Homework 6 (Midterm 2)
Due: Monday, December 14, 2020

Download the MNIST data set (both training and test sets and labels): yann.lecun.com/exdb/mnist/ The labels will tell you which digit it is: $1, 2, 3, 4, 5, 6, 7, 8, 9, 0$. Let each output be denoted by the vector $\mathbf{y}_j$

$$\text{“1”} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \text{“2”} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}, \quad \cdots \quad, \text{“9”} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{bmatrix}, \text{“0”} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$$

Now let $\mathbf{B}$ be the set of output vectors

$$\mathbf{B} = \begin{bmatrix} \mathbf{y}_1 & \mathbf{y}_2 & \mathbf{y}_3 & \cdots & \mathbf{y}_n \end{bmatrix}$$

and let the matrix $\mathbf{A}$ be the corresponding reshaped (vectorized) MNIST images

$$\mathbf{A} = \begin{bmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \mathbf{x}_3 & \cdots & \mathbf{x}_n \end{bmatrix}$$

Thus each vector $\mathrm{x}_j \in \mathbb{R}^{n^2}$ is an a vector reshaped from the $n \times n$ image.

1. Using various $\mathbf{AX} = \mathbf{B}$ solvers, determine a mapping from the image space to the label space.
2. By promoting sparsity, determine and rank which pixels in the MNIST set are most informative for correctly labeling the digits. (You'll have to come up with your own heuristics or empirical rules for this. Use pcolor to help you visualize the results from $\mathbf{X}$)
3. Apply your most important pixels to the test data set to see how accurate you are with as few pixels as possible.
4. Redo the analysis with each digit individually to find the most important pixels for each digit.
5. IMPORTANT: Think about the interpretation of what you are doing with this $\mathbf{AX} = \mathbf{B}$ problem.

This is an exploratory homework. So play around with the data and make sure to make lots of plots. Good luck, and have fun.

Solution to 1. This can be done in many ways, we will use `lasso`, `pinv` and backslash for this problem.

```
X=zeros(784,10);
for i=1:10
```

```
    X(:,i)=lasso(A,B(:,i),'Lambda',0.01);
end
% X=B*pinv(A); % X=A\B;
```

Solution to 2. We test the accuracy of $\mathbf{X}$ from different solvers in the following way

```
D=Atest*X; % apply mappinp to test images
[~,idx]=max(D,[],2); % find index of largest entry on each row
accuracy=sum(test_labels==idx)/100; % compare with actual labels
```

Running the above yields an accuracy of 85.34% for both `pinv` and backslash, and 78.33% for `lasso` with $\lambda = 0.01$. Since our goal here is to achieve high accuracy with as few pixels as possible, we need to pick out the most important pixels. One logical way to do this is to rank the rows of $\mathbf{X}$ by their 2-norms, then we get a 784 by 1 vector, which we can use to visualize $\mathbf{X}$. We can sort the rows in descending order and the first N rows correspond to the first N most important pixels, which we will use for the test images.

```
X1=vecnorm(X,2,2); % 2-norm of each row
[~,idx]=sort(X1,'descend');
```

Solution to 3. We can choose the first N most important pixels and set all others to 0. It is evident from Figure 1 that 71% accuracy can be achieved with the first 200 pixels, which is not far off the maximum value 78%.

```
N=[50 100 200 400 700 783]; % number of pixels
for i=1:length(N)
    X2=zeros(784,10);
    X2(idx(1:N(i)),:)=X(idx(1:N(i)),:); % extract the first N pixels
    D=Atest*X2;
    [~,idx2]=max(D,[],2);
    accuracy(i)=sum(test_labels==idx2)/100;
end
```

Solution to 4. We use a similar method for the analysis of individual digits. Previously we computed the 2-norm of each row of $\mathbf{X}$, we will do this again, but we split $\mathbf{X}$ into ten columns and order each element of a column by its 2-norm, i.e. absolute value. Then we have ten different orderings and we apply each ordering to $\mathbf{X}$ respectively to get a different mapping for each digit. We find that identifying a particular digit is generally more accurate than identifying an arbitrary one, as shown in Table 1. We plot the most important pixels for each digit in Figure 2.

```
for i=1:10
    X1=X(:,i); % consider a single column of X
    [~,idx]=sort(abs(X1),'descend');
    X2=zeros(784,10);
```

```
    X2(idx(1:100),:)=X(idx(1:100),:); % extract the first 100 pixels
    D=Atest*X2;
    [¬,idx2]=max(D,[],2);
    k=find(test_labels==i); % find index of a particular digit
    accuracy(i)=sum(idx2(k)==i)/sum(test_labels==i);
end
```

Digit	Accuracy
1	96.65
2	94.28
3	96.83
4	82.59
5	74.10
6	96.97
7	88.62
8	99.38
9	95.54
0	97.65

Table 1: Identifying each digit using first 100 pixels from `lasso`.

Solution to 5. Our understanding of this $\mathbf{AX} = \mathbf{B}$ problem is similar to that of the "eigenface" problem in the previous homework. In essence, we are trying to extract the most important features from a noisy data set that contains redundant features. The extracted features have a lower rank than the original data, which can reduce the computational cost when we apply our model to new data sets.

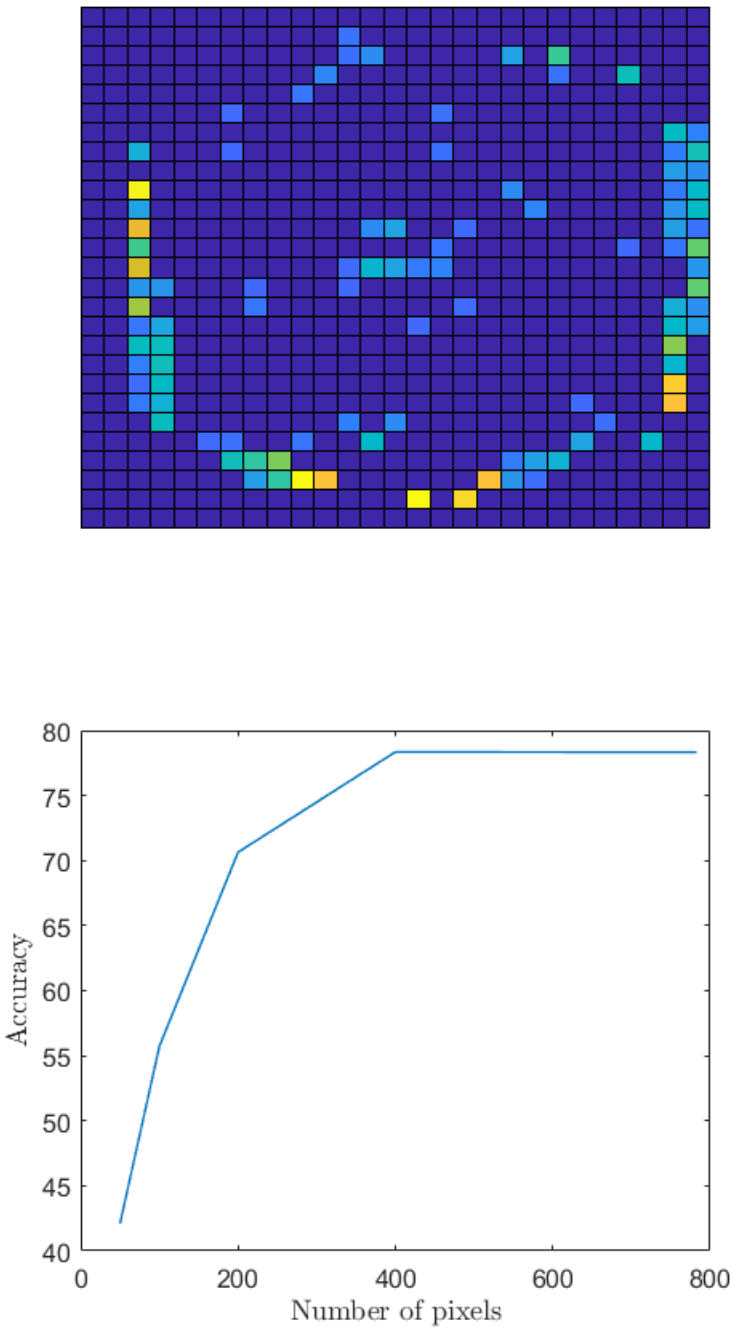

Figure 1: Top: the most important pixels (first 100); Bottom: accuracy of `lasso` as a function of number of pixels

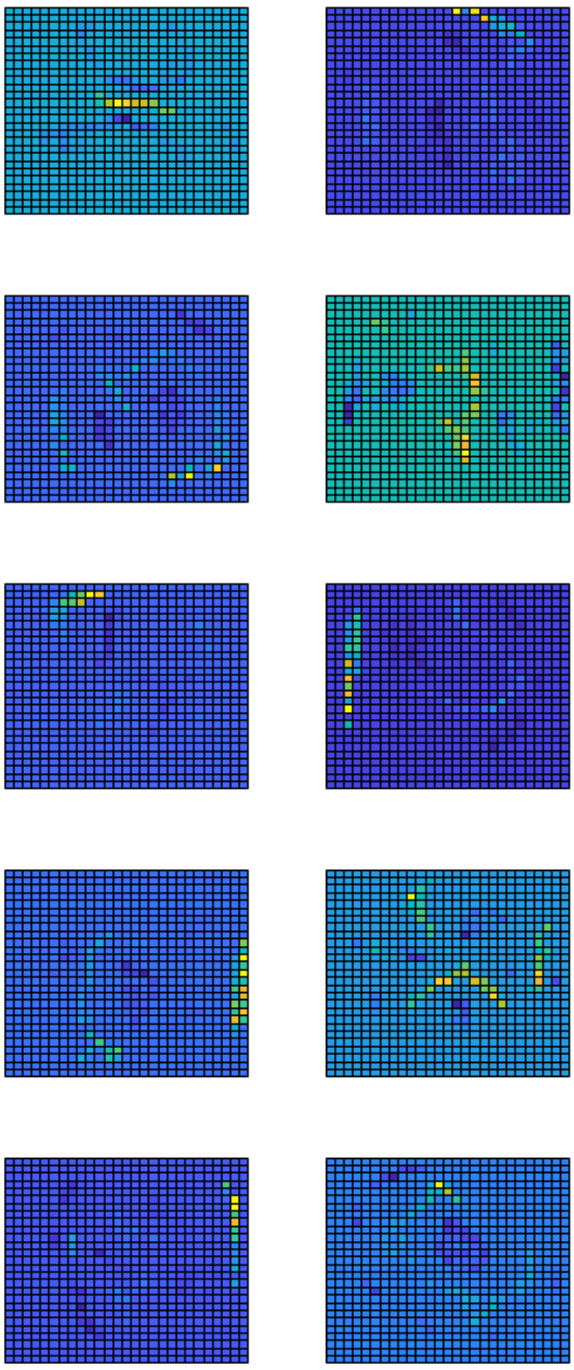

Figure 2: The most important pixels (first 100) for digits 1 to 0

Code

```
% Load the data
train=load('mnist_train.csv');
test=load('mnist_test.csv');
A=train(:,2:785); % train images
Atest=test(:,2:785); % test images
train_labels=train(:,1); % train labels
test_labels=test(:,1); % test labels
test_labels(test_labels==0)=10; % change 0s to 10s
B=zeros(60000,10);
for i=1:60000
    n=train_labels(i);
    if n==0
        n=10;
    end
    B(i,n)=1;
end

% Solve AX=B
X=zeros(784,10);
for i=1:10
    X(:,i)=lasso(A,B(:,i),'Lambda',0.01);
end

% Sort the pixels
X1=vecnorm(X,2,2); % 2-norm of each row
[~,idx]=sort(X1,'descend');
N=[50 100 200 400 700 783];
for i=1:length(N)
    X2=zeros(784,10);
    X2(idx(1:N(i)),:)=X(idx(1:N(i)),:); % pick first N pixels
    D=Atest*X2;
    [~,idx2]=max(D,[],2);
    accuracy(i)=sum(test_labels==idx2)/100;
end

figure(1)
X2=zeros(784,10);
X2(idx(1:100),:)=X(idx(1:100),:);
X1=vecnorm(X2,2,2);
X3=flipud(reshape(X1,28,28));
subplot(2,1,1);
pcolor(X3), axis off
subplot(2,1,2);
plot(N,accuracy,'LineWidth',1)
xlabel('Number of pixels','interpreter','latex')
ylabel('Accuracy','interpreter','latex')
set(gca,'Fontsize',[12])
```

```
48
49 figure(2)
50 for i=1:10
51     X1=X(:,i);
52     [¬,idx]=sort(abs(X1),'descend');
53     X2=zeros(784,1);
54     X2(idx(1:100))=X1(idx(1:100));
55     X3=reshape(X2,28,28);
56     subplot(5,2,i)
57     pcolor(X3), axis off
58 end
59
60 % Test the mapping for each digit
61 for i=1:10
62 X1=X(:,i);
63 [¬,idx]=sort(abs(X1),'descend');
64 X2=zeros(784,10);
65 X2(idx(1:100),:)=X(idx(1:100),:);
66 D=Atest*X2;
67 [¬,idx2]=max(D,[],2);
68 k=find(test_labels==i);
69 accuracy(i)=sum(idx2(k)==i)/sum(test_labels==i);
70 end
```