning: Rank deficient, rank = 712, tol =  2.270267e-09.
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
Warning: Rank deficient, rank = 712, tol =  2.270267e-09.
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
Warning: X is rank deficient, rank = 713
> In statrobustfit (line 47)
In robustfit (line 114)
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
Warning: X is rank deficient, rank = 713
> In statrobustfit (line 47)
In robustfit (line 114)
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
Warning: X is rank deficient, rank = 713
> In statrobustfit (line 47)
In robustfit (line 114)
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
Warning: X is rank deficient, rank = 713
> In statrobustfit (line 47)
In robustfit (line 114)
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
Warning: X is rank deficient, rank = 713
> In statrobustfit (line 47)
In robustfit (line 114)
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
Warning: X is rank deficient, rank = 713
> In statrobustfit (line 47)
In robustfit (line 114)
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
Warning: X is rank deficient, rank = 713
> In statrobustfit (line 47)
In robustfit (line 114)
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
Warning: X is rank deficient, rank = 713
```

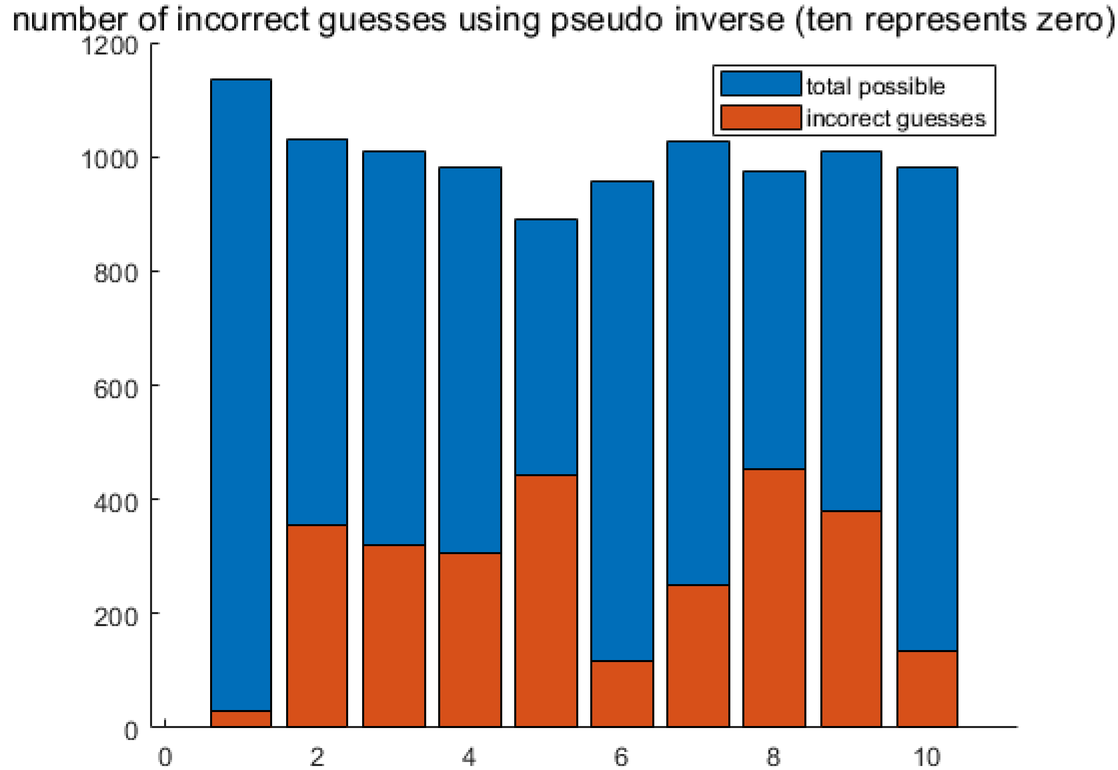
```
> In statrobustfit (line 47)
In robustfit (line 114)
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
Warning: X is rank deficient, rank = 713
> In statrobustfit (line 47)
In robustfit (line 114)
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
Warning: X is rank deficient, rank = 713
> In statrobustfit (line 47)
In robustfit (line 114)
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
     4

     8

Warning: Iteration limit reached.
> In statrobustfit (line 80)
In robustfit (line 114)
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
Warning: Iteration limit reached.
> In statrobustfit (line 80)
In robustfit (line 114)
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
Warning: Iteration limit reached.
> In statrobustfit (line 80)
In robustfit (line 114)
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
Warning: Iteration limit reached.
> In statrobustfit (line 80)
In robustfit (line 114)
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
Warning: Iteration limit reached.
> In statrobustfit (line 80)
In robustfit (line 114)
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
Warning: Iteration limit reached.
> In statrobustfit (line 80)
In robustfit (line 114)
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
Warning: Iteration limit reached.
> In statrobustfit (line 80)
In robustfit (line 114)
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
     6
```

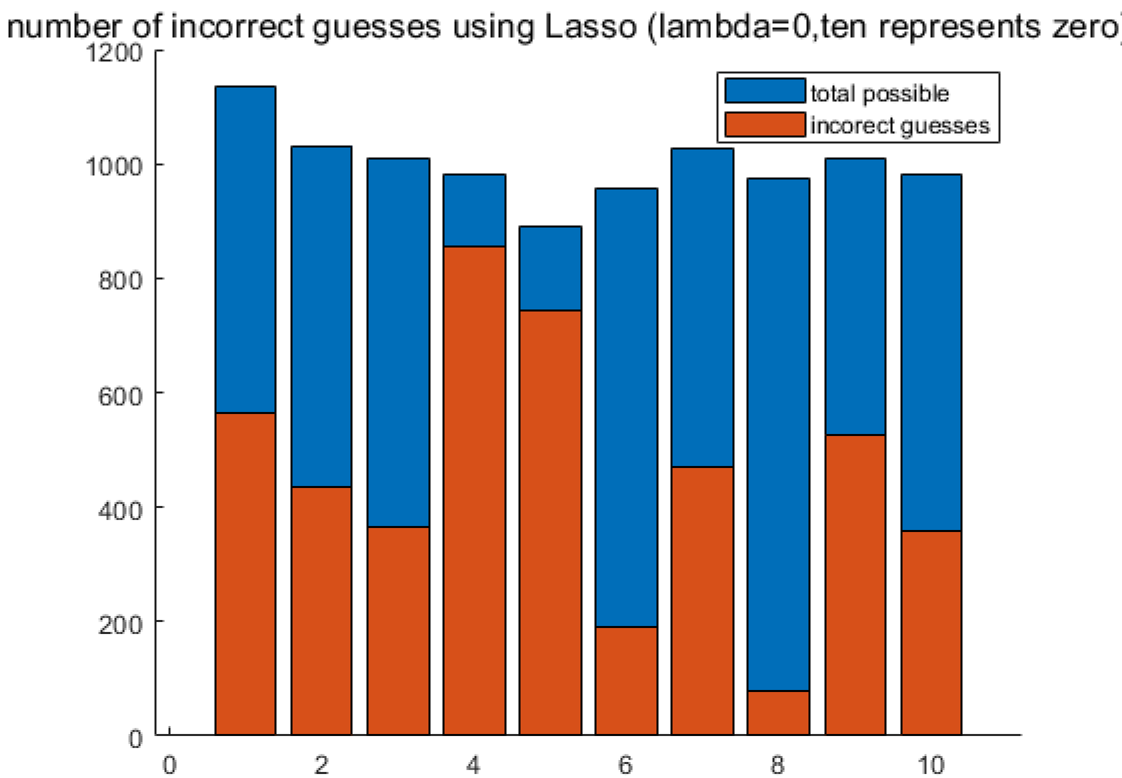
```
     2

Warning: Iteration limit reached.
> In statrobustfit (line 80)
In robustfit (line 114)
In parallel_function>make_general_channel/channel_general (line 832)
In remoteParallelFunction (line 67)
     1

     9

     3

     5

     7

    10
```

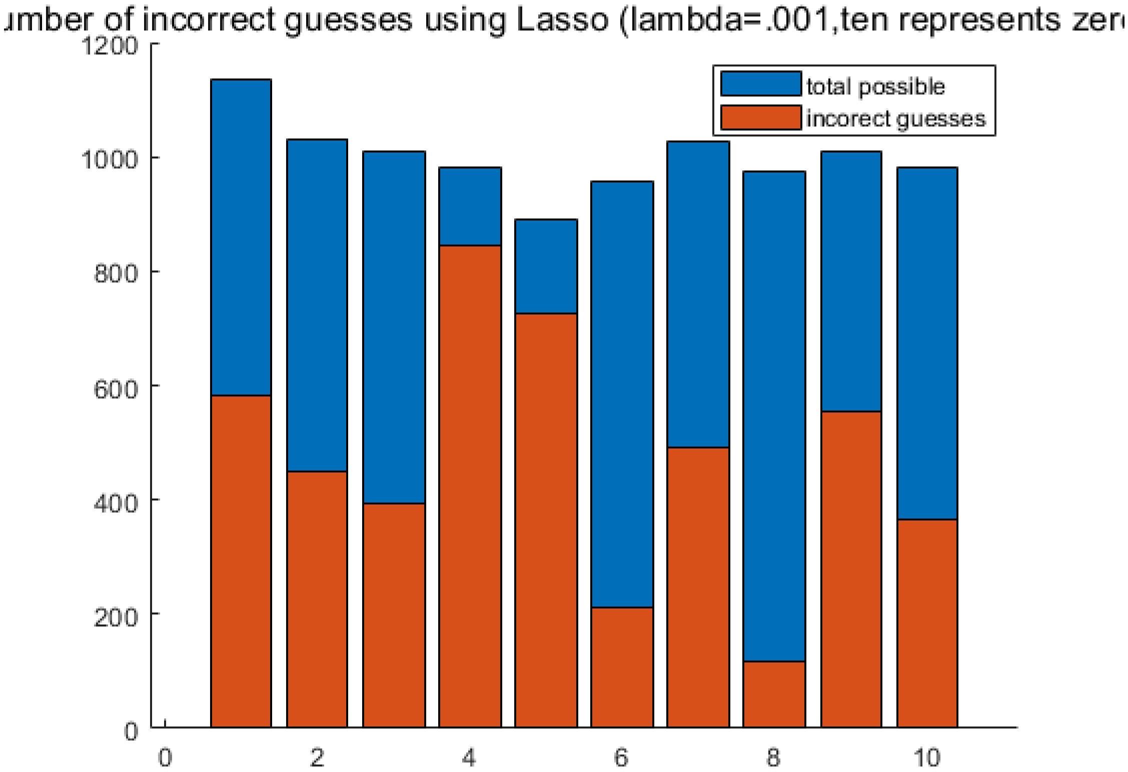
Determining the effectiveness of Maping Algorithms at identifying Digits

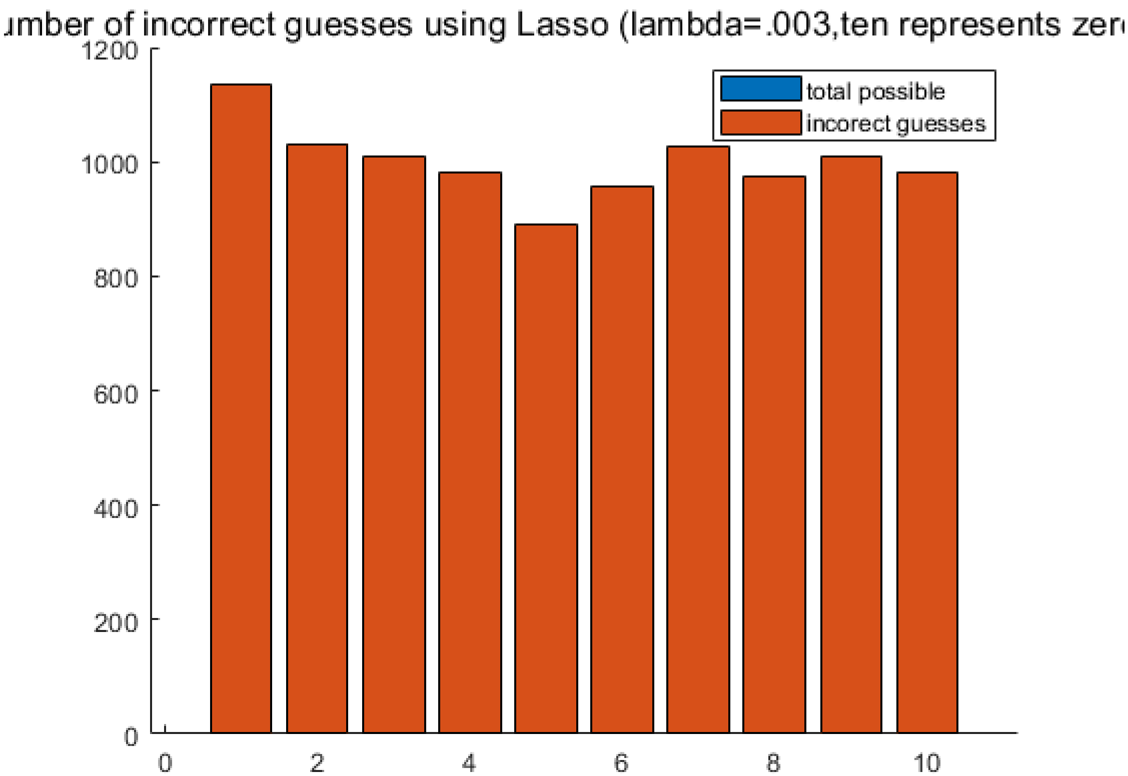
```
figure(); hold on;
bar(totalPossible);
bar(numWrongpInv);
sgtitle('number of incorrect guesses using pseudo inverse (ten
 represents zero)')
legend('total possible','incorect guesses')

figure(); hold on;
bar(totalPossible);
bar(numWrongL0);
sgtitle('number of incorrect guesses using Lasso (lambda=0,ten
 represents zero)')
legend('total possible','incorect guesses')

figure(); hold on;
bar(totalPossible);
bar(numWrongL1);
sgtitle('number of incorrect guesses using Lasso (lambda=.001,ten
 represents zero)')
legend('total possible','incorect guesses')

figure(); hold on;
bar(totalPossible);
bar(numWrongL5);
sgtitle('number of incorrect guesses using Lasso (lambda=.003,ten
 represents zero)')
legend('total possible','incorect guesses')
```

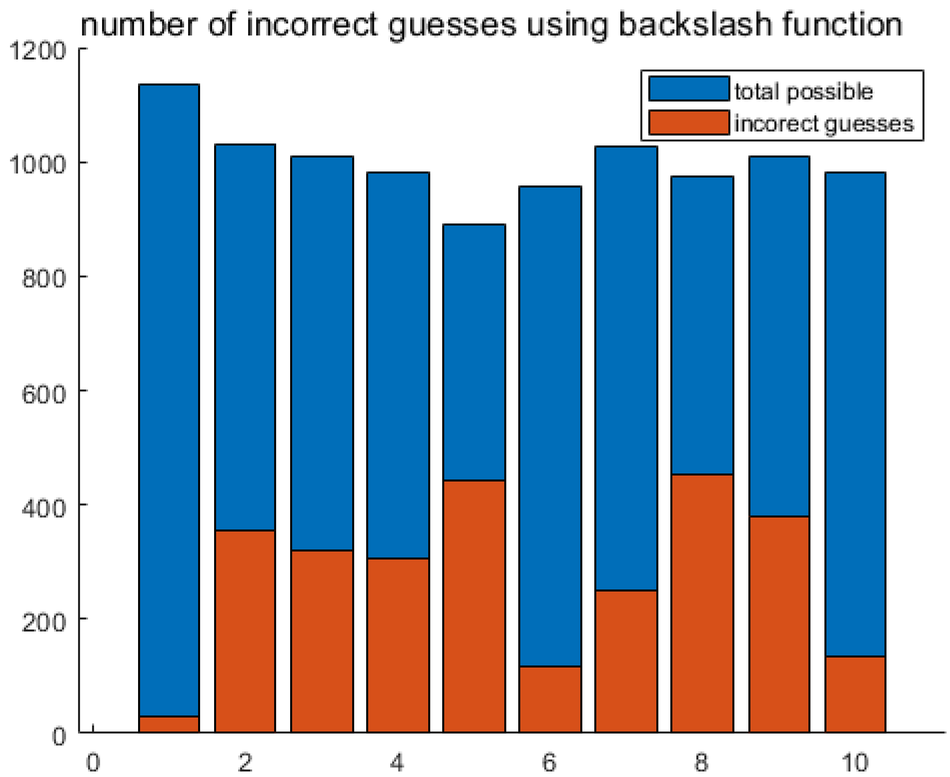
```
figure(); hold on;
bar(totalPossible);
bar(numWrongBsl);
sgtitle('number of incorrect guesses using backslash function')
legend('total possible','incorect guesses')

figure(); hold on;
bar(totalPossible);
bar(numWrongRo);
sgtitle('number of incorrect guesses using robust fit')
legend('total possible','incorect guesses')

figure(); hold on;
bar(totalPossible);
bar(numWrongRi);
sgtitle('number of incorrect guesses using ridge')
legend('total possible','incorect guesses')

%{
In general most solvers do better than random guessing for the digits
 in
the testing data. The backslash solve performs the best of the bunch
 with
lasso (lambda=.003) resulting in the worst guesses. Given the compute
optimization that has been leveraged in Matlab's backslash command,
 this
seems like the best algorith to start with. As the nature of your
 dataset
changes, and the information you hope to extract varies, the other
 solvers
make more sense.


As can be seen from the lasso plots, we can over promote
sparsity in the dataset by changing lambda. Instead we will look at
 different
ways to determine the most important pixels further below. We use the
 Ridge
Regression technique as another way to promote sparsity within the
 dataset
and it is able to generate very meaningful and accurate results. As
 Lambda
increases beyond a small value, we see the accuracy of guesses
 diminishes
dramatically for the lasso method. This is likely due the sparsity
 promoting
 nature of the algorithm and the way guesses are encoded in the
 system.
%}
```

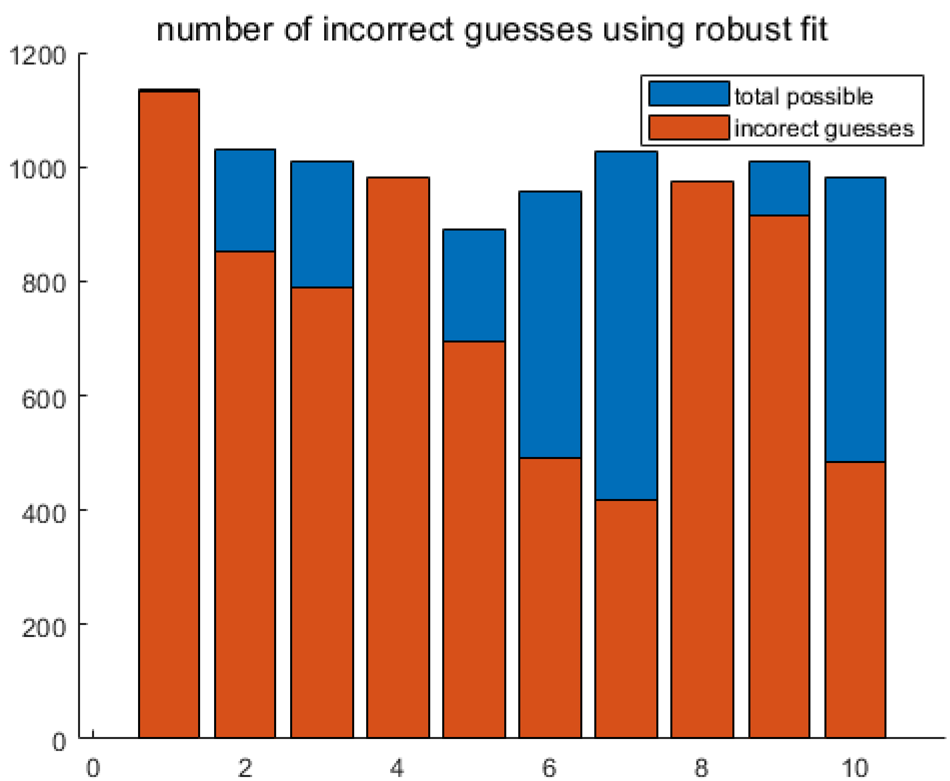

number of incorrect guesses using pseudo inverse (ten represents zero)
total possible
incorect guesses
0
200
400
600
800
1000
1200
0
2
4
6
8
10

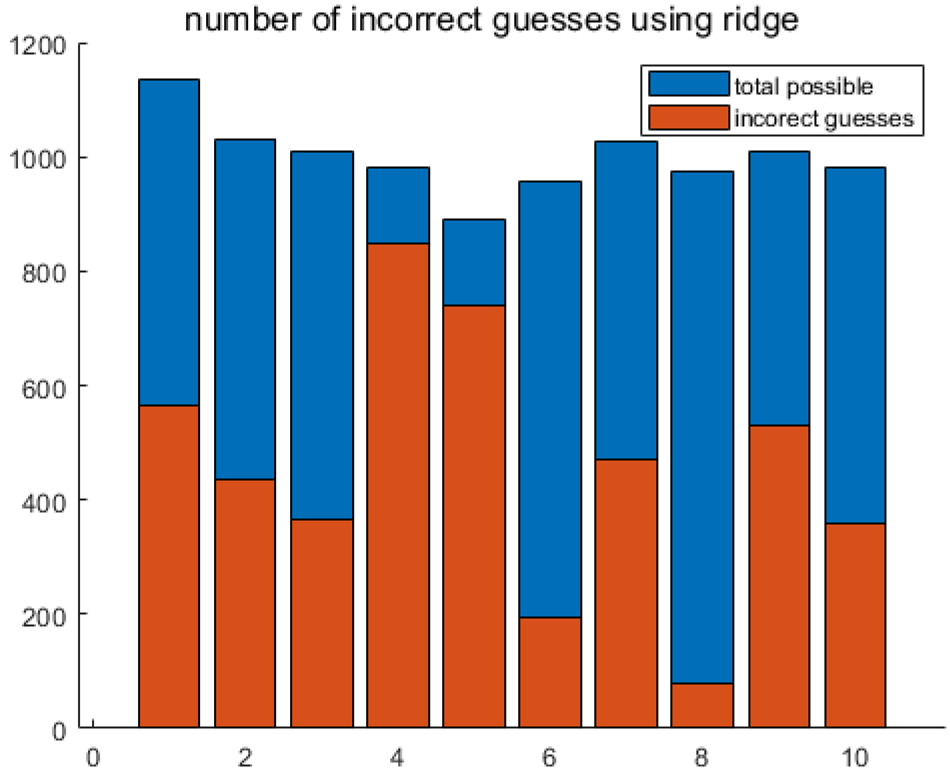

number of incorrect guesses using Lasso (lambda=0,ten represents zero)
total possible
incorect guesses
0
200
400
600
800
1000
1200
0
2
4
6
8
10


umber of incorrect guesses using Lasso (lambda=.001,ten represents zer
total possible
incorect guesses


umber of incorrect guesses using Lasso (lambda=.003,ten represents zer
total possible
incorect guesses


number of incorrect guesses using backslash function
total possible
incorect guesses
1200
1000
800
600
400
200
0
0
2
4
6
8
10


number of incorrect guesses using robust fit
total possible
incorect guesses
1200
1000
800
600
400
200
0
0
2
4
6
8
10

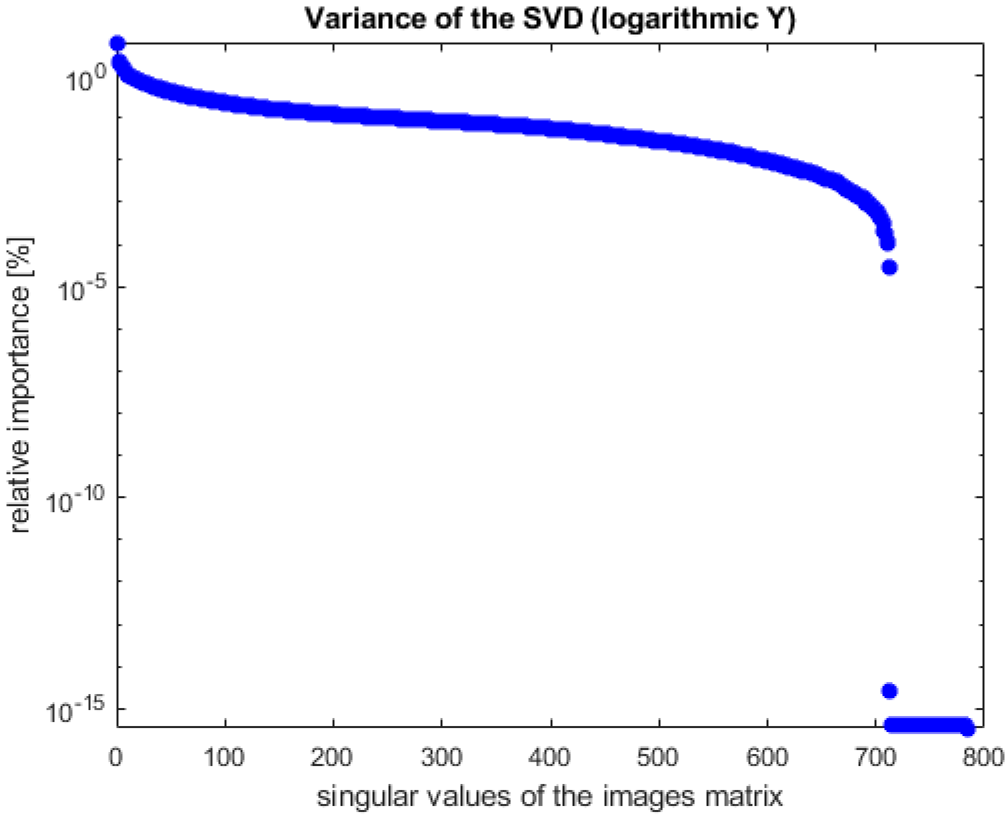


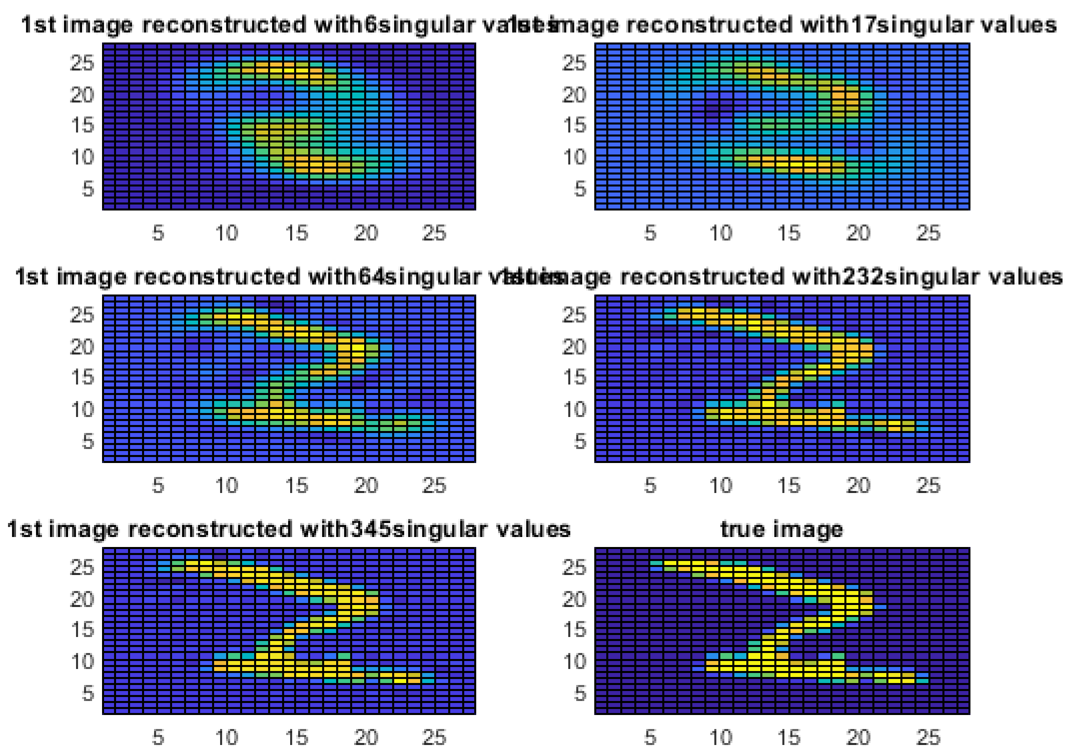
Using SVD to investigate Data

```
[U,S,V] = svd(images,0);

figure()
semilogy(100*diag(S)/sum(diag(S)),'b*','linewidth',[2]);
title('Variance of the SVD (logarithmic Y)')
hold on; xlabel('singular values of the images matrix');
 ylabel('relative importance [%]');

%Code for determining the Percentage covered but the nth-Singular
 Value
testPs = [13,25,50,80,90,99];
pCapture = zeros(1,length(testPs)); %percent covered by the Singular
 values
iter=ones(1,length(testPs));
Svec = diag(S);
for jter=1:length(testPs)
    while pCapture(jter) <testPs(jter);
        pCapture(jter) =100*sum(Svec(1:iter(jter)))/sum(Svec);
        iter(jter)=iter(jter)+1;
    end
    disp(strcat("The number of singular values
 needed for ",num2str(testPs(jter)),'percent coverage
 is',num2str(iter(jter)),'.'));
```

```
end
singValsCap = iter; %store the number of singular values needed for
 future use
%{
 Note: there is a sharp drop off around X=650. We do not see a strong
 correlation with a singular value. The most dominant value contains
 only
 6% of trhe relative importance. By the 13th singular value, the
 relative
 importance has dropped to be less than one per pixel. For 80%
 coverage,
 we need the first 232 Singular Values, for 90% we need 345, and for
 99%
 coverage we need 558 singular values.
%}

%Reconstruct the Images dataset with the fewest pixels possible
    figure();
for iter = 1:length(testPs)
    k=singValsCap(iter);
    imagesRecon(:,:,iter) = U(:,1:k)*diag(Svec(1:k))*V(:,1:k)';
    %plotting the 1st value
    subplot(3,2,iter);
    pcolor(reshape(imagesRecon(:,1,iter),pDim,pDim));
    title(strcat('1st image reconstructed with',num2str(k),'singular
 values'));
end
subplot(3,2,6);
pcolor(reshape(images(:,1),pDim,pDim));
title(strcat('true image'));
%{
from the above plotting, it looks like 6, 17, and 232 singular values
 will
 give the most interesting results while reducing the number of
 singular
 values used.
%}
```

The number of singular values needed for 13percent coverage is6.
The number of singular values needed for 25percent coverage is17.
The number of singular values needed for 50percent coverage is64.
The number of singular values needed for 80percent coverage is232.
The number of singular values needed for 90percent coverage is345.
The number of singular values needed for 99percent coverage is558.

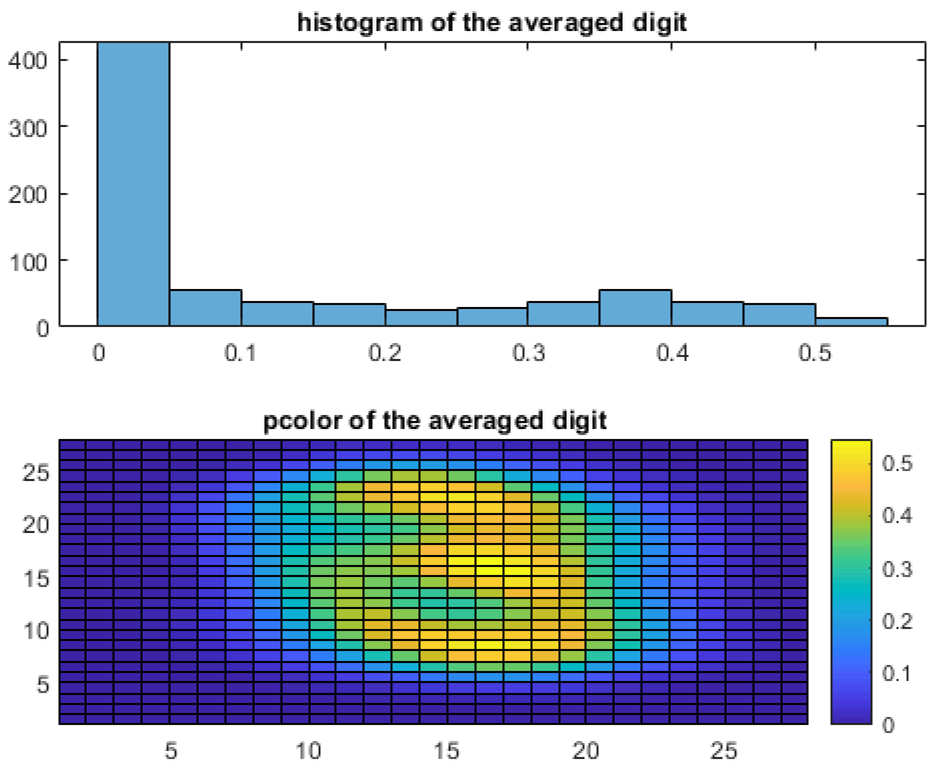

Variance of the SVD (logarithmic Y)
relative importance [%]
singular values of the images matrix
10^{0}
10^{-5}
10^{-10}
10^{-15}
0
100
200
300
400
500
600
700
800

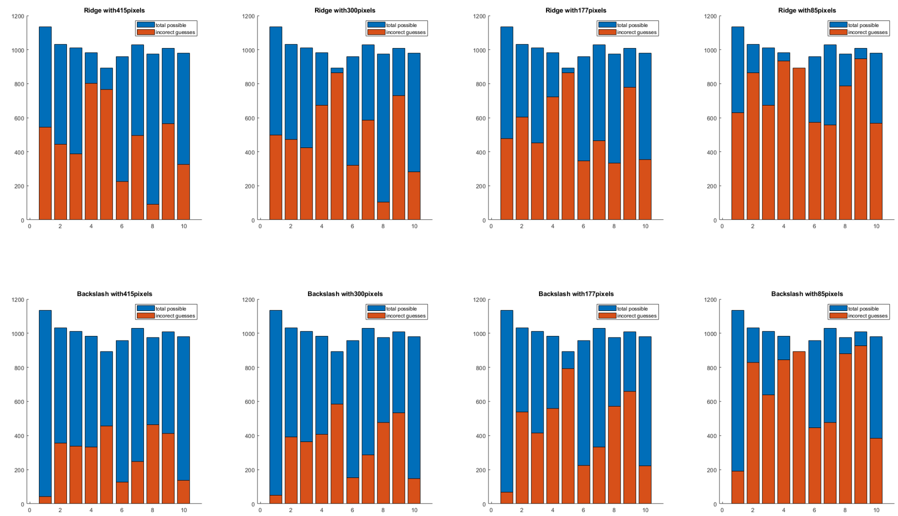

1st image reconstructed with6singular values
1st image reconstructed with17singular values
1st image reconstructed with64singular values
1st image reconstructed with232singular values
1st image reconstructed with345singular values
true image
5
10
15
20
25

Determining the most Important Pixels from the dataset

```
meanImg = mean(reshape(images,28,28,60000),3); %generates the average
 image
figure(); hold on;
subplot(2,1,1); histogram(meanImg); title('histogram of the averaged
 digit')
subplot(2,1,2); pcolor(meanImg); title('pcolor of the averaged
 digit'); colorbar('eastoutside');

numConstraint = [.02,.1,.3,.4];
figure(); hold on;
for jter = 1:length(numConstraint)
    linearIndicesG =find(meanImg>numConstraint(jter));
    imagesReduced = images(linearIndicesG,:);
    imagesTestRed = imagesTest(linearIndicesG,:);

    for iter=1:10
        %now recalculate the guesses on the test data using ridge
        ARidgeRed(:,:,iter) =
 ridge(BLabelsTrain(iter,:).',imagesReduced(2:end,:).',0.5,0);
        labelsTestRiRed(iter,:) =
 round(ARidgeRed(:,:,iter).'*imagesTestRed);
        numWrongRiRed(iter) = sum(BLabelsTest(iter,:)-
labelsTestRiRed(iter,:));
        %now recalculate the guesses on the test data using backslash
        ABslRed =  imagesReduced.'\BLabelsTrain(iter,:).';
        labelsTestBslRed(iter,:) = round(ABslRed.'*imagesTestRed);
        numWrongBslRed(iter) = sum(BLabelsTest(iter,:)-
labelsTestBslRed(iter,:));

    end
    subplot(2,4,jter); hold on; bar(totalPossible);
 bar(numWrongRiRed);set(gcf, 'Position', get(0, 'Screensize'));
    title(strcat('Ridge
 with',num2str(length(linearIndicesG)),'pixels'));
    legend('total possible','incorect guesses')
    subplot(2,4,jter+4); hold on; bar(totalPossible);
 bar(numWrongBslRed);set(gcf, 'Position', get(0, 'Screensize'));
    title(strcat('Backslash
 with',num2str(length(linearIndicesG)),'pixels'));
    legend('total possible','incorect guesses')
    %clear old vars
    clear ARidgeRed
    clear ABslRed
    clear labelsTestRiRed
    clear labelsTestBslRed

end

%{
```

```
We can see a significat improvement in the accuracy of guesses by
 removing
the unimportant parts of the dataset. By dropping digits, we are
 training
the solver on the parts of the picture that are most important so we
 would
expect to see equal or slightly better results until we get to the
 point
where we have removed so many pixels that we can no longer
 differentiate
results. And this holds with the data we record. The values in
numConstraints were tested against many values, with the ones kept
resulting in the most interesting information present.

Note we reduce the number of algorithms tested here to both aid
 compute
time and to reduce the strain on user interpretability of the data.
 Ridge
was kept because it represents a parsimonious model and the backslash
solver was kept because it performed the best on the data. This holds
 for
the rest of the analysis in the paper.
%}
```





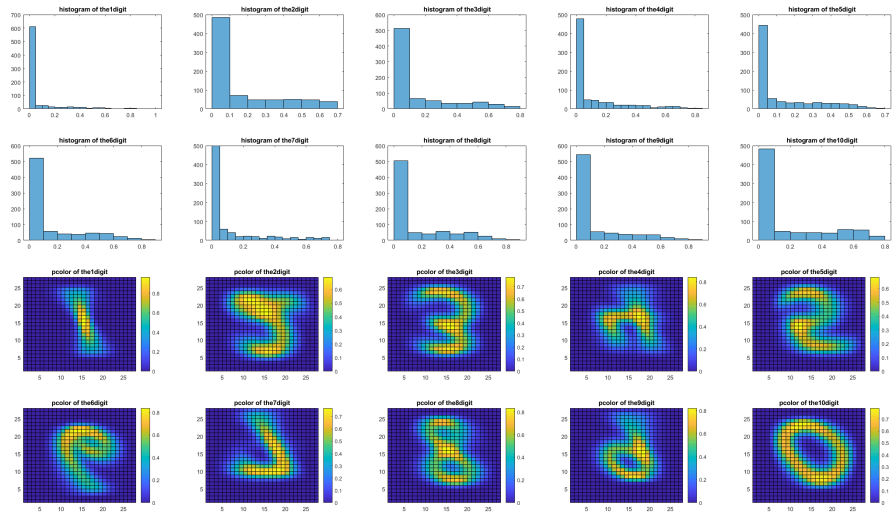
Determine the most important pixels for each number individually

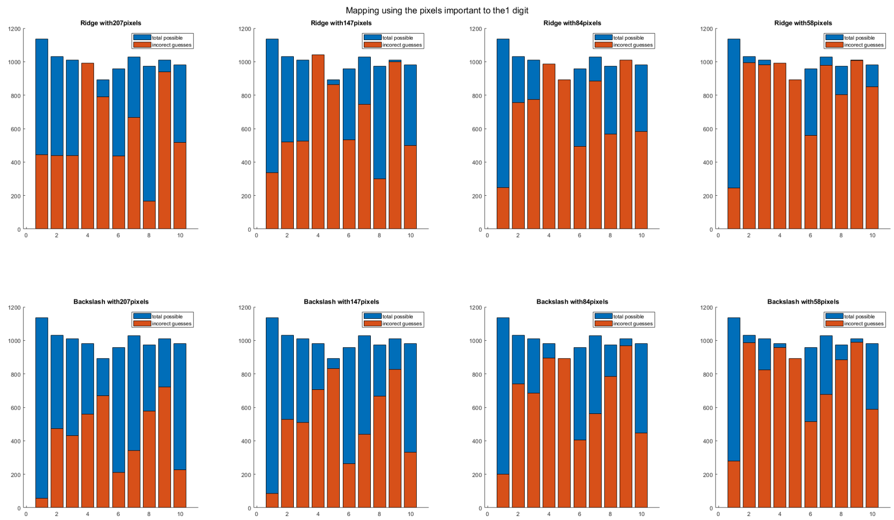
```
figure(); hold on;
for iter=1:10
    %generate the appropriate Mean Images
    [IndRowPerPx,IndColPerPx]
 =ind2sub(size(BLabelsTrain(iter,:)),find(BLabelsTrain(iter,:)>0));
    meanImgPx =
 mean(reshape(images(:,IndColPerPx),28,28,length(IndColPerPx)),3);
    subplot(4,5,iter); histogram(meanImgPx); title(strcat('histogram
 of the',num2str(iter), 'digit'));set(gcf, 'Position',
 get(0, 'Screensize'));
    subplot(4,5,iter+10); pcolor(meanImgPx);
 title(strcat('pcolor of the',num2str(iter), 'digit'));
 colorbar('eastoutside');set(gcf, 'Position', get(0, 'Screensize'));
end
for iter=1:10
    [IndRowPerPx,IndColPerPx]
 =ind2sub(size(BLabelsTrain(iter,:)),find(BLabelsTrain(iter,:)>0));
    meanImgPx =
 mean(reshape(images(:,IndColPerPx),28,28,length(IndColPerPx)),3);
    figure(); hold on; sgtitle(strcat('Mapping using the pixels
 important to the',num2str(iter), ' digit'));
    if iter ==10
        sgtitle(strcat('Mapping using the pixels important to the 0
 digit'));
    end
    %Build the reduced image matrices
    for jter = 1:length(numConstraint)
        linearIndicesPx =find(meanImgPx>numConstraint(jter));
        imagesReducedPx = images(linearIndicesPx,:);
```

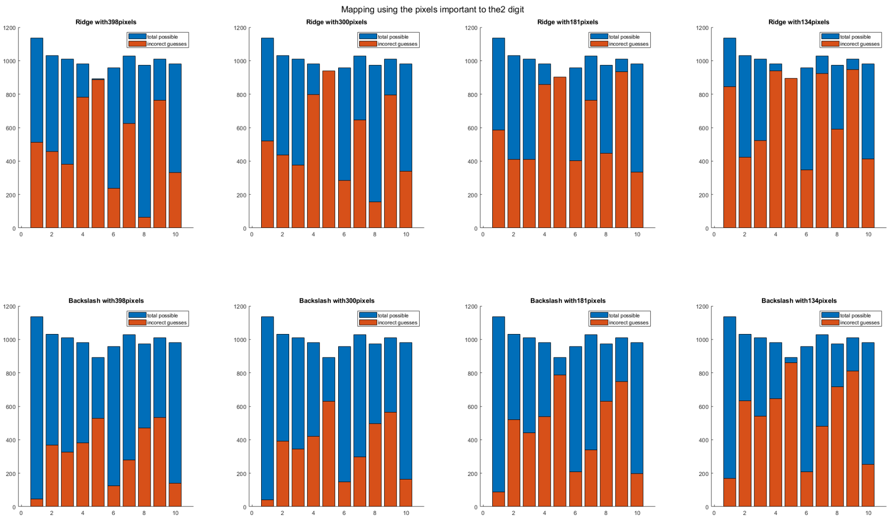
```
        imagesTestRedPx = imagesTest(linearIndicesPx,:);

        for kter=1:10
            %now recalculate the guesses on the test data using ridge
            ARidgeRedPx(:,:,kter) =
 ridge(BLabelsTrain(kter,:).',imagesReducedPx(2:end,:).',0.5,0);
            labelsTestRiRedPx(kter,:) =
 round(ARidgeRedPx(:,:,kter).'*imagesTestRedPx);
            numWrongRiRedPx(kter) = sum(BLabelsTest(kter,:)-
labelsTestRiRedPx(kter,:));
            %now recalculate the guesses on the test data using
 backslash
            ABslRedPx =  imagesReducedPx.'\BLabelsTrain(kter,:).';
            labelsTestBslRedPx(kter,:) =
 round(ABslRedPx.'*imagesTestRedPx);
            numWrongBslRedPx(kter) = sum(BLabelsTest(kter,:)-
labelsTestBslRedPx(kter,:));
            if jter==1 & kter==iter %this jter value was consistently
 good across all digits
            bestcaseWrongBsl(1,iter) = numWrongBslRedPx(kter);
            end
        end
        subplot(2,4,jter); hold on; bar(totalPossible);
 bar(numWrongRiRedPx);set(gcf, 'Position', get(0, 'Screensize'));
        title(strcat('Ridge
 with',num2str(length(linearIndicesPx)),'pixels'));
        legend('total possible','incorect guesses')
        subplot(2,4,jter+4); hold on; bar(totalPossible);
 bar(numWrongBslRedPx);set(gcf, 'Position', get(0, 'Screensize'));
        title(strcat('Backslash
 with',num2str(length(linearIndicesPx)),'pixels'));
        legend('total possible','incorect guesses')
        %clear old vars
        clear ARidgeRedPx
        clear ABslRedPx
        clear labelsTestRiRedPx
        clear labelsTestBslRedPx

    end


end

figure(); hold on;
bar(totalPossible);
bar(bestcaseWrongBsl);
sgtitle('number of incorrect guesses using backslash and the most
 important pixels for the digit we are considering')
legend('total possible','incorect guesses')


%{
There is a lot of data to process here. We see that as we reduce the
number of pixels we are looking at, our ability to correctly guess the
```

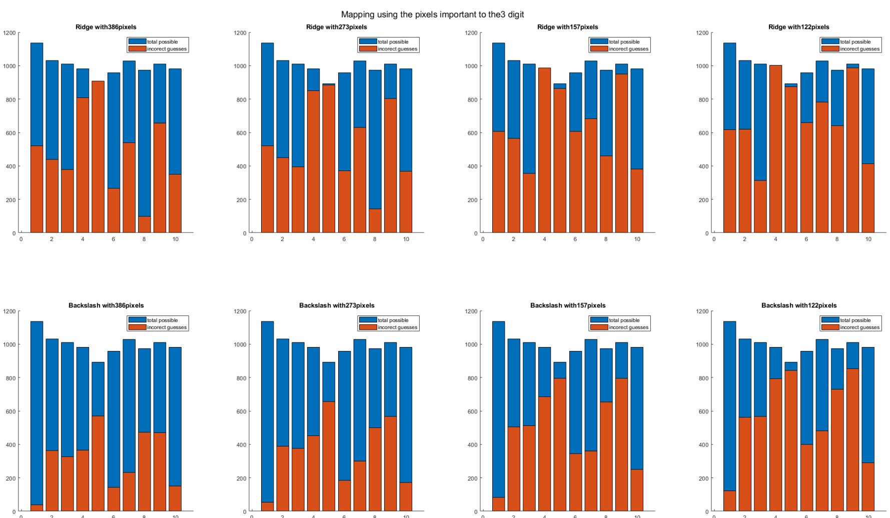
```
digit improves drastically. This makes sense because other digits
 which
could have similar patterns to the one we care about have less room to
 look
like the digit we are searching for. An example would be 3 and 8 which
 when
written by hand look similar. Looking to the Ridge Plot on the 3's
 digit,
we see that we are very accurate in our prediction of the 8s digit
 becuase
they share many of the same important pixels. As we shrink that space
 down,
we become significantly worse at guessing the 8 using the important
 pixels
from the 3.

Another interesting note that this data brings out is our continued
 ability
to properly identify the ones digit. This tracks all of the digits,
 and
especially so using the backslack method of computing A.


On the whole we see equal or better performance by considering only
 the
pixels important to the digit we are guessing at the time. We are far
 off
from a well trained and implimented non-linear nural net (90% plus
identification) but we are significantly better than chance
(10% to guess correctly).
%}

toc
```

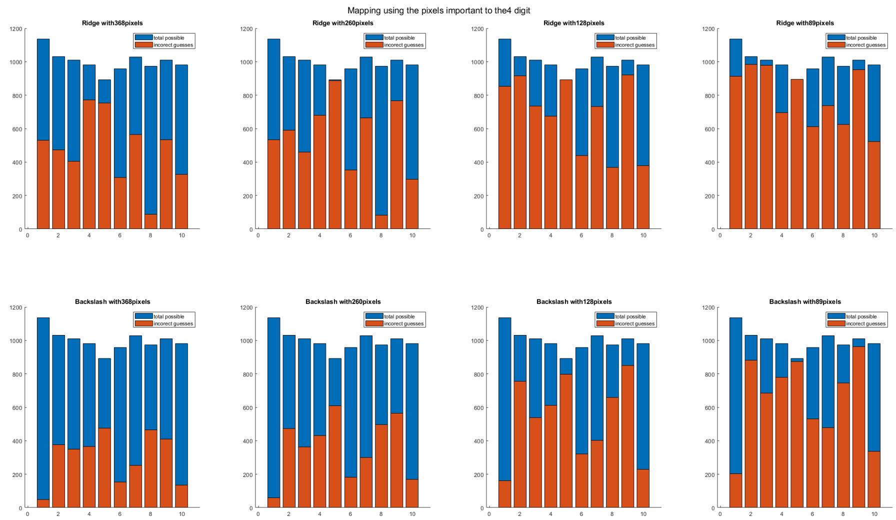
Elapsed time is 2042.359463 seconds.


histogram of the1digit
histogram of the2digit
histogram of the3digit
histogram of the4digit
histogram of the5digit
histogram of the6digit
histogram of the7digit
histogram of the8digit
histogram of the9digit
histogram of the10digit
pcolor of the1digit
pcolor of the2digit
pcolor of the3digit
pcolor of the4digit
pcolor of the5digit
pcolor of the6digit
pcolor of the7digit
pcolor of the8digit
pcolor of the9digit
pcolor of the10digit

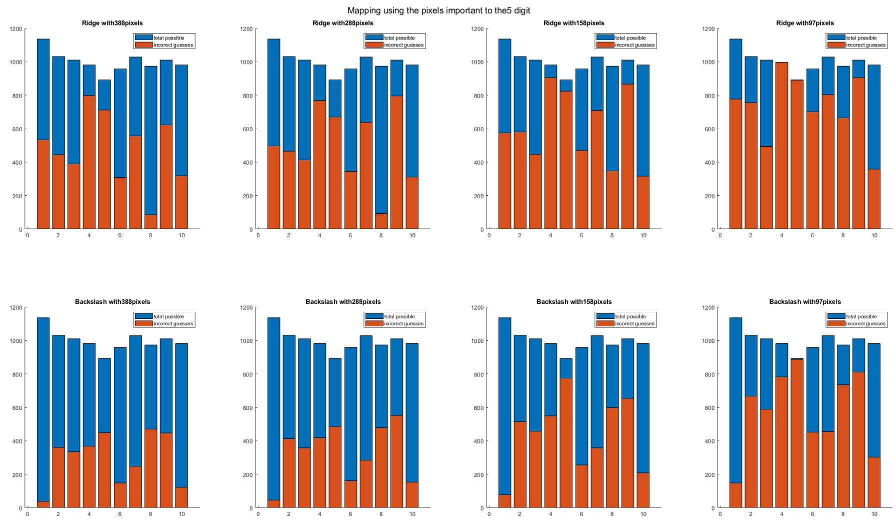

Mapping using the pixels important to the1 digit
Ridge with207pixels
Ridge with147pixels
Ridge with84pixels
Ridge with58pixels
Backslash with207pixels
Backslash with147pixels
Backslash with84pixels
Backslash with58pixels
total possible
incorrect guesses


Mapping using the pixels important to the2 digit
Ridge with398pixels
Ridge with300pixels
Ridge with181pixels
Ridge with134pixels
Backslash with398pixels
Backslash with300pixels
Backslash with181pixels
Backslash with134pixels
total possible
incorrect guesses

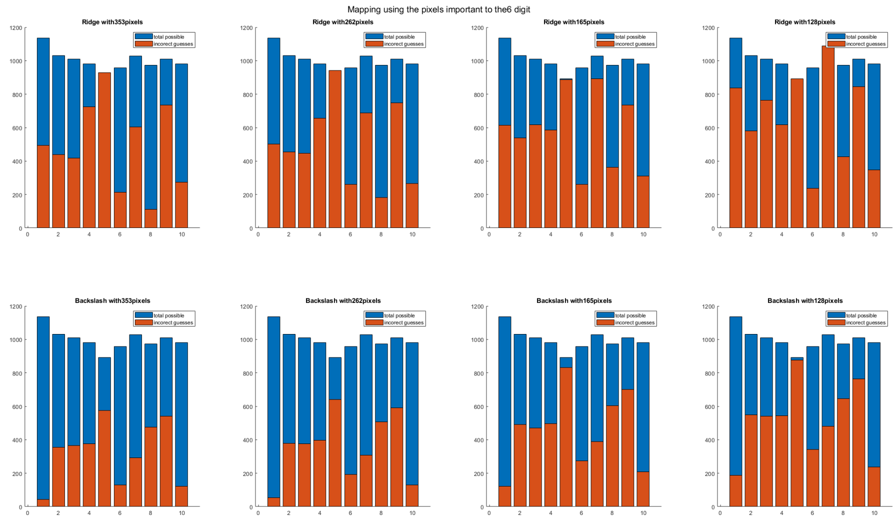

Mapping using the pixels important to the3 digit
Ridge with386pixels
Ridge with273pixels
Ridge with157pixels
Ridge with122pixels
Backslash with386pixels
Backslash with273pixels
Backslash with157pixels
Backslash with122pixels
total possible
incorrect guesses

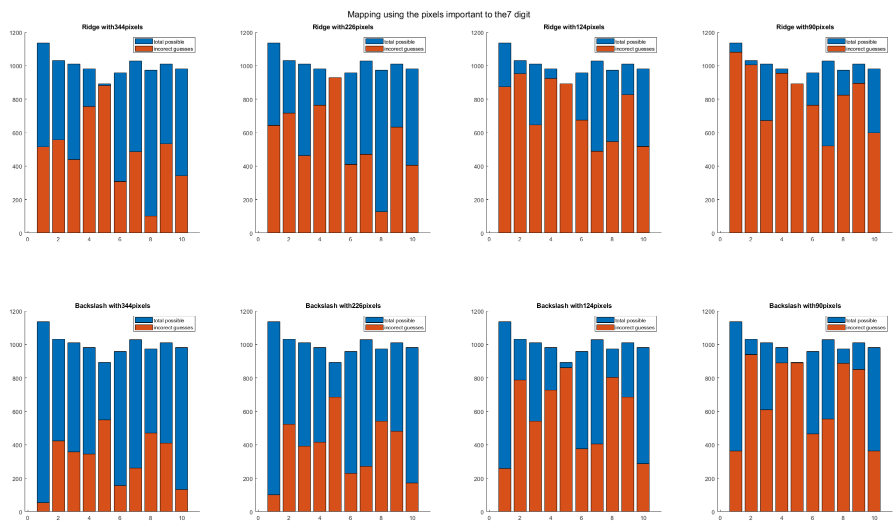


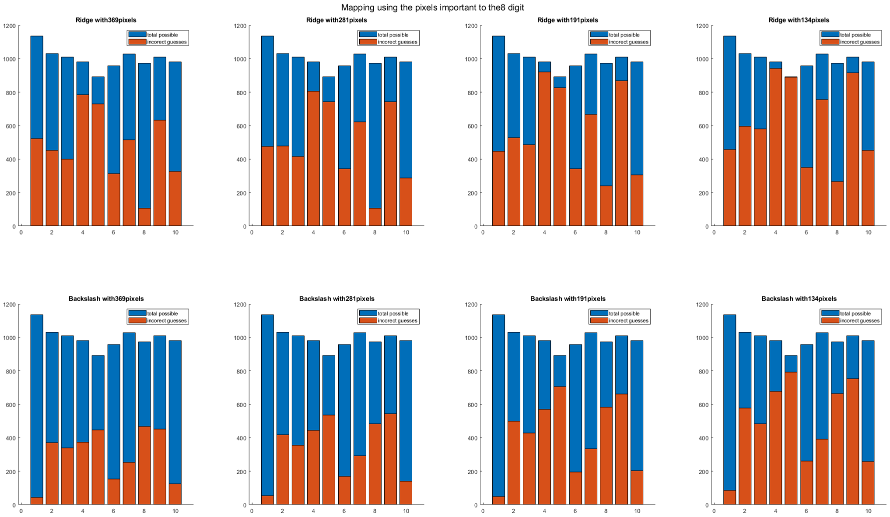
Mapping using the pixels important to the4 digit
Ridge with368pixels
Ridge with260pixels
Ridge with128pixels
Ridge with89pixels
Backslash with368pixels
Backslash with260pixels
Backslash with128pixels
Backslash with89pixels
total possible
incorrect guesses

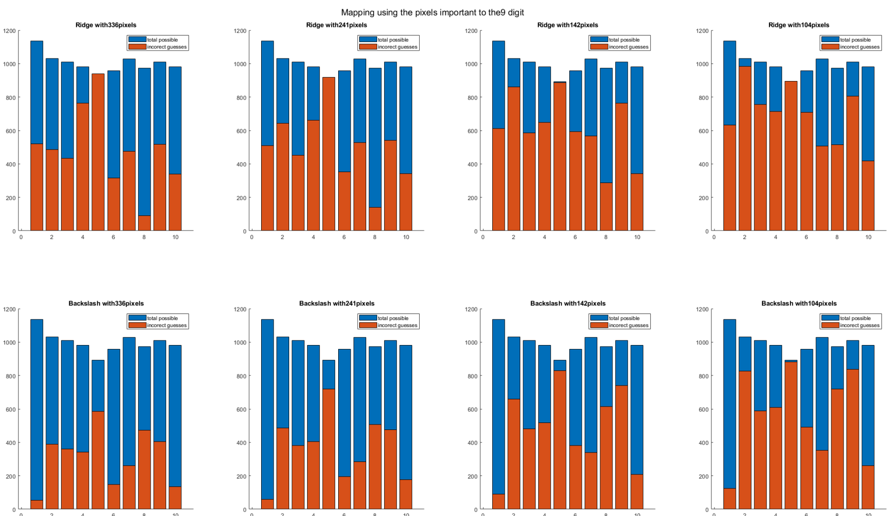


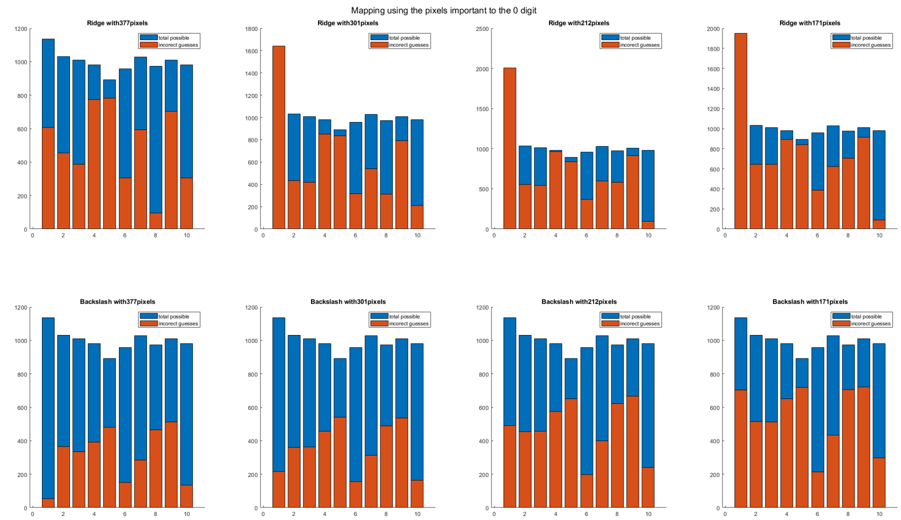
Mapping using the pixels important to the5 digit
Ridge with388pixels
Ridge with288pixels
Ridge with158pixels
Ridge with97pixels
Backslash with388pixels
Backslash with288pixels
Backslash with158pixels
Backslash with97pixels
total possible
incorrect guesses

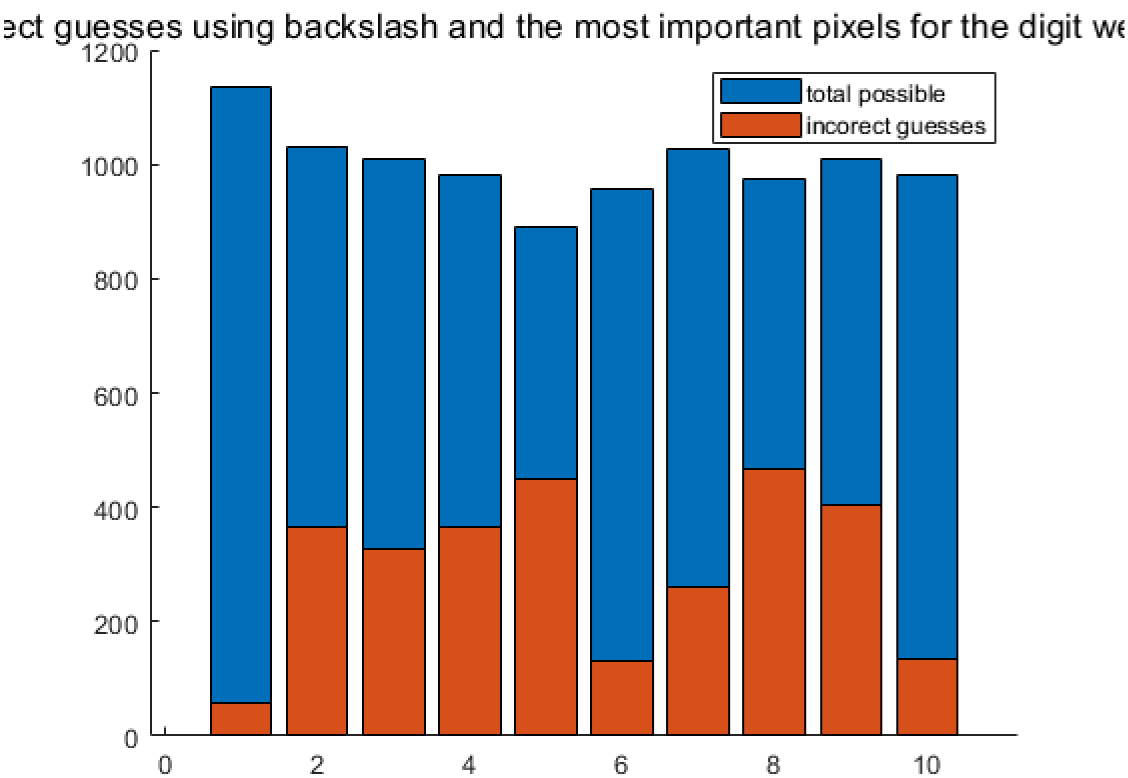

Mapping using the pixels important to the6 digit
Ridge with353pixels
Ridge with262pixels
Ridge with165pixels
Ridge with128pixels
Backslash with353pixels
Backslash with262pixels
Backslash with165pixels
Backslash with128pixels
total possible
incorrect guesses


Mapping using the pixels important to the7 digit
Ridge with344pixels
Ridge with226pixels
Ridge with124pixels
Ridge with90pixels
Backslash with344pixels
Backslash with226pixels
Backslash with124pixels
Backslash with90pixels
total possible
incorrect guesses


Mapping using the pixels important to the8 digit
Ridge with369pixels
Ridge with281pixels
Ridge with191pixels
Ridge with134pixels
Backslash with369pixels
Backslash with281pixels
Backslash with191pixels
Backslash with134pixels
total possible
incorrect guesses


Mapping using the pixels important to the9 digit
Ridge with336pixels
Ridge with241pixels
Ridge with142pixels
Ridge with104pixels
Backslash with336pixels
Backslash with241pixels
Backslash with142pixels
Backslash with104pixels
total possible
incorrect guesses





Published with MATLAB® R2020b